

**RNALib-2.5.1**

Generated by Doxygen 1.9.3



---

|   |           |
|---|-----------|
| <b>1 RNAlib-2.5.1</b>   | <b>1</b>  |
| 1.1 A Library for predicting and comparing RNA secondary structures . . . . . | 1         |
| 1.2 License . . . . .   | 1         |
| 1.3 Contributors . . . . .  | 2         |
| <b>2 Getting Started</b>  | <b>3</b>  |
| 2.1 Installation and Configuration . . . . .                                  | 3         |
| 2.1.1 Installing the ViennaRNA Package . . . . .                              | 3         |
| 2.1.1.1 Quick-start . . . . .   | 3         |
| 2.1.1.2 Installation without root privileges . . . . .                        | 3         |
| 2.1.1.3 Notes for MacOS X users . . . . .                                     | 4         |
| 2.1.2 Configuring RNAlib features . . . . .                                   | 4         |
| 2.1.2.1 Streaming SIMD Extension (SSE) support . . . . .                      | 5         |
| 2.1.2.2 Scripting Interfaces . . . . .  | 5         |
| 2.1.2.3 Cluster Analysis . . . . .  | 5         |
| 2.1.2.4 Kinfold . . . . .   | 5         |
| 2.1.2.5 RNAforester . . . . .   | 6         |
| 2.1.2.6 Kinwalker . . . . .   | 6         |
| 2.1.2.7 Link Time Optimization (LTO) . . . . .                                | 6         |
| 2.1.2.8 OpenMP support . . . . .  | 6         |
| 2.1.2.9 POSIX threads (pthread) support . . . . .                             | 6         |
| 2.1.2.10 SVM Z-score filter in RNAlfold . . . . .                             | 7         |
| 2.1.2.11 GNU Scientific Library . . . . .                                     | 7         |
| 2.1.2.12 Disable C11/C++11 feature support . . . . .                          | 7         |
| 2.1.2.13 Enable warnings for use of deprecated symbols . . . . .              | 7         |
| 2.1.2.14 Single precision partition function . . . . .                        | 7         |
| 2.1.2.15 Help . . . . .   | 8         |
| 2.1.3 Linking against RNAlib . . . . .  | 8         |
| 2.1.3.1 Compiler and Linker flags . . . . .                                   | 8         |
| 2.1.3.2 The pkg-config tool . . . . .   | 9         |
| 2.2 HelloWorld . . . . .  | 10        |
| 2.3 HelloWorld (Perl/Python) . . . . .  | 12        |
| 2.3.1 Perl5 . . . . .   | 12        |
| 2.3.2 Python . . . . .  | 12        |
| <b>3 Concepts and Algorithms</b>  | <b>15</b> |
| 3.1 RNA Structure . . . . .   | 16        |
| 3.1.1 RNA Structures . . . . .  | 16        |
| 3.1.2 Levels of Structure Abstraction . . . . .                               | 16        |
| 3.1.2.1 Primary Structure . . . . .   | 16        |
| 3.1.2.2 Secondary Structure . . . . .   | 16        |
| 3.1.2.3 Tertiary Structure . . . . .  | 16        |
| 3.1.2.4 Quaternary Structure . . . . .  | 16        |

---

|   |    |
|---|----|
| 3.1.2.5 Pseudo-Knots . . . . .  | 16 |
| 3.2 Distance Measures . . . . .   | 16 |
| 3.2.1 Functions for Tree Edit Distances . . . . .                         | 17 |
| 3.2.2 Functions for String Alignment . . . . .                            | 18 |
| 3.2.3 Functions for Comparison of Base Pair Probabilities . . . . .       | 18 |
| 3.3 Free Energy of Secondary Structures . . . . .                         | 18 |
| 3.3.1 Secondary Structure Loop Decomposition . . . . .                    | 19 |
| 3.3.1.1 Free Energy Evaluation API . . . . .                              | 20 |
| 3.3.2 Free Energy Parameters . . . . .                                    | 20 |
| 3.3.2.1 Free Energy Parameters Modification API . . . . .                 | 20 |
| 3.3.3 Fine-tuning of the Energy Evaluation Model . . . . .                | 20 |
| 3.4 Secondary Structure Folding Grammar . . . . .                         | 20 |
| 3.4.1 Secondary Structure Folding Recurrences . . . . .                   | 21 |
| 3.4.2 Additional Structural Domains . . . . .                             | 21 |
| 3.4.2.1 Structured Domains . . . . .                                      | 22 |
| 3.4.2.2 Unstructured Domains . . . . .                                    | 22 |
| 3.4.2.3 Domain Extension API . . . . .                                    | 23 |
| 3.4.3 Constraints on the Folding Grammar . . . . .                        | 23 |
| 3.4.3.1 Hard Constraints API . . . . .                                    | 23 |
| 3.4.3.2 Soft Constraints API . . . . .                                    | 24 |
| 3.5 RNA Secondary Structure Landscapes . . . . .                          | 24 |
| 3.5.1 The Neighborhood of a Secondary Structure . . . . .                 | 24 |
| 3.5.2 The Secondary Structure Landscape API . . . . .                     | 24 |
| 3.6 Minimum Free Energy Algorithm(s) . . . . .                            | 24 |
| 3.6.1 Zuker's Algorithm . . . . .   | 24 |
| 3.6.2 MFE for circular RNAs . . . . .                                     | 24 |
| 3.6.3 MFE Algorithm API . . . . .   | 24 |
| 3.7 Partition Function and Equilibrium Probability Algorithm(s) . . . . . | 25 |
| 3.7.1 Equilibrium Ensemble Statistics . . . . .                           | 25 |
| 3.7.2 Partition Function and Equilibrium Probability API . . . . .        | 25 |
| 3.8 Suboptimals and (other) Representative Structures . . . . .           | 26 |
| 3.8.1 Suboptimal Secondary Structures . . . . .                           | 26 |
| 3.8.2 Sampling Secondary Structures from the Ensemble . . . . .           | 26 |
| 3.8.3 Structure Enumeration and Sampling API . . . . .                    | 26 |
| 3.9 RNA-RNA Interaction . . . . .   | 26 |
| 3.9.1 <br> . . . . .  | 26 |
| 3.9.2 Concatenating RNA sequences . . . . .                               | 26 |
| 3.9.3 RNA-RNA interaction as a Stepwise Process . . . . .                 | 26 |
| 3.9.4 RNA-RNA Interaction API . . . . .                                   | 27 |
| 3.10 Locally Stable Secondary Structures . . . . .                        | 27 |
| 3.10.1 local_intro . . . . .  | 27 |
| 3.10.2 local_mfe . . . . .  | 27 |

|  |           |
|--|-----------|
| 3.10.3 local_pf . . . . .  | 27        |
| 3.10.4 Locally Stable Secondary Structure API . . . . .                            | 27        |
| 3.11 Comparative Structure Prediction . . . . .                                    | 27        |
| 3.11.1 Incorporate Evolutionary Information . . . . .                              | 27        |
| 3.11.2 Comparative Structure Prediction API . . . . .                              | 27        |
| 3.12 Classified DP variations . . . . .  | 27        |
| 3.12.1 The Idea of Classified Dynamic Programming . . . . .                        | 27        |
| 3.12.2 Distance Class Partitioning . . . . .                                       | 27        |
| 3.12.3 Density of States (DOS) . . . . .   | 28        |
| 3.12.4 Classified DP API . . . . .   | 28        |
| 3.13 RNA Sequence Design . . . . .   | 28        |
| 3.13.1 Generate Sequences that fold into particular Secondary Structures . . . . . | 28        |
| 3.13.2 RNA Sequence Design API . . . . .   | 28        |
| 3.14 Experimental Structure Probing Data . . . . .                                 | 28        |
| 3.14.1 Guide the Structure Prediction using Experimental Data . . . . .            | 28        |
| 3.14.1.1 SHAPE reactivities . . . . .  | 28        |
| 3.14.2 Structure Probing Data API . . . . .  | 28        |
| 3.15 Ligand Binding . . . . .  | 28        |
| 3.15.1 Small Molecules and Proteins that bind to specific RNA Structures . . . . . | 28        |
| 3.15.2 ligand_binding_api . . . . .  | 28        |
| 3.16 (Tertiary) Structure Motifs . . . . .   | 29        |
| 3.16.1 Incorporating Higher-Order (Tertiary) Structure Motifs . . . . .            | 29        |
| 3.16.2 RNA G-Quadruplexes . . . . .  | 29        |
| 3.16.3 (Tertiary) Structure Motif API . . . . .                                    | 29        |
| <b>4 I/O Formats</b> . . . . .   | <b>31</b> |
| 4.1 RNA Structure Notations . . . . .  | 31        |
| 4.1.1 Representations of Secondary Structures . . . . .                            | 31        |
| 4.1.1.1 Dot-Bracket Notation (a.k.a. Dot-Parenthesis Notation) . . . . .           | 31        |
| 4.1.1.2 Washington University Secondary Structure (WUSS) notation . . . . .        | 32        |
| 4.1.1.3 Abstract Shapes . . . . .  | 33        |
| 4.1.1.4 Tree Representations of Secondary Structures . . . . .                     | 33        |
| 4.1.2 Examples for Structure Parsing and Conversion . . . . .                      | 34        |
| 4.1.3 Structure Parsing and Conversion API . . . . .                               | 34        |
| 4.2 File Formats . . . . .   | 35        |
| 4.2.1 File formats for Multiple Sequence Alignments (MSA) . . . . .                | 35        |
| 4.2.1.1 ClustalW format . . . . .  | 35        |
| 4.2.1.2 Stockholm 1.0 format . . . . .   | 36        |
| 4.2.1.3 FASTA (Pearson) format . . . . .   | 36        |
| 4.2.1.4 MAF format . . . . .   | 37        |
| 4.2.2 File formats to manipulate the RNA folding grammar . . . . .                 | 38        |
| 4.2.2.1 Command Files . . . . .  | 38        |

|   |            |
|---|------------|
| 4.3 Plotting . . . . .  | 40         |
| 4.3.1 Producing secondary structure graphs . . . . .                      | 40         |
| 4.3.2 Producing (colored) dot plots for base pair probabilities . . . . . | 41         |
| 4.3.3 Producing (colored) alignments . . . . .                            | 42         |
| <b>5 Basic Data Structures</b>  | <b>43</b>  |
| 5.1 Sequence and Structure Data . . . . .                                 | 43         |
| 5.2 The 'Fold Compound'   | 43         |
| 5.3 Model Details . . . . .   | 43         |
| <b>6 API Features</b>   | <b>45</b>  |
| 6.1 RNAlib API v3.0 . . . . .   | 45         |
| 6.1.1 Introduction . . . . .  | 45         |
| 6.1.2 What are the major changes?   | 45         |
| 6.1.3 How to port your program to the new API . . . . .                   | 45         |
| 6.1.4 Some Examples using RNAlib API v3.0 . . . . .                       | 45         |
| 6.2 Callback Functions . . . . .  | 46         |
| 6.2.1 The purpose of Callback mechanisms . . . . .                        | 46         |
| 6.2.2 List of available Callbacks . . . . .                               | 46         |
| 6.3 Scripting Language interface(s)                                       | 47         |
| 6.3.1 Introduction . . . . .  | 47         |
| 6.3.2 Function Renaming . . . . .   | 47         |
| 6.3.2.1 Global Variables . . . . .  | 48         |
| 6.3.3 Object oriented Interface for Data Structures . . . . .             | 48         |
| 6.3.4 Examples . . . . .  | 48         |
| 6.3.5 SWIG generated Wrapper notes . . . . .                              | 48         |
| <b>7 Additional Utilities</b>   | <b>61</b>  |
| <b>8 Examples</b>   | <b>63</b>  |
| 8.1 C Examples . . . . .  | 63         |
| 8.1.1 Hello World Examples . . . . .                                      | 63         |
| 8.1.2 First Steps with the Fold Compound . . . . .                        | 64         |
| 8.1.3 Writing Callback Functions . . . . .                                | 65         |
| 8.1.4 Application of Soft Constraints . . . . .                           | 66         |
| 8.1.5 Other Examples . . . . .  | 66         |
| 8.1.6 Deprecated Examples . . . . .                                       | 67         |
| 8.2 Perl5 Examples . . . . .  | 68         |
| 8.3 Python Examples . . . . .   | 69         |
| <b>9 Contributing to the ViennaRNA Package</b>                            | <b>73</b>  |
| <b>10 Changelog</b>   | <b>75</b>  |
| <b>11 Deprecated List</b>   | <b>107</b> |

---

|  |            |
|--|------------|
| <b>12 Bug List</b>   | <b>119</b> |
| <b>13 Module Index</b>   | <b>121</b> |
| 13.1 The RNAlib API . . . . .                                      | 121        |
| <b>14 Data Structure Index</b>                                     | <b>125</b> |
| 14.1 Data Structures . . . . .                                     | 125        |
| <b>15 File Index</b>   | <b>127</b> |
| 15.1 File List . . . . .   | 127        |
| <b>16 Module Documentation</b>                                     | <b>133</b> |
| 16.1 Free Energy Evaluation . . . . .                              | 133        |
| 16.1.1 Detailed Description . . . . .                              | 133        |
| 16.1.2 Function Documentation . . . . .                            | 136        |
| 16.1.2.1 vrna_eval_structure() . . . . .                           | 136        |
| 16.1.2.2 vrna_eval_covar_structure() . . . . .                     | 136        |
| 16.1.2.3 vrna_eval_structure_verbose() . . . . .                   | 137        |
| 16.1.2.4 vrna_eval_structure_v() . . . . .                         | 137        |
| 16.1.2.5 vrna_eval_structure_pt() . . . . .                        | 138        |
| 16.1.2.6 vrna_eval_structure_pt_verbose() . . . . .                | 138        |
| 16.1.2.7 vrna_eval_structure_pt_v() . . . . .                      | 139        |
| 16.1.2.8 vrna_eval_structure_simple() . . . . .                    | 140        |
| 16.1.2.9 vrna_eval_circ_structure() . . . . .                      | 140        |
| 16.1.2.10 vrna_eval_gquad_structure() . . . . .                    | 141        |
| 16.1.2.11 vrna_eval_circ_gquad_structure() . . . . .               | 141        |
| 16.1.2.12 vrna_eval_structure_simple_verbose() . . . . .           | 142        |
| 16.1.2.13 vrna_eval_structure_simple_v() . . . . .                 | 142        |
| 16.1.2.14 vrna_eval_circ_structure_v() . . . . .                   | 143        |
| 16.1.2.15 vrna_eval_gquad_structure_v() . . . . .                  | 143        |
| 16.1.2.16 vrna_eval_circ_gquad_structure_v() . . . . .             | 144        |
| 16.1.2.17 vrna_eval_consensus_structure_simple() . . . . .         | 145        |
| 16.1.2.18 vrna_eval_circ_consensus_structure() . . . . .           | 145        |
| 16.1.2.19 vrna_eval_gquad_consensus_structure() . . . . .          | 146        |
| 16.1.2.20 vrna_eval_circ_gquad_consensus_structure() . . . . .     | 146        |
| 16.1.2.21 vrna_eval_consensus_structure_simple_verbose() . . . . . | 147        |
| 16.1.2.22 vrna_eval_consensus_structure_simple_v() . . . . .       | 148        |
| 16.1.2.23 vrna_eval_circ_consensus_structure_v() . . . . .         | 148        |
| 16.1.2.24 vrna_eval_gquad_consensus_structure_v() . . . . .        | 149        |
| 16.1.2.25 vrna_eval_circ_gquad_consensus_structure_v() . . . . .   | 150        |
| 16.1.2.26 vrna_eval_structure_pt_simple() . . . . .                | 151        |
| 16.1.2.27 vrna_eval_structure_pt_simple_verbose() . . . . .        | 151        |
| 16.1.2.28 vrna_eval_structure_pt_simple_v() . . . . .              | 152        |
| 16.1.2.29 vrna_eval_consensus_structure_pt_simple() . . . . .      | 152        |

---

|   |     |
|---|-----|
| 16.1.2.30 vrna_eval_consensus_structure_pt_simple_verbose() | 153 |
| 16.1.2.31 vrna_eval_consensus_structure_pt_simple_v()       | 153 |
| 16.2 Energy Evaluation for Individual Loops                 | 153 |
| 16.2.1 Detailed Description                                 | 153 |
| 16.2.2 Function Documentation                               | 154 |
| 16.2.2.1 vrna_eval_loop_pt()                                | 154 |
| 16.2.2.2 vrna_eval_loop_pt_v()                              | 154 |
| 16.3 Energy Evaluation for Atomic Moves                     | 155 |
| 16.3.1 Detailed Description                                 | 155 |
| 16.3.2 Function Documentation                               | 155 |
| 16.3.2.1 vrna_eval_move()                                   | 155 |
| 16.3.2.2 vrna_eval_move_pt()                                | 156 |
| 16.4 Deprecated Interface for Free Energy Evaluation        | 156 |
| 16.4.1 Detailed Description                                 | 156 |
| 16.4.2 Function Documentation                               | 157 |
| 16.4.2.1 energy_of_structure()                              | 157 |
| 16.4.2.2 energy_of_struct_par()                             | 158 |
| 16.4.2.3 energy_of_circ_structure()                         | 158 |
| 16.4.2.4 energy_of_circ_struct_par()                        | 159 |
| 16.4.2.5 energy_of_structure_pt()                           | 160 |
| 16.4.2.6 energy_of_struct_pt_par()                          | 160 |
| 16.4.2.7 energy_of_move()                                   | 161 |
| 16.4.2.8 energy_of_move_pt()                                | 161 |
| 16.4.2.9 loop_energy()                                      | 162 |
| 16.4.2.10 energy_of_struct()                                | 162 |
| 16.4.2.11 energy_of_struct_pt()                             | 163 |
| 16.4.2.12 energy_of_circ_struct()                           | 164 |
| 16.4.2.13 E_Stem()  | 164 |
| 16.4.2.14 exp_E_ExtLoop()                                   | 165 |
| 16.4.2.15 exp_E_Stem()                                      | 166 |
| 16.4.2.16 E_IntLoop()                                       | 166 |
| 16.4.2.17 exp_E_IntLoop()                                   | 167 |
| 16.5 The RNA Folding Grammar                                | 168 |
| 16.5.1 Detailed Description                                 | 168 |
| 16.5.2 Data Structure Documentation                         | 169 |
| 16.5.2.1 struct vrna_gr_aux_s                               | 169 |
| 16.6 Fine-tuning of the Implemented Models                  | 169 |
| 16.6.1 Detailed Description                                 | 169 |
| 16.6.2 Data Structure Documentation                         | 173 |
| 16.6.2.1 struct vrna_md_s                                   | 173 |
| 16.6.3 Macro Definition Documentation                       | 176 |
| 16.6.3.1 VRNA_MODEL_DEFAULT_TEMPERATURE                     | 176 |

---

|  |     |
|--|-----|
| 16.6.3.2 VRNA_MODEL_DEFAULT_PF_SCALE . . . . .         | 176 |
| 16.6.3.3 VRNA_MODEL_DEFAULT_BETA_SCALE . . . . .       | 176 |
| 16.6.3.4 VRNA_MODEL_DEFAULT_DANGLES . . . . .          | 177 |
| 16.6.3.5 VRNA_MODEL_DEFAULT_SPECIAL_HP . . . . .       | 177 |
| 16.6.3.6 VRNA_MODEL_DEFAULT_NO_LP . . . . .            | 177 |
| 16.6.3.7 VRNA_MODEL_DEFAULT_NO_GU . . . . .            | 177 |
| 16.6.3.8 VRNA_MODEL_DEFAULT_NO_GU_CLOSURE . . . . .    | 177 |
| 16.6.3.9 VRNA_MODEL_DEFAULT_CIRC . . . . .             | 177 |
| 16.6.3.10 VRNA_MODEL_DEFAULT_GQUAD . . . . .           | 178 |
| 16.6.3.11 VRNA_MODEL_DEFAULT_UNIQ_ML . . . . .         | 178 |
| 16.6.3.12 VRNA_MODEL_DEFAULT_ENERGY_SET . . . . .      | 178 |
| 16.6.3.13 VRNA_MODEL_DEFAULT_BACKTRACK . . . . .       | 178 |
| 16.6.3.14 VRNA_MODEL_DEFAULT_BACKTRACK_TYPE . . . . .  | 178 |
| 16.6.3.15 VRNA_MODEL_DEFAULT_COMPUTE_BPP . . . . .     | 178 |
| 16.6.3.16 VRNA_MODEL_DEFAULT_MAX_BP_SPAN . . . . .     | 179 |
| 16.6.3.17 VRNA_MODEL_DEFAULT_WINDOW_SIZE . . . . .     | 179 |
| 16.6.3.18 VRNA_MODEL_DEFAULT_LOG_ML . . . . .          | 179 |
| 16.6.3.19 VRNA_MODEL_DEFAULT_ALI_OLD_EN . . . . .      | 179 |
| 16.6.3.20 VRNA_MODEL_DEFAULT_ALI_RIBO . . . . .        | 179 |
| 16.6.3.21 VRNA_MODEL_DEFAULT_ALI_CV_FACT . . . . .     | 179 |
| 16.6.3.22 VRNA_MODEL_DEFAULT_ALI_NC_FACT . . . . .     | 180 |
| 16.6.4 Function Documentation . . . . .                | 180 |
| 16.6.4.1 vrna_md_set_default() . . . . .               | 180 |
| 16.6.4.2 vrna_md_update() . . . . .                    | 180 |
| 16.6.4.3 vrna_md_copy() . . . . .                      | 180 |
| 16.6.4.4 vrna_md_option_string() . . . . .             | 181 |
| 16.6.4.5 vrna_md_defaults_reset() . . . . .            | 181 |
| 16.6.4.6 vrna_md_defaults_temperature() . . . . .      | 181 |
| 16.6.4.7 vrna_md_defaults_temperature_get() . . . . .  | 182 |
| 16.6.4.8 vrna_md_defaults_betaScale() . . . . .        | 182 |
| 16.6.4.9 vrna_md_defaults_betaScale_get() . . . . .    | 182 |
| 16.6.4.10 vrna_md_defaults_dangles() . . . . .         | 183 |
| 16.6.4.11 vrna_md_defaults_dangles_get() . . . . .     | 183 |
| 16.6.4.12 vrna_md_defaults_special_hp() . . . . .      | 183 |
| 16.6.4.13 vrna_md_defaults_special_hp_get() . . . . .  | 183 |
| 16.6.4.14 vrna_md_defaults_noLP() . . . . .            | 184 |
| 16.6.4.15 vrna_md_defaults_noLP_get() . . . . .        | 184 |
| 16.6.4.16 vrna_md_defaults_noGU() . . . . .            | 184 |
| 16.6.4.17 vrna_md_defaults_noGU_get() . . . . .        | 185 |
| 16.6.4.18 vrna_md_defaults_noGUclosure() . . . . .     | 185 |
| 16.6.4.19 vrna_md_defaults_noGUclosure_get() . . . . . | 185 |
| 16.6.4.20 vrna_md_defaults_logML() . . . . .           | 185 |

|   |            |
|---|------------|
| 16.6.4.21 vrna_md_defaults_logML_get()          | 186        |
| 16.6.4.22 vrna_md_defaults_circ()               | 186        |
| 16.6.4.23 vrna_md_defaults_circ_get()           | 186        |
| 16.6.4.24 vrna_md_defaults_gquad()              | 186        |
| 16.6.4.25 vrna_md_defaults_gquad_get()          | 187        |
| 16.6.4.26 vrna_md_defaults_uniq_ML()            | 187        |
| 16.6.4.27 vrna_md_defaults_uniq_ML_get()        | 187        |
| 16.6.4.28 vrna_md_defaults_energy_set()         | 188        |
| 16.6.4.29 vrna_md_defaults_energy_set_get()     | 188        |
| 16.6.4.30 vrna_md_defaults_backtrack()          | 188        |
| 16.6.4.31 vrna_md_defaults_backtrack_get()      | 189        |
| 16.6.4.32 vrna_md_defaults_backtrack_type()     | 189        |
| 16.6.4.33 vrna_md_defaults_backtrack_type_get() | 189        |
| 16.6.4.34 vrna_md_defaults_compute_bpp()        | 189        |
| 16.6.4.35 vrna_md_defaults_compute_bpp_get()    | 190        |
| 16.6.4.36 vrna_md_defaults_max_bp_span()        | 190        |
| 16.6.4.37 vrna_md_defaults_max_bp_span_get()    | 190        |
| 16.6.4.38 vrna_md_defaults_min_loop_size()      | 190        |
| 16.6.4.39 vrna_md_defaults_min_loop_size_get()  | 191        |
| 16.6.4.40 vrna_md_defaults_window_size()        | 191        |
| 16.6.4.41 vrna_md_defaults_window_size_get()    | 191        |
| 16.6.4.42 vrna_md_defaults_oldAliEn()           | 192        |
| 16.6.4.43 vrna_md_defaults_oldAliEn_get()       | 192        |
| 16.6.4.44 vrna_md_defaults_ribo()               | 192        |
| 16.6.4.45 vrna_md_defaults_ribo_get()           | 193        |
| 16.6.4.46 vrna_md_defaults_cv_fact()            | 193        |
| 16.6.4.47 vrna_md_defaults_cv_fact_get()        | 193        |
| 16.6.4.48 vrna_md_defaults_nc_fact()            | 193        |
| 16.6.4.49 vrna_md_defaults_nc_fact_get()        | 194        |
| 16.6.4.50 vrna_md_defaults_sfact()              | 194        |
| 16.6.4.51 vrna_md_defaults_sfact_get()          | 194        |
| 16.6.4.52 set_model_details()                   | 194        |
| <b>16.6.5 Variable Documentation</b>            | <b>195</b> |
| 16.6.5.1 temperature                            | 195        |
| 16.6.5.2 pf_scale                               | 195        |
| 16.6.5.3 dangles                                | 195        |
| 16.6.5.4 tetra_loop                             | 196        |
| 16.6.5.5 noLonelyPairs                          | 196        |
| 16.6.5.6 energy_set                             | 196        |
| 16.6.5.7 do_backtrack                           | 196        |
| 16.6.5.8 backtrack_type                         | 196        |
| 16.6.5.9 nonstandards                           | 196        |

---

|  |     |
|--|-----|
| 16.6.5.10 max_bp_span . . . . .                                      | 196 |
| 16.7 Energy Parameters . . . . .                                     | 196 |
| 16.7.1 Detailed Description . . . . .                                | 197 |
| 16.7.2 Data Structure Documentation . . . . .                        | 198 |
| 16.7.2.1 struct vrna_param_s . . . . .                               | 198 |
| 16.7.2.2 struct vrna_exp_param_s . . . . .                           | 198 |
| 16.7.3 Typedef Documentation . . . . .                               | 199 |
| 16.7.3.1 paramT . . . . .  | 199 |
| 16.7.3.2 pf_paramT . . . . .   | 199 |
| 16.7.4 Function Documentation . . . . .                              | 199 |
| 16.7.4.1 vrna_params() . . . . .                                     | 200 |
| 16.7.4.2 vrna_params_copy() . . . . .                                | 200 |
| 16.7.4.3 vrna_exp_params() . . . . .                                 | 200 |
| 16.7.4.4 vrna_exp_params_comparative() . . . . .                     | 201 |
| 16.7.4.5 vrna_exp_params_copy() . . . . .                            | 201 |
| 16.7.4.6 vrna_params_subst() . . . . .                               | 202 |
| 16.7.4.7 vrna_exp_params_subst() . . . . .                           | 202 |
| 16.7.4.8 vrna_exp_params_rescale() . . . . .                         | 203 |
| 16.7.4.9 vrna_params_reset() . . . . .                               | 203 |
| 16.7.4.10 vrna_exp_params_reset() . . . . .                          | 204 |
| 16.7.4.11 get_scaled_pf_parameters() . . . . .                       | 204 |
| 16.7.4.12 get_boltzmann_factors() . . . . .                          | 205 |
| 16.7.4.13 get_boltzmann_factor_copy() . . . . .                      | 205 |
| 16.7.4.14 get_scaled_alipf_parameters() . . . . .                    | 206 |
| 16.7.4.15 get_boltzmann_factors_ali() . . . . .                      | 206 |
| 16.7.4.16 scale_parameters() . . . . .                               | 206 |
| 16.7.4.17 get_scaled_parameters() . . . . .                          | 206 |
| 16.8 Extending the Folding Grammar with Additional Domains . . . . . | 207 |
| 16.8.1 Detailed Description . . . . .                                | 207 |
| 16.9 Unstructured Domains . . . . .                                  | 207 |
| 16.9.1 Detailed Description . . . . .                                | 207 |
| 16.9.2 Data Structure Documentation . . . . .                        | 209 |
| 16.9.2.1 struct vrna_unstructured_domain_s . . . . .                 | 209 |
| 16.9.3 Typedef Documentation . . . . .                               | 210 |
| 16.9.3.1 vrna_callback_ud_energy . . . . .                           | 210 |
| 16.9.3.2 vrna_callback_ud_exp_energy . . . . .                       | 210 |
| 16.9.3.3 vrna_callback_ud_production . . . . .                       | 211 |
| 16.9.3.4 vrna_callback_ud_exp_production . . . . .                   | 211 |
| 16.9.3.5 vrna_callback_ud_probs_add . . . . .                        | 211 |
| 16.9.3.6 vrna_callback_ud_probs_get . . . . .                        | 211 |
| 16.9.4 Function Documentation . . . . .                              | 211 |
| 16.9.4.1 vrna_ud_motifs_centroid() . . . . .                         | 212 |

---

|   |     |
|---|-----|
| 16.9.4.2 vrna_ud_motifs_MEA()                   | 212 |
| 16.9.4.3 vrna_ud_motifs_MFE()                   | 213 |
| 16.9.4.4 vrna_ud_add_motif()                    | 213 |
| 16.9.4.5 vrna_ud_remove()                       | 214 |
| 16.9.4.6 vrna_ud_set_data()                     | 214 |
| 16.9.4.7 vrna_ud_set_prod_rule_cb()             | 214 |
| 16.9.4.8 vrna_ud_set_exp_prod_rule_cb()         | 215 |
| 16.10 Structured Domains                        | 216 |
| 16.10.1 Detailed Description                    | 216 |
| 16.11 Constraining the RNA Folding Grammar      | 216 |
| 16.11.1 Detailed Description                    | 216 |
| 16.11.2 Macro Definition Documentation          | 219 |
| 16.11.2.1 VRNA_CONSTRAINT_FILE                  | 220 |
| 16.11.2.2 VRNA_CONSTRAINT_SOFT_MFE              | 220 |
| 16.11.2.3 VRNA_CONSTRAINT_SOFT_PF               | 220 |
| 16.11.2.4 VRNA_DECOMP_PAIR_HP                   | 220 |
| 16.11.2.5 VRNA_DECOMP_PAIR_IL                   | 221 |
| 16.11.2.6 VRNA_DECOMP_PAIR_DL                   | 221 |
| 16.11.2.7 VRNA_DECOMP_DL_DL_DL                  | 222 |
| 16.11.2.8 VRNA_DECOMP_DL_STEM                   | 222 |
| 16.11.2.9 VRNA_DECOMP_DL_DL                     | 223 |
| 16.11.2.10 VRNA_DECOMP_DL_UP                    | 223 |
| 16.11.2.11 VRNA_DECOMP_DL_DL_STEM               | 224 |
| 16.11.2.12 VRNA_DECOMP_DL_COAXIAL               | 224 |
| 16.11.2.13 VRNA_DECOMP_DL_COAXIAL_ENC           | 224 |
| 16.11.2.14 VRNA_DECOMP_EXT_EXT                  | 225 |
| 16.11.2.15 VRNA_DECOMP_EXT_UP                   | 225 |
| 16.11.2.16 VRNA_DECOMP_EXT_STEM                 | 225 |
| 16.11.2.17 VRNA_DECOMP_EXT_EXT_EXT              | 226 |
| 16.11.2.18 VRNA_DECOMP_EXT_STEM_EXT             | 226 |
| 16.11.2.19 VRNA_DECOMP_EXT_EXT_STEM             | 226 |
| 16.11.2.20 VRNA_DECOMP_EXT_EXT_STEM1            | 227 |
| 16.11.3 Function Documentation                  | 227 |
| 16.11.3.1 vrna_constraints_add()                | 227 |
| 16.11.3.2 vrna_message_constraint_options()     | 228 |
| 16.11.3.3 vrna_message_constraint_options_all() | 228 |
| 16.12 Hard Constraints                          | 229 |
| 16.12.1 Detailed Description                    | 229 |
| 16.12.2 Data Structure Documentation            | 230 |
| 16.12.2.1 struct vrna_hc_s                      | 230 |
| 16.12.2.2 struct vrna_hc_up_s                   | 231 |
| 16.12.3 Macro Definition Documentation          | 231 |

---

|   |     |
|---|-----|
| 16.12.3.1 VRNA_CONSTRAINT_DB . . . . .                | 232 |
| 16.12.3.2 VRNA_CONSTRAINT_DB_ENFORCE_BP . . . . .     | 232 |
| 16.12.3.3 VRNA_CONSTRAINT_DB_PIPE . . . . .           | 232 |
| 16.12.3.4 VRNA_CONSTRAINT_DB_DOT . . . . .            | 232 |
| 16.12.3.5 VRNA_CONSTRAINT_DB_X . . . . .              | 232 |
| 16.12.3.6 VRNA_CONSTRAINT_DB_RND_BRACK . . . . .      | 233 |
| 16.12.3.7 VRNA_CONSTRAINT_DB_INTRAMOL . . . . .       | 233 |
| 16.12.3.8 VRNA_CONSTRAINT_DB_INTERMOL . . . . .       | 233 |
| 16.12.3.9 VRNA_CONSTRAINT_DB_GQUAD . . . . .          | 233 |
| 16.12.3.10 VRNA_CONSTRAINT_DB_WUSS . . . . .          | 233 |
| 16.12.3.11 VRNA_CONSTRAINT_DB_DEFAULT . . . . .       | 234 |
| 16.12.4 Typedef Documentation . . . . .               | 234 |
| 16.12.4.1 vrna_callback_hc_evaluate . . . . .         | 234 |
| 16.12.5 Function Documentation . . . . .              | 235 |
| 16.12.5.1 vrna_hc_init() . . . . .                    | 235 |
| 16.12.5.2 vrna_hc_add_up() . . . . .                  | 235 |
| 16.12.5.3 vrna_hc_add_up_batch() . . . . .            | 235 |
| 16.12.5.4 vrna_hc_add_bp() . . . . .                  | 236 |
| 16.12.5.5 vrna_hc_add_bp_nonspecific() . . . . .      | 236 |
| 16.12.5.6 vrna_hc_free() . . . . .                    | 237 |
| 16.12.5.7 vrna_hc_add_from_db() . . . . .             | 237 |
| 16.13 Soft Constraints . . . . .                      | 237 |
| 16.13.1 Detailed Description . . . . .                | 237 |
| 16.13.2 Data Structure Documentation . . . . .        | 238 |
| 16.13.2.1 struct vrna_sc_s . . . . .                  | 238 |
| 16.13.3 Typedef Documentation . . . . .               | 240 |
| 16.13.3.1 vrna_callback_sc_energy . . . . .           | 240 |
| 16.13.3.2 vrna_callback_sc_exp_energy . . . . .       | 240 |
| 16.13.3.3 vrna_callback_sc_backtrack . . . . .        | 241 |
| 16.13.4 Function Documentation . . . . .              | 242 |
| 16.13.4.1 vrna_sc_init() . . . . .                    | 242 |
| 16.13.4.2 vrna_sc_set_bp() . . . . .                  | 242 |
| 16.13.4.3 vrna_sc_add_bp() . . . . .                  | 243 |
| 16.13.4.4 vrna_sc_set_up() . . . . .                  | 243 |
| 16.13.4.5 vrna_sc_add_up() . . . . .                  | 244 |
| 16.13.4.6 vrna_sc_remove() . . . . .                  | 244 |
| 16.13.4.7 vrna_sc_free() . . . . .                    | 245 |
| 16.13.4.8 vrna_sc_add_data() . . . . .                | 245 |
| 16.13.4.9 vrna_sc_add_f() . . . . .                   | 245 |
| 16.13.4.10 vrna_sc_add_bt() . . . . .                 | 246 |
| 16.13.4.11 vrna_sc_add_exp_f() . . . . .              | 246 |
| 16.14 The RNA Secondary Structure Landscape . . . . . | 247 |

|   |     |
|---|-----|
| 16.14.1 Detailed Description                                  | 247 |
| 16.15 Minimum Free Energy (MFE) Algorithms                    | 247 |
| 16.15.1 Detailed Description                                  | 247 |
| 16.16 Partition Function and Equilibrium Properties           | 248 |
| 16.16.1 Detailed Description                                  | 248 |
| 16.16.2 Function Documentation                                | 248 |
| 16.16.2.1 vrna_pf_float_precision()                           | 249 |
| 16.17 Global MFE Prediction                                   | 249 |
| 16.17.1 Detailed Description                                  | 249 |
| 16.17.2 Function Documentation                                | 250 |
| 16.17.2.1 vrna_mfe()  | 250 |
| 16.17.2.2 vrna_mfe_dimer()                                    | 250 |
| 16.17.2.3 vrna_fold()   | 251 |
| 16.17.2.4 vrna_circfold()                                     | 252 |
| 16.17.2.5 vrna_alifold()                                      | 252 |
| 16.17.2.6 vrna_circalifold()                                  | 253 |
| 16.17.2.7 vrna_cofold()                                       | 253 |
| 16.18 Local (sliding window) MFE Prediction                   | 254 |
| 16.18.1 Detailed Description                                  | 254 |
| 16.18.2 Typedef Documentation                                 | 255 |
| 16.18.2.1 vrna_mfe_window_callback                            | 255 |
| 16.18.3 Function Documentation                                | 256 |
| 16.18.3.1 vrna_mfe_window()                                   | 256 |
| 16.18.3.2 vrna_mfe_window_zscore()                            | 256 |
| 16.18.3.3 vrna_Lfold()  | 257 |
| 16.18.3.4 vrna_Lfoldz()                                       | 257 |
| 16.19 Backtracking MFE structures                             | 258 |
| 16.19.1 Detailed Description                                  | 258 |
| 16.19.2 Function Documentation                                | 259 |
| 16.19.2.1 vrna_backtrack5()                                   | 259 |
| 16.19.2.2 vrna_BT_hp_loop()                                   | 259 |
| 16.19.2.3 vrna_BT_mb_loop()                                   | 260 |
| 16.20 Global Partition Function and Equilibrium Probabilities | 260 |
| 16.20.1 Detailed Description                                  | 260 |
| 16.20.2 Data Structure Documentation                          | 261 |
| 16.20.2.1 struct vrna_dimer_pf_s                              | 261 |
| 16.20.2.2 struct vrna_multimer_pf_s                           | 262 |
| 16.20.3 Function Documentation                                | 262 |
| 16.20.3.1 vrna_pf()   | 262 |
| 16.20.3.2 vrna_pf_dimer()                                     | 263 |
| 16.20.3.3 vrna_pf_fold()                                      | 263 |
| 16.20.3.4 vrna_pf_circfold()                                  | 264 |

---

|   |     |
|---|-----|
| 16.20.3.5 vrna_pf_alifold()   | 264 |
| 16.20.3.6 vrna_pf_circalifold()   | 265 |
| 16.20.3.7 vrna_plist_from_probs()   | 266 |
| 16.20.3.8 vrna_pf_co_fold()   | 266 |
| 16.21 Local (sliding window) Partition Function and Equilibrium Probabilities | 267 |
| 16.21.1 Detailed Description  | 267 |
| 16.21.2 Macro Definition Documentation  | 268 |
| 16.21.2.1 VRNA_PROBS_WINDOW_BPP   | 268 |
| 16.21.2.2 VRNA_PROBS_WINDOW_UP  | 268 |
| 16.21.2.3 VRNA_PROBS_WINDOW_STACKP  | 269 |
| 16.21.2.4 VRNA_PROBS_WINDOW_UP_SPLIT  | 269 |
| 16.21.2.5 VRNA_PROBS_WINDOW_PF  | 269 |
| 16.21.3 Typedef Documentation   | 269 |
| 16.21.3.1 vrna_probs_window_callback  | 270 |
| 16.21.4 Function Documentation  | 270 |
| 16.21.4.1 vrna_probs_window()   | 270 |
| 16.21.4.2 vrna_pfl_fold()   | 271 |
| 16.21.4.3 vrna_pfl_fold_cb()  | 272 |
| 16.21.4.4 vrna_pfl_fold_up()  | 272 |
| 16.21.4.5 vrna_pfl_fold_up_cb()   | 273 |
| 16.22 Suboptimals and Representative Structures                               | 274 |
| 16.22.1 Detailed Description  | 274 |
| 16.23 Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989    | 274 |
| 16.23.1 Detailed Description  | 274 |
| 16.23.2 Function Documentation  | 275 |
| 16.23.2.1 zukersubopt()   | 275 |
| 16.23.2.2 zukersubopt_par()   | 275 |
| 16.23.2.3 vrna_subopt_zuker()   | 275 |
| 16.24 Suboptimal Structures within an Energy Band around the MFE              | 276 |
| 16.24.1 Detailed Description  | 276 |
| 16.24.2 Typedef Documentation   | 276 |
| 16.24.2.1 vrna_subopt_callback  | 276 |
| 16.24.3 Function Documentation  | 277 |
| 16.24.3.1 vrna_subopt()   | 277 |
| 16.24.3.2 vrna_subopt_cb()  | 277 |
| 16.24.3.3 subopt()  | 278 |
| 16.24.3.4 subopt_circ()   | 279 |
| 16.25 Random Structure Samples from the Ensemble                              | 279 |
| 16.25.1 Detailed Description  | 279 |
| 16.25.2 Macro Definition Documentation  | 281 |
| 16.25.2.1 VRNA_PBACKTRACK_DEFAULT   | 281 |
| 16.25.2.2 VRNA_PBACKTRACK_NON_REDUNDANT                                       | 281 |

---

|  |     |
|--|-----|
| 16.25.3 Typedef Documentation . . . . .  | 281 |
| 16.25.3.1 vrna_boltzmann_sampling_callback . . . . .                           | 281 |
| 16.25.3.2 vrna_pbacktrack_mem_t . . . . .                                      | 281 |
| 16.25.4 Function Documentation . . . . .                                       | 282 |
| 16.25.4.1 vrna_pbacktrack5() . . . . .   | 282 |
| 16.25.4.2 vrna_pbacktrack5_num() . . . . .                                     | 283 |
| 16.25.4.3 vrna_pbacktrack5_cb() . . . . .                                      | 284 |
| 16.25.4.4 vrna_pbacktrack5_resume() . . . . .                                  | 285 |
| 16.25.4.5 vrna_pbacktrack5_resume_cb() . . . . .                               | 286 |
| 16.25.4.6 vrna_pbacktrack() . . . . .  | 288 |
| 16.25.4.7 vrna_pbacktrack_num() . . . . .                                      | 288 |
| 16.25.4.8 vrna_pbacktrack_cb() . . . . .                                       | 289 |
| 16.25.4.9 vrna_pbacktrack_resume() . . . . .                                   | 291 |
| 16.25.4.10 vrna_pbacktrack_resume_cb() . . . . .                               | 292 |
| 16.25.4.11 vrna_pbacktrack_sub() . . . . .                                     | 293 |
| 16.25.4.12 vrna_pbacktrack_sub_num() . . . . .                                 | 294 |
| 16.25.4.13 vrna_pbacktrack_sub_cb() . . . . .                                  | 295 |
| 16.25.4.14 vrna_pbacktrack_sub_resume() . . . . .                              | 296 |
| 16.25.4.15 vrna_pbacktrack_sub_resume_cb() . . . . .                           | 298 |
| 16.25.4.16 vrna_pbacktrack_mem_free() . . . . .                                | 299 |
| 16.26 Compute the Structure with Maximum Expected Accuracy (MEA) . . . . .     | 299 |
| 16.26.1 Detailed Description . . . . .   | 299 |
| 16.26.2 Function Documentation . . . . .                                       | 300 |
| 16.26.2.1 vrna_MEAl . . . . .  | 300 |
| 16.26.2.2 vrna_MEAl_from_plist() . . . . .                                     | 300 |
| 16.26.2.3 MEAl . . . . .   | 301 |
| 16.27 Compute the Centroid Structure . . . . .                                 | 301 |
| 16.27.1 Detailed Description . . . . .   | 301 |
| 16.27.2 Function Documentation . . . . .                                       | 302 |
| 16.27.2.1 vrna_centroid() . . . . .  | 302 |
| 16.27.2.2 vrna_centroid_from_plist() . . . . .                                 | 302 |
| 16.27.2.3 vrna_centroid_from_probs() . . . . .                                 | 303 |
| 16.28 RNA-RNA Interaction . . . . .  | 303 |
| 16.28.1 Detailed Description . . . . .   | 303 |
| 16.29 Classified Dynamic Programming Variants . . . . .                        | 303 |
| 16.29.1 Detailed Description . . . . .   | 303 |
| 16.30 Distance Based Partitioning of the Secondary Structure Space . . . . .   | 304 |
| 16.30.1 Detailed Description . . . . .   | 304 |
| 16.31 Computing MFE representatives of a Distance Based Partitioning . . . . . | 304 |
| 16.31.1 Detailed Description . . . . .   | 304 |
| 16.31.2 Data Structure Documentation . . . . .                                 | 305 |
| 16.31.2.1 struct vrna_sol_TwoD_t . . . . .                                     | 305 |

---

|  |     |
|--|-----|
| 16.31.2.2 struct TwoDfold_vars . . . . .   | 305 |
| 16.31.3 Typedef Documentation . . . . .  | 306 |
| 16.31.3.1 vrna_sol_TwoD_t . . . . .  | 306 |
| 16.31.3.2 TwoDfold_vars . . . . .  | 306 |
| 16.31.4 Function Documentation . . . . .   | 307 |
| 16.31.4.1 vrna_mfe_TwoD() . . . . .  | 307 |
| 16.31.4.2 vrna_backtrack5_TwoD() . . . . .   | 307 |
| 16.31.4.3 get_TwoDfold_variables() . . . . .   | 308 |
| 16.31.4.4 destroy_TwoDfold_variables() . . . . .                                       | 308 |
| 16.31.4.5 TwoDfoldList() . . . . .   | 309 |
| 16.31.4.6 TwoDfold_backtrack_f5() . . . . .  | 309 |
| 16.32 Computing Partition Functions of a Distance Based Partitioning . . . . .         | 310 |
| 16.32.1 Detailed Description . . . . .   | 310 |
| 16.32.2 Data Structure Documentation . . . . .   | 310 |
| 16.32.2.1 struct vrna_sol_TwoD_pf_t . . . . .  | 310 |
| 16.32.3 Typedef Documentation . . . . .  | 310 |
| 16.32.3.1 vrna_sol_TwoD_pf_t . . . . .   | 310 |
| 16.32.4 Function Documentation . . . . .   | 311 |
| 16.32.4.1 vrna_pf_TwoD() . . . . .   | 311 |
| 16.33 Stochastic Backtracking of Structures from Distance Based Partitioning . . . . . | 311 |
| 16.33.1 Detailed Description . . . . .   | 311 |
| 16.33.2 Function Documentation . . . . .   | 312 |
| 16.33.2.1 vrna_pbacktrack_TwoD() . . . . .   | 312 |
| 16.33.2.2 vrna_pbacktrack5_TwoD() . . . . .  | 312 |
| 16.34 Predicting various thermodynamic properties . . . . .                            | 313 |
| 16.34.1 Detailed Description . . . . .   | 313 |
| 16.34.2 Data Structure Documentation . . . . .   | 315 |
| 16.34.2.1 struct vrna_heat_capacity_s . . . . .  | 315 |
| 16.34.3 Typedef Documentation . . . . .  | 315 |
| 16.34.3.1 vrna_heat_capacity_callback . . . . .  | 315 |
| 16.34.3.2 vrna_heat_capacity_t . . . . .   | 315 |
| 16.34.4 Function Documentation . . . . .   | 315 |
| 16.34.4.1 vrna_mean_bp_distance_pr() . . . . .   | 315 |
| 16.34.4.2 vrna_mean_bp_distance() . . . . .  | 316 |
| 16.34.4.3 vrna_ensemble_defect_pt() . . . . .  | 316 |
| 16.34.4.4 vrna_ensemble_defect() . . . . .   | 317 |
| 16.34.4.5 vrna_positional_entropy() . . . . .  | 318 |
| 16.34.4.6 vrna_stack_prob() . . . . .  | 318 |
| 16.34.4.7 vrna_pf_dimer_probs() . . . . .  | 319 |
| 16.34.4.8 vrna_pr_structure() . . . . .  | 319 |
| 16.34.4.9 vrna_pr_energy() . . . . .   | 320 |
| 16.34.4.10 vrna_heat_capacity() . . . . .  | 320 |

---

|  |     |
|--|-----|
| 16.34.4.11 <code>vrna_heat_capacity_cb()</code>                    | 321 |
| 16.34.4.12 <code>vrna_heat_capacity_simple()</code>                | 321 |
| 16.35 Compute the Density of States                                | 322 |
| 16.35.1 Detailed Description                                       | 322 |
| 16.35.2 Variable Documentation                                     | 322 |
| 16.35.2.1 <code>density_of_states</code>                           | 322 |
| 16.36 Inverse Folding (Design)                                     | 323 |
| 16.36.1 Detailed Description                                       | 323 |
| 16.36.2 Function Documentation                                     | 323 |
| 16.36.2.1 <code>inverse_fold()</code>                              | 323 |
| 16.36.2.2 <code>inverse_pf_fold()</code>                           | 324 |
| 16.36.3 Variable Documentation                                     | 324 |
| 16.36.3.1 <code>final_cost</code>                                  | 324 |
| 16.36.3.2 <code>give_up</code>                                     | 324 |
| 16.36.3.3 <code>inv_verbose</code>                                 | 324 |
| 16.37 Neighborhood Relation and Move Sets for Secondary Structures | 324 |
| 16.37.1 Detailed Description                                       | 324 |
| 16.37.2 Data Structure Documentation                               | 327 |
| 16.37.2.1 <code>struct vrna_move_s</code>                          | 327 |
| 16.37.3 Macro Definition Documentation                             | 328 |
| 16.37.3.1 <code>VRNA_MOVESET_INSERTION</code>                      | 328 |
| 16.37.3.2 <code>VRNA_MOVESET_DELETION</code>                       | 328 |
| 16.37.3.3 <code>VRNA_MOVESET_SHIFT</code>                          | 328 |
| 16.37.3.4 <code>VRNA_MOVESET_NO_LP</code>                          | 328 |
| 16.37.3.5 <code>VRNA_MOVESET_DEFAULT</code>                        | 329 |
| 16.37.3.6 <code>VRNA_NEIGHBOR_CHANGE</code>                        | 329 |
| 16.37.3.7 <code>VRNA_NEIGHBOR_INVALID</code>                       | 329 |
| 16.37.3.8 <code>VRNA_NEIGHBOR_NEW</code>                           | 329 |
| 16.37.4 Typedef Documentation                                      | 329 |
| 16.37.4.1 <code>vrna_callback_move_update</code>                   | 329 |
| 16.37.5 Function Documentation                                     | 330 |
| 16.37.5.1 <code>vrna_move_init()</code>                            | 330 |
| 16.37.5.2 <code>vrna_move_list_free()</code>                       | 330 |
| 16.37.5.3 <code>vrna_move_apply()</code>                           | 330 |
| 16.37.5.4 <code>vrna_move_is_removal()</code>                      | 330 |
| 16.37.5.5 <code>vrna_move_is_insertion()</code>                    | 331 |
| 16.37.5.6 <code>vrna_move_is_shift()</code>                        | 331 |
| 16.37.5.7 <code>vrna_move_compare()</code>                         | 331 |
| 16.37.5.8 <code>vrna_loopidx_update()</code>                       | 332 |
| 16.37.5.9 <code>vrna_neighbors()</code>                            | 332 |
| 16.37.5.10 <code>vrna_neighbors_successive()</code>                | 333 |
| 16.37.5.11 <code>vrna_move_neighbor_diff_cb()</code>               | 333 |

---

|  |     |
|--|-----|
| 16.37.5.12 <code>vrna_move_neighbor_diff()</code>                          | 334 |
| 16.38 (Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima | 335 |
| 16.38.1 Detailed Description   | 335 |
| 16.38.2 Data Structure Documentation                                       | 336 |
| 16.38.2.1 <code>struct vrna_path_s</code>                                  | 336 |
| 16.38.3 Macro Definition Documentation                                     | 337 |
| 16.38.3.1 <code>VRNA_PATH_TYPE_DOT_BRACKET</code>                          | 337 |
| 16.38.3.2 <code>VRNA_PATH_TYPE_MOVES</code>                                | 337 |
| 16.38.4 Function Documentation   | 337 |
| 16.38.4.1 <code>vrna_path_free()</code>                                    | 337 |
| 16.38.4.2 <code>vrna_path_options_free()</code>                            | 338 |
| 16.39 Direct Refolding Paths between two Secondary Structures              | 338 |
| 16.39.1 Detailed Description   | 338 |
| 16.39.2 Function Documentation   | 338 |
| 16.39.2.1 <code>vrna_path_findpath_saddle()</code>                         | 338 |
| 16.39.2.2 <code>vrna_path_findpath_saddle_ub()</code>                      | 339 |
| 16.39.2.3 <code>vrna_path_findpath()</code>                                | 340 |
| 16.39.2.4 <code>vrna_path_findpath_ub()</code>                             | 340 |
| 16.39.2.5 <code>vrna_path_options_findpath()</code>                        | 341 |
| 16.39.2.6 <code>vrna_path_direct()</code>                                  | 342 |
| 16.39.2.7 <code>vrna_path_direct_ub()</code>                               | 342 |
| 16.40 Folding Paths that start at a single Secondary Structure             | 343 |
| 16.40.1 Detailed Description   | 343 |
| 16.40.2 Macro Definition Documentation                                     | 344 |
| 16.40.2.1 <code>VRNA_PATH_STEEPEST_DESCENT</code>                          | 344 |
| 16.40.2.2 <code>VRNA_PATH_RANDOM</code>                                    | 344 |
| 16.40.2.3 <code>VRNA_PATH_NO_TRANSITION_OUTPUT</code>                      | 344 |
| 16.40.2.4 <code>VRNA_PATH_DEFAULT</code>                                   | 344 |
| 16.40.3 Function Documentation   | 344 |
| 16.40.3.1 <code>vrna_path()</code>   | 345 |
| 16.40.3.2 <code>vrna_path_gradient()</code>                                | 345 |
| 16.40.3.3 <code>vrna_path_random()</code>                                  | 346 |
| 16.41 Experimental Structure Probing Data                                  | 347 |
| 16.41.1 Detailed Description   | 347 |
| 16.42 SHAPE Reactivity Data  | 347 |
| 16.42.1 Detailed Description   | 347 |
| 16.42.2 Function Documentation   | 348 |
| 16.42.2.1 <code>vrna_sc_add_SHAPE_deigan()</code>                          | 348 |
| 16.42.2.2 <code>vrna_sc_add_SHAPE_deigan_ali()</code>                      | 348 |
| 16.42.2.3 <code>vrna_sc_add_SHAPE_zarringhalam()</code>                    | 349 |
| 16.42.2.4 <code>vrna_sc_SHAPE_to_pr()</code>                               | 350 |
| 16.43 Generate Soft Constraints from Data                                  | 350 |

---

|  |     |
|--|-----|
| 16.43.1 Detailed Description . . . . .   | 350 |
| 16.43.2 Macro Definition Documentation . . . . .   | 351 |
| 16.43.2.1 VRNA_OBJECTIVE_FUNCTION_QUADRATIC . . . . .  | 351 |
| 16.43.2.2 VRNA_OBJECTIVE_FUNCTION_ABSOLUTE . . . . .   | 351 |
| 16.43.2.3 VRNA_MINIMIZER_CONJUGATE_FR . . . . .  | 351 |
| 16.43.2.4 VRNA_MINIMIZER_CONJUGATE_PR . . . . .  | 352 |
| 16.43.2.5 VRNA_MINIMIZER_VECTOR_BFGS . . . . .   | 352 |
| 16.43.2.6 VRNA_MINIMIZER_VECTOR_BFGS2 . . . . .  | 352 |
| 16.43.2.7 VRNA_MINIMIZER_STEEPEST_DESCENT . . . . .  | 352 |
| 16.43.3 Typedef Documentation . . . . .  | 352 |
| 16.43.3.1 progress_callback . . . . .  | 352 |
| 16.43.4 Function Documentation . . . . .   | 352 |
| 16.43.4.1 vrna_sc_minimize_perturbation() . . . . .  | 352 |
| 16.44 Ligands Binding to RNA Structures . . . . .  | 354 |
| 16.44.1 Detailed Description . . . . .   | 354 |
| 16.45 Ligands Binding to Unstructured Domains . . . . .  | 354 |
| 16.46 Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints . . . . . | 354 |
| 16.46.1 Detailed Description . . . . .   | 354 |
| 16.46.2 Data Structure Documentation . . . . .   | 355 |
| 16.46.2.1 struct vrna_sc_motif_s . . . . .   | 355 |
| 16.46.3 Function Documentation . . . . .   | 355 |
| 16.46.3.1 vrna_sc_add_hi_motif() . . . . .   | 355 |
| 16.47 Structure Modules and Pseudoknots . . . . .  | 356 |
| 16.47.1 Detailed Description . . . . .   | 356 |
| 16.48 Pseudoknots . . . . .  | 356 |
| 16.48.1 Detailed Description . . . . .   | 356 |
| 16.48.2 Data Structure Documentation . . . . .   | 357 |
| 16.48.2.1 struct vrna_pk_plex_result_s . . . . .   | 357 |
| 16.48.3 Typedef Documentation . . . . .  | 358 |
| 16.48.3.1 vrna_callback_pk_plex_score . . . . .  | 358 |
| 16.48.3.2 vrna_pk_plex_opt_t . . . . .   | 358 |
| 16.48.3.3 vrna_pk_plex_t . . . . .   | 358 |
| 16.48.4 Function Documentation . . . . .   | 359 |
| 16.48.4.1 vrna_pk_plex() . . . . .   | 359 |
| 16.48.4.2 vrna_pk_plex_accessibility() . . . . .   | 359 |
| 16.48.4.3 vrna_pk_plex_opt_defaults() . . . . .  | 360 |
| 16.48.4.4 vrna_pk_plex_opt() . . . . .   | 360 |
| 16.48.4.5 vrna_pk_plex_opt_fun() . . . . .   | 360 |
| 16.49 G-Quadruplexes . . . . .   | 361 |
| 16.49.1 Detailed Description . . . . .   | 361 |
| 16.49.2 Function Documentation . . . . .   | 361 |
| 16.49.2.1 get_gquad_matrix() . . . . .   | 361 |

---

|  |     |
|--|-----|
| 16.49.2.2 parse_gquad()                | 362 |
| 16.49.2.3 backtrack_GQuad_IntLoop()    | 362 |
| 16.49.2.4 backtrack_GQuad_IntLoop_L()  | 362 |
| 16.50 Utilities                        | 363 |
| 16.50.1 Detailed Description           | 363 |
| 16.50.2 Macro Definition Documentation | 365 |
| 16.50.2.1 VRNA_INPUT_FASTA_HEADER      | 366 |
| 16.50.3 Function Documentation         | 366 |
| 16.50.3.1 vrna_alloc()                 | 366 |
| 16.50.3.2 vrna_realloc()               | 366 |
| 16.50.3.3 vrna_init_rand()             | 366 |
| 16.50.3.4 vrna_init_rand_seed()        | 367 |
| 16.50.3.5 vrna_urn()                   | 367 |
| 16.50.3.6 vrna_int_urn()               | 367 |
| 16.50.3.7 vrna_time_stamp()            | 368 |
| 16.50.3.8 get_input_line()             | 368 |
| 16.50.3.9 vrna_idx_row_wise()          | 368 |
| 16.50.3.10 vrna_idx_col_wise()         | 369 |
| 16.50.4 Variable Documentation         | 369 |
| 16.50.4.1 xsubi                        | 369 |
| 16.51 Exterior Loops                   | 369 |
| 16.51.1 Detailed Description           | 370 |
| 16.51.2 Typedef Documentation          | 370 |
| 16.51.2.1 vrna_mx_pf_aux_el_t          | 370 |
| 16.51.3 Function Documentation         | 370 |
| 16.51.3.1 vrna_E_ext_stem()            | 370 |
| 16.51.3.2 vrna_eval_ext_stem()         | 371 |
| 16.51.3.3 vrna_exp_E_ext_stem()        | 371 |
| 16.52 Hairpin Loops                    | 372 |
| 16.52.1 Detailed Description           | 372 |
| 16.52.2 Function Documentation         | 373 |
| 16.52.2.1 vrna_E_hp_loop()             | 373 |
| 16.52.2.2 vrna_E_ext_hp_loop()         | 373 |
| 16.52.2.3 vrna_eval_hp_loop()          | 373 |
| 16.52.2.4 E_Hairpin()                  | 374 |
| 16.52.2.5 exp_E_Hairpin()              | 375 |
| 16.52.2.6 vrna_exp_E_hp_loop()         | 376 |
| 16.53 Internal Loops                   | 376 |
| 16.53.1 Detailed Description           | 376 |
| 16.53.2 Function Documentation         | 376 |
| 16.53.2.1 vrna_eval_int_loop()         | 377 |
| 16.54 Multibranch Loops                | 377 |

---

|   |     |
|---|-----|
| 16.54.1 Detailed Description . . . . .  | 377 |
| 16.54.2 Typedef Documentation . . . . .   | 377 |
| 16.54.2.1 vrna_mx_pf_aux_ml_t . . . . .   | 378 |
| 16.54.3 Function Documentation . . . . .  | 378 |
| 16.54.3.1 vrna_E_mb_loop_stack() . . . . .  | 378 |
| 16.55 Partition Function for Two Hybridized Sequences . . . . .                       | 378 |
| 16.55.1 Detailed Description . . . . .  | 378 |
| 16.55.2 Function Documentation . . . . .  | 379 |
| 16.55.2.1 vrna_pf_co_fold() . . . . .   | 379 |
| 16.55.2.2 vrna_pf_dimer_concentrations() . . . . .                                    | 380 |
| 16.56 Partition Function for two Hybridized Sequences as a Stepwise Process . . . . . | 380 |
| 16.56.1 Detailed Description . . . . .  | 380 |
| 16.56.2 Function Documentation . . . . .  | 381 |
| 16.56.2.1 pf_unstru() . . . . .   | 381 |
| 16.56.2.2 pf_interact() . . . . .   | 381 |
| 16.57 Reading/Writing Energy Parameter Sets from/to File . . . . .                    | 382 |
| 16.57.1 Detailed Description . . . . .  | 382 |
| 16.57.2 Macro Definition Documentation . . . . .                                      | 383 |
| 16.57.2.1 VRNA_PARAMETER_FORMAT_DEFAULT . . . . .                                     | 383 |
| 16.57.3 Function Documentation . . . . .  | 383 |
| 16.57.3.1 vrna_params_load() . . . . .  | 383 |
| 16.57.3.2 vrna_params_save() . . . . .  | 384 |
| 16.57.3.3 vrna_params_load_from_string() . . . . .                                    | 384 |
| 16.57.3.4 vrna_params_load_defaults() . . . . .                                       | 385 |
| 16.57.3.5 vrna_params_load_RNA_Turner2004() . . . . .                                 | 385 |
| 16.57.3.6 vrna_params_load_RNA_Turner1999() . . . . .                                 | 386 |
| 16.57.3.7 vrna_params_load_RNA_Andronescu2007() . . . . .                             | 386 |
| 16.57.3.8 vrna_params_load_RNA_Langdon2018() . . . . .                                | 386 |
| 16.57.3.9 vrna_params_load_RNA_misc_special_hairpins() . . . . .                      | 387 |
| 16.57.3.10 vrna_params_load_DNA_Mathews2004() . . . . .                               | 387 |
| 16.57.3.11 vrna_params_load_DNA_Mathews1999() . . . . .                               | 387 |
| 16.57.3.12 last_parameter_file() . . . . .  | 388 |
| 16.57.3.13 read_parameter_file() . . . . .  | 388 |
| 16.57.3.14 write_parameter_file() . . . . .   | 388 |
| 16.58 Converting Energy Parameter Files . . . . .                                     | 388 |
| 16.58.1 Detailed Description . . . . .  | 388 |
| 16.58.2 Macro Definition Documentation . . . . .                                      | 389 |
| 16.58.2.1 VRNA_CONVERT_OUTPUT_ALL . . . . .   | 389 |
| 16.58.2.2 VRNA_CONVERT_OUTPUT_HP . . . . .  | 389 |
| 16.58.2.3 VRNA_CONVERT_OUTPUT_STACK . . . . .   | 389 |
| 16.58.2.4 VRNA_CONVERT_OUTPUT_MM_HP . . . . .   | 390 |
| 16.58.2.5 VRNA_CONVERT_OUTPUT_MM_INT . . . . .  | 390 |

---

|   |     |
|---|-----|
| 16.58.2.6 VRNA_CONVERT_OUTPUT_MM_INT_1N . . . . .           | 390 |
| 16.58.2.7 VRNA_CONVERT_OUTPUT_MM_INT_23 . . . . .           | 390 |
| 16.58.2.8 VRNA_CONVERT_OUTPUT_MM_MULTI . . . . .            | 390 |
| 16.58.2.9 VRNA_CONVERT_OUTPUT_MM_EXT . . . . .              | 390 |
| 16.58.2.10 VRNA_CONVERT_OUTPUT_DANGLE5 . . . . .            | 390 |
| 16.58.2.11 VRNA_CONVERT_OUTPUT_DANGLE3 . . . . .            | 390 |
| 16.58.2.12 VRNA_CONVERT_OUTPUT_INT_11 . . . . .             | 390 |
| 16.58.2.13 VRNA_CONVERT_OUTPUT_INT_21 . . . . .             | 391 |
| 16.58.2.14 VRNA_CONVERT_OUTPUT_INT_22 . . . . .             | 391 |
| 16.58.2.15 VRNA_CONVERT_OUTPUT_BULGE . . . . .              | 391 |
| 16.58.2.16 VRNA_CONVERT_OUTPUT_INT . . . . .                | 391 |
| 16.58.2.17 VRNA_CONVERT_OUTPUT_ML . . . . .                 | 391 |
| 16.58.2.18 VRNA_CONVERT_OUTPUT_MISC . . . . .               | 391 |
| 16.58.2.19 VRNA_CONVERT_OUTPUT_SPECIAL_HP . . . . .         | 391 |
| 16.58.2.20 VRNA_CONVERT_OUTPUT_VANILLA . . . . .            | 391 |
| 16.58.2.21 VRNA_CONVERT_OUTPUT_NINIO . . . . .              | 392 |
| 16.58.2.22 VRNA_CONVERT_OUTPUT_DUMP . . . . .               | 392 |
| 16.58.3 Function Documentation . . . . .                    | 392 |
| 16.58.3.1 convert_parameter_file() . . . . .                | 392 |
| 16.59 Utilities to deal with Nucleotide Alphabets . . . . . | 392 |
| 16.59.1 Detailed Description . . . . .                      | 393 |
| 16.59.2 Data Structure Documentation . . . . .              | 393 |
| 16.59.2.1 struct vrna_sequence_s . . . . .                  | 393 |
| 16.59.2.2 struct vrna_alignment_s . . . . .                 | 394 |
| 16.59.3 Enumeration Type Documentation . . . . .            | 394 |
| 16.59.3.1 vrna_seq_type_e . . . . .                         | 394 |
| 16.59.4 Function Documentation . . . . .                    | 394 |
| 16.59.4.1 vrna_ptypes() . . . . .                           | 394 |
| 16.59.4.2 vrna_seq_encode() . . . . .                       | 394 |
| 16.59.4.3 vrna_nucleotide_encode() . . . . .                | 395 |
| 16.59.4.4 vrna_nucleotide_decode() . . . . .                | 395 |
| 16.60 (Nucleic Acid Sequence) String Utilities . . . . .    | 396 |
| 16.60.1 Detailed Description . . . . .                      | 396 |
| 16.60.2 Macro Definition Documentation . . . . .            | 397 |
| 16.60.2.1 FILENAME_MAX_LENGTH . . . . .                     | 397 |
| 16.60.2.2 FILENAME_ID_LENGTH . . . . .                      | 397 |
| 16.60.2.3 VRNA_TRIM.LEADING . . . . .                       | 397 |
| 16.60.2.4 VRNA_TRIM.TRAILING . . . . .                      | 398 |
| 16.60.2.5 VRNA_TRIM.IN_BETWEEN . . . . .                    | 398 |
| 16.60.2.6 VRNA_TRIM.SUBST_BY_FIRST . . . . .                | 398 |
| 16.60.2.7 VRNA_TRIM.DEFAULT . . . . .                       | 398 |
| 16.60.2.8 VRNA_TRIM.ALL . . . . .                           | 398 |

---

|  |     |
|--|-----|
| 16.60.3 Function Documentation . . . . .                     | 398 |
| 16.60.3.1 vrna_strdup_printf() . . . . .                     | 399 |
| 16.60.3.2 vrna_strdup_vprintf() . . . . .                    | 399 |
| 16.60.3.3 vrna_strcat_printf() . . . . .                     | 400 |
| 16.60.3.4 vrna_strcat_vprintf() . . . . .                    | 400 |
| 16.60.3.5 vrna_strtrim() . . . . .                           | 401 |
| 16.60.3.6 vrna_strsplit() . . . . .                          | 402 |
| 16.60.3.7 vrna_random_string() . . . . .                     | 402 |
| 16.60.3.8 vrna_hamming_distance() . . . . .                  | 403 |
| 16.60.3.9 vrna_hamming_distance_bound() . . . . .            | 403 |
| 16.60.3.10 vrna_seq_toRNA() . . . . .                        | 403 |
| 16.60.3.11 vrna_seq_toupper() . . . . .                      | 404 |
| 16.60.3.12 vrna_seq_reverse() . . . . .                      | 404 |
| 16.60.3.13 vrna_DNA_complement() . . . . .                   | 404 |
| 16.60.3.14 vrna_seq_ungapped() . . . . .                     | 405 |
| 16.60.3.15 vrna_cut_point_insert() . . . . .                 | 405 |
| 16.60.3.16 vrna_cut_point_remove() . . . . .                 | 405 |
| 16.61 Secondary Structure Utilities . . . . .                | 406 |
| 16.61.1 Detailed Description . . . . .                       | 406 |
| 16.61.2 Function Documentation . . . . .                     | 407 |
| 16.61.2.1 vrna_refBPcnt_matrix() . . . . .                   | 407 |
| 16.61.2.2 vrna_refBPDist_matrix() . . . . .                  | 407 |
| 16.61.2.3 vrna_db_from_probs() . . . . .                     | 407 |
| 16.61.2.4 vrna_db_from_bp_stack() . . . . .                  | 407 |
| 16.62 Dot-Bracket Notation of Secondary Structures . . . . . | 408 |
| 16.62.1 Detailed Description . . . . .                       | 408 |
| 16.62.2 Macro Definition Documentation . . . . .             | 409 |
| 16.62.2.1 VRNA_BRACKETS_ALPHA . . . . .                      | 409 |
| 16.62.2.2 VRNA_BRACKETS_RND . . . . .                        | 409 |
| 16.62.2.3 VRNA_BRACKETS_CLY . . . . .                        | 409 |
| 16.62.2.4 VRNA_BRACKETS_ANG . . . . .                        | 409 |
| 16.62.2.5 VRNA_BRACKETS_SQR . . . . .                        | 410 |
| 16.62.2.6 VRNA_BRACKETS_DEFAULT . . . . .                    | 410 |
| 16.62.2.7 VRNA_BRACKETS_ANY . . . . .                        | 410 |
| 16.62.3 Function Documentation . . . . .                     | 410 |
| 16.62.3.1 vrna_db_pack() . . . . .                           | 410 |
| 16.62.3.2 vrna_db_unpack() . . . . .                         | 411 |
| 16.62.3.3 vrna_db_flatten() . . . . .                        | 411 |
| 16.62.3.4 vrna_db_flatten_to() . . . . .                     | 412 |
| 16.62.3.5 vrna_db_from_ptable() . . . . .                    | 412 |
| 16.62.3.6 vrna_db_from plist() . . . . .                     | 413 |
| 16.62.3.7 vrna_db_to_element_string() . . . . .              | 413 |

---

|   |     |
|---|-----|
| 16.62.3.8 vrna_db_pk_remove()                                   | 413 |
| 16.63 Washington University Secondary Structure (WUSS) notation | 414 |
| 16.63.1 Detailed Description                                    | 414 |
| 16.63.2 Function Documentation                                  | 415 |
| 16.63.2.1 vrna_db_from_WUSS()                                   | 415 |
| 16.64 Pair Table Representation of Secondary Structures         | 415 |
| 16.64.1 Detailed Description                                    | 415 |
| 16.64.2 Function Documentation                                  | 416 |
| 16.64.2.1 vrna_ptable()   | 416 |
| 16.64.2.2 vrna_ptable_from_string()                             | 416 |
| 16.64.2.3 vrna_pt_pk_get()                                      | 418 |
| 16.64.2.4 vrna_ptable_copy()                                    | 418 |
| 16.64.2.5 vrna_pt_snoop_get()                                   | 419 |
| 16.64.2.6 vrna_pt_pk_remove()                                   | 419 |
| 16.65 Pair List Representation of Secondary Structures          | 419 |
| 16.65.1 Detailed Description                                    | 419 |
| 16.65.2 Data Structure Documentation                            | 420 |
| 16.65.2.1 struct vrna_elem_prob_s                               | 420 |
| 16.65.3 Function Documentation                                  | 420 |
| 16.65.3.1 vrna_plist()  | 420 |
| 16.66 Abstract Shapes Representation of Secondary Structures    | 421 |
| 16.66.1 Detailed Description                                    | 421 |
| 16.66.2 Function Documentation                                  | 422 |
| 16.66.2.1 vrna_abstract_shapes()                                | 422 |
| 16.66.2.2 vrna_abstract_shapes_pt()                             | 422 |
| 16.67 Helix List Representation of Secondary Structures         | 423 |
| 16.67.1 Detailed Description                                    | 423 |
| 16.67.2 Data Structure Documentation                            | 423 |
| 16.67.2.1 struct vrna_hx_s                                      | 423 |
| 16.67.3 Function Documentation                                  | 423 |
| 16.67.3.1 vrna_hx_from_ptable()                                 | 423 |
| 16.68 Tree Representation of Secondary Structures               | 424 |
| 16.68.1 Detailed Description                                    | 424 |
| 16.68.2 Macro Definition Documentation                          | 425 |
| 16.68.2.1 VRNA_STRUCTURE_TREE_HIT                               | 425 |
| 16.68.2.2 VRNA_STRUCTURE_TREE_SHAPIRO_SHORT                     | 426 |
| 16.68.2.3 VRNA_STRUCTURE_TREE_SHAPIRO                           | 426 |
| 16.68.2.4 VRNA_STRUCTURE_TREE_SHAPIRO_EXT                       | 426 |
| 16.68.2.5 VRNA_STRUCTURE_TREE_SHAPIRO_WEIGHT                    | 426 |
| 16.68.2.6 VRNA_STRUCTURE_TREE_EXPANDED                          | 426 |
| 16.68.3 Function Documentation                                  | 426 |
| 16.68.3.1 vrna_db_to_tree_string()                              | 427 |

---

|  |     |
|--|-----|
| 16.68.3.2 vrna_tree_string_unweight()                | 427 |
| 16.68.3.3 vrna_tree_string_to_db()                   | 428 |
| 16.69 Distance measures between Secondary Structures | 428 |
| 16.69.1 Detailed Description                         | 428 |
| 16.69.2 Function Documentation                       | 428 |
| 16.69.2.1 vrna_bp_distance_pt()                      | 428 |
| 16.69.2.2 vrna_bp_distance()                         | 429 |
| 16.70 Multiple Sequence Alignment Utilities          | 429 |
| 16.70.1 Detailed Description                         | 429 |
| 16.70.2 Data Structure Documentation                 | 431 |
| 16.70.2.1 struct vrna_pinfo_s                        | 431 |
| 16.70.3 Macro Definition Documentation               | 431 |
| 16.70.3.1 VRNA_MEASURE_SHANNON_ENTROPY               | 431 |
| 16.70.4 Function Documentation                       | 431 |
| 16.70.4.1 vrna_aln_mpi()                             | 431 |
| 16.70.4.2 vrna_aln_pinfo()                           | 432 |
| 16.70.4.3 vrna_aln_slice()                           | 432 |
| 16.70.4.4 vrna_aln_free()                            | 433 |
| 16.70.4.5 vrna_aln_uppercase()                       | 433 |
| 16.70.4.6 vrna_aln_toRNA()                           | 433 |
| 16.70.4.7 vrna_aln_copy()                            | 434 |
| 16.70.4.8 vrna_aln_conservation_struct()             | 434 |
| 16.70.4.9 vrna_aln_conservation_col()                | 434 |
| 16.70.4.10 vrna_aln_consensus_sequence()             | 435 |
| 16.70.4.11 vrna_aln_consensus_mis()                  | 435 |
| 16.71 Files and I/O                                  | 436 |
| 16.71.1 Detailed Description                         | 436 |
| 16.71.2 Function Documentation                       | 437 |
| 16.71.2.1 vrna_read_line()                           | 437 |
| 16.71.2.2 vrna_filename_sanitize()                   | 437 |
| 16.71.2.3 vrna_file_exists()                         | 438 |
| 16.72 Nucleic Acid Sequences and Structures          | 438 |
| 16.72.1 Detailed Description                         | 438 |
| 16.72.2 Macro Definition Documentation               | 439 |
| 16.72.2.1 VRNA_OPTION_MULTILINE                      | 439 |
| 16.72.2.2 VRNA_CONSTRAINT_MULTILINE                  | 439 |
| 16.72.3 Function Documentation                       | 439 |
| 16.72.3.1 vrna_file_helixlist()                      | 440 |
| 16.72.3.2 vrna_file_connect()                        | 440 |
| 16.72.3.3 vrna_file_bpseq()                          | 441 |
| 16.72.3.4 vrna_file_json()                           | 441 |
| 16.72.3.5 vrna_file_fasta_read_record()              | 441 |

---

|   |     |
|---|-----|
| 16.72.3.6 vrna_extract_record_rest_structure()  | 442 |
| 16.72.3.7 vrna_file_SHAPE_read()                | 443 |
| 16.72.3.8 vrna_extract_record_rest_constraint() | 443 |
| 16.72.3.9 read_record()                         | 444 |
| 16.73 Multiple Sequence Alignments              | 444 |
| 16.73.1 Detailed Description                    | 444 |
| 16.73.2 Macro Definition Documentation          | 445 |
| 16.73.2.1 VRNA_FILE_FORMAT_MSA_CLUSTAL          | 445 |
| 16.73.2.2 VRNA_FILE_FORMAT_MSA_STOCKHOLM        | 445 |
| 16.73.2.3 VRNA_FILE_FORMAT_MSA_FASTA            | 446 |
| 16.73.2.4 VRNA_FILE_FORMAT_MSA_MAF              | 446 |
| 16.73.2.5 VRNA_FILE_FORMAT_MSA_MIS              | 446 |
| 16.73.2.6 VRNA_FILE_FORMAT_MSA_DEFAULT          | 446 |
| 16.73.2.7 VRNA_FILE_FORMAT_MSA_NOCHECK          | 446 |
| 16.73.2.8 VRNA_FILE_FORMAT_MSA_UNKNOWN          | 447 |
| 16.73.2.9 VRNA_FILE_FORMAT_MSA_APPEND           | 447 |
| 16.73.2.10 VRNA_FILE_FORMAT_MSA QUIET           | 447 |
| 16.73.2.11 VRNA_FILE_FORMAT_MSA_SILENT          | 447 |
| 16.73.3 Function Documentation                  | 447 |
| 16.73.3.1 vrna_file_msa_read()                  | 447 |
| 16.73.3.2 vrna_file_msa_read_record()           | 448 |
| 16.73.3.3 vrna_file_msa_detect_format()         | 450 |
| 16.73.3.4 vrna_file_msa_write()                 | 450 |
| 16.74 Command Files                             | 451 |
| 16.74.1 Detailed Description                    | 451 |
| 16.74.2 Macro Definition Documentation          | 452 |
| 16.74.2.1 VRNA_CMD_PARSE_HC                     | 452 |
| 16.74.2.2 VRNA_CMD_PARSE_SC                     | 452 |
| 16.74.2.3 VRNA_CMD_PARSE_UD                     | 452 |
| 16.74.2.4 VRNA_CMD_PARSE_SD                     | 453 |
| 16.74.2.5 VRNA_CMD_PARSE_DEFAULTS               | 453 |
| 16.74.3 Function Documentation                  | 453 |
| 16.74.3.1 vrna_file_commands_read()             | 453 |
| 16.74.3.2 vrna_file_commands_apply()            | 453 |
| 16.74.3.3 vrna_commands_apply()                 | 454 |
| 16.74.3.4 vrna_commands_free()                  | 454 |
| 16.75 Plotting                                  | 455 |
| 16.75.1 Detailed Description                    | 455 |
| 16.75.2 Data Structure Documentation            | 456 |
| 16.75.2.1 struct vrna_dotplot_auxdata_t         | 456 |
| 16.75.3 Function Documentation                  | 456 |
| 16.75.3.1 PS_dot_plot_list()                    | 456 |

---

|  |     |
|--|-----|
| 16.75.3.2 PS_dot_plot()                      | 457 |
| 16.75.3.3 vrna_file_PS_rnplot()              | 457 |
| 16.75.3.4 vrna_file_PS_rnplot_a()            | 457 |
| 16.75.3.5 gmlRNA()                           | 459 |
| 16.75.3.6 ssv_rna_plot()                     | 459 |
| 16.75.3.7 svg_rna_plot()                     | 460 |
| 16.75.3.8 xrna_plot()                        | 460 |
| 16.75.3.9 PS_rna_plot()                      | 460 |
| 16.75.3.10 PS_rna_plot_a()                   | 461 |
| 16.75.3.11 PS_rna_plot_a_gquad()             | 461 |
| 16.76 Layouts and Coordinates                | 461 |
| 16.76.1 Detailed Description                 | 461 |
| 16.76.2 Data Structure Documentation         | 462 |
| 16.76.2.1 struct vrna_plot_layout_s          | 462 |
| 16.76.2.2 struct vrna_plot_options_puzzler_t | 462 |
| 16.76.3 Macro Definition Documentation       | 463 |
| 16.76.3.1 VRNA_PLOT_TYPE_SIMPLE              | 463 |
| 16.76.3.2 VRNA_PLOT_TYPE_NAVIEW              | 463 |
| 16.76.3.3 VRNA_PLOT_TYPE_CIRCULAR            | 463 |
| 16.76.4 Typedef Documentation                | 463 |
| 16.76.4.1 vrna_plot_layout_t                 | 463 |
| 16.76.5 Function Documentation               | 463 |
| 16.76.5.1 vrna_plot_layout()                 | 464 |
| 16.76.5.2 vrna_plot_layout_simple()          | 464 |
| 16.76.5.3 vrna_plot_layout_circular()        | 465 |
| 16.76.5.4 vrna_plot_layout_turtle()          | 465 |
| 16.76.5.5 vrna_plot_layout_puzzler()         | 466 |
| 16.76.5.6 vrna_plot_layout_free()            | 466 |
| 16.76.5.7 vrna_plot_coords()                 | 466 |
| 16.76.5.8 vrna_plot_coords_pt()              | 467 |
| 16.76.5.9 vrna_plot_coords_simple()          | 468 |
| 16.76.5.10 vrna_plot_coords_simple_pt()      | 469 |
| 16.76.5.11 vrna_plot_coords_circular()       | 469 |
| 16.76.5.12 vrna_plot_coords_circular_pt()    | 470 |
| 16.76.5.13 vrna_plot_coords_puzzler()        | 470 |
| 16.76.5.14 vrna_plot_coords_puzzler_pt()     | 471 |
| 16.76.5.15 vrna_plot_options_puzzler()       | 472 |
| 16.76.5.16 vrna_plot_options_puzzler_free()  | 472 |
| 16.76.5.17 vrna_plot_coords_turtle()         | 473 |
| 16.76.5.18 vrna_plot_coords_turtle_pt()      | 473 |
| 16.77 Annotation                             | 474 |
| 16.77.1 Detailed Description                 | 474 |

---

|  |     |
|--|-----|
| 16.78 Alignment Plots . . . . .                        | 474 |
| 16.78.1 Detailed Description . . . . .                 | 474 |
| 16.78.2 Function Documentation . . . . .               | 475 |
| 16.78.2.1 vrna_file_PS_aln() . . . . .                 | 475 |
| 16.78.2.2 vrna_file_PS_aln_slice() . . . . .           | 475 |
| 16.79 Search Algorithms . . . . .                      | 476 |
| 16.79.1 Detailed Description . . . . .                 | 476 |
| 16.79.2 Function Documentation . . . . .               | 476 |
| 16.79.2.1 vrna_search_BMH_num() . . . . .              | 477 |
| 16.79.2.2 vrna_search_BMH() . . . . .                  | 477 |
| 16.79.2.3 vrna_search_BM_BCT_num() . . . . .           | 478 |
| 16.79.2.4 vrna_search_BM_BCT() . . . . .               | 478 |
| 16.80 Combinatorics Algorithms . . . . .               | 479 |
| 16.80.1 Detailed Description . . . . .                 | 479 |
| 16.80.2 Function Documentation . . . . .               | 479 |
| 16.80.2.1 vrna_enumerate_necklaces() . . . . .         | 480 |
| 16.80.2.2 vrna_rotational_symmetry_num() . . . . .     | 480 |
| 16.80.2.3 vrna_rotational_symmetry_pos_num() . . . . . | 480 |
| 16.80.2.4 vrna_rotational_symmetry() . . . . .         | 481 |
| 16.80.2.5 vrna_rotational_symmetry_pos() . . . . .     | 482 |
| 16.80.2.6 vrna_rotational_symmetry_db() . . . . .      | 482 |
| 16.80.2.7 vrna_rotational_symmetry_db_pos() . . . . .  | 483 |
| 16.80.2.8 vrna_n_multichoose_k() . . . . .             | 484 |
| 16.80.2.9 vrna_boustrophedon() . . . . .               | 484 |
| 16.80.2.10 vrna_boustrophedon_pos() . . . . .          | 485 |
| 16.81 (Abstract) Data Structures . . . . .             | 485 |
| 16.81.1 Detailed Description . . . . .                 | 485 |
| 16.81.2 Data Structure Documentation . . . . .         | 487 |
| 16.81.2.1 struct vrna_basepair_s . . . . .             | 487 |
| 16.81.2.2 struct vrna_cpair_s . . . . .                | 487 |
| 16.81.2.3 struct vrna_color_s . . . . .                | 487 |
| 16.81.2.4 struct vrna_data_linear_s . . . . .          | 487 |
| 16.81.2.5 struct vrna_sect_s . . . . .                 | 487 |
| 16.81.2.6 struct vrna_bp_stack_s . . . . .             | 487 |
| 16.81.2.7 struct pu_contrib . . . . .                  | 487 |
| 16.81.2.8 struct interact . . . . .                    | 488 |
| 16.81.2.9 struct pu_out . . . . .                      | 488 |
| 16.81.2.10 struct constrain . . . . .                  | 489 |
| 16.81.2.11 struct duplexT . . . . .                    | 489 |
| 16.81.2.12 struct node . . . . .                       | 489 |
| 16.81.2.13 struct snoopT . . . . .                     | 489 |
| 16.81.2.14 struct dupVar . . . . .                     | 489 |

---

|   |     |
|---|-----|
| 16.81.3 Typedef Documentation . . . . .             | 489 |
| 16.81.3.1 PAIR . . . . .                            | 489 |
| 16.81.3.2 plist . . . . .                           | 489 |
| 16.81.3.3 cpair . . . . .                           | 489 |
| 16.81.3.4 sect . . . . .                            | 490 |
| 16.81.3.5 bondT . . . . .                           | 490 |
| 16.81.4 Function Documentation . . . . .            | 490 |
| 16.81.4.1 vrna_C11_features() . . . . .             | 490 |
| 16.82 Messages . . . . .                            | 490 |
| 16.82.1 Detailed Description . . . . .              | 490 |
| 16.82.2 Function Documentation . . . . .            | 491 |
| 16.82.2.1 vrna_message_error() . . . . .            | 491 |
| 16.82.2.2 vrna_message_verror() . . . . .           | 491 |
| 16.82.2.3 vrna_message_warning() . . . . .          | 492 |
| 16.82.2.4 vrna_message_vwarning() . . . . .         | 492 |
| 16.82.2.5 vrna_message_info() . . . . .             | 492 |
| 16.82.2.6 vrna_message_vinfo() . . . . .            | 493 |
| 16.82.2.7 vrna_message_input_seq_simple() . . . . . | 493 |
| 16.82.2.8 vrna_message_input_seq() . . . . .        | 493 |
| 16.83 Unit Conversion . . . . .                     | 494 |
| 16.83.1 Detailed Description . . . . .              | 494 |
| 16.83.2 Enumeration Type Documentation . . . . .    | 494 |
| 16.83.2.1 vrna_unit_energy_e . . . . .              | 494 |
| 16.83.2.2 vrna_unit_temperature_e . . . . .         | 495 |
| 16.83.3 Function Documentation . . . . .            | 495 |
| 16.83.3.1 vrna_convert_energy() . . . . .           | 495 |
| 16.83.3.2 vrna_convert_temperature() . . . . .      | 496 |
| 16.83.3.3 vrna_convert_kcal_to_dcal() . . . . .     | 496 |
| 16.83.3.4 vrna_convert_dcal_to_kcal() . . . . .     | 497 |
| 16.84 The Fold Compound . . . . .                   | 497 |
| 16.84.1 Detailed Description . . . . .              | 497 |
| 16.84.2 Data Structure Documentation . . . . .      | 499 |
| 16.84.2.1 struct vrna_fc_s . . . . .                | 499 |
| 16.84.3 Macro Definition Documentation . . . . .    | 504 |
| 16.84.3.1 VRNA_STATUS_MFE_PRE . . . . .             | 504 |
| 16.84.3.2 VRNA_STATUS_MFE_POST . . . . .            | 505 |
| 16.84.3.3 VRNA_STATUS_PF_PRE . . . . .              | 505 |
| 16.84.3.4 VRNA_STATUS_PF_POST . . . . .             | 505 |
| 16.84.3.5 VRNA_OPTION_MFE . . . . .                 | 505 |
| 16.84.3.6 VRNA_OPTION_PF . . . . .                  | 505 |
| 16.84.3.7 VRNA_OPTION_EVAL_ONLY . . . . .           | 506 |
| 16.84.4 Typedef Documentation . . . . .             | 506 |

---

|  |     |
|--|-----|
| 16.84.4.1 <code>vrna_callback_free_auxdata</code>        | 506 |
| 16.84.4.2 <code>vrna_callback_recursion_status</code>    | 506 |
| 16.84.5 Enumeration Type Documentation                   | 507 |
| 16.84.5.1 <code>vrna_fc_type_e</code>                    | 507 |
| 16.84.6 Function Documentation                           | 507 |
| 16.84.6.1 <code>vrna_fold_compound()</code>              | 507 |
| 16.84.6.2 <code>vrna_fold_compound_comparative()</code>  | 508 |
| 16.84.6.3 <code>vrna_fold_compound_free()</code>         | 509 |
| 16.84.6.4 <code>vrna_fold_compound_add_auxdata()</code>  | 509 |
| 16.84.6.5 <code>vrna_fold_compound_add_callback()</code> | 510 |
| 16.85 The Dynamic Programming Matrices                   | 510 |
| 16.85.1 Detailed Description                             | 510 |
| 16.85.2 Data Structure Documentation                     | 511 |
| 16.85.2.1 <code>struct vrna_mx_mfe_s</code>              | 511 |
| 16.85.2.2 <code>struct vrna_mx_pf_s</code>               | 512 |
| 16.85.3 Enumeration Type Documentation                   | 512 |
| 16.85.3.1 <code>vrna_mx_type_e</code>                    | 512 |
| 16.85.4 Function Documentation                           | 513 |
| 16.85.4.1 <code>vrna_mx_add()</code>                     | 513 |
| 16.85.4.2 <code>vrna_mx_mfe_free()</code>                | 514 |
| 16.85.4.3 <code>vrna_mx_pf_free()</code>                 | 514 |
| 16.86 Hash Tables  | 514 |
| 16.86.1 Detailed Description                             | 514 |
| 16.86.2 Data Structure Documentation                     | 515 |
| 16.86.2.1 <code>struct vrna_ht_entry_db_t</code>         | 515 |
| 16.86.3 Typedef Documentation                            | 516 |
| 16.86.3.1 <code>vrna_hash_table_t</code>                 | 516 |
| 16.86.3.2 <code>vrna_callback_ht_compare_entries</code>  | 516 |
| 16.86.3.3 <code>vrna_callback_ht_hash_function</code>    | 516 |
| 16.86.3.4 <code>vrna_callback_ht_free_entry</code>       | 517 |
| 16.86.4 Function Documentation                           | 517 |
| 16.86.4.1 <code>vrna_ht_init()</code>                    | 517 |
| 16.86.4.2 <code>vrna_ht_size()</code>                    | 518 |
| 16.86.4.3 <code>vrna_ht_collisions()</code>              | 518 |
| 16.86.4.4 <code>vrna_ht_get()</code>                     | 518 |
| 16.86.4.5 <code>vrna_ht_insert()</code>                  | 519 |
| 16.86.4.6 <code>vrna_ht_remove()</code>                  | 519 |
| 16.86.4.7 <code>vrna_ht_clear()</code>                   | 519 |
| 16.86.4.8 <code>vrna_ht_free()</code>                    | 520 |
| 16.86.4.9 <code>vrna_ht_db_comp()</code>                 | 520 |
| 16.86.4.10 <code>vrna_ht_db_hash_func()</code>           | 520 |
| 16.86.4.11 <code>vrna_ht_db_free_entry()</code>          | 521 |

|  |     |
|--|-----|
| 16.87 Heaps . . . . .  | 521 |
| 16.87.1 Detailed Description . . . . .                         | 521 |
| 16.87.2 Typedef Documentation . . . . .                        | 522 |
| 16.87.2.1 vrna_heap_t . . . . .                                | 522 |
| 16.87.2.2 vrna_callback_heap_cmp . . . . .                     | 522 |
| 16.87.2.3 vrna_callback_heap_get_pos . . . . .                 | 524 |
| 16.87.2.4 vrna_callback_heap_set_pos . . . . .                 | 524 |
| 16.87.3 Function Documentation . . . . .                       | 524 |
| 16.87.3.1 vrna_heap_init() . . . . .                           | 524 |
| 16.87.3.2 vrna_heap_free() . . . . .                           | 525 |
| 16.87.3.3 vrna_heap_size() . . . . .                           | 525 |
| 16.87.3.4 vrna_heap_insert() . . . . .                         | 526 |
| 16.87.3.5 vrna_heap_pop() . . . . .                            | 526 |
| 16.87.3.6 vrna_heap_top() . . . . .                            | 526 |
| 16.87.3.7 vrna_heap_remove() . . . . .                         | 527 |
| 16.87.3.8 vrna_heap_update() . . . . .                         | 527 |
| 16.88 Buffers . . . . .  | 528 |
| 16.88.1 Detailed Description . . . . .                         | 528 |
| 16.88.2 Typedef Documentation . . . . .                        | 529 |
| 16.88.2.1 vrna_callback_stream_output . . . . .                | 529 |
| 16.88.3 Function Documentation . . . . .                       | 529 |
| 16.88.3.1 vrna_cstr() . . . . .                                | 529 |
| 16.88.3.2 vrna_cstr_discard() . . . . .                        | 530 |
| 16.88.3.3 vrna_cstr_free() . . . . .                           | 530 |
| 16.88.3.4 vrna_cstr_close() . . . . .                          | 530 |
| 16.88.3.5 vrna_cstr_fflush() . . . . .                         | 530 |
| 16.88.3.6 vrna_ostream_init() . . . . .                        | 531 |
| 16.88.3.7 vrna_ostream_free() . . . . .                        | 531 |
| 16.88.3.8 vrna_ostream_request() . . . . .                     | 532 |
| 16.88.3.9 vrna_ostream_provide() . . . . .                     | 532 |
| 16.89 Deprecated Interface for Global MFE Prediction . . . . . | 532 |
| 16.89.1 Detailed Description . . . . .                         | 532 |
| 16.89.2 Function Documentation . . . . .                       | 534 |
| 16.89.2.1 alifold() . . . . .                                  | 534 |
| 16.89.2.2 cofold() . . . . .                                   | 534 |
| 16.89.2.3 cofold_par() . . . . .                               | 534 |
| 16.89.2.4 free_co_arrays() . . . . .                           | 535 |
| 16.89.2.5 update_cofold_params() . . . . .                     | 535 |
| 16.89.2.6 update_cofold_params_par() . . . . .                 | 535 |
| 16.89.2.7 export_cofold_arrays_gq() . . . . .                  | 535 |
| 16.89.2.8 export_cofold_arrays() . . . . .                     | 536 |
| 16.89.2.9 initialize_cofold() . . . . .                        | 537 |

---

|  |     |
|--|-----|
| 16.89.2.10 fold_par()  | 537 |
| 16.89.2.11 fold()  | 538 |
| 16.89.2.12 circfold()  | 538 |
| 16.89.2.13 free_arrays()   | 539 |
| 16.89.2.14 update_fold_params()                                      | 539 |
| 16.89.2.15 update_fold_params_par()                                  | 539 |
| 16.89.2.16 export_fold_arrays()                                      | 539 |
| 16.89.2.17 export_fold_arrays_par()                                  | 540 |
| 16.89.2.18 export_circfold_arrays()                                  | 540 |
| 16.89.2.19 export_circfold_arrays_par()                              | 540 |
| 16.89.2.20 LoopEnergy()  | 541 |
| 16.89.2.21 HairpinE()  | 541 |
| 16.89.2.22 initialize_fold()   | 541 |
| 16.89.2.23 circlifold()  | 541 |
| 16.89.2.24 free_alifold_arrays()                                     | 542 |
| 16.90 Deprecated Interface for Local (Sliding Window) MFE Prediction | 542 |
| 16.90.1 Detailed Description   | 542 |
| 16.90.2 Function Documentation                                       | 542 |
| 16.90.2.1 Lfold()  | 542 |
| 16.90.2.2 Lfoldz()   | 543 |
| 16.91 Deprecated Interface for Global Partition Function Computation | 543 |
| 16.91.1 Detailed Description   | 543 |
| 16.91.2 Function Documentation                                       | 544 |
| 16.91.2.1 alipf_fold_par()   | 544 |
| 16.91.2.2 pf_fold_par()  | 545 |
| 16.91.2.3 pf_fold()  | 546 |
| 16.91.2.4 pf_circ_fold()   | 547 |
| 16.91.2.5 free_pf_arrays()   | 548 |
| 16.91.2.6 update_pf_params()   | 548 |
| 16.91.2.7 update_pf_params_par()                                     | 548 |
| 16.91.2.8 export_bppm()  | 548 |
| 16.91.2.9 get_pf_arrays()  | 549 |
| 16.91.2.10 mean_bp_distance()  | 549 |
| 16.91.2.11 mean_bp_distance_pr()                                     | 550 |
| 16.91.2.12 stackProb()   | 550 |
| 16.91.2.13 init_pf_fold()  | 550 |
| 16.91.2.14 co_pf_fold()  | 551 |
| 16.91.2.15 co_pf_fold_par()  | 551 |
| 16.91.2.16 compute_probabilities()                                   | 552 |
| 16.91.2.17 init_co_pf_fold()   | 552 |
| 16.91.2.18 export_co_bppm()  | 552 |
| 16.91.2.19 free_co_pf_arrays()                                       | 553 |

---

|  |     |
|--|-----|
| 16.91.2.20 update_co_pf_params()   | 553 |
| 16.91.2.21 update_co_pf_params_par()   | 553 |
| 16.91.2.22 assign_plist_from_db()  | 554 |
| 16.91.2.23 assign_plist_from_pr()  | 554 |
| 16.91.2.24 alipf_fold()  | 554 |
| 16.91.2.25 alipf_circ_fold()   | 555 |
| 16.91.2.26 export_ali_bppm()   | 555 |
| 16.91.2.27 free_alipf_arrays()   | 556 |
| 16.91.2.28 alipbacktrack()   | 556 |
| 16.91.2.29 get_alipf_arrays()  | 556 |
| 16.92 Deprecated Interface for Local (Sliding Window) Partition Function Computation | 557 |
| 16.92.1 Detailed Description   | 557 |
| 16.92.2 Function Documentation   | 558 |
| 16.92.2.1 update_pf_paramsLP()   | 558 |
| 16.92.2.2 pfl_fold()   | 558 |
| 16.92.2.3 putoutpU_prob()  | 559 |
| 16.92.2.4 putoutpU_prob_bin()  | 559 |
| 16.93 Deprecated Interface for Stochastic Backtracking                               | 559 |
| 16.93.1 Detailed Description   | 559 |
| 16.93.2 Function Documentation   | 560 |
| 16.93.2.1 pbacktrack()   | 560 |
| 16.93.2.2 pbacktrack_circ()  | 560 |
| 16.93.3 Variable Documentation   | 561 |
| 16.93.3.1 st_back  | 561 |
| 16.94 Deprecated Interface for Multiple Sequence Alignment Utilities                 | 561 |
| 16.94.1 Detailed Description   | 561 |
| 16.94.2 Typedef Documentation  | 561 |
| 16.94.2.1 pair_info  | 562 |
| 16.94.3 Function Documentation   | 562 |
| 16.94.3.1 get_mpi()  | 562 |
| 16.94.3.2 encode_ali_sequence()  | 562 |
| 16.94.3.3 alloc_sequence_arrays()  | 563 |
| 16.94.3.4 free_sequence_arrays()   | 563 |
| 16.95 Deprecated Interface for Secondary Structure Utilities                         | 564 |
| 16.95.1 Detailed Description   | 564 |
| 16.95.2 Function Documentation   | 565 |
| 16.95.2.1 b2HIT()  | 565 |
| 16.95.2.2 b2C()  | 566 |
| 16.95.2.3 b2Shapiro()  | 566 |
| 16.95.2.4 add_root()   | 566 |
| 16.95.2.5 expand_Shapiro()   | 567 |
| 16.95.2.6 expand_Full()  | 567 |

---

|   |            |
|---|------------|
| 16.95.2.7 unexpand_Full()   | 567        |
| 16.95.2.8 unweight()  | 567        |
| 16.95.2.9 unexpand_aligned_F()  | 568        |
| 16.95.2.10 parse_structure()  | 568        |
| 16.95.2.11 pack_structure()   | 568        |
| 16.95.2.12 unpack_structure()   | 569        |
| 16.95.2.13 make_pair_table()  | 569        |
| 16.95.2.14 copy_pair_table()  | 569        |
| 16.95.2.15 alimake_pair_table()   | 570        |
| 16.95.2.16 make_pair_table_snoop()  | 570        |
| 16.95.2.17 bp_distance()  | 570        |
| 16.95.2.18 make_referenceBP_array()   | 570        |
| 16.95.2.19 compute_BPdifferences()  | 571        |
| 16.95.2.20 parenthesis_structure()  | 571        |
| 16.95.2.21 parenthesis_zuker()  | 571        |
| 16.95.2.22 bppm_to_structure()  | 572        |
| 16.95.2.23 bppm_symbol()  | 572        |
| 16.96 Deprecated Interface for Plotting Utilities                                     | 572        |
| 16.96.1 Detailed Description  | 572        |
| 16.96.2 Data Structure Documentation  | 572        |
| 16.96.2.1 struct COORDINATE   | 572        |
| 16.96.3 Function Documentation  | 572        |
| 16.96.3.1 PS_color_aln()  | 573        |
| 16.96.3.2 aliPS_color_aln()   | 573        |
| 16.96.3.3 simple_xy_coordinates()   | 573        |
| 16.96.3.4 simple_circplot_coordinates()   | 573        |
| 16.96.4 Variable Documentation  | 574        |
| 16.96.4.1 rna_plot_type   | 574        |
| 16.97 Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers | 574        |
| 16.97.1 Detailed Description  | 574        |
| 16.97.2 Typedef Documentation   | 575        |
| 16.97.2.1 path_t  | 575        |
| 16.97.3 Function Documentation  | 575        |
| 16.97.3.1 find_saddle()   | 575        |
| 16.97.3.2 free_path()   | 576        |
| 16.97.3.3 get_path()  | 576        |
| <b>17 Data Structure Documentation</b>  | <b>577</b> |
| 17.1 _struct_en Struct Reference  | 577        |
| 17.1.1 Detailed Description   | 577        |
| 17.2 LIST Struct Reference  | 577        |
| 17.3 LST_BUCKET Struct Reference  | 577        |

---

|   |            |
|---|------------|
| 17.4 Postorder_list Struct Reference . . . . .                    | 577        |
| 17.4.1 Detailed Description . . . . .                             | 577        |
| 17.5 swString Struct Reference . . . . .                          | 577        |
| 17.5.1 Detailed Description . . . . .                             | 578        |
| 17.6 Tree Struct Reference . . . . .                              | 578        |
| 17.6.1 Detailed Description . . . . .                             | 578        |
| 17.7 TwoDpfold_vars Struct Reference . . . . .                    | 578        |
| 17.7.1 Detailed Description . . . . .                             | 579        |
| 17.8 vrna_dimer_conc_s Struct Reference . . . . .                 | 579        |
| 17.8.1 Detailed Description . . . . .                             | 579        |
| 17.9 vrna_sc_bp_storage_t Struct Reference . . . . .              | 579        |
| 17.9.1 Detailed Description . . . . .                             | 579        |
| 17.10 vrna_structured_domains_s Struct Reference . . . . .        | 579        |
| 17.11 vrna_subopt_sol_s Struct Reference . . . . .                | 579        |
| 17.11.1 Detailed Description . . . . .                            | 580        |
| 17.12 vrna_unstructured_domain_motif_s Struct Reference . . . . . | 580        |
| <b>18 File Documentation</b> . . . . .                            | <b>581</b> |
| 18.1 ViennaRNA/2Dfold.h File Reference . . . . .                  | 581        |
| 18.2 2Dfold.h . . . . .   | 581        |
| 18.3 ViennaRNA/2Dpfold.h File Reference . . . . .                 | 583        |
| 18.3.1 Detailed Description . . . . .                             | 584        |
| 18.3.2 Function Documentation . . . . .                           | 584        |
| 18.3.2.1 get_TwoDpfold_variables() . . . . .                      | 584        |
| 18.3.2.2 destroy_TwoDpfold_variables() . . . . .                  | 584        |
| 18.3.2.3 TwoDpfoldList() . . . . .                                | 585        |
| 18.3.2.4 TwoDpfold_pbacktrack() . . . . .                         | 585        |
| 18.3.2.5 TwoDpfold_pbacktrack5() . . . . .                        | 586        |
| 18.4 2Dpfold.h . . . . .  | 587        |
| 18.5 ali_plex.h . . . . .   | 589        |
| 18.6 ViennaRNA/alifold.h File Reference . . . . .                 | 589        |
| 18.6.1 Detailed Description . . . . .                             | 590        |
| 18.6.2 Function Documentation . . . . .                           | 590        |
| 18.6.2.1 energy_of_alistruct() . . . . .                          | 591        |
| 18.6.2.2 update_alifold_params() . . . . .                        | 591        |
| 18.6.3 Variable Documentation . . . . .                           | 591        |
| 18.6.3.1 cv_fact . . . . .  | 591        |
| 18.6.3.2 nc_fact . . . . .  | 591        |
| 18.7 alifold.h . . . . .  | 592        |
| 18.8 ViennaRNA/aln_util.h File Reference . . . . .                | 593        |
| 18.8.1 Detailed Description . . . . .                             | 593        |
| 18.9 aln_util.h . . . . .   | 593        |

---

|   |     |
|---|-----|
| 18.10 ViennaRNA/alphabet.h File Reference . . . . .                   | 593 |
| 18.10.1 Detailed Description . . . . .                                | 594 |
| 18.11 alphabet.h . . . . .  | 594 |
| 18.12 ViennaRNA/boltzmann_sampling.h File Reference . . . . .         | 595 |
| 18.12.1 Detailed Description . . . . .                                | 596 |
| 18.13 boltzmann_sampling.h . . . . .                                  | 596 |
| 18.14 ViennaRNA/centroid.h File Reference . . . . .                   | 598 |
| 18.14.1 Detailed Description . . . . .                                | 598 |
| 18.14.2 Function Documentation . . . . .                              | 598 |
| 18.14.2.1 get_centroid_struct_pl() . . . . .                          | 599 |
| 18.14.2.2 get_centroid_struct_pr() . . . . .                          | 599 |
| 18.15 centroid.h . . . . .  | 599 |
| 18.16 ViennaRNA/char_stream.h File Reference . . . . .                | 600 |
| 18.16.1 Detailed Description . . . . .                                | 600 |
| 18.17 char_stream.h . . . . .   | 600 |
| 18.18 ViennaRNA/datastructures/char_stream.h File Reference . . . . . | 600 |
| 18.18.1 Detailed Description . . . . .                                | 600 |
| 18.19 char_stream.h . . . . .   | 600 |
| 18.20 ViennaRNA/cofold.h File Reference . . . . .                     | 603 |
| 18.20.1 Detailed Description . . . . .                                | 603 |
| 18.21 cofold.h . . . . .  | 603 |
| 18.22 ViennaRNA/combinatorics.h File Reference . . . . .              | 604 |
| 18.22.1 Detailed Description . . . . .                                | 605 |
| 18.23 combinatorics.h . . . . .                                       | 605 |
| 18.24 ViennaRNA/commands.h File Reference . . . . .                   | 606 |
| 18.24.1 Detailed Description . . . . .                                | 606 |
| 18.25 commands.h . . . . .  | 606 |
| 18.26 ViennaRNA/concentrations.h File Reference . . . . .             | 607 |
| 18.26.1 Detailed Description . . . . .                                | 607 |
| 18.26.2 Function Documentation . . . . .                              | 608 |
| 18.26.2.1 get_concentrations() . . . . .                              | 608 |
| 18.27 concentrations.h . . . . .                                      | 608 |
| 18.28 ViennaRNA/constraints.h File Reference . . . . .                | 609 |
| 18.28.1 Detailed Description . . . . .                                | 609 |
| 18.29 constraints.h . . . . .   | 609 |
| 18.30 ViennaRNA/constraints/hard.h File Reference . . . . .           | 610 |
| 18.30.1 Detailed Description . . . . .                                | 612 |
| 18.30.2 Macro Definition Documentation . . . . .                      | 612 |
| 18.30.2.1 VRNA_CONSTRAINT_NO_HEADER . . . . .                         | 612 |
| 18.30.2.2 VRNA_CONSTRAINT_DB_ANG_BRACK . . . . .                      | 612 |
| 18.30.3 Enumeration Type Documentation . . . . .                      | 612 |
| 18.30.3.1 vrna_hc_type_e . . . . .                                    | 612 |

---

|   |     |
|---|-----|
| 18.30.4 Function Documentation . . . . .                              | 612 |
| 18.30.4.1 vrna_hc_add_data() . . . . .                                | 613 |
| 18.30.4.2 print_tty_constraint() . . . . .                            | 613 |
| 18.30.4.3 print_tty_constraint_full() . . . . .                       | 613 |
| 18.30.4.4 constrain_ptypes() . . . . .                                | 613 |
| 18.31 hard.h . . . . .  | 614 |
| 18.32 ViennaRNA/constraints/ligand.h File Reference . . . . .         | 617 |
| 18.32.1 Detailed Description . . . . .                                | 617 |
| 18.33 ligand.h . . . . .  | 617 |
| 18.34 ViennaRNA/constraints/SHAPE.h File Reference . . . . .          | 618 |
| 18.34.1 Detailed Description . . . . .                                | 618 |
| 18.34.2 Function Documentation . . . . .                              | 618 |
| 18.34.2.1 vrna_sc_SHAPE_parse_method() . . . . .                      | 618 |
| 18.35 SHAPE.h . . . . .   | 619 |
| 18.36 ViennaRNA/constraints/soft.h File Reference . . . . .           | 619 |
| 18.36.1 Detailed Description . . . . .                                | 620 |
| 18.36.2 Enumeration Type Documentation . . . . .                      | 620 |
| 18.36.2.1 vrna_sc_type_e . . . . .                                    | 620 |
| 18.37 soft.h . . . . .  | 621 |
| 18.38 ViennaRNA/constraints_hard.h File Reference . . . . .           | 623 |
| 18.38.1 Detailed Description . . . . .                                | 623 |
| 18.39 constraints_hard.h . . . . .                                    | 623 |
| 18.40 ViennaRNA/constraints_ligand.h File Reference . . . . .         | 623 |
| 18.40.1 Detailed Description . . . . .                                | 624 |
| 18.41 constraints_ligand.h . . . . .                                  | 624 |
| 18.42 ViennaRNA/constraints_SHAPE.h File Reference . . . . .          | 624 |
| 18.42.1 Detailed Description . . . . .                                | 624 |
| 18.43 constraints_SHAPE.h . . . . .                                   | 624 |
| 18.44 ViennaRNA/constraints_soft.h File Reference . . . . .           | 624 |
| 18.44.1 Detailed Description . . . . .                                | 624 |
| 18.45 constraints_soft.h . . . . .                                    | 624 |
| 18.46 ViennaRNA/convert_epars.h File Reference . . . . .              | 625 |
| 18.46.1 Detailed Description . . . . .                                | 625 |
| 18.47 convert_epars.h . . . . .                                       | 625 |
| 18.48 ViennaRNA/data_structures.h File Reference . . . . .            | 625 |
| 18.48.1 Detailed Description . . . . .                                | 625 |
| 18.49 data_structures.h . . . . .                                     | 625 |
| 18.50 ViennaRNA/datastructures/hash_tables.h File Reference . . . . . | 626 |
| 18.50.1 Detailed Description . . . . .                                | 626 |
| 18.51 hash_tables.h . . . . .   | 627 |
| 18.52 ViennaRNA/datastructures/heap.h File Reference . . . . .        | 628 |
| 18.52.1 Detailed Description . . . . .                                | 628 |

---

|  |     |
|--|-----|
| 18.53 heap.h . . . . .   | 628 |
| 18.54 lists.h . . . . .  | 629 |
| 18.55 ViennaRNA/dist_vars.h File Reference . . . . .           | 630 |
| 18.55.1 Detailed Description . . . . .                         | 630 |
| 18.55.2 Variable Documentation . . . . .                       | 630 |
| 18.55.2.1 edit_backtrack . . . . .                             | 630 |
| 18.55.2.2 cost_matrix . . . . .                                | 631 |
| 18.56 dist_vars.h . . . . .                                    | 631 |
| 18.57 ViennaRNA/dp_matrices.h File Reference . . . . .         | 631 |
| 18.57.1 Detailed Description . . . . .                         | 632 |
| 18.58 dp_matrices.h . . . . .                                  | 632 |
| 18.59 ViennaRNA/duplex.h File Reference . . . . .              | 635 |
| 18.59.1 Detailed Description . . . . .                         | 635 |
| 18.60 duplex.h . . . . .                                       | 635 |
| 18.61 ViennaRNA/edit_cost.h File Reference . . . . .           | 636 |
| 18.61.1 Detailed Description . . . . .                         | 636 |
| 18.62 edit_cost.h . . . . .                                    | 636 |
| 18.63 ViennaRNA/energy_const.h File Reference . . . . .        | 637 |
| 18.63.1 Detailed Description . . . . .                         | 637 |
| 18.64 energy_const.h . . . . .                                 | 637 |
| 18.65 ViennaRNA/energy_par.h File Reference . . . . .          | 637 |
| 18.65.1 Detailed Description . . . . .                         | 637 |
| 18.66 energy_par.h . . . . .                                   | 638 |
| 18.67 ViennaRNA/equilibrium_probs.h File Reference . . . . .   | 638 |
| 18.67.1 Detailed Description . . . . .                         | 638 |
| 18.68 equilibrium_probs.h . . . . .                            | 639 |
| 18.69 ViennaRNA/eval.h File Reference . . . . .                | 639 |
| 18.69.1 Detailed Description . . . . .                         | 642 |
| 18.70 eval.h . . . . .   | 642 |
| 18.71 ViennaRNA/exterior_loops.h File Reference . . . . .      | 646 |
| 18.71.1 Detailed Description . . . . .                         | 647 |
| 18.72 exterior_loops.h . . . . .                               | 647 |
| 18.73 ViennaRNA/file_formats.h File Reference . . . . .        | 647 |
| 18.73.1 Detailed Description . . . . .                         | 647 |
| 18.74 file_formats.h . . . . .                                 | 647 |
| 18.75 ViennaRNA/io/file_formats.h File Reference . . . . .     | 647 |
| 18.75.1 Detailed Description . . . . .                         | 648 |
| 18.76 file_formats.h . . . . .                                 | 648 |
| 18.77 ViennaRNA/file_formats_msa.h File Reference . . . . .    | 649 |
| 18.77.1 Detailed Description . . . . .                         | 650 |
| 18.78 file_formats_msa.h . . . . .                             | 650 |
| 18.79 ViennaRNA/io/file_formats_msa.h File Reference . . . . . | 650 |

---

|   |     |
|---|-----|
| 18.79.1 Detailed Description . . . . .                        | 651 |
| 18.80 file_formats_msa.h . . . . .                            | 651 |
| 18.81 ViennaRNA/file_utils.h File Reference . . . . .         | 652 |
| 18.81.1 Detailed Description . . . . .                        | 652 |
| 18.82 file_utils.h . . . . .                                  | 652 |
| 18.83 ViennaRNA/findpath.h File Reference . . . . .           | 652 |
| 18.83.1 Detailed Description . . . . .                        | 652 |
| 18.84 findpath.h . . . . .                                    | 652 |
| 18.85 ViennaRNA/landscape/findpath.h File Reference . . . . . | 653 |
| 18.85.1 Detailed Description . . . . .                        | 653 |
| 18.86 findpath.h . . . . .                                    | 653 |
| 18.87 ViennaRNA/fold.h File Reference . . . . .               | 654 |
| 18.87.1 Detailed Description . . . . .                        | 655 |
| 18.88 fold.h . . . . .  | 655 |
| 18.89 ViennaRNA/fold_compound.h File Reference . . . . .      | 656 |
| 18.89.1 Detailed Description . . . . .                        | 657 |
| 18.90 fold_compound.h . . . . .                               | 658 |
| 18.91 ViennaRNA/fold_vars.h File Reference . . . . .          | 660 |
| 18.91.1 Detailed Description . . . . .                        | 660 |
| 18.91.2 Variable Documentation . . . . .                      | 661 |
| 18.91.2.1 RibosumFile . . . . .                               | 661 |
| 18.91.2.2 james_rule . . . . .                                | 661 |
| 18.91.2.3 logML . . . . .                                     | 661 |
| 18.91.2.4 cut_point . . . . .                                 | 661 |
| 18.91.2.5 base_pair . . . . .                                 | 661 |
| 18.91.2.6 pr . . . . .  | 661 |
| 18.91.2.7 iindx . . . . .                                     | 661 |
| 18.92 fold_vars.h . . . . .                                   | 661 |
| 18.93 ViennaRNA/gquad.h File Reference . . . . .              | 662 |
| 18.93.1 Detailed Description . . . . .                        | 662 |
| 18.94 gquad.h . . . . .                                       | 662 |
| 18.95 ViennaRNA/grammar.h File Reference . . . . .            | 682 |
| 18.95.1 Detailed Description . . . . .                        | 682 |
| 18.96 grammar.h . . . . .                                     | 682 |
| 18.97 ViennaRNA/hairpin_loops.h File Reference . . . . .      | 683 |
| 18.97.1 Detailed Description . . . . .                        | 683 |
| 18.98 hairpin_loops.h . . . . .                               | 683 |
| 18.99 ViennaRNA/heat_capacity.h File Reference . . . . .      | 684 |
| 18.99.1 Detailed Description . . . . .                        | 684 |
| 18.100 heat_capacity.h . . . . .                              | 684 |
| 18.101 ViennaRNA/interior_loops.h File Reference . . . . .    | 685 |
| 18.101.1 Detailed Description . . . . .                       | 685 |

---

|   |     |
|---|-----|
| 18.102 interior_loops.h . . . . .                             | 685 |
| 18.103 ViennaRNA/inverse.h File Reference . . . . .           | 685 |
| 18.103.1 Detailed Description . . . . .                       | 686 |
| 18.104 inverse.h . . . . .                                    | 686 |
| 18.105 ViennaRNA/landscape/move.h File Reference . . . . .    | 686 |
| 18.105.1 Detailed Description . . . . .                       | 687 |
| 18.106 move.h . . . . .                                       | 687 |
| 18.107 ViennaRNA/landscape/paths.h File Reference . . . . .   | 688 |
| 18.107.1 Detailed Description . . . . .                       | 689 |
| 18.108 paths.h . . . . .                                      | 689 |
| 18.109 ViennaRNA/Lfold.h File Reference . . . . .             | 690 |
| 18.109.1 Detailed Description . . . . .                       | 690 |
| 18.110 Lfold.h . . . . .                                      | 690 |
| 18.111 ViennaRNA/loop_energies.h File Reference . . . . .     | 690 |
| 18.111.1 Detailed Description . . . . .                       | 691 |
| 18.112 loop_energies.h . . . . .                              | 691 |
| 18.113 ViennaRNA/loops/all.h File Reference . . . . .         | 691 |
| 18.113.1 Detailed Description . . . . .                       | 691 |
| 18.114 all.h . . . . .  | 691 |
| 18.115 ViennaRNA/loops/external.h File Reference . . . . .    | 691 |
| 18.115.1 Detailed Description . . . . .                       | 692 |
| 18.116 external.h . . . . .                                   | 692 |
| 18.117 ViennaRNA/loops/hairpin.h File Reference . . . . .     | 694 |
| 18.117.1 Detailed Description . . . . .                       | 694 |
| 18.118 hairpin.h . . . . .                                    | 694 |
| 18.119 ViennaRNA/loops/internal.h File Reference . . . . .    | 697 |
| 18.119.1 Detailed Description . . . . .                       | 697 |
| 18.120 internal.h . . . . .                                   | 697 |
| 18.121 ViennaRNA/loops/multibranch.h File Reference . . . . . | 705 |
| 18.121.1 Detailed Description . . . . .                       | 705 |
| 18.122 multibranch.h . . . . .                                | 705 |
| 18.123 ViennaRNA/LPfold.h File Reference . . . . .            | 707 |
| 18.123.1 Detailed Description . . . . .                       | 708 |
| 18.123.2 Function Documentation . . . . .                     | 708 |
| 18.123.2.1 init_pf_foldLP() . . . . .                         | 708 |
| 18.124 LPfold.h . . . . .                                     | 708 |
| 18.125 ViennaRNA/MEA.h File Reference . . . . .               | 709 |
| 18.125.1 Detailed Description . . . . .                       | 710 |
| 18.126 MEA.h . . . . .  | 710 |
| 18.127 ViennaRNA/mfe.h File Reference . . . . .               | 710 |
| 18.127.1 Detailed Description . . . . .                       | 711 |
| 18.128 mfe.h . . . . .  | 711 |

---

|   |     |
|---|-----|
| 18.129 ViennaRNA/mfe_window.h File Reference . . . . .            | 712 |
| 18.129.1 Detailed Description . . . . .                           | 713 |
| 18.130 mfe_window.h . . . . .                                     | 713 |
| 18.131 ViennaRNA/mm.h File Reference . . . . .                    | 714 |
| 18.131.1 Detailed Description . . . . .                           | 714 |
| 18.131.2 Function Documentation . . . . .                         | 714 |
| 18.131.2.1 vrna_maximum_matching() . . . . .                      | 714 |
| 18.131.2.2 vrna_maximum_matching_simple() . . . . .               | 714 |
| 18.132 mm.h . . . . .   | 715 |
| 18.133 ViennaRNA/model.h File Reference . . . . .                 | 715 |
| 18.133.1 Detailed Description . . . . .                           | 719 |
| 18.134 model.h . . . . .  | 719 |
| 18.135 move_set.h . . . . .                                       | 723 |
| 18.136 ViennaRNA/multibranch_loops.h File Reference . . . . .     | 725 |
| 18.136.1 Detailed Description . . . . .                           | 725 |
| 18.137 multibranch_loops.h . . . . .                              | 725 |
| 18.138 ViennaRNA/naview.h File Reference . . . . .                | 725 |
| 18.138.1 Detailed Description . . . . .                           | 725 |
| 18.139 naview.h . . . . .   | 725 |
| 18.140 ViennaRNA/landscape/neighbor.h File Reference . . . . .    | 726 |
| 18.140.1 Detailed Description . . . . .                           | 726 |
| 18.141 neighbor.h . . . . .                                       | 726 |
| 18.142 ViennaRNA/neighbor.h File Reference . . . . .              | 727 |
| 18.142.1 Detailed Description . . . . .                           | 727 |
| 18.143 neighbor.h . . . . .                                       | 727 |
| 18.144 pair_mat.h . . . . .                                       | 728 |
| 18.145 ViennaRNA/params.h File Reference . . . . .                | 730 |
| 18.145.1 Detailed Description . . . . .                           | 730 |
| 18.146 params.h . . . . .   | 730 |
| 18.147 ViennaRNA/params/1.8.4_epars.h File Reference . . . . .    | 730 |
| 18.147.1 Detailed Description . . . . .                           | 730 |
| 18.148 1.8.4_epars.h . . . . .                                    | 730 |
| 18.149 ViennaRNA/params/1.8.4_intloops.h File Reference . . . . . | 734 |
| 18.149.1 Detailed Description . . . . .                           | 734 |
| 18.150 1.8.4_intloops.h . . . . .                                 | 735 |
| 18.151 ViennaRNA/constraints/basic.h File Reference . . . . .     | 882 |
| 18.151.1 Detailed Description . . . . .                           | 883 |
| 18.152 basic.h . . . . .  | 883 |
| 18.153 ViennaRNA/datastructures/basic.h File Reference . . . . .  | 883 |
| 18.153.1 Detailed Description . . . . .                           | 885 |
| 18.154 basic.h . . . . .  | 885 |
| 18.155 ViennaRNA/params/basic.h File Reference . . . . .          | 888 |

---

|  |      |
|--|------|
| 18.155.1 Detailed Description . . . . .                      | 889  |
| 18.156 basic.h . . . . .                                     | 889  |
| 18.157 ViennaRNA/utils/basic.h File Reference . . . . .      | 892  |
| 18.157.1 Detailed Description . . . . .                      | 894  |
| 18.157.2 Function Documentation . . . . .                    | 894  |
| 18.157.2.1 get_line() . . . . .                              | 894  |
| 18.157.2.2 print_tty_input_seq() . . . . .                   | 894  |
| 18.157.2.3 print_tty_input_seq_str() . . . . .               | 894  |
| 18.157.2.4 warn_user() . . . . .                             | 894  |
| 18.157.2.5 nerror() . . . . .                                | 895  |
| 18.157.2.6 space() . . . . .                                 | 895  |
| 18.157.2.7 xrealloc() . . . . .                              | 895  |
| 18.157.2.8 init_rand() . . . . .                             | 895  |
| 18.157.2.9 urn() . . . . .                                   | 895  |
| 18.157.2.10 int_urn() . . . . .                              | 896  |
| 18.157.2.11 filecopy() . . . . .                             | 896  |
| 18.157.2.12 time_stamp() . . . . .                           | 896  |
| 18.158 basic.h . . . . .                                     | 896  |
| 18.159 ViennaRNA/params/constants.h File Reference . . . . . | 898  |
| 18.159.1 Detailed Description . . . . .                      | 899  |
| 18.159.2 Macro Definition Documentation . . . . .            | 899  |
| 18.159.2.1 GASCONST . . . . .                                | 899  |
| 18.159.2.2 K0 . . . . .                                      | 899  |
| 18.159.2.3 INF . . . . .                                     | 899  |
| 18.159.2.4 FORBIDDEN . . . . .                               | 899  |
| 18.159.2.5 BONUS . . . . .                                   | 899  |
| 18.159.2.6 NBPAIRS . . . . .                                 | 899  |
| 18.159.2.7 TURN . . . . .                                    | 899  |
| 18.159.2.8 MAXLOOP . . . . .                                 | 899  |
| 18.160 constants.h . . . . .                                 | 900  |
| 18.161 ViennaRNA/params/convert.h File Reference . . . . .   | 900  |
| 18.161.1 Detailed Description . . . . .                      | 900  |
| 18.162 convert.h . . . . .                                   | 901  |
| 18.163 default.h . . . . .                                   | 901  |
| 18.164 intl11.h . . . . .                                    | 902  |
| 18.165 intl11dH.h . . . . .                                  | 907  |
| 18.166 intl21.h . . . . .                                    | 911  |
| 18.167 intl21dH.h . . . . .                                  | 934  |
| 18.168 intl22.h . . . . .                                    | 957  |
| 18.169 intl22dH.h . . . . .                                  | 1072 |
| 18.170 ViennaRNA/params/io.h File Reference . . . . .        | 1187 |
| 18.170.1 Detailed Description . . . . .                      | 1188 |

---

|   |      |
|---|------|
| 18.171 io.h . . . . .   | 1188 |
| 18.172 ViennaRNA/part_func.h File Reference . . . . .         | 1189 |
| 18.172.1 Detailed Description . . . . .                       | 1191 |
| 18.172.2 Function Documentation . . . . .                     | 1191 |
| 18.172.2.1 centroid() . . . . .                               | 1191 |
| 18.172.2.2 get_centroid_struct_gquad_pr() . . . . .           | 1191 |
| 18.172.2.3 mean_bp_dist() . . . . .                           | 1192 |
| 18.172.2.4 expLoopEnergy() . . . . .                          | 1192 |
| 18.172.2.5 expHairpinEnergy() . . . . .                       | 1192 |
| 18.173 part_func.h . . . . .                                  | 1192 |
| 18.174 ViennaRNA/part_func_co.h File Reference . . . . .      | 1195 |
| 18.174.1 Detailed Description . . . . .                       | 1196 |
| 18.174.2 Function Documentation . . . . .                     | 1196 |
| 18.174.2.1 get plist() . . . . .                              | 1196 |
| 18.175 part_func_co.h . . . . .                               | 1196 |
| 18.176 ViennaRNA/part_func_up.h File Reference . . . . .      | 1197 |
| 18.176.1 Detailed Description . . . . .                       | 1197 |
| 18.177 part_func_up.h . . . . .                               | 1197 |
| 18.178 ViennaRNA/part_func_window.h File Reference . . . . .  | 1198 |
| 18.178.1 Detailed Description . . . . .                       | 1199 |
| 18.179 part_func_window.h . . . . .                           | 1199 |
| 18.180 ViennaRNA/perturbation_fold.h File Reference . . . . . | 1200 |
| 18.180.1 Detailed Description . . . . .                       | 1201 |
| 18.181 perturbation_fold.h . . . . .                          | 1201 |
| 18.182 pf_multifold.h . . . . .                               | 1202 |
| 18.183 ViennaRNA/pk_plex.h File Reference . . . . .           | 1202 |
| 18.183.1 Detailed Description . . . . .                       | 1202 |
| 18.184 pk_plex.h . . . . .                                    | 1203 |
| 18.185 PKplex.h . . . . .                                     | 1203 |
| 18.186 plex.h . . . . .                                       | 1204 |
| 18.187 ViennaRNA/plot_aln.h File Reference . . . . .          | 1205 |
| 18.187.1 Detailed Description . . . . .                       | 1205 |
| 18.188 plot_aln.h . . . . .                                   | 1205 |
| 18.189 ViennaRNA/plot_layouts.h File Reference . . . . .      | 1205 |
| 18.189.1 Detailed Description . . . . .                       | 1205 |
| 18.190 plot_layouts.h . . . . .                               | 1205 |
| 18.191 ViennaRNA/plot_structure.h File Reference . . . . .    | 1205 |
| 18.191.1 Detailed Description . . . . .                       | 1206 |
| 18.192 plot_structure.h . . . . .                             | 1206 |
| 18.193 ViennaRNA/plot_utils.h File Reference . . . . .        | 1206 |
| 18.193.1 Detailed Description . . . . .                       | 1206 |
| 18.194 plot_utils.h . . . . .                                 | 1206 |

---

|  |      |
|--|------|
| 18.195 ViennaRNA/plotting/alignments.h File Reference . . . . .            | 1206 |
| 18.195.1 Detailed Description . . . . .                                    | 1207 |
| 18.196 alignments.h . . . . .  | 1207 |
| 18.197 ViennaRNA/utils/alignments.h File Reference . . . . .               | 1207 |
| 18.197.1 Detailed Description . . . . .                                    | 1209 |
| 18.198 alignments.h . . . . .  | 1209 |
| 18.199 ViennaRNA/plotting/layouts.h File Reference . . . . .               | 1211 |
| 18.199.1 Detailed Description . . . . .                                    | 1212 |
| 18.200 layouts.h . . . . .   | 1212 |
| 18.201 ViennaRNA/plotting/probabilities.h File Reference . . . . .         | 1214 |
| 18.201.1 Detailed Description . . . . .                                    | 1214 |
| 18.202 probabilities.h . . . . .   | 1214 |
| 18.203 ViennaRNA/plotting/RNApuzzler/RNApuzzler.h File Reference . . . . . | 1216 |
| 18.203.1 Detailed Description . . . . .                                    | 1216 |
| 18.204 RNApuzzler.h . . . . .  | 1216 |
| 18.205 ViennaRNA/plotting/RNApuzzler/RNAturtle.h File Reference . . . . .  | 1217 |
| 18.205.1 Detailed Description . . . . .                                    | 1217 |
| 18.206 RNAturtle.h . . . . .   | 1217 |
| 18.207 ViennaRNA/plotting/structures.h File Reference . . . . .            | 1218 |
| 18.207.1 Detailed Description . . . . .                                    | 1218 |
| 18.208 structures.h . . . . .  | 1218 |
| 18.209 ViennaRNA/utils/structures.h File Reference . . . . .               | 1219 |
| 18.209.1 Detailed Description . . . . .                                    | 1223 |
| 18.210 structures.h . . . . .  | 1223 |
| 18.211 ProfileAln.h . . . . .  | 1227 |
| 18.212 ViennaRNA/profiledist.h File Reference . . . . .                    | 1227 |
| 18.212.1 Detailed Description . . . . .                                    | 1227 |
| 18.212.2 Function Documentation . . . . .                                  | 1227 |
| 18.212.2.1 profile_edit_distance() . . . . .                               | 1228 |
| 18.212.2.2 Make_bp_profile_bppm() . . . . .                                | 1228 |
| 18.212.2.3 free_profile() . . . . .  | 1228 |
| 18.212.2.4 Make_bp_profile() . . . . .                                     | 1228 |
| 18.213 profiledist.h . . . . .   | 1228 |
| 18.214 ViennaRNA/PS_dot.h File Reference . . . . .                         | 1229 |
| 18.214.1 Detailed Description . . . . .                                    | 1229 |
| 18.215 PS_dot.h . . . . .  | 1229 |
| 18.216 ViennaRNA/read_epars.h File Reference . . . . .                     | 1229 |
| 18.216.1 Detailed Description . . . . .                                    | 1229 |
| 18.217 read_epars.h . . . . .  | 1230 |
| 18.218 ViennaRNA/ribo.h File Reference . . . . .                           | 1230 |
| 18.218.1 Detailed Description . . . . .                                    | 1230 |
| 18.219 ribo.h . . . . .  | 1230 |

---

|  |      |
|--|------|
| 18.220 ViennaRNA/RNAstruct.h File Reference . . . . .                    | 1230 |
| 18.220.1 Detailed Description . . . . .                                  | 1231 |
| 18.221 RNAstruct.h . . . . .   | 1231 |
| 18.222 ViennaRNA/search/BoyerMoore.h File Reference . . . . .            | 1232 |
| 18.222.1 Detailed Description . . . . .                                  | 1233 |
| 18.223 BoyerMoore.h . . . . .  | 1233 |
| 18.224 ViennaRNA/sequence.h File Reference . . . . .                     | 1233 |
| 18.224.1 Detailed Description . . . . .                                  | 1234 |
| 18.225 sequence.h . . . . .  | 1234 |
| 18.226 snofold.h . . . . .   | 1235 |
| 18.227 snoop.h . . . . .   | 1236 |
| 18.228 special_const.h . . . . .   | 1239 |
| 18.229 ViennaRNA/datastructures/stream_output.h File Reference . . . . . | 1239 |
| 18.229.1 Detailed Description . . . . .                                  | 1239 |
| 18.230 stream_output.h . . . . .   | 1239 |
| 18.231 ViennaRNA/stream_output.h File Reference . . . . .                | 1240 |
| 18.231.1 Detailed Description . . . . .                                  | 1240 |
| 18.232 stream_output.h . . . . .   | 1240 |
| 18.233 ViennaRNA/string_utils.h File Reference . . . . .                 | 1240 |
| 18.233.1 Detailed Description . . . . .                                  | 1240 |
| 18.234 string_utils.h . . . . .  | 1240 |
| 18.235 ViennaRNA/stringdist.h File Reference . . . . .                   | 1241 |
| 18.235.1 Detailed Description . . . . .                                  | 1241 |
| 18.235.2 Function Documentation . . . . .                                | 1241 |
| 18.235.2.1 Make_swString() . . . . .                                     | 1241 |
| 18.235.2.2 string_edit_distance() . . . . .                              | 1241 |
| 18.236 stringdist.h . . . . .  | 1242 |
| 18.237 ViennaRNA/structure_utils.h File Reference . . . . .              | 1242 |
| 18.237.1 Detailed Description . . . . .                                  | 1242 |
| 18.238 structure_utils.h . . . . .                                       | 1242 |
| 18.239 ViennaRNA/structured_domains.h File Reference . . . . .           | 1242 |
| 18.239.1 Detailed Description . . . . .                                  | 1242 |
| 18.240 structured_domains.h . . . . .                                    | 1242 |
| 18.241 ViennaRNA/subopt.h File Reference . . . . .                       | 1243 |
| 18.241.1 Detailed Description . . . . .                                  | 1244 |
| 18.241.2 Typedef Documentation . . . . .                                 | 1244 |
| 18.241.2.1 SOLUTION . . . . .  | 1244 |
| 18.242 subopt.h . . . . .  | 1244 |
| 18.243 subopt_zuker.h . . . . .  | 1245 |
| 18.244 ViennaRNA/svm_utils.h File Reference . . . . .                    | 1245 |
| 18.244.1 Detailed Description . . . . .                                  | 1245 |
| 18.245 svm_utils.h . . . . .   | 1245 |

---

|  |      |
|--|------|
| 18.246 ViennaRNA/treedist.h File Reference . . . . .             | 1246 |
| 18.246.1 Detailed Description . . . . .                          | 1246 |
| 18.246.2 Function Documentation . . . . .                        | 1246 |
| 18.246.2.1 make_tree() . . . . .                                 | 1246 |
| 18.246.2.2 tree_edit_distance() . . . . .                        | 1246 |
| 18.246.2.3 free_tree() . . . . .                                 | 1247 |
| 18.247 treedist.h . . . . .                                      | 1247 |
| 18.248 ViennaRNA/units.h File Reference . . . . .                | 1247 |
| 18.248.1 Detailed Description . . . . .                          | 1247 |
| 18.249 units.h . . . . .   | 1247 |
| 18.250 ViennaRNA/utils/units.h File Reference . . . . .          | 1248 |
| 18.250.1 Detailed Description . . . . .                          | 1248 |
| 18.251 units.h . . . . .   | 1248 |
| 18.252 ViennaRNA/unstructured_domains.h File Reference . . . . . | 1249 |
| 18.252.1 Detailed Description . . . . .                          | 1250 |
| 18.252.2 Function Documentation . . . . .                        | 1251 |
| 18.252.2.1 vrna_ud_set_prob_cb() . . . . .                       | 1251 |
| 18.253 unstructured_domains.h . . . . .                          | 1251 |
| 18.254 ViennaRNA/io/utils.h File Reference . . . . .             | 1253 |
| 18.254.1 Detailed Description . . . . .                          | 1253 |
| 18.255 utils.h . . . . .   | 1253 |
| 18.256 ViennaRNA/plotting/utils.h File Reference . . . . .       | 1254 |
| 18.256.1 Detailed Description . . . . .                          | 1254 |
| 18.257 utils.h . . . . .   | 1254 |
| 18.258 ViennaRNA/utils.h File Reference . . . . .                | 1254 |
| 18.258.1 Detailed Description . . . . .                          | 1254 |
| 18.259 utils.h . . . . .   | 1255 |
| 18.260 cpu.h . . . . .   | 1255 |
| 18.261 higher_order_functions.h . . . . .                        | 1255 |
| 18.262 ViennaRNA/utils/strings.h File Reference . . . . .        | 1255 |
| 18.262.1 Detailed Description . . . . .                          | 1257 |
| 18.262.2 Function Documentation . . . . .                        | 1257 |
| 18.262.2.1 str_uppercase() . . . . .                             | 1257 |
| 18.262.2.2 str_DNA2RNA() . . . . .                               | 1257 |
| 18.262.2.3 random_string() . . . . .                             | 1257 |
| 18.262.2.4 hamming() . . . . .                                   | 1258 |
| 18.262.2.5 hamming_bound() . . . . .                             | 1258 |
| 18.263 strings.h . . . . .                                       | 1258 |
| 18.264 svm.h . . . . .   | 1260 |
| 18.265 vrna_config.h . . . . .                                   | 1260 |
| 18.266 ViennaRNA/landscape/walk.h File Reference . . . . .       | 1261 |
| 18.266.1 Detailed Description . . . . .                          | 1262 |

---

|  |             |
|--|-------------|
| 18.267 walk.h . . . . .                          | 1262        |
| 18.268 ViennaRNA/walk.h File Reference . . . . . | 1263        |
| 18.268.1 Detailed Description . . . . .          | 1263        |
| 18.269 walk.h . . . . .                          | 1263        |
| 18.270 wrap_dlib.h . . . . .                     | 1263        |
| 18.271 zscore.h . . . . .                        | 1263        |
| <b>Bibliography</b>                              | <b>1266</b> |
| <b>Index</b>                                     | <b>1267</b> |

# Chapter 1

## RNALib-2.5.1

### 1.1 A Library for predicting and comparing RNA secondary structures

The core of the ViennaRNA Package ([14], [12]) is formed by a collection of routines for the prediction and comparison of RNA secondary structures. These routines can be accessed through stand-alone programs, such as RNAfold, RNAdistance etc., which should be sufficient for most users. For those who wish to develop their own programs we provide a library which can be linked to your own code.

This document describes the library and will be primarily useful to programmers. However, it also contains details about the implementation that may be of interest to advanced users. The stand-alone programs are described in separate man pages. The latest version of the package including source code and html versions of the documentation can be found at

<http://www.tbi.univie.ac.at/RNA>

#### Date

1994-2020

#### Authors

Ivo Hofacker, Peter Stadler, Ronny Lorenz, and so many more

### 1.2 License

#### Disclaimer and Copyright

The programs, library and source code of the Vienna RNA Package are free software. They are distributed in the hope that they will be useful but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Permission is granted for research, educational, and commercial use and modification so long as 1) the package and any derived works are not redistributed for any fee, other than media costs, 2) proper credit is given to the authors and the Institute for Theoretical Chemistry of the University of Vienna.

If you want to include this software in a commercial product, please contact the authors.

### 1.3 Contributors

Over the past decades since the ViennaRNA Package first sprang to life as part of Ivo Hofackers PhD project, several different authors contributed more and more algorithm implementations. In 2008, Ronny Lorenz took over the extensive task to harmonize and simplify the already existing implementations for the sake of easier feature addition. This eventually lead to version 2.0 of the ViennaRNA Package. Since then, he (re-)implemented a large portion of the currently existing library features, such as the new, generalized constraints framework, RNA folding grammar domain extensions, and the major part of the scripting language interface. Below is a list of most people who contributed larger parts of the implementations:

- Daniel Wiegreffe (RNAturtle and RNApuzzler secondary structure layouts)
- Andreas Gruber (first approach on RNALfold Z-score filtering)
- Juraj Michalik (non-redundant Boltzmann sampling)
- Gregor Entzian (neighbor, walk)
- Mario Koestl (worked on SWIG interface and related unit testing)
- Dominik Luntzer (perturbation fold)
- Stefan Badelt (cofold evaluation, RNAdesign.pl, cofold findpath extensions)
- Stefan Hammer (parts of SWIG interface and corresponding unit tests)
- Ronny Lorenz (circfold, version 2.0, generic constraints, grammar extensions, and much more)
- Hakim Tafer (RNAPlex, RNAsnoop)
- Ulrike Mueckstein (RNAup)
- Stephan Bernhart (RNACofold, RNAPlfold, unpaired probabilities, alifold, and so many more)
- Stefan Wuchty (RNAsubopt)
- Ivo Hofacker, Peter Stadler, and Christoph Flamm (almost every implementation up to version 1.8.5)

We also want to thank the following people:

- Sebastian Bonhoeffer's implementation of partition function folding served as a precursor to our part\_func.c
- Manfred Tacker hacked constrained folding into fold.c for the first time
- Martin Fekete made the first attempts at "alignment folding"
- Andrea Tanzer and Martin Raden (Mann) for not stopping to report bugs found through comprehensive usage of our applications and RNAlib
- Thanks also to everyone else who helped testing and finding bugs, especially Christoph Flamm, Martijn Huynen, Baerbel Krakhofer, and many more

If you want to get involved in the development of the ViennaRNA Package yourself, please read the [Contributing page](#).

# Chapter 2

## Getting Started

- [Installation and Configuration](#) describes how to install and configure RNAlib for your requirements
- [HelloWorld](#) presents some small example programs to get a first impression on how to use this library
- [HelloWorld \(Perl/Python\)](#) contains small examples that show how to use RNAlib even without C/C++ programming skills from within your favorite scripting language

### 2.1 Installation and Configuration

A documentation on how to configure the different features of RNAlib, how to install the ViennaRNA Package, and finally, how to link your own programs against RNAlib.

#### 2.1.1 Installing the ViennaRNA Package

For best portability the ViennaRNA package uses the GNU autoconf and automake tools. The instructions below are for installing the ViennaRNA package from source. However, pre-compiled binaries for various Linux distributions, as well as for Windows users are available from Download section of the [main ViennaRNA homepage](#).

##### 2.1.1.1 Quick-start

Usually you'll just unpack, configure and make. To do this type:

```
tar -zxvf ViennaRNA-2.5.1.tar.gz
cd ViennaRNA-2.5.1
./configure
make
sudo make install
```

##### 2.1.1.2 Installation without root privileges

If you do not have root privileges on your computer, you might want to install the ViennaRNA Package to a location where you actually have write access to. To do so, you can set the installation prefix of the ./configure script like so:

```
./configure --prefix=/home/username/ViennaRNA
make install
```

This will install the entire ViennaRNA Package into a new directory ViennaRNA directly into the users username home directory.

### 2.1.1.3 Notes for MacOS X users

**2.1.1.3.1 Compilation** Although users will find /usr/bin/gcc and /usr/bin/g++ executables in their directory tree, these programs are not at all what they pretend to be. Instead of including the GNU programs, Apple decided to install clang/llvm in disguise. Unfortunately, the default version of clang/llvm does not support OpenMP (yet), but only complains at a late stage of the build process when this support is required. Therefore, it seems necessary to deactivate OpenMP support by passing the option `--disable-openmp` to the `./configure` script.

**2.1.1.3.2 Missing EXTERN.h include file** Furthermore, as far as we are informed, users are discouraged to use the Perl 5 interpreter that is shipped with Mac OS X. Instead, one should install a more recent version from another source, e.g. homebrew. If, however, for any reason you do not want to install your own Perl 5 interpreter but use the one from Apple, you need to specify its include path to enable building the ViennaRNA Perl interface. Otherwise, the file `EXTERN.h` will be missing at compile time. To fix this problem, you first need to find out where `EXTERN.h` is located:

```
sudo find /Library -type f -name EXTERN.h
```

Then choose the one that corresponds to your default perl interpreter (find out the version number with `perl -v | grep version`), simply execute the following before running the `./configure` script, e.g.:

```
export CPATH=/Library/Developer/CommandLineTools/SDKs/MacOSX10.15.sdk/System/Library/Perl/5.18/darwin-thread-m
```

if your default perl is v5.18 running on MacOSX10.15. Change the paths according to your current setup. After that, running `./configure` and compilation should run fine.

See also <https://stackoverflow.com/questions/52682304/fatal-error-extern-h-file-not-found>

**2.1.1.3.3 Universal binaries** Additionally, if you intend to build the ViennaRNA such that it runs on both, x86\_64 and the armv8 (such as for the M1 processors in recent MacBooks), architectures, you need to build a so-called universal binary. Note, however, that to accomplish this task, you might need to deactivate any third-party library dependency as in most cases, only one architecture will be available at link time. This includes the Perl 5 and Python interfaces but also MPFR and GSL support, possibly even more. In order to compile and link the programs, library, and scripting language interfaces of the ViennaRNA Package for multiple architectures, we've added a new configure switch that sets up the required changes automatically:

```
./configure --enable-universal-binary
```

#### Note

Note, that with link time optimization turned on, MacOS X's default compiler (llvm/clang) generates an intermediary binary format that can not easily be combined into a multi-architecture library. Therefore, the `--enable-universal-binary` switch turns off link time optimization!

## 2.1.2 Configuring RNAlib features

The ViennaRNA Package includes additional executable programs such as RNAforester, Kinfold, and Kinwalker. Furthermore, we include several features in our C-library that may be activated by default, or have to be explicitly turned on at configure-time. Below we list a selection of the available configure options that affect the features included in all executable programs, the RNAlib C-library, and the corresponding scripting language interface(s).

### 2.1.2.1 Streaming SIMD Extension (SSE) support

Since version 2.3.5 our sources contain code that implements a faster multibranch loop decomposition in global MFE predictions, as used e.g. in RNAfold. This implementation makes use of modern processors capability to execute particular instructions on multiple data simultaneously (SIMD - single instruction multiple data, thanks to W. B. Langdon for providing the modified code). Consequently, the time required to assess the minimum of all multibranch loop decompositions is reduced up to about one half compared to the runtime of the original implementation. This feature is enabled by default since version 2.4.11 and a dispatcher ensures that the correct implementation will be selected at runtime. If for any reason you want to disable this feature at compile-time use the following configure flag:

```
./configure --disable-simd
```

### 2.1.2.2 Scripting Interfaces

The ViennaRNA Package comes with scripting language interfaces for Perl 5, Python 3.x, and Python 2.x (provided by swig), that allow one to use the implemented algorithms directly without the need of calling an executable program. The interfaces are build by default whenever the autoconf tool-chain detects the required build tools on your system. You may, however, explicitly turn off particular scripting language interface support at configure-time, for instance for Perl 5 and Python 2, before the actual installation.

Example:

```
./configure --without-perl --without-python2
```

Disabling the scripting language support all-together can be accomplished using the following switch:

```
./configure --without-swig
```

### 2.1.2.3 Cluster Analysis

The programs AnalyseSeqs and AnalyseDists offer some cluster analysis tools (split decomposition, statistical geometry, neighbor joining, Ward's method) for sequences and distance data. To also build these programs add

```
--with-cluster
```

to your configure options.

### 2.1.2.4 Kinfold

The Kinfold program can be used to simulate the folding dynamics of an RNA molecule, and is compiled by default. Use the

```
--without-kinfold
```

option to skip compilation and installation of Kinfold.

### 2.1.2.5 RNAforester

The RNAforester program is used for comparing secondary structures using tree alignment. Similar to Kinfold, use the

```
--without-forester
```

option to skip compilation and installation of RNAforester.

### 2.1.2.6 Kinwalker

The Kinwalker algorithm performs co-transcriptional folding of RNAs, starting at a user specified structure (default: open chain) and ending at the minimum free energy structure. Compilation and installation of this program is deactivated by default. Use the

```
--with-kinwalker
```

option to enable building and installation of Kinwalker.

### 2.1.2.7 Link Time Optimization (LTO)

To increase the performance of our implementations, the ViennaRNA Package tries to make use of the Link Time Optimization (LTO) feature of modern C-compilers. If you are experiencing any troubles at make-time or run-time, or the configure script for some reason detects that your compiler supports this feature although it doesn't, you can deactivate it using the flag

```
./configure --disable-lto
```

Note, that GCC before version 5 is known to produce unreliable LTO code, especially in combination with SIMD (see [Streaming SIMD Extension \(SSE\) support](#)). We therefore recommend using a more recent compiler (GCC 5 or above) or to turn off one of the two features, LTO or SIMD optimized code.

### 2.1.2.8 OpenMP support

To enable concurrent computation of our implementations and in some cases parallelization of the algorithms we make use of the OpenMP API. This interface is well understood by most modern compilers. However, in some cases it might be necessary to deactivate OpenMP support and therefore transform *RNAlib* into a C-library that is not entirely *thread-safe*. To do so, add the following configure option

```
./configure --disable-openmp
```

### 2.1.2.9 POSIX threads (pthread) support

To enable concurrent computation of multiple input data in RNAfold, and for our implementation of the concurrent unordered insert, ordered output flush data structure *vrna\_ostream\_t* we make use of POSIX threads. This should be supported on all modern platforms and usually does not pose any problems. Unfortunately, we use a threadpool implementation that is not compatible with Microsoft Windows yet. Thus, POSIX thread support can not be activated for Windows builds until we have fixed this problem. If you want to compile RNAfold and RNAlib without POSIX threads support for any other reasons, add the following configure option

```
./configure --disable-pthreads
```

### 2.1.2.10 SVM Z-score filter in RNALfold

By default, RNALfold that comes with the ViennaRNA Package allows for z-score filtering of its predicted results using a support vector machine (SVM). However, the library we use to implement this feature (libsvm) is statically linked to our own RNAlib. If this introduces any problems for your own third-party programs that link against RNAlib, you can safely switch off the z-scoring implementation using

```
./configure --without-svm
```

### 2.1.2.11 GNU Scientific Library

The new program RNApmin computes a pseudo-energy perturbation vector that aims to minimize the discrepancy of predicted, and observed pairing probabilities. For that purpose it implements several methods to solve the optimization problem. Many of them are provided by the GNU Scientific Library, which is why the RNApmin program, and the RNAlib C-library are required to be linked against libgsl. If this introduces any problems in your own third-party programs that link against RNAlib, you can turn off a larger portion of available minimizers in RNApmin and linking against libgsl all-together, using the switch

```
./configure --without-gsl
```

### 2.1.2.12 Disable C11/C++11 feature support

By default, we use C11/C++11 features in our implementations. This mainly accounts for unnamed unions/structs within *RNAlib*. The configure script automatically detects whether or not your compiler understands these features. In case you are using an older compiler, these features will be deactivated by setting a specific pre-processor directive. If for some reason you want to deactivate C11/C++11 features despite the capabilities of your compiler, use the following configure option:

```
./configure --disable-c11
```

### 2.1.2.13 Enable warnings for use of deprecated symbols

Since version 2.2 we are in the process of transforming the API of our *RNAlib*. Hence, several symbols are marked as *deprecated* whenever they have been replaced by the new API. By default, deprecation warnings at compile time are deactivated. If you want to get your terminal spammed by tons of deprecation warnings, enable them using:

```
./configure --enable-warn-deprecated
```

### 2.1.2.14 Single precision partition function

Calculation of partition functions (via RNAfold -p) uses double precision floats by default, to avoid overflow errors on longer sequences. If your machine has little memory and you don't plan to fold sequences over 1000 bases in length you can compile the package to do the computations in single precision by running

```
./configure --enable-floatpf
```

#### Note

Using this option is discouraged and not necessary on most modern computers.

### 2.1.2.15 Help

For a complete list of all ./configure options and important environment variables, type

```
./configure --help
```

For more general information on the build process see the INSTALL file.

## 2.1.3 Linking against RNAlib

In order to use our implemented algorithms you simply need to link your program to our *RNAlib* C-library that usually comes along with the ViennaRNA Package installation. If you've installed the ViennaRNA Package as a pre-build binary package, you probably need the corresponding development package, e.g. *viennarna-devel*, or *viennarna-dev*. The only thing that is left is to include the ViennaRNA header files into your source code, e.g.:

```
#include <ViennaRNA/mfe.h>
```

and start using our fast and efficient algorithm implementations.

### See also

In the [C Examples](#) and [Some Examples using RNAlib API v3.0](#) sections, we list a small set of example code that usually is a good starting point for your application.

### 2.1.3.1 Compiler and Linker flags

Of course, simply adding the ViennaRNA header files into your source code is usually not enough. You probably need to tell your compiler where to find the header files, and sometimes add additional pre-processor directives. Whenever your installation of *RNAlib* was build with default settings and the header files were installed into their default location, a simple

```
-I/usr/include
```

pre-processor/compile flag should suffice. It can even be omitted in this case, since your compiler should search this directory by default anyway. You only need to change the path from */usr/include* to the correct location whenever the header files have been installed into a non-standard directory.

On the other hand, if you've compiled *RNAlib* with some non-default settings then you probably need to define some additional pre-processor macros:

- *VRNA\_DISABLE\_C11\_FEATURES* . . . Disable C11/C++11 features.

#### Warning

Add this directive to your pre-processor/compile flags only if *RNAlib* was build with the *--disable-c11* configure option.

**See also**

[Disable C11/C++11 feature support](#) and [vrna\\_C11\\_features\(\)](#)

- **VRNA\_WARN\_DEPRECATED** ... Enable warnings for using deprecated symbols.

**Note**

Adding this directive enables compiler warnings whenever you use symbols in *RNALib* that are marked *deprecated*.

**See also**

[Enable warnings for use of deprecated symbols](#) and [Deprecated List](#)

- **USE\_FLOAT\_PF** ... Use single precision floating point operations instead of double precision in partition function computations.

**Warning**

Define this macro only if *RNALib* was build with the `--enable-floatpf` configure option!

**See also**

[Single precision partition function](#)

Simply add the corresponding definition(s) to your pre-processor/compile flags, for instance:

`-DVRNA_DISABLE_C11_FEATURES`

Finally, linking against *RNALib* is achieved by adding the following linker flag

`-L/usr/lib -lRNA -fopenmp`

Again, the path to the library, `/usr/lib`, may be omitted if this path is searched for libraries by default. The second flag tells the linker to include `libRNA.a`, and the remaining two flags activate [Link Time Optimization \(LTO\)](#) and [OpenMP support](#) support, respectively.

**Note**

Depending on your linker, the last two flags may differ.

Depending on your configure time decisions, you can drop one or both of the last flags.

In case you've compiled *RNALib* with LTO support (See [Link Time Optimization \(LTO\)](#)) and you are using the same compiler for your third-party project that links against our library, you may add the `-fleo`

flag to enable Link Time Optimization.

### 2.1.3.2 The `pkg-config` tool

Instead of hard-coding the required compiler and linker flags, you can also let the `pkg-config` tool automatically determine the required flags. This tool is usually packaged for any Linux distribution and should be available for MacOS X and MinGW as well. We ship a file `RNALib2.pc` which is installed along with the static `libRNA.a` C-library and populated with all required compiler and linker flags that correspond to your configure time decisions.

The compiler flags required for properly building your code that uses *RNALib* can be easily obtained via

```
pkg-config --cflags RNALib2
```

You get the corresponding linker flags using

```
pkg-config --libs RNALib2
```

With this widely accepted standard it is also very easy to integrate *RNALib* in your *autotools* project, just have a look at the `PKG_CHECK_MODULES` macro.

## 2.2 HelloWorld

Below, you'll find some more or less simple C programs showing first steps into using *RNAlib*. A complete list of example C programs can be found in the [C Examples](#) section.

### Simple MFE prediction for a given sequence

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ViennaRNA/fold.h>
#include <ViennaRNA/utils/basic.h>
int
main()
{
    /* The RNA sequence */
    char *seq = "GAGUAGUGGAACCAGGCCUAUGUUUGUGACUCGCAGACUAACA";
    /* allocate memory for MFE structure (length + 1) */
    char *structure = (char *)vrna_alloc(sizeof(char) * (strlen(seq) + 1));
    /* predict Minimum Free Energy and corresponding secondary structure */
    float mfe = vrna_fold(seq, structure);
    /* print sequence, structure and MFE */
    printf("%s\n%s [ %6.2f ]\n", seq, structure, mfe);
    /* cleanup memory */
    free(structure);
    return 0;
}
```

#### See also

[examples/helloworld\\_mfe.c](#) in the source code tarball

### Simple MFE prediction for a multiple sequence alignment

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ViennaRNA/alifold.h>
#include <ViennaRNA/utils/basic.h>
#include <ViennaRNA/utils/alignments.h>
int
main()
{
    /* The RNA sequence alignment */
    const char *sequences[] = {
        "CUGCCUCACAACGUUUUGGCCUCAGUUACCCGUAGAUGUAGUGAGGGU",
        "CUGCCUCACAACAUUUGGCCUCAGUUACAUAGAUGUAGUGAGGGU",
        "---CUCGACACCAACU---GCCUCGGUUACCCAUCGGUGCAGUGCAGGGU",
        NULL /* indicates end of alignment */
    };
    /* compute the consensus sequence */
    char *cons = consensus(sequences);
    /* allocate memory for MFE consensus structure (length + 1) */
    char *structure = (char *)vrna_alloc(sizeof(char) * (strlen(sequences[0]) + 1));
    /* predict Minimum Free Energy and corresponding secondary structure */
    float mfe = vrna_alifold(sequences, structure);
    /* print consensus sequence, structure and MFE */
    printf("%s\n%s [ %6.2f ]\n", cons, structure, mfe);
    /* cleanup memory */
    free(cons);
    free(structure);
    return 0;
}
```

#### See also

[examples/helloworld\\_mfe\\_comparative.c](#) in the source code tarball

## Simple Base Pair Probability computation

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ViennaRNA/fold.h>
#include <ViennaRNA/part_func.h>
#include <ViennaRNA/utils/basic.h>
int
main()
{
    /* The RNA sequence */
    char     *seq = "GAGUAGUGGAACCAGGCUAUGUUUGUGACUCGCAGACUAACA";
    /* allocate memory for pairing propensity string (length + 1) */
    char     *propensity = (char *)vrna_alloc(sizeof(char) * (strlen(seq) + 1));
    /* pointers for storing and navigating through base pair probabilities */
    vrna_ep_t *ptr, *pair_probabilities = NULL;
    float    en = vrna_pf_fold(seq, propensity, &pair_probabilities);
    /* print sequence, pairing propensity string and ensemble free energy */
    printf("%s\n%s [ %6.2f ]\n", seq, propensity, en);
    /* print all base pairs with probability above 50% */
    for (ptr = pair_probabilities; ptr->i != 0; ptr++)
        if (ptr->p > 0.5)
            printf("p(%d, %d) = %g\n", ptr->i, ptr->j, ptr->p);
    /* cleanup memory */
    free(pair_probabilities);
    free(propensity);
    return 0;
}
```

### See also

[examples/helloworld\\_probabilities.c](#) in the source code tarball

## Deviating from the Default Model

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ViennaRNA/model.h>
#include <ViennaRNA/fold_compound.h>
#include <ViennaRNA/utils/basic.h>
#include <ViennaRNA/utils/strings.h>
#include <ViennaRNA/mfe.h>
int
main()
{
    /* initialize random number generator */
    vrna_init_rand();
    /* Generate a random sequence of 50 nucleotides */
    char     *seq = vrna_random_string(50, "ACGU");
    /* allocate memory for MFE structure (length + 1) */
    char     *structure = (char *)vrna_alloc(sizeof(char) * (strlen(seq) + 1));
    /* create a new model details structure to store the Model Settings */
    vrna_md_t md;
    /* ALWAYS set default model settings first! */
    vrna_md_set_default(&md);
    /* change temperature and activate G-Quadruplex prediction */
    md.temperature = 25.0; /* 25 Deg Celcius */
    md.gquad      = 1;    /* Turn-on G-Quadruples support */
    /* create a fold compound */
    vrna_fold_compound_t *fc = vrna_fold_compound(seq, &md, VRNA_OPTION_DEFAULT);
    /* predict Minimum Free Energy and corresponding secondary structure */
    float    mfe = vrna_mfe(fc, structure);
    /* print sequence, structure and MFE */
    printf("%s\n%s [ %6.2f ]\n", seq, structure, mfe);
    /* cleanup memory */
    free(structure);
    vrna_fold_compound_free(fc);
    return 0;
}
```

### See also

[examples/fold\\_compound\\_md.c](#) in the source code tarball

## 2.3 HelloWorld (Perl/Python)

### 2.3.1 Perl5

#### Simple MFE prediction for a given sequence

```
use RNA;
# The RNA sequence
my $seq = "GAGUAGUGGAACCAGGCUAUGUUUGUGACUCGCAGACUAACA";
# compute minimum free energy (MFE) and corresponding structure
my ($ss, $mfe) = RNA::fold($seq);
# print output
printf "%s\n%s [ %6.2f ]\n", $seq, $ss, $mfe;
```

#### Simple MFE prediction for a multiple sequence alignment

```
use RNA;
# The RNA sequence alignment
my @sequences = (
    "CUGCCUCACAACGUUUGUGCCUCAGGUUACCCGUAGAUGUAGUGAGGGU",
    "CUGCCUCACAACAUUUGUGCCUCAGGUUACUCAUAGAUGUAGUGAGGGU",
    "---CUCGACACCACU---GCCUCGGUUAACCAUCGGUGCAGUGCGGGU"
);
# compute the consensus sequence
my $cons = RNA::consensus(\@sequences);
# predict Minimum Free Energy and corresponding secondary structure
my ($ss, $mfe) = RNA::alifold(\@sequences);
# print output
printf "%s\n%s [ %6.2f ]\n", $cons, $ss, $mfe;
```

### Deviating from the Default Model

```
use RNA;
# The RNA sequence
my $seq = "GAGUAGUGGAACCAGGCUAUGUUUGUGACUCGCAGACUAACA";
# create a new model details structure
my $md = new RNA::md();
# change temperature and dangle model
$md->{temperature} = 20.0; # 20 Deg Celcius
$md->{dangles} = 1; # Dangle Model 1
# create a fold compound
my $fc = new RNA::fold_compound($seq, $md);
# predict Minimum Free Energy and corresponding secondary structure
my ($ss, $mfe) = $fc->mfe();
# print sequence, structure and MFE
printf "%s\n%s [ %6.2f ]\n", $seq, $ss, $mfe;
```

### 2.3.2 Python

#### Simple MFE prediction for a given sequence

```
import RNA
# The RNA sequence
seq = "GAGUAGUGGAACCAGGCUAUGUUUGUGACUCGCAGACUAACA"
# compute minimum free energy (MFE) and corresponding structure
(ss, mfe) = RNA.fold(seq)
# print output
print("{}\n{} [ {:.2f} ]".format(seq, ss, mfe))
```

#### Simple MFE prediction for a multiple sequence alignment

```
import RNA
# The RNA sequence alignment
sequences = [
    "CUGCCUCACAACGUUUGUGCCUCAGGUUACCCGUAGAUGUAGUGAGGGU",
    "CUGCCUCACAACAUUUGUGCCUCAGGUUACUCAUAGAUGUAGUGAGGGU",
    "---CUCGACACCACU---GCCUCGGUUAACCAUCGGUGCAGUGCGGGU"
]
# compute the consensus sequence
cons = RNA.consensus(sequences)
# predict Minimum Free Energy and corresponding secondary structure
(ss, mfe) = RNA.alifold(sequences);
# print output
print("{}\n{} [ {:.2f} ]".format(cons, ss, mfe))
```

## Deviating from the Default Model

```
import RNA
# The RNA sequence
seq = "GAGUAGUGGAACCAGGCCAUGUUUGUGACUCGCAGACUAACA"
# create a new model details structure
md = RNA.md()
# change temperature and dangle model
md.temperature = 20.0 # 20 Deg Celcius
md.dangles = 1 # Dangle Model 1
# create a fold compound
fc = RNA.fold_compound(seq, md)
# predict Minimum Free Energy and corresponding secondary structure
(ss, mfe) = fc.mfe()
# print sequence, structure and MFE
print("{}\n{} [ {:.2f} ]".format(seq, ss, mfe))
```



## Chapter 3

# Concepts and Algorithms

This is an overview of the concepts and algorithms for which implementations can be found in this library.

Almost all of them rely on the physics based Nearest Neighbor Model for RNA secondary structure prediction.

- [RNA Structure](#) gives an introduction into the different layers of abstraction for RNA structures
- [Distance Measures](#) introduces different metrics to allow for the comparison of secondary structures
- [Free Energy of Secondary Structures](#) shows how the stability of a secondary structure can be quantified in terms of free energy
- [Secondary Structure Folding Grammar](#) explains the basic recursive decomposition scheme that is applied in secondary structure prediction
- [RNA Secondary Structure Landscapes](#) describes how transition paths between secondary structures span a landscape like graph
- [Minimum Free Energy Algorithm\(s\)](#) compute the most stable conformation in thermodynamic equilibrium
- [Partition Function and Equilibrium Probability Algorithm\(s\)](#) enable one to apply statistical mechanics to derive equilibrium probabilities of structure features
- [Suboptimals and \(other\) Representative Structures](#) allow for alternative description and enumeration of the structure ensemble
- [RNA-RNA Interaction](#) introduces how to model the interaction between RNA molecules
- [Locally Stable Secondary Structures](#) offer insights into structuredness of long sequences and entire genomes
- [Comparative Structure Prediction](#) augment structure prediction with evolutionary conservation of homologous sequences
- [Classified DP variations](#) perform an *a priori* partitioning of the structure ensemble and compute various properties for the resulting classes.
- [RNA Sequence Design](#) constitutes the inverse problem of structure prediction
- [Experimental Structure Probing Data](#) can be used to guide structure prediction, for instance using SHAPE reactivity data
- [Ligand Binding](#) adds more complexity to structure prediction by modelling the interaction between small chemical compounds or proteins and the RNA
- [\(Tertiary\) Structure Motifs](#) extend the abstraction of secondary structure beyond canonical base pair formation

## 3.1 RNA Structure

### 3.1.1 RNA Structures

### 3.1.2 Levels of Structure Abstraction

#### 3.1.2.1 Primary Structure

#### 3.1.2.2 Secondary Structure

#### 3.1.2.3 Tertiary Structure

#### 3.1.2.4 Quarternary Structure

#### 3.1.2.5 Pseudo-Knots

## 3.2 Distance Measures

A simple measure of dissimilarity between secondary structures of equal length is the base pair distance, given by the number of pairs present in only one of the two structures being compared. I.e. the number of base pairs that have to be opened or closed to transform one structure into the other. It is therefore particularly useful for comparing structures on the same sequence. It is implemented by

```
int bp_distance(const char *str1,
                const char *str2)
```

Compute the "base pair" distance between two secondary structures s1 and s2.

For other cases a distance measure that allows for gaps is preferable. We can define distances between structures as edit distances between trees or their string representations. In the case of string distances this is the same as "sequence alignment". Given a set of edit operations and edit costs, the edit distance is given by the minimum sum of the costs along an edit path converting one object into the other. Edit distances like these always define a metric. The edit operations used by us are insertion, deletion and replacement of nodes. String editing does not pay attention to the matching of brackets, while in tree editing matching brackets represent a single node of the tree. Tree editing is therefore usually preferable, although somewhat slower. String edit distances are always smaller or equal to tree edit distances.

The different level of detail in the structure representations defined above naturally leads to different measures of distance. For full structures we use a cost of 1 for deletion or insertion of an unpaired base and 2 for a base pair. Replacing an unpaired base for a pair incurs a cost of 1.

Two cost matrices are provided for coarse grained structures:

```

/* Null, H, B, I, M, S, E */
{ 0, 2, 2, 2, 1, 1}, /* Null replaced */
{ 2, 0, 2, 2, INF, INF}, /* H replaced */
{ 2, 2, 0, 1, 2, INF, INF}, /* B replaced */
{ 2, 2, 1, 0, 2, INF, INF}, /* I replaced */
{ 2, 2, 2, 2, 0, INF, INF}, /* M replaced */
{ 1, INF, INF, INF, 0, INF}, /* S replaced */
{ 1, INF, INF, INF, INF, 0}, /* E replaced */

```

```

/* Null, H, B, I, M, S, E */
{ 0, 100, 5, 5, 75, 5, 5}, /* Null replaced */
{ 100, 0, 8, 8, 8, INF, INF}, /* H replaced */
{ 5, 8, 0, 3, 8, INF, INF}, /* B replaced */
{ 5, 8, 3, 0, 8, INF, INF}, /* I replaced */
{ 75, 8, 8, 8, 0, INF, INF}, /* M replaced */
{ 5, INF, INF, INF, INF, 0, INF}, /* S replaced */
{ 5, INF, INF, INF, INF, INF, 0}, /* E replaced */

```

The lower matrix uses the costs given in [22]. All distance functions use the following global variables:

```
int cost_matrix;
```

Specify the cost matrix to be used for distance calculations.

```
int edit_backtrack;
```

Produce an alignment of the two structures being compared by tracing the editing path giving the minimum distance.

```
char *aligned_line[4];
```

Contains the two aligned structures after a call to one of the distance functions with `edit_backtrack` set to 1.

#### See also

[utils.h](#), [dist\\_vars.h](#) and [stringdist.h](#) for more details

### 3.2.1 Functions for Tree Edit Distances

```
Tree *make_tree (char *struc)
```

Constructs a `Tree` (essentially the postorder list) of the structure 'struc', for use in [tree\\_edit\\_distance\(\)](#).

```
float tree_edit_distance (Tree *T1,
                         Tree *T2)
```

Calculates the edit distance of the two trees.

```
void free_tree(Tree *t)
```

Free the memory allocated for `Tree` t.

#### See also

[dist\\_vars.h](#) and [treedist.h](#) for prototypes and more detailed descriptions

### 3.2.2 Functions for String Alignment

```
swString *Make_swString (char *string)
```

Convert a structure into a format suitable for [string\\_edit\\_distance\(\)](#).

```
float string_edit_distance (swString *T1,
                           swString *T2)
```

Calculate the string edit distance of T1 and T2.

**See also**

[dist\\_vars.h](#) and [stringdist.h](#) for prototypes and more detailed descriptions

### 3.2.3 Functions for Comparison of Base Pair Probabilities

For comparison of base pair probability matrices, the matrices are first condensed into probability profiles which are then compared by alignment.

```
float *Make_bp_profile_bppm ( double *bppm,
                             int length)
```

condense pair probability matrix into a vector containing probabilities for unpaired, upstream paired and downstream paired.

```
float profile_edit_distance ( const float *T1,
                             const float *T2)
```

Align the 2 probability profiles T1, T2

**See also**

[ProfileDist.h](#) for prototypes and more details of the above functions

## 3.3 Free Energy of Secondary Structures

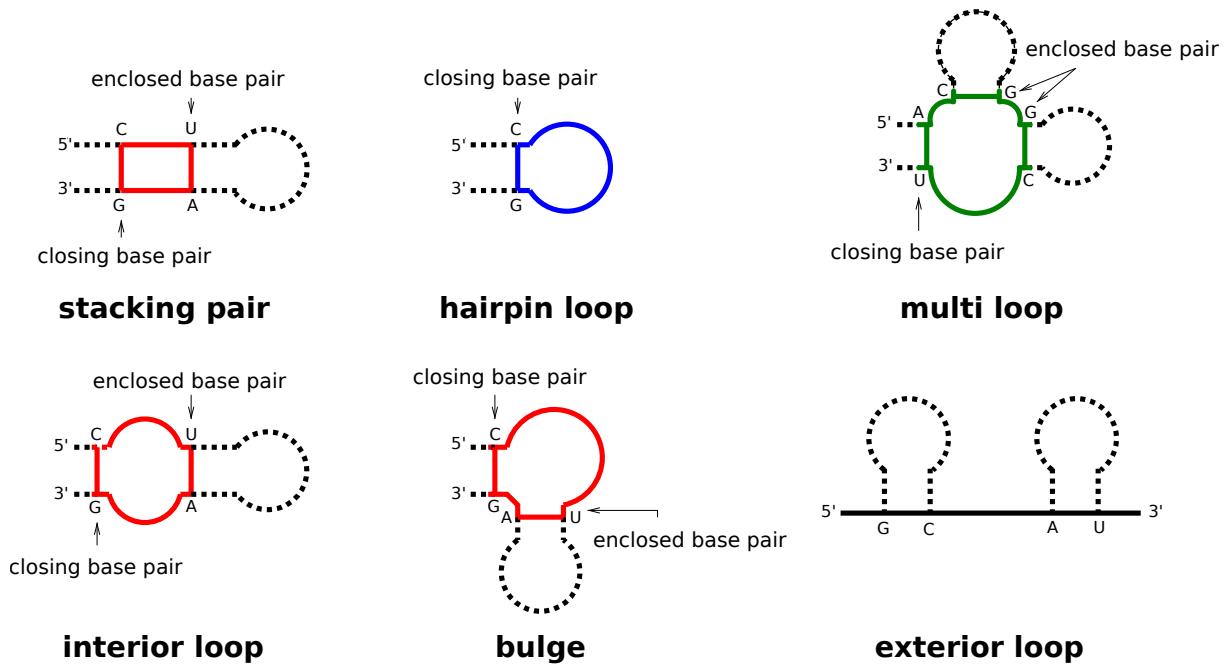
A description on how secondary structures are decomposed into individual loops to eventually evaluate their stability in terms of free energy.

### 3.3.1 Secondary Structure Loop Decomposition

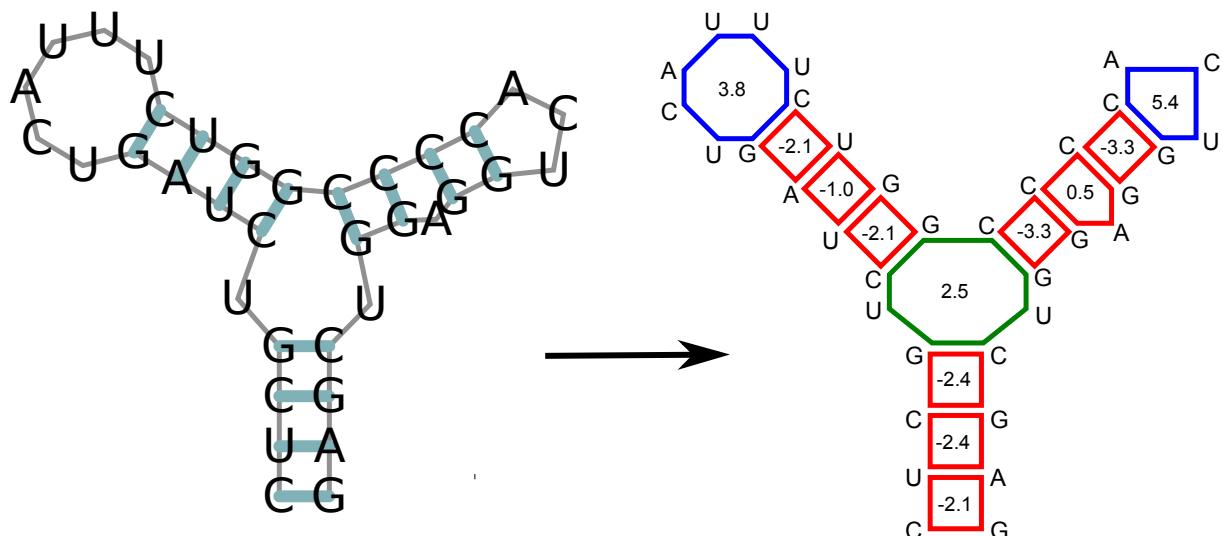
Each base pair in a secondary structure closes a loop, thereby directly enclosing unpaired nucleotides, and/or further base pairs. Our implementation distinguishes four basic types of loops:

- hairpin loops
- interior loops
- multibranch loops
- exterior loop

While the exterior loop is a special case without a closing pair, the other loops are determined by the number of base pairs involved in the loop formation, i.e. hairpin loops are 1-loops, since only a single base pair delimits the loop. interior loops are 2-loops due to their enclosing, and enclosed base pair. All loops where more than two base pairs are involved, are termed multibranch loops.



Any secondary structure can be decomposed into its loops. Each of the loops then can be scored in terms of free energy, and the free energy of an entire secondary structure is simply the sum of free energies of its loops.



### 3.3.1.1 Free Energy Evaluation API

While we implement some functions that decompose a secondary structure into its individual loops, the majority of methods provided in **RNALib** are dedicated to free energy evaluation. The corresponding modules are:

See also

[Free Energy Evaluation, Energy Evaluation for Individual Loops](#)

## 3.3.2 Free Energy Parameters

For secondary structure free energy evaluation we usually utilize the set of Nearest Neighbor Parameters also used in other software, such as *UNAFold* and *RNAstructure*. While the *RNALib* already contains a compiled-in set of the latest *Turner 2004 Free Energy Parameters*, we defined a file format that allows to change these parameters at runtime. The ViennaRNA Package already comes with a set of parameter files containing

- Turner 1999 RNA parameters
- Mathews 1999 DNA parameters
- Andronescu 2007 RNA parameters
- Mathews 2004 DNA parameters

### 3.3.2.1 Free Energy Parameters Modification API

See also

[Energy Parameters, Reading/Writing Energy Parameter Sets from/to File](#)

## 3.3.3 Fine-tuning of the Energy Evaluation Model

See also

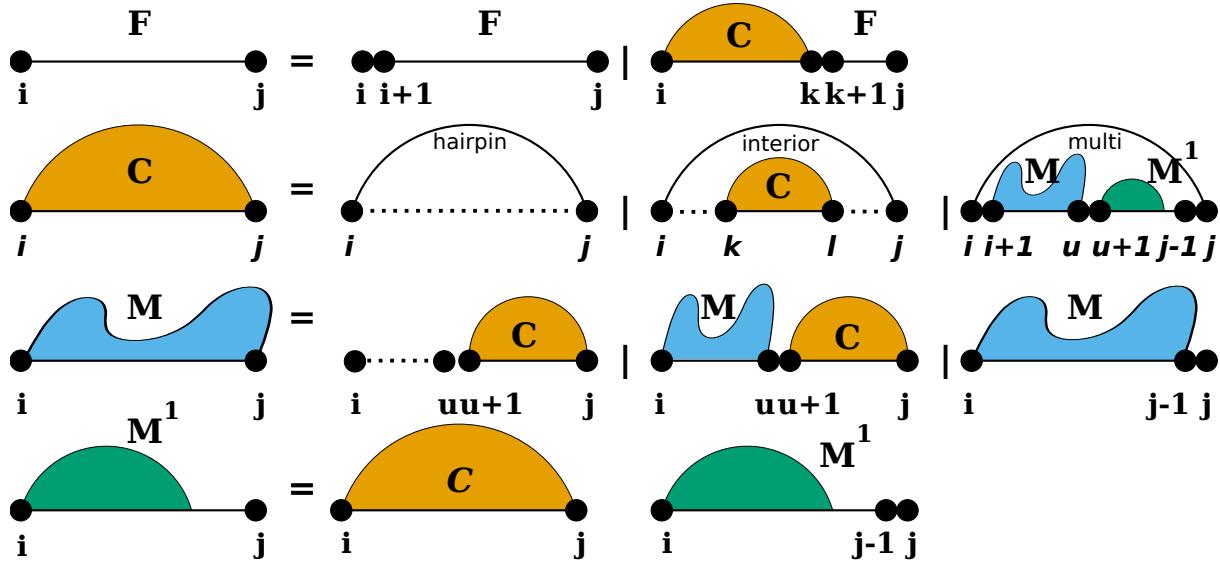
[Fine-tuning of the Implemented Models](#)

## 3.4 Secondary Structure Folding Grammar

A description of the basic grammar to generate secondary structures, used for almost all prediction algorithms in our library and how to modify it.

### 3.4.1 Secondary Structure Folding Recurrences

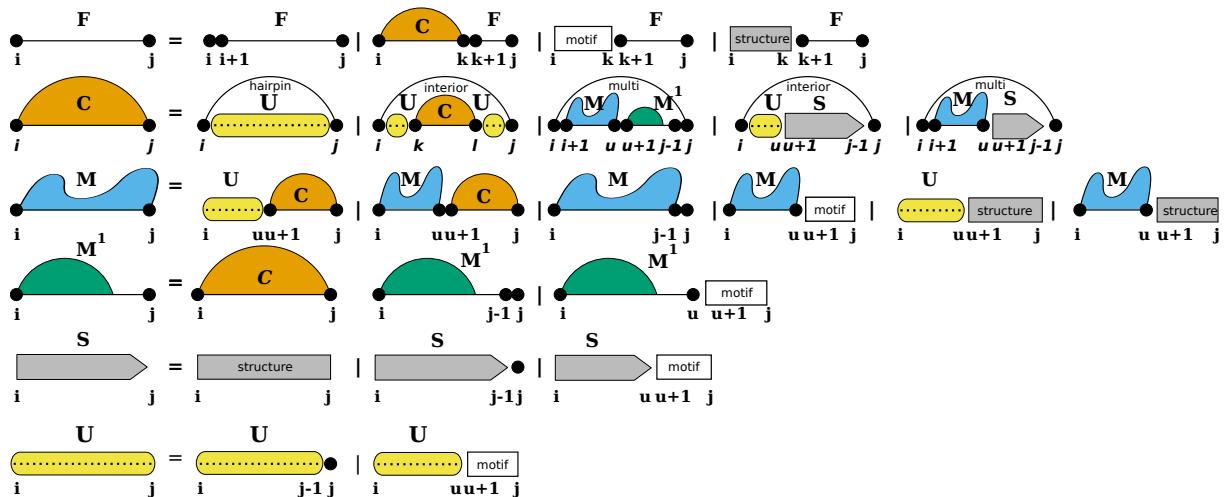
To predict secondary structures composed of the four distinguished loop types introduced before, all algorithms implemented in *RNAlib* follow a specific decomposition scheme, also known as the *RNA folding grammar*, or *Secondary Structure Folding Recurrences*.



However, compared to other RNA secondary structure prediction libraries, our implementation allows for a fine-grained control of the above recursions by constraining both, the individual derivations of the grammar as well as the evaluation of particular loop contributions. Furthermore, we provide a mechanism to extend the above grammar with additional derivation rules, so-called *Domains*.

### 3.4.2 Additional Structural Domains

Some applications of RNA secondary structure prediction require an extension of the *regular RNA folding grammar*. For instance one would like to include proteins and other ligands binding to unpaired loop regions while competing with conventional base pairing. Another application could be that one may want to include the formation of self-enclosed structural modules, such as *G-quadruplexes*. For such applications, we provide a pair of additional domains that extend the regular RNA folding grammar, [Structured Domains](#) and [Unstructured Domains](#).



While unstructured domains are usually determined by a more or less precise sequence motif, e.g. the binding site for a protein, structured domains are considered self-enclosed modules with a more or less complex pairing pattern. Our extension with these two domains introduces two production rules to fill additional dynamic processing matrices  $S$  and  $U$  where we store the pre-computed contributions of structured domains ( $S$ ), and unstructured domains ( $U$ ).

### 3.4.2.1 Structured Domains

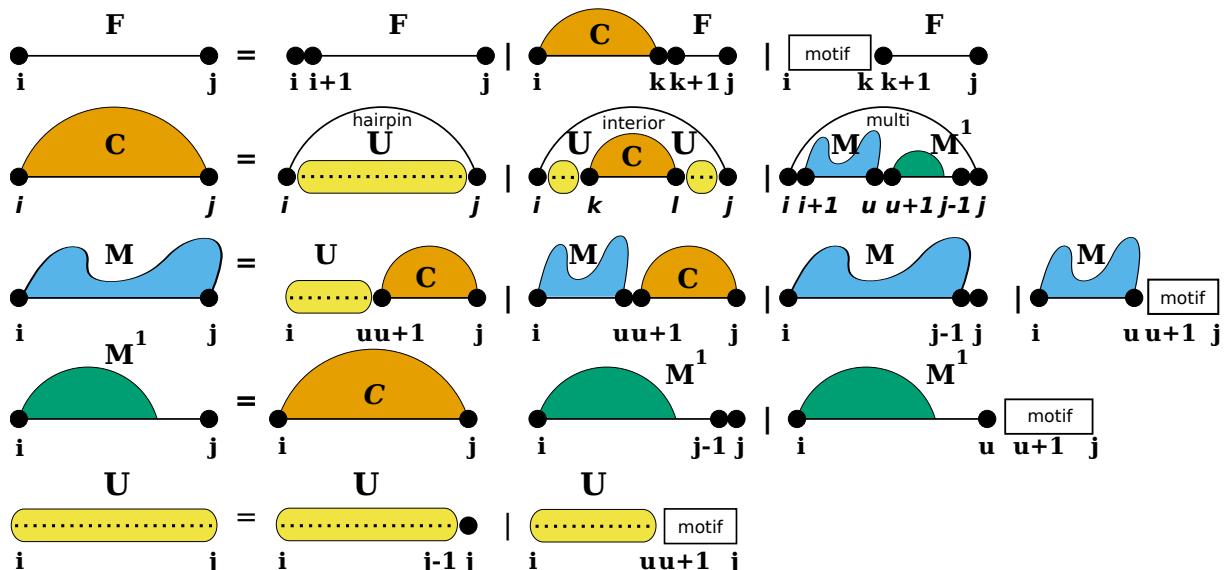
Usually, structured domains represent self-enclosed structural modules that exhibit a more or less complex base pairing pattern. This can be more or less well-defined 3D motifs, such as *G-Quadruplexes*, or loops with additional non-canonical base pair interactions, such as *kink-turns*.

#### Note

Currently, our implementation only provides the specialized case of *G-Quadruplexes*.

### 3.4.2.2 Unstructured Domains

Unstructured domains appear in the production rules of the RNA folding grammar wherever new unpaired nucleotides are attached to a growing substructure (see also [16]):



The white boxes represent the stretch of RNA bound to the ligand and represented by a more or less specific sequence motif. The motif itself is considered unable to form base pairs. The additional production rule  $U$  is used to precompute the contribution of unpaired stretches possibly bound by one or more ligands. The auxiliary DP matrix for this production rule is filled right before processing the other (regular) production rules of the RNA folding grammar.

### 3.4.2.3 Domain Extension API

For the sake of flexibility, each of the domains is associated with a specific data structure serving as an abstract interface to the extension. The interface uses callback functions to

- pre-compute arbitrary data, e.g. filling up additional dynamic programming matrices, and
- evaluate the contribution of a paired or unpaired structural feature of the RNA.

Implementations of these callbacks are separate for regular free energy evaluation, e.g. MFE prediction, and partition function applications. A data structure holding arbitrary data required for the callback functions can be associated to the domain as well. While *RNAlib* comes with a default implementation for structured and unstructured domains, the system is entirely user-customizable.

#### See also

[Unstructured Domains](#), [Structured Domains](#), [G-Quadruplexes](#), [Ligands Binding to Unstructured Domains](#)

### 3.4.3 Constraints on the Folding Grammar

Secondary Structure constraints can be subdivided into two groups:

- Hard Constraints
- Soft Constraints

While Hard-Constraints directly influence the production rules used in the folding recursions by allowing, disallowing, or enforcing certain decomposition steps, Soft-constraints on the other hand are used to change position specific contributions in the recursions by adding bonuses/penalties in form of pseudo free energies to certain loop configurations.

#### Note

Secondary structure constraints are always applied at decomposition level, i.e. in each step of the recursive structure decomposition, for instance during MFE prediction.

#### 3.4.3.1 Hard Constraints API

Hard constraints as implemented in our library can be specified for individual loop types, i.e. the atomic derivations of the RNA folding grammar rules. Hence, the pairing behavior of both, single nucleotides and pairs of bases, can be constrained in every loop context separately. Additionally, an abstract implementation using a callback mechanism allows for full control of more complex hard constraints.

#### See also

[Hard Constraints](#)

### 3.4.3.2 Soft Constraints API

For the sake of memory efficiency, we do not implement a loop context aware version of soft constraints. The *static* soft constraints as implemented only distinguish unpaired from paired nucleotides. This is usually sufficient for most use-case scenarios. However, similar to hard constraints, an abstract soft constraints implementation using a callback mechanism exists, that allows for any soft constraint that is compatible with the RNA folding grammar. Thus, loop contexts and even individual derivation rules can be addressed separately for maximum flexibility in soft-constraints application.

See also

[Soft Constraints, Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints, SHAPE Reactivity Data](#)

## 3.5 RNA Secondary Structure Landscapes

A description of the implicit landscape-like network of structures that appears upon modelling the transition of one structure into another.

### 3.5.1 The Neighborhood of a Secondary Structure

### 3.5.2 The Secondary Structure Landscape API

## 3.6 Minimum Free Energy Algorithm(s)

Computing the Minimum Free Energy (MFE), i.e. the most stable conformation in thermodynamic equilibrium.

### 3.6.1 Zuker's Algorithm

Our library provides fast dynamic programming Minimum Free Energy (MFE) folding algorithms derived from the decomposition scheme as described by "Zuker & Stiegler (1981)" [28].

### 3.6.2 MFE for circular RNAs

Folding of *circular* RNA sequences is handled as a post-processing step of the forward recursions. See [13] for further details.

### 3.6.3 MFE Algorithm API

We provide interfaces for the prediction of

- MFE and corresponding secondary structure for single sequences,
- consensus MFE structures of sequence alignments, and
- MFE structure for two hybridized RNA strands

See also

[Minimum Free Energy \(MFE\) Algorithms, RNA-RNA Interaction, Computing MFE representatives of a Distance Based Partition](#)

## 3.7 Partition Function and Equilibrium Probabilitiy Algorithm(s)

### 3.7.1 Equilibrium Ensemble Statistics

In contrast to methods that compute the property of a single structure in the ensemble, e.g. [Minimum Free Energy Algorithm\(s\)](#), the partition function algorithms always consider the entire equilibrium ensemble. For that purpose, the McCaskill algorithm [18] and its variants can be used to efficiently compute

- the partition function, and from that
- various equilibrium probabilities, for instance base pair probabilities, probabilities of individual structure motifs, and many more.

The principal idea behind this approach is that in equilibrium, statistical mechanics and polymer theory tells us that the frequency or probability  $p(s)$  of a particular state  $s$  depends on its energy  $E(s)$  and follows a Boltzmann distribution, i.e.

$$p(s) \propto e^{-\beta E(s)} \text{ with } \beta = \frac{1}{kT}$$

where  $k \approx 1.987 \cdot 10^{-3} \frac{\text{kcal}}{\text{mol K}}$  is the Boltzmann constant, and  $T$  the thermodynamic temperature. From that relation, the actual probability of state  $s$  can then be obtained using a proper scaling factor, the *canonical partition function*

$$Z = \sum_{s \in \Omega} e^{-\beta E(s)}$$

where  $\Omega$  is the finite set of all states. Finally, the equilibrium probability of state  $s$  can be computed as

$$p(s) = \frac{e^{-\beta E(s)}}{Z}$$

Instead of enumerating all states exhaustively to compute  $Z$  one can apply the [Secondary Structure Folding Recurrences](#) again for an efficient computation in cubic time. An *outside* variant of the same recursions is then used to compute probabilities for base pairs, stretches of consecutive unpaired nucleotides, or structural motifs.

#### See also

Further details of the Partition function and Base Pair Probability algorithm can be obtained from McCaskill 1990 [18]

### 3.7.2 Partition Function and Equilibrium Probability API

We implement a wide variety of variants of the partition function algorithm according to McCaskill 1990 [18]. See the corresponding submodules for specific implementation details.

#### See also

[Partition Function and Equilibrium Properties](#), [RNA-RNA Interaction](#), [Partition Function for two Hybridized Sequences as a Step Computing Partition Functions of a Distance Based Partitioning](#)

## 3.8 Suboptimals and (other) Representative Structures

### 3.8.1 Suboptimal Secondary Structures

### 3.8.2 Sampling Secondary Structures from the Ensemble

### 3.8.3 Structure Enumeration and Sampling API

See also

[Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989, Suboptimal Structures within an Energy Band around the MEA](#),  
[Random Structure Samples from the Ensemble, Compute the Structure with Maximum Expected Accuracy \(MEA\)](#),  
[Compute the Centroid Structure](#)

## 3.9 RNA-RNA Interaction

### 3.9.1 <br>

The function of an RNA molecule often depends on its interaction with other RNAs. The following routines therefore allows one to predict structures formed by two RNA molecules upon hybridization.

### 3.9.2 Concatenating RNA sequences

One approach to co-folding two RNAs consists of concatenating the two sequences and keeping track of the concatenation point in all energy evaluations. Correspondingly, many of the `cofold()` and `co_pf_fold()` routines take one sequence string as argument and use the global variable `cut_point` to mark the concatenation point. Note that while the *RNAcofold* program uses the '&' character to mark the chain break in its input, you should not use an '&' when using the library routines (set `cut_point` instead).

### 3.9.3 RNA-RNA interaction as a Stepwise Process

In a second approach to co-folding two RNAs, cofolding is seen as a stepwise process. In the first step the probability of an unpaired region is calculated and in a second step this probability of an unpaired region is multiplied with the probability of an interaction between the two RNAs. This approach is implemented for the interaction between a long target sequence and a short ligand RNA. Function `pf_unstru()` calculates the partition function over all unpaired regions in the input sequence. Function `pf_interact()`, which calculates the partition function over all possible interactions between two sequences, needs both sequence as separate strings as input.

### 3.9.4 RNA-RNA Interaction API

## 3.10 Locally Stable Secondary Structures

### 3.10.1 local\_intro

### 3.10.2 local\_mfe

### 3.10.3 local\_pf

### 3.10.4 Locally Stable Secondary Structure API

## 3.11 Comparative Structure Prediction

### 3.11.1 Incorporate Evolutionary Information

Consensus structures can be predicted by a modified version of the [fold\(\)](#) algorithm that takes a set of aligned sequences instead of a single sequence. The energy function consists of the mean energy averaged over the sequences, plus a covariance term that favors pairs with consistent and compensatory mutations and penalizes pairs that cannot be formed by all structures. For details see [11] and [1].

### 3.11.2 Comparative Structure Prediction API

## 3.12 Classified DP variations

### 3.12.1 The Idea of Classified Dynamic Programming

Usually, thermodynamic properties using the basic recursions for [Minimum Free Energy Algorithm\(s\)](#), [Partition Function](#) and [Equilibrium](#) and so forth, are computed over the entire structure space. However, sometimes it is desired to partition the structure space *a priori* and compute the above properties for each of the resulting partitions. This approach directly leads to *Classified Dynamic Programming*.

### 3.12.2 Distance Class Partitioning

The secondary structure space is divided into partitions according to the base pair distance to two given reference structures and all relevant properties are calculated for each of the resulting partitions.

#### See also

For further details, we refer to Lorenz et al. 2009 [15]

### 3.12.3 Density of States (DOS)

#### 3.12.4 Classified DP API

## 3.13 RNA Sequence Design

### 3.13.1 Generate Sequences that fold into particular Secondary Structures

### 3.13.2 RNA Sequence Design API

See also

[Inverse Folding \(Design\)](#)

## 3.14 Experimental Structure Probing Data

### 3.14.1 Guide the Structure Prediction using Experimental Data

#### 3.14.1.1 SHAPE reactivities

### 3.14.2 Structure Probing Data API

See also

[Experimental Structure Probing Data](#), [SHAPE Reactivity Data](#), [Generate Soft Constraints from Data](#)

## 3.15 Ligand Binding

### 3.15.1 Small Molecules and Proteins that bind to specific RNA Structures

### 3.15.2 ligand\_binding\_api

In our library, we provide two different ways to incorporate ligand binding to RNA structures:

- [Ligands Binding to Unstructured Domains](#), and
- [Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints](#)

The first approach is implemented as an actual extension of the folding grammar. It adds auxiliary derivation rules for each case when consecutive unpaired nucleotides are evaluated. Therefore, this model is applicable to ligand binding to any loop context.

The second approach, on the other hand, uses the soft-constraints feature to change the energy evaluation of hairpin- or interior-loops. Hence, it can only be applied when a ligand binds to a hairpin-like, or interior-loop like motif.

See also

[Ligands Binding to Unstructured Domains](#), [Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints](#)

## 3.16 (Tertiary) Structure Motifs

### 3.16.1 Incorporating Higher-Order (Tertiary) Structure Motifs

### 3.16.2 RNA G-Quadruplexes

### 3.16.3 (Tertiary) Structure Motif API



# Chapter 4

## I/O Formats

Below, you'll find a listing of different sections that introduce the most common notations of sequence and structure data, specifications of bioinformatics sequence and structure file formats, and various output file formats produced by our library.

- [RNA Structure Notations](#) describes the different notations and representations of RNA secondary structures
- [File Formats](#) gives an overview of the file formats compatible with our library
- [Plotting](#) shows the different (PostScript) plotting functions for RNA secondary structures, feature probabilities, and multiple sequence alignments

### 4.1 RNA Structure Notations

#### 4.1.1 Representations of Secondary Structures

The standard representation of a secondary structure in our library is the [Dot-Bracket Notation \(a.k.a. Dot-Parenthesis Notation\)](#), where matching brackets symbolize base pairs and unpaired bases are shown as dots. Based on that notation, more elaborate representations have been developed to include additional information, such as the loop context a nucleotide belongs to and to annotated pseudo-knots.

##### 4.1.1.1 Dot-Bracket Notation (a.k.a. Dot-Parenthesis Notation)

The Dot-Bracket notation as introduced already in the early times of the ViennaRNA Package denotes base pairs by matching pairs of parenthesis () and unpaired nucleotides by dots ..

As a simple example, consider a helix of size 4 enclosing a hairpin of size 4. In dot-bracket notation, this is annotated as

(( ( ( . . . ) ) ) )

##### Extended Dot-Bracket Notation

A more generalized version of the original Dot-Bracket notation may use additional pairs of brackets, such as <>, {}, and [], and matching pairs of uppercase/lowercase letters. This allows for annotating pseudo-knots, since different pairs of brackets are not required to be nested.

The following annotations of a simple structure with two crossing helices of size 4 are equivalent:

```
<<<< [ [ [ [ . . . ] >>> ] ] ] ]  
(( ( ( AAAA . . . ) ) ) ) aaaa  
AAAA { { { . . . aaaa } } }
```

##### See also

[vrna\\_db\\_pack\(\)](#), [vrna\\_db\\_unpack\(\)](#), [vrna\\_db\\_flatten\(\)](#), [vrna\\_db\\_flatten\\_to\(\)](#), [vrna\\_db\\_from\\_ptable\(\)](#), [vrna\\_db\\_from\\_plist\(\)](#), [vrna\\_db\\_to\\_element\\_string\(\)](#), [vrna\\_db\\_pk\\_remove\(\)](#)

#### 4.1.1.2 Washington University Secondary Structure (WUSS) notation

The WUSS notation, as frequently used for consensus secondary structures in [Stockholm 1.0 format](#).

This notation allows for a fine-grained annotation of base pairs and unpaired nucleotides, including pseudo-knots. Below, you'll find a list of secondary structure elements and their corresponding WUSS annotation (See also the infernal user guide at <http://eddylab.org/infernal/Userguide.pdf>)

- **Base pairs**

Nested base pairs are annotated by matching pairs of the symbols <>, (), {}, and []. Each of the matching pairs of parenthesis have their special meaning, however, when used as input in our programs, e.g. structure constraint, these details are usually ignored. Furthermore, base pairs that constitute as pseudo-knot are denoted by letters from the latin alphabet and are, if not denoted otherwise, ignored entirely in our programs.

- **Hairpin loops**

Unpaired nucleotides that constitute the hairpin loop are indicated by underscores, \_.

Example: <<<<\_\_\_\_\_>>>>

- **Bulges and interior loops**

Residues that constitute a bulge or interior loop are denoted by dashes, -.

Example: ( ( --<<\_\_\_\_\_>>- ) )

- **Multibranch loops**

Unpaired nucleotides in multibranch loops are indicated by commas ,.

Example: ( ( ,<<\_\_\_\_\_>>,<<\_\_\_\_\_>>) )

- **External residues**

Single stranded nucleotides in the exterior loop, i.e. not enclosed by any other pair are denoted by colons, ::.

Example: <<<\_\_\_\_\_>>>:::

- **Insertions**

In cases where an alignment represents the consensus with a known structure, insertions relative to the known structure are denoted by periods, ... Regions where local structural alignment was invoked, leaving regions of both target and query sequence unaligned, are indicated by tildes, ~.

**Note**

These symbols only appear in alignments of a known (query) structure annotation to a target sequence of unknown structure.

- **Pseudo-knots**

The WUSS notation allows for annotation of pseudo-knots using pairs of upper-case/lower-case letters.

**Note**

Our programs and library functions usually ignore pseudo-knots entirely treating them as unpaired nucleotides, if not stated otherwise.

Example: <<<\_AAA\_\_\_\_>>>aaa

**See also**

[vrna\\_db\\_from\\_WUSS\(\)](#)

### 4.1.1.3 Abstract Shapes

Abstract Shapes, introduced by Giegerich et al. in (2004) [10], collapse the secondary structure while retaining the nestedness of helices and hairpin loops.

The abstract shapes representation abstracts the structure from individual base pairs and their corresponding location in the sequence, while retaining the inherent nestedness of helices and hairpin loops.

Below is a description of what is included in the abstract shapes abstraction for each respective level together with an example structure:

```
CGUCUAAAACUCAUCACCGUGUGGAGCUGCGACCCUUCCUAGAUUCGAAGACGAG
((((((...((..((....)))))))...((..((....))...))))))))..
```

| Shape Level | Description  | Result                  |
|-------------|--|-------------------------|
| 1           | Most accurate - all loops and all unpaired   | [__[_[]]_[_[]_<]<br>]]_ |
| 2           | Nesting pattern for all loop types and unpaired regions in external loop and multiloop | [[_[]][_[]_]]           |
| 3           | Nesting pattern for all loop types but no unpaired regions                             | [[[[]][[[[]]]]          |
| 4           | Helix nesting pattern in external loop and multiloop                                   | [[[[]][[[[]]]]          |
| 5           | Most abstract - helix nesting pattern and no unpaired regions                          | [[[[]]]]                |

#### Note

Our implementations also provide the special Shape Level 0, which does not collapse any structural features but simply convert base pairs and unpaired nucleotides into their corresponding set of symbols for abstract shapes.

#### See also

[vrna\\_abstract\\_shapes\(\)](#), [vrna\\_abstract\\_shapes\\_pt\(\)](#)

### 4.1.1.4 Tree Representations of Secondary Structures

Secondary structures can be readily represented as trees, where internal nodes represent base pairs, and leaves represent unpaired nucleotides. The dot-bracket structure string already is a tree represented by a string of parenthesis (base pairs) and dots for the leaf nodes (unpaired nucleotides).

Alternatively, one may find representations with two types of node labels, P for paired and U for unpaired; a dot is then replaced by (U), and each closed bracket is assigned an additional identifier P. We call this the expanded notation. In [8] a condensed representation of the secondary structure is proposed, the so-called homeomorphically irreducible tree (HIT) representation. Here a stack is represented as a single pair of matching brackets labeled P and weighted by the number of base pairs. Correspondingly, a contiguous strain of unpaired bases is shown as one pair of matching brackets labeled U and weighted by its length. Generally any string consisting of matching brackets and identifiers is equivalent to a plane tree with as many different types of nodes as there are identifiers.

Bruce Shapiro proposed a coarse grained representation [21], which, does not retain the full information of the secondary structure. He represents the different structure elements by single matching brackets and labels them as

- H (hairpin loop),
- I (interior loop),
- B (bulge),
- M (multi-loop), and
- S (stack).

We extend his alphabet by an extra letter for external elements `E`. Again these identifiers may be followed by a weight corresponding to the number of unpaired bases or base pairs in the structure element. All tree representations (except for the dot-bracket form) can be encapsulated into a virtual root (labeled `R`).

The following example illustrates the different linear tree representations used by the package:

Consider the secondary structure represented by the dot-bracket string (full tree) `.((...(((....))))...((...))).` which is the most convenient condensed notation used by our programs and library functions.

Then, the following tree representations are equivalent:

- Expanded tree:  
`((U)((((U)(U)((((U)(U)(P)P)P)(U)(U)((((U)(U)P)P)P)P)(U)R))`
- HIT representation (Fontana et al. 1993 [8]):  
`((U1)((U2)((U3)P3)(U2)((U2)P2)P2)(U1)R)`
- Coarse Grained Tree Representation (Shapiro 1988 [21]):  
  - Short (with root node `R`, without stem nodes `S`):  
`((H)((HM)R))`
  - Full (with root node `R`):  
`(((((H)S)((HS)M)S)R))`
  - Extended (with root node `R`, with external nodes `E`):  
`(((((H)S)((HS)M)S)E)R))`
  - Weighted (with root node `R`, with external nodes `E`):  
`(((((H3)S3)((H2)S2)M4)S2)E2)R))`

The Expanded tree is rather clumsy and mostly included for the sake of completeness. The different versions of Coarse Grained Tree Representations are variations of Shapiro's linear tree notation.

For the output of aligned structures from string editing, different representations are needed, where we put the label on both sides. The above examples for tree representations would then look like:

```
* a) ((UU)(P(P(P(P(UU)(UU)(P(P(P(UU)(UU)(UU)P)P)P)(UU)(UU)(P(P(UU)(U...
* b) ((UU)(P2(P2(U2U2)(P2(U3U3)P3)(U2U2)(P2(U2U2)P2)P2)(UU)P2)(UU)
* c) ((B(M(HH)(HH)M)B)
*     ((S(B(S(M(S(HH)S)(S(HH)S)M)S)B)S)
*     ((E(S(B(S(M(S(HH)S)(S(HH)S)M)S)B)S)E)
* d) ((R(E2(S2(B1(S2(M4(S3(H3)S3)((H2)S2)M4)S2)B1)S2)E2)R)
*
```

Aligned structures additionally contain the gap character `_`.

See also

[vrna\\_db\\_to\\_tree\\_string\(\)](#), [vrna\\_tree\\_string\\_unweight\(\)](#), [vrna\\_tree\\_string\\_to\\_db\(\)](#)

#### 4.1.2 Examples for Structure Parsing and Conversion

#### 4.1.3 Structure Parsing and Conversion API

Several functions are provided for parsing structures and converting to different representations.

```
char *expand_Full(const char *structure)
```

Converts the full structure from bracket notation to the expanded notation including root.

```
char *b2HIT (const char *structure)
```

Converts the full structure from bracket notation to the HIT notation including root.

```
char *b2C (const char *structure)
```

Converts the full structure from bracket notation to the a coarse grained notation using the 'H' 'B' 'I' 'M' and 'R' identifiers.

```
char *b2Shapiro (const char *structure)
```

Converts the full structure from bracket notation to the *weighted* coarse grained notation using the 'H' 'B' 'I' 'M' 'S' 'E' and 'R' identifiers.

```
char *expand_Shapiro (const char *coarse);
```

Inserts missing 'S' identifiers in unweighted coarse grained structures as obtained from [b2C\(\)](#).

```
char *add_root (const char *structure)
```

Adds a root to an un-rooted tree in any except bracket notation.

```
char *unexpand_Full (const char *ffull)
```

Restores the bracket notation from an expanded full or HIT tree, that is any tree using only identifiers 'U' 'P' and 'R'.

```
char *unweight (const char *wcoarse)
```

Strip weights from any weighted tree.

```
void unexpand_aligned_F (char *align[2])
```

Converts two aligned structures in expanded notation.

```
void parse_structure (const char *structure)
```

Collects a statistic of structure elements of the full structure in bracket notation.

See also

[RNAAstruct.h](#) for prototypes and more detailed description

## 4.2 File Formats

### 4.2.1 File formats for Multiple Sequence Alignments (MSA)

#### 4.2.1.1 ClustalW format

The *ClustalW* format is a relatively simple text file containing a single multiple sequence alignment of DNA, RNA, or protein sequences. It was first used as an output format for the *clustalw* programs, but nowadays it may also be generated by various other sequence alignment tools. The specification is straight forward:

- The first line starts with the words

```
CLUSTAL W
```

or

```
CLUSTALW
```

- After the above header there is at least one empty line
- Finally, one or more blocks of sequence data are following, where each block is separated by at least one empty line

Each line in a blocks of sequence data consists of the sequence name followed by the sequence symbols, separated by at least one whitespace character. Usually, the length of a sequence in one block does not exceed 60 symbols. Optionally, an additional whitespace separated cumulative residue count may follow the sequence symbols. Optionally, a block may be followed by a line depicting the degree of conservation of the respective alignment columns.

Note

Sequence names and the sequences must not contain whitespace characters! Allowed gap symbols are the hyphen ("‐"), and dot (".").

## Warning

Please note that many programs that output this format tend to truncate the sequence names to a limited number of characters, for instance the first 15 characters. This can destroy the uniqueness of identifiers in your MSA.

Here is an example alignment in ClustalW format:

```
CLUSTAL W (1.83) multiple sequence alignment
```

|                            |  |
|----------------------------|--|
| AL031296.1/85969-86120     | CUGCCUCACAACGUUUGGCCUCAGUUACCGUAGAUGUAGUGAGGGUAACAAUACUUAC   |
| AANU01225121.1/438-603     | CUGCCUCACAACAUUUGGCCUCAGUUACUAGAUGUAGUGAGGGUGACAAUACUUAC     |
| AAWR02037329.1/29294-29150 | ---CUCGACACCACU---GCCUCGGUUACCCAUCGGUGCAGUGCGGGUAGUAGUACCAAU |
| AL031296.1/85969-86120     | UCUCGUUGGUGAUAGGAACAGCU                                      |
| AANU01225121.1/438-603     | UCUCGUUGGUGAUAGGAACAGCU                                      |
| AAWR02037329.1/29294-29150 | GCUAUUAGUUGUGAGGACCAACU                                      |

### 4.2.1.2 Stockholm 1.0 format

Here is an example alignment in Stockholm 1.0 format:

```
# STOCKHOLM 1.0

#=GF AC RF01293
#=GF ID ACA59
#=GF DE Small nucleolar RNA ACA59
#=GF AU Wilkinson A
#=GF SE Predicted; WAR; Wilkinson A
#=GF SS Predicted; WAR; Wilkinson A
#=GF GA 43.00
#=GF TC 44.90
#=GF NC 40.30
#=GF TP Gene; snRNA; snoRNA; HACA-box;
#=GF BM cmbuild -F CM SEED
#=GF CB cmcalibrate --mpi CM
#=GF SM cmsearch --cpu 4 --verbose --nohmmonly -E 1000 -Z 549862.597050 CM SEQDB
#=GF DR snoRNABase; ACA59;
#=GF DR SO; 0001263; ncRNA_gene;
#=GF DR GO; 0006396; RNA processing;
#=GF DR GO; 0005730; nucleolus;
#=GF RN [1]
#=GF RM 15199136
#=GF RT Human box H/ACA pseudouridylation guide RNA machinery.
#=GF RA Kiss AM, Jady BE, Bertrand E, Kiss T
#=GF RL Mol Cell Biol. 2004;24:5797-5807.
#=GF WK Small_nucleolar_RNA
#=GF SQ 3

AL031296.1/85969-86120 CUGCCUCACAACGUUUGGCCUCAGUUACCGUAGAUGUAGUGAGGGUAACAAUACUUACUCUCGUUGGUGAUAGGAACAGG
AANU01225121.1/438-603 CUGCCUCACAACAUUUGGCCUCAGUUACUAGAUGUAGUGAGGGUGACAAUACUUACUCUCGUUGGUGAUAGGAACAGG
AAWR02037329.1/29294-29150 ---CUCGACACCACU---GCCUCGGUUACCCAUCGGUGCAGUGCGGGUAGUAGUACCAUUGCIAAUUAGUUGUGAGGACCAAC
#=GC SS_cons -----((((<<<<<<_____>>>>>>,,,<<<<<<_____>>>>>,,,,))::::::::::
#=GC RF CUGCcccaCaaCacuuguGCCUCAGUUACcCauaggGuAGUGaGgGuggcAaUACccaCcCucgUUgGuggUaAGGAaCAgG
//
```

## See also

[Washington University Secondary Structure \(WUSS\) notation](#) on legal characters for the consensus secondary structure line *SS\_cons* and their interpretation

### 4.2.1.3 FASTA (Pearson) format

#### Note

Sequence names must not contain whitespace characters. Otherwise, the parts after the first whitespace will be dropped. The only allowed gap character is the hyphen ("").

Here is an example alignment in FASTA format:

```
>AL031296.1/85969-86120
CUGCCUCACAACGUUUGUGCCUCAGUUACCGUAGAUGUAGUGAGGGUAACAAUACUUAC
UCUCGUUGGUGAUAGGAACAGCU
>AANU01225121.1/438-603
CUGCCUCACAACAUUUGUGCCUCAGUUACAUAGAUGUAGUGAGGGUGACAAUACUUAC
UCUCGUUGGUGAUAGGAACAGCU
>AAWR02037329.1/29294-29150
---CUCGACACCACU---GCCUCGGUUAACCAUCGGUGCAGUGCAGGGUAGUAGUACCAAU
GCUAAUUAGUUGUGAGGACCAACU
```

#### 4.2.1.4 MAF format

The multiple alignment format (MAF) is usually used to store multiple alignments on DNA level between entire genomes. It consists of independent blocks of aligned sequences which are annotated by their genomic location. Consequently, an MAF formatted MSA file may contain multiple records. MAF files start with a line

```
##maf
```

which is optionally extended by whitespace delimited key=value pairs. Lines starting with the character ("#") are considered comments and usually ignored.

A MAF block starts with character ("a") at the beginning of a line, optionally followed by whitespace delimited key=value pairs. The next lines start with character ("s") and contain sequence information of the form

```
s src start size strand srcSize sequence
```

where

- *src* is the name of the sequence source
- *start* is the start of the aligned region within the source (0-based)
- *size* is the length of the aligned region without gap characters
- *strand* is either ("+" or "-"), depicting the location of the aligned region relative to the source
- *srcSize* is the size of the entire sequence source, e.g. the full chromosome
- *sequence* is the aligned sequence including gaps depicted by the hyphen (-)

Here is an example alignment in MAF format (bluntly taken from the [UCSC Genome browser website](#)):

```
##maf version=1 scoring=tba.v8
# tba.v8 ((human chimp) baboon) (mouse rat)
# multiz.v7
# maf_project.v5 _tba_right.maf3 mouse _tba_C
# single_cov2.v4 single_cov2 /dev/stdin

a score=23262.0
s hg16.chr7    27578828 38 + 158545518 AAA-GGGAATGTTAACCAAATGA---ATTGTCTCTTACGGTG
s panTro1.chr6 28741140 38 + 161576975 AAA-GGGAATGTTAACCAAATGA---ATTGTCTCTTACGGTG
s baboon       116834 38 +   4622798 AAA-GGGAATGTTAACCAAATGA---GTTGTCTCTTATGGTG
s mm4.chr6     53215344 38 + 151104725 -AATGGGAATGTTAACCAAACGA---ATTGTCTCTCAGTGTG
s rn3.chr4     81344243 40 + 187371129 -AA-GGGGATGCTAACCCAATGAGTTGTTCTCAATGTG

a score=5062.0
s hg16.chr7    27699739 6 + 158545518 TAAAGA
s panTro1.chr6 28862317 6 + 161576975 TAAAGA
s baboon       241163 6 +   4622798 TAAAGA
s mm4.chr6     53303881 6 + 151104725 TAAAGA
s rn3.chr4     81444246 6 + 187371129 taagga

a score=6636.0
s hg16.chr7    27707221 13 + 158545518 gcagctaaaaaca
s panTro1.chr6 28869787 13 + 161576975 gcagctaaaaaca
s baboon       249182 13 +   4622798 gcagctaaaaaca
s mm4.chr6     53310102 13 + 151104725 ACAGCTGAAAATA
```

## 4.2.2 File formats to manipulate the RNA folding grammar

### 4.2.2.1 Command Files

The RNAlib and many programs of the ViennaRNA Package can parse and apply data from so-called command files. These commands may refer to structure constraints or even extensions of the RNA folding grammar (such as [Unstructured Domains](#)). Commands are given as a line of whitespace delimited data fields. The syntax we use extends the constraint definitions used in the `mfold` / `UNAfold` software, where each line begins with a command character followed by a set of positions.

However, we introduce several new commands, and allow for an optional loop type context specifier in form of a sequence of characters, and an orientation flag that enables one to force a nucleotide to pair upstream, or downstream.

#### 4.2.2.1.1 Constraint commands

The following set of commands is recognized:

- F ... Force
- P ... Prohibit
- C ... Conflicts/Context dependency
- A ... Allow (for non-canonical pairs)
- E ... Soft constraints for unpaired position(s), or base pair(s)

#### 4.2.2.1.2 RNA folding grammar exnsions

- UD ... Add ligand binding using the [Unstructured Domains](#) feature

#### 4.2.2.1.3 Specification of the loop type context

The optional loop type context specifier [LOOP] may be a combination of the following:

- E ... Exterior loop
- H ... Hairpin loop
- I ... Interior loop
- M ... Multibranch loop
- A ... All loops

For structure constraints, we additionally allow one to address base pairs enclosed by a particular kind of loop, which results in the specifier [WHERE] which consists of [LOOP] plus the following character:

- i ... enclosed pair of an Interior loop
- m ... enclosed pair of a Multibranch loop

If no [LOOP] or [WHERE] flags are set, all contexts are considered (equivalent to A )

#### 4.2.2.1.4 Controlling the orientation of base pairing

For particular nucleotides that are forced to pair, the following [ORIENTATION] flags may be used:

- U ... Upstream
- D ... Downstream

If no [ORIENTATION] flag is set, both directions are considered.

#### 4.2.2.1.5 Sequence coordinates

Sequence positions of nucleotides/base pairs are 1– based and consist of three positions  $i$ ,  $j$ , and  $k$ . Alternatively, four positions may be provided as a pair of two position ranges  $[i : j]$ , and  $[k : l]$  using the '-' sign as delimiter within each range, i.e.  $i - j$ , and  $k - l$ .

**4.2.2.1.6 Valid constraint commands** Below are resulting general cases that are considered *valid* constraints:

1. **"Forcing a range of nucleotide positions to be paired"**:

Syntax:

```
F i 0 k [WHERE] [ORIENTATION]
```

Description:

Enforces the set of  $k$  consecutive nucleotides starting at position  $i$  to be paired. The optional loop type specifier [WHERE] allows to force them to appear as closing/enclosed pairs of certain types of loops.

2. **"Forcing a set of consecutive base pairs to form"**:

Syntax:

```
F i j k [WHERE]
```

Description:

Enforces the base pairs  $(i, j), \dots, (i + (k - 1), j - (k - 1))$  to form. The optional loop type specifier [WHERE] allows to specify in which loop context the base pair must appear.

3. **"Prohibiting a range of nucleotide positions to be paired"**:

Syntax:

```
P i 0 k [WHERE]
```

Description:

Prohibit a set of  $k$  consecutive nucleotides to participate in base pairing, i.e. make these positions unpaired. The optional loop type specifier [WHERE] allows to force the nucleotides to appear within the loop of specific types.

4. **"Prohibiting a set of consecutive base pairs to form"**:

Syntax:

```
P i j k [WHERE]
```

Description:

Prohibit the base pairs  $(i, j), \dots, (i + (k - 1), j - (k - 1))$  to form. The optional loop type specifier [WHERE] allows to specify the type of loop they are disallowed to be the closing or an enclosed pair of.

5. **"Prohibiting two ranges of nucleotides to pair with each other"**:

Syntax:

```
P i-j k-l [WHERE]
```

Description:

Prohibit any nucleotide  $p \in [i : j]$  to pair with any other nucleotide  $q \in [k : l]$ . The optional loop type specifier [WHERE] allows to specify the type of loop they are disallowed to be the closing or an enclosed pair of.

6. **"Enforce a loop context for a range of nucleotide positions"**:

Syntax:

```
C i 0 k [WHERE]
```

Description:

This command enforces nucleotides to be unpaired similar to *prohibiting* nucleotides to be paired, as described above. It too marks the corresponding nucleotides to be unpaired, however, the [WHERE] flag can be used to enforce specific loop types the nucleotides must appear in.

7. **"Remove pairs that conflict with a set of consecutive base pairs"**:

Syntax:

```
C i j k
```

Description:

Remove all base pairs that conflict with a set of consecutive base pairs  $(i, j), \dots, (i + (k - 1), j - (k - 1))$ . Two base pairs  $(i, j)$  and  $(p, q)$  conflict with each other if  $i < p < j < q$ , or  $p < i < q < j$ .

#### 8. "Allow a set of consecutive (non-canonical) base pairs to form":

Syntax:

A i j k [WHERE]

Description:

This command enables the formation of the consecutive base pairs  $(i, j), \dots, (i + (k - 1), j - (k - 1))$ , no matter if they are *canonical*, or *non-canonical*. In contrast to the above F and W commands, which remove conflicting base pairs, the A command does not. Therefore, it may be used to allow *non-canonical* base pair interactions. Since the RNAlib does not contain free energy contributions  $E_{ij}$  for non-canonical base pairs  $(i, j)$ , they are scored as the *maximum* of similar, known contributions. In terms of a *Nussinov* like scoring function the free energy of non-canonical base pairs is therefore estimated as

$$E_{ij} = \min \left[ \max_{(i,k) \in \{GC,CG,AU,UA,GU,UG\}} E_{ik}, \max_{(k,j) \in \{GC,CG,AU,UA,GU,UG\}} E_{kj} \right].$$

The optional loop type specifier [WHERE] allows to specify in which loop context the base pair may appear.

#### 9. "Apply pseudo free energy to a range of unpaired nucleotide positions":

Syntax:

E i 0 k e

Description:

Use this command to apply a pseudo free energy of  $e$  to the set of  $k$  consecutive nucleotides, starting at position  $i$ . The pseudo free energy is applied only if these nucleotides are considered unpaired in the recursions, or evaluations, and is expected to be given in *kcal/mol*.

#### 10. "Apply pseudo free energy to a set of consecutive base pairs":

Syntax

E i j k e

Use this command to apply a pseudo free energy of  $e$  to the set of base pairs  $(i, j), \dots, (i + (k - 1), j - (k - 1))$ . Energies are expected to be given in *kcal/mol*.

### 4.2.2.1.7 Valid domain extensions commands

#### 1. "Add ligand binding to unpaired motif (a.k.a. unstructured domains)":

Syntax:

UD m e [LOOP]

Description:

Add ligand binding to unpaired sequence motif  $m$  (given in IUPAC format, capital letters) with binding energy  $e$  in particular loop type(s).

Example:

UD AAA -5.0 A

The above example applies a binding free energy of  $-5\text{kcal/mol}$  for a motif AAA that may be present in all loop types.

## 4.3 Plotting

Create Plots of Secondary Structures, Feature Motifs, and Sequence Alignments

### 4.3.1 Producing secondary structure graphs

```
int PS_rna_plot ( char *string,
                  char *structure,
                  char *file)
```

Produce a secondary structure graph in PostScript and write it to 'filename'.

```
int PS_rna_plot_a (
    char *string,
    char *structure,
    char *file,
    char *pre,
    char *post)
```

Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename'.

```
int gmlRNA (char *string,
            char *structure,
            char *ssfile,
            char option)
```

Produce a secondary structure graph in Graph Meta Language (gml) and write it to a file.

```
int ssv_rna_plot (char *string,
                  char *structure,
                  char *ssfile)
```

Produce a secondary structure graph in SStructView format.

```
int svg_rna_plot (char *string,
                  char *structure,
                  char *ssfile)
```

Produce a secondary structure plot in SVG format and write it to a file.

```
int xrna_plot ( char *string,
                char *structure,
                char *ssfile)
```

Produce a secondary structure plot for further editing in XRNA.

```
int rna_plot_type
```

Switch for changing the secondary structure layout algorithm.

Two low-level functions provide direct access to the graph lauyouting algorithms:

```
int simple_xy_coordinates ( short *pair_table,
                            float *X,
                            float *Y)
```

Calculate nucleotide coordinates for secondary structure plot the *Simple way*

```
int naview_xy_coordinates ( short *pair_table,
                            float *X,
                            float *Y)
```

#### See also

[PS\\_dot.h](#) and [naview.h](#) for more detailed descriptions.

### 4.3.2 Producing (colored) dot plots for base pair probabilities

```
int PS_color_dot_plot ( char *string,
                        cpair *pi,
                        char *filename)

int PS_color_dot_plot_turn (char *seq,
                           cpair *pi,
                           char *filename,
                           int winSize)

int PS_dot_plot_list (char *seq,
                     char *filename,
                     plist *pl,
                     plist *mf,
                     char *comment)
```

Produce a postscript dot-plot from two pair lists.

```
int PS_dot_plot_turn (char *seq,
                     struct plist *pl,
                     char *filename,
                     int winSize)
```

**See also**

[PS\\_dot.h](#) for more detailed descriptions.

### 4.3.3 Producing (colored) alignments

```
int PS_color_aln (
    const char *structure,
    const char *filename,
    const char *seqs[],
    const char *names[])
```

Produce PostScript sequence alignment color-annotated by consensus structure.

# Chapter 5

## Basic Data Structures

- [Sequence and Structure Data](#) shows the most common types for sequence or structure data
- [The 'Fold Compound'](#) is the basic, central container for our implementations of prediction-, evaluation, and other algorithms
- [Model Details](#) provides the means to store the different model parameters

### 5.1 Sequence and Structure Data

See also

[Secondary Structure Utilities](#)

### 5.2 The 'Fold Compound'

See also

[The Fold Compound](#)

### 5.3 Model Details

See also

[Fine-tuning of the Implemented Models](#)



# Chapter 6

## API Features

- RNAlib API v3.0
- Callback Functions
- Scripting Language interface(s)

### 6.1 RNAlib API v3.0

#### 6.1.1 Introduction

With version 2.2 we introduce the new API that will take over the old one in the future version 3.0. By then, backwards compatibility will be broken, and third party applications using RNAlib need to be ported. This switch of API became necessary, since many new features found their way into the RNAlib where a balance between threadsafety and easy-to-use library functions is hard or even impossible to establish. Furthermore, many old functions of the library are present as slightly modified copies of themself to provide a crude way to overload functions.

Therefore, we introduce the new v3.0 API very early in our development stage such that developers have enough time to migrate to the new functions and interfaces. We also started to provide encapsulation of the RNAlib functions, data structures, typedefs, and macros by prefixing them with `vrna_` and `VRNA_`, respectively. Header files should also be included using the `ViennaRNA/` namespace, e.g.

```
#include <ViennaRNA/fold.h>  
instead of just using  
#include <fold.h>
```

as required for RNAlib 1.x and 2.x.

This eases the work for programmers of third party applications that would otherwise need to put much effort into renaming functions and data types in their own implementations if their names appear in our library. Since we still provide backward compatibility up to the last version of RNAlib 2.x, this advantage may be fully exploited only starting from v3.0 which will be released in the future. However, our plan is to provide the possibility for an early switch-off mechanism of the backward compatibility in one of our next releases of ViennaRNA Package 2.x.

#### 6.1.2 What are the major changes?

...

#### 6.1.3 How to port your program to the new API

...

#### 6.1.4 Some Examples using RNAlib API v3.0

Examples on how to use the new v3.0 API can be found in the [First Steps with the Fold Compound](#) section.

## 6.2 Callback Functions

With the new [RNAlib API v3.0](#) we introduce so-called callback mechanisms for several functions.

### 6.2.1 The purpose of Callback mechanisms

Using callback mechanisms, our library enables users not only to retrieve computed data without the need for parsing complicated data structures, but also allows one to tweak our implementation to do additional tasks without the requirement of a re-implementation of basic algorithms.

Our implementation of the callback mechanisms always follows the same scheme: The user:

- defines a function that complies with the interface we've defined, and
- passes a pointer to said function to our implementations

In addition to the specific arguments of our callback interfaces, virtually all callbacks receive an additional *pass-through-pointer* as their last argument. This enables one to:

- encapsulate data, and
- provide thread-safe operations,

since this pointer is simply passed through by our library functions. It may therefore hold the address of an arbitrary, user-defined data structure.

### 6.2.2 List of available Callbacks

Below, you find an enumeration of the individual callback functions that are available in *RNAlib*.

#### Global `vrna_boltzmann_sampling_callback (const char *structure, void *data)`

This function will be called for each secondary structure that has been successfully backtraced from the partition function DP matrices.

#### Global `vrna_callback_free_auxdata (void *data)`

This callback is supposed to free memory occupied by an auxiliary data structure. It will be called when the `vrna_fold_compound_t` is erased from memory through a call to `vrna_fold_compound_free()` and will be passed the address of memory previously bound to the `vrna_fold_compound_t` via `vrna_fold_compound_add_auxdata()`.

#### Global `vrna_callback_hc_evaluate (int i, int j, int k, int l, unsigned char d, void *data)`

This callback enables one to over-rule default hard constraints in secondary structure decompositions.

#### Global `vrna_callback_recursion_status (unsigned char status, void *data)`

This function will be called to notify a third-party implementation about the status of a currently ongoing recursion. The purpose of this callback mechanism is to provide users with a simple way to ensure pre- and post conditions for auxiliary mechanisms attached to our implementations.

#### Global `vrna_callback_sc_backtrack (int i, int j, int k, int l, unsigned char d, void *data)`

This callback enables one to add auxiliary base pairs in the backtracking steps of hairpin- and interior loops.

#### Global `vrna_callback_sc_energy (int i, int j, int k, int l, unsigned char d, void *data)`

This callback enables one to add (pseudo-)energy contributions to individual decompositions of the secondary structure.

#### Global `vrna_callback_sc_exp_energy (int i, int j, int k, int l, unsigned char d, void *data)`

This callback enables one to add (pseudo-)energy contributions to individual decompositions of the secondary structure (Partition function variant, i.e. contributions must be returned as Boltzmann factors).

#### Global `vrna_callback_ud_energy (vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, void *data)`

This function will be called to determine the additional energy contribution of a specific unstructured domain, e.g. the binding free energy of some ligand.

**Global `vrna_callback_ud_exp_energy` (`vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, void *data`)**

This function will be called to determine the additional energy contribution of a specific unstructured domain, e.g. the binding free energy of some ligand (Partition function variant, i.e. the Boltzmann factors instead of actual free energies).

**Global `vrna_callback_ud_exp_production` (`vrna_fold_compound_t *vc, void *data`)**

The production rule for the unstructured domain grammar extension (Partition function variant)

**Global `vrna_callback_ud_probs_add` (`vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, FLT_OR_DBL exp_energy, void *data`)**

A callback function to store equilibrium probabilities for the unstructured domain feature

**Global `vrna_callback_ud_probs_get` (`vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, int motif, void *data`)**

A callback function to retrieve equilibrium probabilities for the unstructured domain feature

**Global `vrna_callback_ud_production` (`vrna_fold_compound_t *vc, void *data`)**

The production rule for the unstructured domain grammar extension

**Global `vrna_heat_capacity_callback` (`float temp, float heat_capacity, void *data`)**

This function will be called for each evaluated temperature in the heat capacity prediction.

**Global `vrna_mfe_window_callback` (`int start, int end, const char *structure, float en, void *data`)**

This function will be called for each hit in a sliding window MFE prediction.

**Global `vrna_probs_window_callback` (`FLT_OR_DBL *pr, int pr_size, int i, int max, unsigned int type, void *data`)**

This function will be called for each probability data set in the sliding window probability computation implementation of `vrna_probs_window()`. The argument `type` specifies the type of probability that is passed to this function.

**Global `vrna_subopt_callback` (`const char *structure, float energy, void *data`)**

This function will be called for each suboptimal secondary structure that is successfully backtraced.

## 6.3 Scripting Language interface(s)

### 6.3.1 Introduction

For an easy integration into scripting languages, we provide an automatically generated interface to the RNAlib C-library, generated with SWIG.

### 6.3.2 Function Renaming

To provide a namespace-like separation of function symbols from our C library and third-party code, we use the prefix `vrna_` or `VRNA_` whenever possible. This, however, is not necessary for the scripting language interface, as it uses the separate namespace or package `RNA` anyway. Consequently, symbols that appear to have the `vrna_` or `VRNA_` prefix in the C-library have the corresponding prefix stripped away.

For instance, the C code

```
mfe = vrna_fold(sequence, structure);
translates to
my ($structure, $mfe) = RNA::fold($sequence)
in the Perl 5 interface, and
structure,mfe = RNA::fold(sequence)
```

for Python. Note, that in this example we also make use of the possibility to return multiple data at once in the scripting language, while the C library function uses additional parameters to return multiple data.

Functions that are dedicated to work on specific data structures only, e.g. the `vrna_fold_compound_t`, are usually not exported at all. Instead, they are attached as object methods of a corresponding class (see [Object oriented Interface for Data Structures](#) for detailed information).

### 6.3.2.1 Global Variables

For the Python interface(s) SWIG places global variables of the C-library into an additional namespace `cvar`. For instance, changing the global temperature variable thus becomes

```
RNA.cvar.temperature = 25
```

### 6.3.3 Object oriented Interface for Data Structures

For data structures, typedefs, and enumerations the `vrna_` prefixes are dropped as well, together with their suffixes `_s`, `_t`, and `_e`, respectively. Furthermore, data structures are usually transformed into classes and relevant functions of the C-library are attached as methods.

### 6.3.4 Examples

Examples on the basic usage of the scripting language interfaces can be found in the [Perl5 Examples](#) and [Python Examples](#) section.

### 6.3.5 SWIG generated Wrapper notes

Special notes on how functions, structures, enums, and macro definitions are actually wrapped, can be found below

#### Global `vrna_abstract_shapes (const char *structure, unsigned int level)`

This function is available as an overloaded function `abstract_shapes()` where the optional second parameter `level` defaults to 5.

#### Global `vrna_abstract_shapes_pt (const short *pt, unsigned int level)`

This function is available as an overloaded function `abstract_shapes()` where the optional second parameter `level` defaults to 5.

#### Global `vrna_aln_conservation_col (const char **alignment, const vrna_md_t *md_p, unsigned int options)`

This function is available in an overloaded form where the last two parameters may be omitted, indicating `md = NULL`, and `options = VRNA_MEASURE_SHANNON_ENTROPY`, respectively.

#### Global `vrna_aln_conservation_struct (const char **alignment, const char *structure, const vrna_md_t *md)`

This function is available in an overloaded form where the last parameter may be omitted, indicating `md = NULL`.

#### Global `vrna_backtrack5 (vrna_fold_compound_t *fc, unsigned int length, char *structure)`

This function is attached as overloaded method `backtrack()` to objects of type `fold_compound` with default parameter `length` equal to the total length of the RNA.

#### Global `vrna_boustrophedon (size_t start, size_t end)`

This function is available as overloaded global function `boustrophedon()`.

#### Global `vrna_boustrophedon_pos (size_t start, size_t end, size_t pos)`

This function is available as overloaded global function `boustrophedon()`. Omitting the `pos` argument yields the entire sequence from `start` to `end`.

#### Global `vrna_bp_distance (const char *str1, const char *str2)`

This function is available as an overloaded method `bp_distance()`. Note that the SWIG wrapper takes two structure in dot-bracket notation and converts them into pair tables using `vrna_ptable_from_string()`. The resulting pair tables are then internally passed to `vrna_bp_distance_pt()`. To control which kind of matching brackets will be used during conversion, the optional argument `options` can be used. See also the description of `vrna_ptable_from_string()` for available options. (default: `VRNA_BRACKETS_RND`).

#### Global `vrna_bp_distance_pt (const short *pt1, const short *pt2)`

This function is available as an overloaded method `bp_distance()`.

#### Global `vrna_db_flatten (char *structure, unsigned int options)`

This function flattens an input structure string in-place! The second parameter is optional and defaults to `VRNA_BRACKETS_DEFAULT`.

An overloaded version of this function exists, where an additional second parameter can be passed to specify the target brackets, i.e. the type of matching pair characters all brackets will be flattened to. Therefore, in the scripting language interface this function is a replacement for `vrna_db_flatten_to()`.

**Global `vrna_db_flatten_to` (`char *string, const char target[3], unsigned int options`)**

This function is available as an overloaded version of `vrna_db_flatten()`

**Global `vrna_db_from_probs` (`const FLT_OR_DBL *pr, unsigned int length`)**

This function is available as parameter-less method `db_from_probs()` bound to objects of type `fold_compound`. Parameters `pr` and `length` are implicitly taken from the `fold_compound` object the method is bound to. Upon missing base pair probabilities, this method returns an empty string.

**Global `vrna_db_pk_remove` (`const char *structure, unsigned int options`)**

This function is available as an overloaded function `db_pk_remove()` where the optional second parameter `options` defaults to `VRNA_BRACKETS_ANY`.

**Global `vrna_ensemble_defect` (`vrna_fold_compound_t *fc, const char *structure`)**

This function is attached as method `ensemble_defect()` to objects of type `fold_compound`. Note that the SWIG wrapper takes a structure in dot-bracket notation and converts it into a pair table using `vrna_ptable_from_string()`. The resulting pair table is then internally passed to `vrna_ensemble_defect_pt()`. To control which kind of matching brackets will be used during conversion, the optional argument `options` can be used. See also the description of `vrna_ptable_from_string()` for available options. (default: `VRNA_BRACKETS_RND`).

**Global `vrna_ensemble_defect_pt` (`vrna_fold_compound_t *fc, const short *pt`)**

This function is attached as overloaded method `ensemble_defect()` to objects of type `fold_compound`.

**Global `vrna_enumerate_necklaces` (`const unsigned int *type_counts`)**

This function is available as global function `enumerate_necklaces()` which accepts lists input, and produces list of lists output.

**Global `vrna_eval_circ_consensus_structure` (`const char **alignment, const char *structure`)**

This function is available through an overloaded version of `vrna_eval_circ_structure()`. Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

**Global `vrna_eval_circ_consensus_structure_v` (`const char **alignment, const char *structure, int verbosity_level, FILE *file`)**

This function is available through an overloaded version of `vrna_eval_circ_structure()`. Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to `VRNA_VERBOSITY QUIET` and NULL, respectively.

**Global `vrna_eval_circ_gquad_consensus_structure` (`const char **alignment, const char *structure`)**

This function is available through an overloaded version of `vrna_eval_circ_gquad_structure()`. Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

**Global `vrna_eval_circ_gquad_consensus_structure_v` (`const char **alignment, const char *structure, int verbosity_level, FILE *file`)**

This function is available through an overloaded version of `vrna_eval_circ_gquad_structure()`. Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to `VRNA_VERBOSITY QUIET` and NULL, respectively.

**Global `vrna_eval_circ_gquad_structure` (`const char *string, const char *structure`)**

In the target scripting language, this function serves as a wrapper for `vrna_eval_circ_gquad_structure_v()` and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to `VRNA_VERBOSITY QUIET` and NULL, respectively.

**Global `vrna_eval_circ_gquad_structure_v` (`const char *string, const char *structure, int verbosity_level, FILE *file`)**

This function is available through an overloaded version of `vrna_eval_circ_gquad_structure()`. The last two arguments for this function are optional and default to `VRNA_VERBOSITY QUIET` and NULL, respectively.

**Global `vrna_eval_circ_structure` (`const char *string, const char *structure`)**

In the target scripting language, this function serves as a wrapper for `vrna_eval_circ_structure_v()` and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to `VRNA_VERBOSITY QUIET` and NULL, respectively.

**Global `vrna_eval_circ_structure_v` (`const char *string, const char *structure, int verbosity_level, FILE *file`)**

This function is available through an overloaded version of `vrna_eval_circ_structure()`. The last two arguments for this function are optional and default to `VRNA_VERBOSITY QUIET` and `NULL`, respectively.

**Global `vrna_eval_consensus_structure_pt_simple` (`const char **alignment, const short *pt`)**

This function is available through an overloaded version of `vrna_eval_structure_pt_simple()`. Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

**Global `vrna_eval_consensus_structure_pt_simple_v` (`const char **alignment, const short *pt, int verbosity_level, FILE *file`)**

This function is available through an overloaded version of `vrna_eval_structure_pt_simple()`. Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to `VRNA_VERBOSITY QUIET` and `NULL`, respectively.

**Global `vrna_eval_consensus_structure_pt_simple_verbose` (`const char **alignment, const short *pt, FILE *file`)**

This function is not available. Use `vrna_eval_consensus_structure_pt_v()` instead!

**Global `vrna_eval_consensus_structure_simple` (`const char **alignment, const char *structure`)**

This function is available through an overloaded version of `vrna_eval_structure_simple()`. Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

**Global `vrna_eval_consensus_structure_simple_v` (`const char **alignment, const char *structure, int verbosity_level, FILE *file`)**

This function is available through an overloaded version of `vrna_eval_structure_simple()`. Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to `VRNA_VERBOSITY QUIET` and `NULL`, respectively.

**Global `vrna_eval_consensus_structure_simple_verbose` (`const char **alignment, const char *structure, FILE *file`)**

This function is not available. Use `vrna_eval_consensus_structure_simple_v()` instead!

**Global `vrna_eval_covar_structure` (`vrna_fold_compound_t *fc, const char *structure`)**

This function is attached as method `eval_covar_structure()` to objects of type `fold_compound`

**Global `vrna_eval_gquad_consensus_structure` (`const char **alignment, const char *structure`)**

This function is available through an overloaded version of `vrna_eval_gquad_structure()`. Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

**Global `vrna_eval_gquad_consensus_structure_v` (`const char **alignment, const char *structure, int verbosity_level, FILE *file`)**

This function is available through an overloaded version of `vrna_eval_gquad_structure()`. Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to `VRNA_VERBOSITY QUIET` and `NULL`, respectively.

**Global `vrna_eval_gquad_structure` (`const char *string, const char *structure`)**

In the target scripting language, this function serves as a wrapper for `vrna_eval_gquad_structure_v()` and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to `VRNA_VERBOSITY QUIET` and `NULL`, respectively.

**Global `vrna_eval_gquad_structure_v` (`const char *string, const char *structure, int verbosity_level, FILE *file`)**

This function is available through an overloaded version of `vrna_eval_gquad_structure()`. The last two arguments for this function are optional and default to `VRNA_VERBOSITY QUIET` and `NULL`, respectively.

**Global `vrna_eval_hp_loop` (`vrna_fold_compound_t *fc, int i, int j`)**

This function is attached as method `eval_hp_loop()` to objects of type `fold_compound`

**Global `vrna_eval_int_loop` (`vrna_fold_compound_t *fc, int i, int j, int k, int l`)**

This function is attached as method `eval_int_loop()` to objects of type `fold_compound`

**Global `vrna_eval_loop_pt` (`vrna_fold_compound_t *fc, int i, const short *pt`)**

This function is attached as method `eval_loop_pt()` to objects of type `fold_compound`

**Global `vrna_eval_move` (`vrna_fold_compound_t *fc, const char *structure, int m1, int m2`)**

This function is attached as method `eval_move()` to objects of type `fold_compound`

**Global `vrna_eval_move_pt` (`vrna_fold_compound_t *fc, short *pt, int m1, int m2`)**

This function is attached as method `eval_move_pt()` to objects of type `fold_compound`

**Global `vrna_eval_structure` (`vrna_fold_compound_t *fc, const char *structure`)**

This function is attached as method `eval_structure()` to objects of type `fold_compound`

**Global `vrna_eval_structure_pt` (`vrna_fold_compound_t *fc, const short *pt`)**

This function is attached as method `eval_structure_pt()` to objects of type `fold_compound`

**Global `vrna_eval_structure_pt_simple` (`const char *string, const short *pt`)**

In the target scripting language, this function serves as a wrapper for `vrna_eval_structure_pt_v()` and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to `VRNA_VERBOSITY QUIET` and `NULL`, respectively.

**Global `vrna_eval_structure_pt_verbose` (`vrna_fold_compound_t *fc, const short *pt, FILE *file`)**

This function is attached as method `eval_structure_pt_verbose()` to objects of type `fold_compound`

**Global `vrna_eval_structure_simple` (`const char *string, const char *structure`)**

In the target scripting language, this function serves as a wrapper for `vrna_eval_structure_simple_v()` and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to `VRNA_VERBOSITY QUIET` and `NULL`, respectively.

**Global `vrna_eval_structure_simple_v` (`const char *string, const char *structure, int verbosity_level, FILE *file`)**

This function is available through an overloaded version of `vrna_eval_structure_simple()`. The last two arguments for this function are optional and default to `VRNA_VERBOSITY QUIET` and `NULL`, respectively.

**Global `vrna_eval_structure_simple_verbose` (`const char *string, const char *structure, FILE *file`)**

This function is not available. Use `vrna_eval_structure_simple_v()` instead!

**Global `vrna_eval_structure_verbose` (`vrna_fold_compound_t *fc, const char *structure, FILE *file`)**

This function is attached as method `eval_structure_verbose()` to objects of type `fold_compound`

**Global `vrna_exp_params_rescale` (`vrna_fold_compound_t *vc, double *mfe`)**

This function is attached to `vrna_fc_s` objects as overloaded `exp_params_rescale()` method.

When no parameter is passed to this method, the resulting action is the same as passing `NULL` as second parameter to `vrna_exp_params_rescale()`, i.e. default scaling of the partition function. Passing an energy in kcal/mol, e.g. as retrieved by a previous call to the `mfe()` method, instructs all subsequent calls to scale the partition function accordingly.

**Global `vrna_exp_params_reset` (`vrna_fold_compound_t *vc, vrna_md_t *md_p`)**

This function is attached to `vrna_fc_s` objects as overloaded `exp_params_reset()` method.

When no parameter is passed to this method, the resulting action is the same as passing `NULL` as second parameter to `vrna_exp_params_reset()`, i.e. global default model settings are used. Passing an object of type `vrna_md_s` resets the fold compound according to the specifications stored within the `vrna_md_s` object.

**Global `vrna_exp_params_subst` (`vrna_fold_compound_t *vc, vrna_exp_param_t *params`)**

This function is attached to `vrna_fc_s` objects as overloaded `exp_params_subst()` method.

When no parameter is passed, the resulting action is the same as passing `NULL` as second parameter to `vrna_exp_params_subst()`, i.e. resetting the parameters to the global defaults.

**Class `vrna_fc_s`**

This data structure is wrapped as an object `fold_compound` with several related functions attached as methods.

A new `fold_compound` can be obtained by calling one of its constructors:

- `fold_compound(seq)` – Initialize with a single sequence, or two concatenated sequences separated by an ampersand character '`&`' (for cofolding)

- *fold\_compound(aln)* – Initialize with a sequence alignment *aln* stored as a list of sequences (with gap characters)

The resulting object has a list of attached methods which in most cases directly correspond to functions that mainly operate on the corresponding C data structure:

- *type()* – Get the type of the *fold\_compound* (See [vrna\\_fc\\_type\\_e](#))
- *length()* – Get the length of the sequence(s) or alignment stored within the *fold\_compound*

#### Global [vrna\\_file\\_commands\\_apply](#) (*vrna\_fold\_compound\_t* \**vc*, *const char* \**filename*, *unsigned int* *options*)

This function is attached as method [file\\_commands\\_apply\(\)](#) to objects of type *fold\_compound*

#### Global [vrna\\_file\\_msa\\_detect\\_format](#) (*const char* \**filename*, *unsigned int* *options*)

This function exists as an overloaded version where the *options* parameter may be omitted! In that case, the *options* parameter defaults to [VRNA\\_FILE\\_FORMAT\\_MSA\\_DEFAULT](#).

#### Global [vrna\\_file\\_msa\\_read](#) (*const char* \**filename*, *char \*\*\*names*, *char \*\*\*aln*, *char \*\*id*, *char \*\*structure*, *unsigned int* *options*)

In the target scripting language, only the first and last argument, *filename* and *options*, are passed to the corresponding function. The other arguments, which serve as output in the C-library, are available as additional return values. Hence, a function call in python may look like this:

#### Global [vrna\\_file\\_msa\\_read\\_record](#) (*FILE* \**fp*, *char \*\*\*names*, *char \*\*\*aln*, *char \*\*id*, *char \*\*structure*, *unsigned int* *options*)

In the target scripting language, only the first and last argument, *fp* and *options*, are passed to the corresponding function. The other arguments, which serve as output in the C-library, are available as additional return values. Hence, a function call in python may look like this:

#### Global [vrna\\_file\\_msa\\_write](#) (*const char* \**filename*, *const char \*\*names*, *const char \*\*aln*, *const char \*id*, *const char \*structure*, *const char \*source*, *unsigned int* *options*)

In the target scripting language, this function exists as a set of overloaded versions, where the last four parameters may be omitted. If the *options* parameter is missing the options default to ([VRNA\\_FILE\\_FORMAT\\_MSA\\_STOCKHOLM](#) | [VRNA\\_FILE\\_FORMAT\\_MSA\\_APPEND](#)).

#### Global [vrna\\_file\\_PS\\_aln](#) (*const char* \**filename*, *const char \*\*seqs*, *const char \*\*names*, *const char \*structure*, *unsigned int columns*)

This function is available as overloaded function [file\\_PS\\_aln\(\)](#) with three additional parameters *start*, *end*, and *offset* before the *columns* argument. Thus, it resembles the [vrna\\_file\\_PS\\_aln\\_slice\(\)](#) function. The last four arguments may be omitted, indicating the default of *start* = 0, *end* = 0, *offset* = 0, and *columns* = 60.

#### Global [vrna\\_file\\_PS\\_aln\\_slice](#) (*const char* \**filename*, *const char \*\*seqs*, *const char \*\*names*, *const char \*structure*, *unsigned int start*, *unsigned int end*, *int offset*, *unsigned int columns*)

This function is available as overloaded function [file\\_PS\\_aln\(\)](#) where the last four parameter may be omitted, indicating *start* = 0, *end* = 0, *offset* = 0, and *columns* = 60.

#### Global [vrna\\_hc\\_add\\_from\\_db](#) (*vrna\_fold\_compound\_t* \**vc*, *const char* \**constraint*, *unsigned int* *options*)

This function is attached as method [hc\\_add\\_from\\_db\(\)](#) to objects of type *fold\_compound*

#### Global [vrna\\_hc\\_init](#) (*vrna\_fold\_compound\_t* \**vc*)

This function is attached as method [hc\\_init\(\)](#) to objects of type *fold\_compound*

#### Global [vrna\\_heat\\_capacity](#) (*vrna\_fold\_compound\_t* \**fc*, *float T\_min*, *float T\_max*, *float T\_increment*, *unsigned int mpoints*)

This function is attached as overloaded method [heat\\_capacity\(\)](#) to objects of type *fold\_compound*. If the optional function arguments *T\_min*, *T\_max*, *T\_increment*, and *mpoints* are omitted, they default to 0.0, 100.0, 1.0 and 2, respectively.

#### Global [vrna\\_heat\\_capacity\\_cb](#) (*vrna\_fold\_compound\_t* \**fc*, *float T\_min*, *float T\_max*, *float T\_increment*, *unsigned int mpoints*, *vrna\_heat\_capacity\_callback* \**cb*, *void* \**data*)

This function is attached as method [heat\\_capacity\\_cb\(\)](#) to objects of type *fold\_compound*

**Global `vrna_heat_capacity_simple` (`const char *sequence, float T_min, float T_max, float T_increment, unsigned int mpoints`)**

This function is available as overloaded function `heat_capacity()`. If the optional function arguments `T_min`, `T_max`, `T_increment`, and `mpoints` are omitted, they default to 0.0, 100.0, 1.0 and 2, respectively.

**Global `vrna_init_rand_seed` (`unsigned int seed`)**

This function is available as an overloaded function `init_rand()` where the argument `seed` is optional.

**Global `vrna_maximum_matching` (`vrna_fold_compound_t *fc`)**

This function is attached as method `maximum_matching()` to objects of type `fold_compound` (i.e. `vrna_fold_compound_t`).

**Global `vrna_maximum_matching_simple` (`const char *sequence`)**

This function is available as global function `maximum_matching()`.

**Class `vrna_md_s`**

This data structure is wrapped as an object `md` with multiple related functions attached as methods.

A new set of default parameters can be obtained by calling the constructor of `md`:

- `md()` – Initialize with default settings

The resulting object has a list of attached methods which directly correspond to functions that mainly operate on the corresponding `C` data structure:

- `reset()` – `vrna_md_set_default()`
- `set_from_globals()` – `set_model_details()`
- `option_string()` – `vrna_md_option_string()`

Note, that default parameters can be modified by directly setting any of the following global variables. Internally, getting/setting default parameters using their global variable representative translates into calls of the following functions, therefore these wrappers for these functions do not exist in the scripting language interface(s):

**Global `vrna_MEA` (`vrna_fold_compound_t *fc, double gamma, float *mea`)**

This function is attached as overloaded method `MEA(gamma = 1.)` to objects of type `fold_compound`. Note, that it returns the MEA structure and MEA value as a tuple (MEA\_structure, MEA)

**Global `vrna_MEA_from_plist` (`vrna_ep_t *plist, const char *sequence, double gamma, vrna_md_t *md, float *mea`)**

This function is available as overloaded function `MEA_from_plist(gamma = 1., md = NULL)`. Note, that it returns the MEA structure and MEA value as a tuple (MEA\_structure, MEA)

**Global `vrna_mean_bp_distance` (`vrna_fold_compound_t *vc`)**

This function is attached as method `mean_bp_distance()` to objects of type `fold_compound`

**Global `vrna_mfe` (`vrna_fold_compound_t *vc, char *structure`)**

This function is attached as method `mfe()` to objects of type `fold_compound`

**Global `vrna_mfe_dimer` (`vrna_fold_compound_t *vc, char *structure`)**

This function is attached as method `mfe_dimer()` to objects of type `fold_compound`

**Global `vrna_mfe_window` (`vrna_fold_compound_t *vc, FILE *file`)**

This function is attached as method `mfe_window()` to objects of type `fold_compound`

**Global `vrna_neighbors` (`vrna_fold_compound_t *vc, const short *pt, unsigned int options`)**

This function is attached as an overloaded method `neighbors()` to objects of type `fold_compound`. The optional parameter `options` defaults to `VRNA_MOVESET_DEFAULT` if it is omitted.

**Global `vrna_params_load` (`const char fname[], unsigned int options`)**

This function is available as overloaded function `params_load(fname="", options=VRNA_PARAMETER_FORMAT_DEFAULT)`. Here, the empty filename string indicates to load default RNA parameters, i.e. this is equivalent to calling `vrna_params_load_defaults()`.

**Global `vrna_params_load_defaults` (void)**

This function is available as overloaded function `params_load()`.

**Global `vrna_params_load_DNA_Mathews1999` (void)**

This function is available as function `params_load_DNA_Mathews1999()`.

**Global `vrna_params_load_DNA_Mathews2004` (void)**

This function is available as function `params_load_DNA_Mathews2004()`.

**Global `vrna_params_load_from_string` (const char \*string, const char \*name, unsigned int options)**

This function is available as overloaded function `params_load_from_string(string, name="", options=VRNA_PARAMETER_FORMAT_DEFAULT)`.

**Global `vrna_params_load_RNA_Andronescu2007` (void)**

This function is available as function `params_load_RNA_Andronescu2007()`.

**Global `vrna_params_load_RNA_Langdon2018` (void)**

This function is available as function `params_load_RNA_Langdon2018()`.

**Global `vrna_params_load_RNA_misc_special_hairpins` (void)**

This function is available as function `params_load_RNA_misc_special_hairpins()`.

**Global `vrna_params_load_RNA_Turner1999` (void)**

This function is available as function `params_load_RNA_Turner1999()`.

**Global `vrna_params_load_RNA_Turner2004` (void)**

This function is available as function `params_load_RNA_Turner2004()`.

**Global `vrna_params_reset` (vrna\_fold\_compound\_t \*vc, vrna\_md\_t \*md\_p)**

This function is attached to `vrna_fc_s` objects as overloaded `params_reset()` method.

When no parameter is passed to this method, the resulting action is the same as passing `NULL` as second parameter to `vrna_params_reset()`, i.e. global default model settings are used. Passing an object of type `vrna_md_s` resets the fold compound according to the specifications stored within the `vrna_md_s` object.

**Global `vrna_params_save` (const char fname[], unsigned int options)**

This function is available as overloaded function `params_save(fname, options=VRNA_PARAMETER_FORMAT_DEFAULT)`.

**Global `vrna_params_subst` (vrna\_fold\_compound\_t \*vc, vrna\_param\_t \*par)**

This function is attached to `vrna_fc_s` objects as overloaded `params_subst()` method.

When no parameter is passed, the resulting action is the same as passing `NULL` as second parameter to `vrna_params_subst()`, i.e. resetting the parameters to the global defaults.

**Global `vrna_path` (vrna\_fold\_compound\_t \*vc, short \*pt, unsigned int steps, unsigned int options)**

This function is attached as an overloaded method `path()` to objects of type `fold_compound`. The optional parameter `options` defaults to `VRNA_PATH_DEFAULT` if it is omitted.

**Global `vrna_path_direct` (vrna\_fold\_compound\_t \*fc, const char \*s1, const char \*s2, vrna\_path\_options\_t options)**

This function is attached as an overloaded method `path_direct()` to objects of type `fold_compound`. The optional parameter `options` defaults to `NULL` if it is omitted.

**Global `vrna_path_direct_ub` (vrna\_fold\_compound\_t \*fc, const char \*s1, const char \*s2, int maxE, vrna\_path\_options\_t options)**

This function is attached as an overloaded method `path_direct()` to objects of type `fold_compound`. The optional parameter `maxE` defaults to `#INT_MAX - 1` if it is omitted, while the optional parameter `options` defaults to `NULL`. In case the function did not find a path with  $E_{saddle} < E_{max}$  it returns an empty list.

**Global `vrna_path_findpath` (vrna\_fold\_compound\_t \*fc, const char \*s1, const char \*s2, int width)**

This function is attached as an overloaded method `path_findpath()` to objects of type `fold_compound`. The optional parameter `width` defaults to 1 if it is omitted.

**Global `vrna_path_findpath_saddle` (vrna\_fold\_compound\_t \*fc, const char \*s1, const char \*s2, int width)**

This function is attached as an overloaded method `path_findpath_saddle()` to objects of type `fold_compound`. The optional parameter `width` defaults to 1 if it is omitted.

**Global `vrna_path_findpath_saddle_ub` (`vrna_fold_compound_t *fc, const char *s1, const char *s2, int width, int maxE`)**

This function is attached as an overloaded method `path_findpath_saddle()` to objects of type `fold_compound`. The optional parameter `width` defaults to 1 if it is omitted, while the optional parameter `maxE` defaults to `INF`. In case the function did not find a path with  $E_{saddle} < E_{max}$  the function returns a `NULL` object, i.e. `undef` for Perl and `None` for Python.

**Global `vrna_path_findpath_ub` (`vrna_fold_compound_t *fc, const char *s1, const char *s2, int width, int maxE`)**

This function is attached as an overloaded method `path_findpath()` to objects of type `fold_compound`. The optional parameter `width` defaults to 1 if it is omitted, while the optional parameter `maxE` defaults to `INF`. In case the function did not find a path with  $E_{saddle} < E_{max}$  the function returns an empty list.

**Global `vrna_path_gradient` (`vrna_fold_compound_t *vc, short *pt, unsigned int options`)**

This function is attached as an overloaded method `path_gradient()` to objects of type `fold_compound`. The optional parameter `options` defaults to `VRNA_PATH_DEFAULT` if it is omitted.

**Global `vrna_path_options_findpath` (`int width, unsigned int type`)**

This function is available as overloaded function `path_options_findpath()`. The optional parameter `width` defaults to 10 if omitted, while the optional parameter `type` defaults to `VRNA_PATH_TYPE_DOT_BRACKET`.

**Global `vrna_path_random` (`vrna_fold_compound_t *vc, short *pt, unsigned int steps, unsigned int options`)**

This function is attached as an overloaded method `path_random()` to objects of type `fold_compound`. The optional parameter `options` defaults to `VRNA_PATH_DEFAULT` if it is omitted.

**Global `vrna_pbacktrack` (`vrna_fold_compound_t *fc`)**

This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound`. See also [Python Examples - Boltzmann Sampling](#)

**Global `vrna_pbacktrack5` (`vrna_fold_compound_t *fc, unsigned int length`)**

This function is attached as overloaded method `pbacktrack5()` to objects of type `fold_compound`. See also [Python Examples - Boltzmann Sampling](#)

**Global `vrna_pbacktrack5_cb` (`vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int length, vrna_boltzmann_sampling_callback *cb, void *data, unsigned int options`)**

This function is attached as overloaded method `pbacktrack5()` to objects of type `fold_compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

**Global `vrna_pbacktrack5_num` (`vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int length, unsigned int options`)**

This function is attached as overloaded method `pbacktrack5()` to objects of type `fold_compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

**Global `vrna_pbacktrack5_resume` (`vrna_fold_compound_t *vc, unsigned int num_samples, unsigned int length, vrna_pbacktrack_mem_t *nr_mem, unsigned int options`)**

This function is attached as overloaded method `pbacktrack5()` to objects of type `fold_compound`. In addition to the list of structures, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

**Global `vrna_pbacktrack5_resume_cb` (`vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int length, vrna_boltzmann_sampling_callback *cb, void *data, vrna_pbacktrack_mem_t *nr_mem, unsigned int options`)**

This function is attached as overloaded method `pbacktrack5()` to objects of type `fold_compound`. In addition to the number of structures backtraced, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

**Global `vrna_pbacktrack_cb` (`vrna_fold_compound_t *fc, unsigned int num_samples, vrna_boltzmann_sampling_callback *cb, void *data, unsigned int options`)**

This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

**Global `vrna_pbacktrack_num` (`vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int options`)**

This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

**Global `vrna_pbacktrack_resume` (`vrna_fold_compound_t *fc, unsigned int num_samples, vrna_pbacktrack_mem_t *nr_mem, unsigned int options`)**

This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound`. In addition to the list of structures, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

**Global `vrna_pbacktrack_resume_cb` (`vrna_fold_compound_t *fc, unsigned int num_samples, vrna_boltzmann_sampling_callback *cb, void *data, vrna_pbacktrack_mem_t *nr_mem, unsigned int options`)**

This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound`. In addition to the number of structures backtraced, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

**Global `vrna_pbacktrack_sub` (`vrna_fold_compound_t *fc, unsigned int start, unsigned int end`)**

This function is attached as overloaded method `pbacktrack_sub()` to objects of type `fold_compound`. See also [Python Examples - Boltzmann Sampling](#)

**Global `vrna_pbacktrack_sub_cb` (`vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int start, unsigned int end, vrna_boltzmann_sampling_callback *cb, void *data, unsigned int options`)**

This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

**Global `vrna_pbacktrack_sub_num` (`vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int start, unsigned int end, unsigned int options`)**

This function is attached as overloaded method `pbacktrack_sub()` to objects of type `fold_compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

**Global `vrna_pbacktrack_sub_resume` (`vrna_fold_compound_t *vc, unsigned int num_samples, unsigned int start, unsigned int end, vrna_pbacktrack_mem_t *nr_mem, unsigned int options`)**

This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound`. In addition to the list of structures, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

**Global `vrna_pbacktrack_sub_resume_cb` (`vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int start, unsigned int end, vrna_boltzmann_sampling_callback *bs_cb, void *data, vrna_pbacktrack_mem_t *nr_mem, unsigned int options`)**

This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound`. In addition to the number of structures backtraced, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

**Global `vrna_pf` (`vrna_fold_compound_t *vc, char *structure`)**

This function is attached as method `pf()` to objects of type `fold_compound`

**Global `vrna_pf_dimer` (`vrna_fold_compound_t *vc, char *structure`)**

This function is attached as method `pf_dimer()` to objects of type `fold_compound`

**Global `vrna_positional_entropy` (`vrna_fold_compound_t *fc`)**

This function is attached as method `positional_entropy()` to objects of type `fold_compound`

**Global `vrna_pr_energy` (`vrna_fold_compound_t *vc, double e`)**

This function is attached as method `pr_energy()` to objects of type `fold_compound`

**Global `vrna_pr_structure` (`vrna_fold_compound_t *fc, const char *structure`)**

This function is attached as method `pr_structure()` to objects of type `fold_compound`

**Global `vrna_ptable` (`const char *structure`)**

This function is wrapped as overloaded function `ptable()` that takes an optional argument `options` to specify which type of matching brackets should be considered during conversion. The default set is round brackets, i.e. `VRNA_BRACKETS_RND`.

**Global `vrna_ptable_from_string` (`const char *structure, unsigned int options`)**

This function is wrapped as overloaded function `ptable()` that takes an optional argument `options` to specify which type of matching brackets should be considered during conversion. The default set is round brackets, i.e. `VRNA_BRACKETS_RND`.

**Global `vrna_rotational_symmetry` (`const char *string`)**

This function is available as global function `rotational_symmetry()`. See `vrna_rotational_symmetry_pos()` for details.

**Global `vrna_rotational_symmetry_db` (`vrna_fold_compound_t *fc, const char *structure`)**

This function is attached as method `rotational_symmetry_db()` to objects of type `fold_compound` (i.e. `vrna_fold_compound_t`). See `vrna_rotational_symmetry_db_pos()` for details.

**Global `vrna_rotational_symmetry_db_pos` (`vrna_fold_compound_t *fc, const char *structure, unsigned int **positions`)**

This function is attached as method `rotational_symmetry_db()` to objects of type `fold_compound` (i.e. `vrna_fold_compound_t`). Thus, the first argument must be omitted. In contrast to our C-implementation, this function doesn't simply return the order of rotational symmetry of the secondary structure, but returns the list `position` of cyclic permutation shifts that result in a rotationally symmetric structure. The length of the list then determines the order of rotational symmetry.

**Global `vrna_rotational_symmetry_num` (`const unsigned int *string, size_t string_length`)**

This function is available as global function `rotational_symmetry()`. See `vrna_rotational_symmetry_pos()` for details. Note, that in the target language the length of the list `string` is always known a-priori, so the parameter `string_length` must be omitted.

**Global `vrna_rotational_symmetry_pos` (`const char *string, unsigned int **positions`)**

This function is available as overloaded global function `rotational_symmetry()`. It merges the functionalities of `vrna_rotational_symmetry()`, `vrna_rotational_symmetry_pos()`, `vrna_rotational_symmetry_num()`, and `vrna_rotational_symmetry_pos_num()`. In contrast to our C-implementation, this function doesn't return the order of rotational symmetry as a single value, but returns a list of cyclic permutation shifts that result in a rotationally symmetric string. The length of the list then determines the order of rotational symmetry.

**Global `vrna_rotational_symmetry_pos_num` (`const unsigned int *string, size_t string_length, unsigned int **positions`)**

This function is available as global function `rotational_symmetry()`. See `vrna_rotational_symmetry_pos()` for details. Note, that in the target language the length of the list `string` is always known a-priori, so the parameter `string_length` must be omitted.

**Global `vrna_sc_add_bp` (`vrna_fold_compound_t *vc, int i, int j, FLT_OR_DBL energy, unsigned int options`)**

This function is attached as an overloaded method `sc_add_bp()` to objects of type `fold_compound`. The method either takes arguments for a single base pair (i,j) with the corresponding energy value:

**Global `vrna_sc_add_bt` (`vrna_fold_compound_t *vc, vrna_callback_sc_backtrack *f`)**

This function is attached as method `sc_add_bt()` to objects of type `fold_compound`

**Global `vrna_sc_add_data` (`vrna_fold_compound_t *vc, void *data, vrna_callback_free_auxdata *free_data`)**

This function is attached as method `sc_add_data()` to objects of type `fold_compound`

**Global `vrna_sc_add_exp_f` (`vrna_fold_compound_t *vc, vrna_callback_sc_exp_energy *exp_f`)**

This function is attached as method `sc_add_exp_f()` to objects of type `fold_compound`

**Global `vrna_sc_add_f` (`vrna_fold_compound_t *vc, vrna_callback_sc_energy *f`)**

This function is attached as method `sc_add_f()` to objects of type `fold_compound`

**Global `vrna_sc_add_hi_motif` (`vrna_fold_compound_t *fc, const char *seq, const char *structure, FLT_OR_DBL energy, unsigned int options`)**

This function is attached as method `sc_add_hi_motif()` to objects of type `fold_compound`

**Global `vrna_sc_add_SHAPE_deigan` (`vrna_fold_compound_t *vc, const double *reactivities, double m, double b, unsigned int options`)**

This function is attached as method `sc_add_SHAPE_deigan()` to objects of type `fold_compound`

**Global `vrna_sc_add_SHAPE_deigan_ali` (`vrna_fold_compound_t *vc, const char **shape_files, const int *shape_file_association, double m, double b, unsigned int options`)**

This function is attached as method `sc_add_SHAPE_deigan_ali()` to objects of type `fold_compound`

**Global `vrna_sc_add_SHAPE_zarringhalam` (`vrna_fold_compound_t *vc, const double *reactivities, double b, double default_value, const char *shape_conversion, unsigned int options`)**

This function is attached as method `sc_add_SHAPE_zarringhalam()` to objects of type `fold_compound`

**Global `vrna_sc_add_up` (`vrna_fold_compound_t *vc, int i, FLT_OR_DBL energy, unsigned int options`)**

This function is attached as an overloaded method `sc_add_up()` to objects of type `fold_compound`. The method either takes arguments for a single nucleotide *i* with the corresponding energy value:

**Global `vrna_sc_init` (`vrna_fold_compound_t *vc`)**

This function is attached as method `sc_init()` to objects of type `fold_compound`

**Global `vrna_sc_remove` (`vrna_fold_compound_t *vc`)**

This function is attached as method `sc_remove()` to objects of type `fold_compound`

**Global `vrna_sc_set_bp` (`vrna_fold_compound_t *vc, const FLT_OR_DBL **constraints, unsigned int options`)**

This function is attached as method `sc_set_bp()` to objects of type `fold_compound`

**Global `vrna_sc_set_up` (`vrna_fold_compound_t *vc, const FLT_OR_DBL *constraints, unsigned int options`)**

This function is attached as method `sc_set_up()` to objects of type `fold_compound`

**Global `vrna_seq_encode` (`const char *sequence, vrna_md_t *md`)**

In the target scripting language, this function is wrapped as overloaded function `seq_encode()` where the last parameter, the `model_details` data structure, is optional. If it is omitted, default model settings are applied, i.e. default nucleotide letter conversion. The wrapped function returns a list/tuple of integer representations of the input sequence.

**Global `vrna_strtrim` (`char *string, const char *delimiters, unsigned int keep, unsigned int options`)**

Since many scripting languages treat strings as immutable objects, this function does not modify the input string directly. Instead, it returns the modified string as second return value, together with the number of removed delimiters.

The scripting language interface provides an overloaded version of this function, with default parameters `delimiters=NULL`, `keep=0`, and `options=VRNA_TRIM_DEFAULT`.

**Global `vrna_subopt` (`vrna_fold_compound_t *fc, int delta, int sorted, FILE *fp`)**

This function is attached as method `subopt()` to objects of type `fold_compound`

**Global `vrna_subopt_cb` (`vrna_fold_compound_t *fc, int delta, vrna_subopt_callback *cb, void *data`)**

This function is attached as method `subopt_cb()` to objects of type `fold_compound`

**Global `vrna_subopt_zuker` (`vrna_fold_compound_t *fc`)**

This function is attached as method `subopt_zuker()` to objects of type `fold_compound`

**Global `vrna_ud_remove` (`vrna_fold_compound_t *vc`)**

This function is attached as method `ud_remove()` to objects of type `fold_compound`

**Global `vrna_ud_set_data` (`vrna_fold_compound_t *vc, void *data, vrna_callback_free_auxdata *free_cb`)**

This function is attached as method `ud_set_data()` to objects of type `fold_compound`

**Global `vrna_ud_set_exp_prod_rule_cb` (`vrna_fold_compound_t *vc, vrna_callback_ud_exp_production *pre_cb, vrna_callback_ud_exp_energy *exp_e_cb`)**

This function is attached as method `ud_set_exp_prod_rule_cb()` to objects of type `fold_compound`

**Global `vrna_ud_set_prob_cb` (`vrna_fold_compound_t *vc, vrna_callback_ud_probs_add *setter, vrna_callback_ud_probs_get *getter`)**

This function is attached as method `ud_set_prob_cb()` to objects of type `fold_compound`

Global `vrna_ud_set_prod_rule_cb` (`vrna_fold_compound_t *vc, vrna_callback_ud_production *pre_cb, vrna_callback_ud_energy *e_cb`)

This function is attached as method `ud_set_prod_rule_cb()` to objects of type *fold\_compound*



## **Chapter 7**

# **Additional Utilities**



# Chapter 8

## Examples

- [C Examples](#)
- [Perl5 Examples](#)
- [Python Examples](#)

### 8.1 C Examples

#### 8.1.1 Hello World Examples

##### **helloworld\_mfe.c**

The following is an example showing the minimal requirements to compute the Minimum Free Energy (MFE) and corresponding secondary structure of an RNA sequence

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ViennaRNA/fold.h>
#include <ViennaRNA/utils/basic.h>
int
main()
{
    /* The RNA sequence */
    char *seq = "GAGUAGUGGAACCAGGCCAUGUUUGUGACUCGCAGACUAACA";
    /* allocate memory for MFE structure (length + 1) */
    char *structure = (char *)vrna_alloc(sizeof(char) * (strlen(seq) + 1));
    /* predict Minimum Free Energy and corresponding secondary structure */
    float mfe = vrna_fold(seq, structure);
    /* print sequence, structure and MFE */
    printf("%s\n%s [ %6.2f ]\n", seq, structure, mfe);
    /* cleanup memory */
    free(structure);
    return 0;
}
```

##### See also

[examples/helloworld\\_mfe.c](#) in the source code tarball

##### **helloworld\_mfe\_comparative.c**

Instead of using a single sequence as done above, this example predicts a consensus structure for a multiple sequence alignment

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ViennaRNA/alifold.h>
#include <ViennaRNA/utils/basic.h>
#include <ViennaRNA/utils/alignments.h>
int
main()
{
    /* The RNA sequence alignment */
    const char *sequences[] = {
        "CUGCCUCACAAACGUUUGUGCCUCAGUUACCCGUAGAUGUAGUGAGGGU",
        "CUGCCUCACAAACAUUUGUGCCUCAGUUACUAGAUGUAGUGAGGGU",
    }
```

```

"---CUCGACACCACU---GCCUCGGGUUACCAUCGGUGGCAGUGCGGGU",
NULL /* indicates end of alignment */
};

/* compute the consensus sequence */
char *consensus(sequences);
/* allocate memory for MFE consensus structure (length + 1) */
char *structure = (char *)vrna_alloc(sizeof(char) * (strlen(sequences[0]) + 1));
/* predict Minimum Free Energy and corresponding secondary structure */
float mfe = vrna_alifold(sequences, structure);
/* print consensus sequence, structure and MFE */
printf("%s\n%s [ %6.2f ]\n", cons, structure, mfe);
/* cleanup memory */
free(cons);
free(structure);
return 0;
}

```

**See also**

`examples/helloworld_mfe_comparative.c` in the source code tarball

**helloworld\_probabilities.c**

This example shows how to compute the partition function and base pair probabilities with minimal implementation effort.

```

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ViennaRNA/fold.h>
#include <ViennaRNA/part_func.h>
#include <ViennaRNA/utils/basic.h>
int
main()
{
    /* The RNA sequence */
    char *seq = "GAGUAGUGGAACCAGCUAUGUUGACUCGGAGACUAACA";
    /* allocate memory for pairing propensity string (length + 1) */
    char *propensity = (char *)vrna_alloc(sizeof(char) * (strlen(seq) + 1));
    /* pointers for storing and navigating through base pair probabilities */
    vrna_ep_t *ptr, *pair_probabilities = NULL;
    float en = vrna_pf_fold(seq, propensity, &pair_probabilities);
    /* print sequence, pairing propensity string and ensemble free energy */
    printf("%s\n%s [ %6.2f ]\n", seq, propensity, en);
    /* print all base pairs with probability above 50% */
    for (ptr = pair_probabilities; ptr->i != 0; ptr++)
        if (ptr->p > 0.5)
            printf("p(%d, %d) = %g\n", ptr->i, ptr->j, ptr->p);
    /* cleanup memory */
    free(pair_probabilities);
    free(propensity);
    return 0;
}

```

**See also**

`examples/helloworld_probabilities.c` in the source code tarball

**8.1.2 First Steps with the Fold Compound****fold\_compound\_mfe.c**

Instead of calling the simple MFE folding interface `vrna_fold()`, this example shows how to first create a `vrna_fold_compound_t` container with the RNA sequence to finally compute the MFE using this container. This is especially useful if non-default model settings are applied or the dynamic programming (DP) matrices of the MFE prediction are required for post-processing operations, or other tasks on the same sequence will be performed.

```

#include <stdlib.h>
#include <stdio.h>
#include <ViennaRNA/fold_compound.h>
#include <ViennaRNA/utils/basic.h>
#include <ViennaRNA/utils/strings.h>
#include <ViennaRNA/mfe.h>
int
main()
{
    /* initialize random number generator */
    vrna_init_rand();
    /* Generate a random sequence of 50 nucleotides */
    char *seq = vrna_random_string(50, "ACGU");
    /* Create a fold compound for the sequence */
    vrna_fold_compound_t *fc = vrna_fold_compound(seq, NULL, VRNA_OPTION_DEFAULT);

```

```

/* allocate memory for MFE structure (length + 1) */
char           *structure = (char *)vrna_alloc(sizeof(char) * (strlen(seq) + 1));
/* predict Minimum Free Energy and corresponding secondary structure */
float          mfe = vrna_mfe(fc, structure);
/* print sequence, structure and MFE */
printf("%s\n%s [ %6.2f ]\n", seq, structure, mfe);
/* cleanup memory */
free(seq);
free(structure);
vrna_fold_compound_free(fc);
return 0;
}

```

**See also**

`examples/fold_compound_mfe.c` in the source code tarball

**fold\_compound\_md.c**

In the following, we change the model settings (model details) to a temperature of 25 Degree Celcius, and activate G-Quadruplex precition.

```

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ViennaRNA/model.h>
#include <ViennaRNA/fold_compound.h>
#include <ViennaRNA/utils/basic.h>
#include <ViennaRNA/utils/strings.h>
#include <ViennaRNA/mfe.h>
int
main()
{
    /* initialize random number generator */
    vrna_init_rand();
    /* Generate a random sequence of 50 nucleotides */
    char     *seq = vrna_random_string(50, "ACGU");
    /* allocate memory for MFE structure (length + 1) */
    char     *structure = (char *)vrna_alloc(sizeof(char) * (strlen(seq) + 1));
    /* create a new model details structure to store the Model Settings */
    vrna_md_t md;
    /* ALWAYS set default model settings first! */
    vrna_md_set_default(&md);
    /* change temperature and activate G-Quadruplex prediction */
    md.temperature = 25.0; /* 25 Deg Celcius */
    md.gquad = 1; /* Turn-on G-Quadruples support */
    /* create a fold compound */
    vrna_fold_compound_t *fc = vrna_fold_compound(seq, &md, VRNA_OPTION_DEFAULT);
    /* predict Minimum Free Energy and corresponding secondary structure */
    float          mfe = vrna_mfe(fc, structure);
    /* print sequence, structure and MFE */
    printf("%s\n%s [ %6.2f ]\n", seq, structure, mfe);
    /* cleanup memory */
    free(structure);
    vrna_fold_compound_free(fc);
    return 0;
}

```

**See also**

`examples/fold_compound_md.c` in the source code tarball

**8.1.3 Writing Callback Functions****callback\_subopt.c**

Here is a basic example how to use the callback mechanism in `vrna_subopt_cb()`. It simply defines a callback function (see interface definition for `vrna_subopt_callback`) that prints the result and increases a counter variable.

```

#include <stdlib.h>
#include <stdio.h>
#include <ViennaRNA/fold_compound.h>
#include <ViennaRNA/utils/basic.h>
#include <ViennaRNA/utils/strings.h>
#include <ViennaRNA/subopt.h>
void
subopt_callback(const char  *structure,
               float      energy,
               void       *data)
{
    /* simply print the result and increase the counter variable by 1 */
    if (structure)
        printf("%d.\t%s\t%6.2f\n", ((int *)data))++, structure, energy);
}

```

```

}

int
main()
{
    /* initialize random number generator */
    vrna_init_rand();
    /* Generate a random sequence of 50 nucleotides */
    char           *seq = vrna_random_string(50, "ACGU");
    /* Create a fold compound for the sequence */
    vrna_fold_compound_t *fc = vrna_fold_compound(seq, NULL, VRNA_OPTION_DEFAULT);
    int             counter = 0;
    /*
     * call subopt to enumerate all secondary structures in an energy band of
     * 5 kcal/mol of the MFE and pass it the address of the callback and counter
     * variable
     */
    vrna_subopt_cb(fc, 500, &subopt_callback, (void *)&counter);
    /* cleanup memory */
    free(seq);
    vrna_fold_compound_free(fc);
    return 0;
}

```

**See also**

`examples/callback_subopt.c` in the source code tarball

### 8.1.4 Application of Soft Constraints

**soft\_constraints\_up.c**

In this example, a random RNA sequence is generated to predict its MFE under the constraint that a particular nucleotide receives an additional bonus energy if it remains unpaired.

```

#include <stdlib.h>
#include <stdio.h>
#include <ViennaRNA/fold_compound.h>
#include <ViennaRNA/utils/basic.h>
#include <ViennaRNA/utils/strings.h>
#include <ViennaRNA/constraints/soft.h>
#include <ViennaRNA/mfe.h>
int
main()
{
    /* initialize random number generator */
    vrna_init_rand();
    /* Generate a random sequence of 50 nucleotides */
    char           *seq = vrna_random_string(50, "ACGU");
    /* Create a fold compound for the sequence */
    vrna_fold_compound_t *fc = vrna_fold_compound(seq, NULL, VRNA_OPTION_DEFAULT);
    /* Add soft constraint of -1.7 kcal/mol to nucleotide 5 whenever it appears in an unpaired context */
    vrna_sc_add_up(fc, 5, -1.7, VRNA_OPTION_DEFAULT);
    /* allocate memory for MFE structure (length + 1) */
    char           *structure = (char *)vrna_alloc(sizeof(char) * 51);
    /* predict Minimum Free Energy and corresponding secondary structure */
    float          mfe = vrna_mfe(fc, structure);
    /* print sequence, structure and MFE */
    printf("%s\n%s [ %6.2f ]\n", seq, structure, mfe);
    /* cleanup memory */
    free(seq);
    free(structure);
    vrna_fold_compound_free(fc);
    return 0;
}

```

**See also**

`examples/soft_constraints_up.c` in the source code tarball

### 8.1.5 Other Examples

**example1.c**

A more extensive example including MFE, Partition Function, and Centroid structure prediction.

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ViennaRNA/data_structures.h>
#include <ViennaRNA/params/basic.h>
#include <ViennaRNA/utils/basic.h>
#include <ViennaRNA/eval.h>
#include <ViennaRNA/fold.h>

```

```
#include <ViennaRNA/part_func.h>
int
main(int argc,
     char *argv[])
{
    char           *seq =
        "AGACGACAAGGUUGAAUCGCCACAGCUAUGAGUCGGUGACAACAUUACGAAAGGCUGUAAAUAUUAUCACCACAGGGGGCCCCGUGUCUAG";
    char           *mfe_structure = vrna_alloc(sizeof(char) * (strlen(seq) + 1));
    char           *prob_string = vrna_alloc(sizeof(char) * (strlen(seq) + 1));
    /* get a vrna_fold_compound with default settings */
    vrna_fold_compound_t *vc = vrna_fold_compound(seq, NULL, VRNA_OPTION_DEFAULT);
    /* call MFE function */
    double         mfe = (double)vrna_mfe(vc, mfe_structure);
    printf("%s\n% (%.2f)\n", seq, mfe_structure, mfe);
    /* rescale parameters for Boltzmann factors */
    vrna_exp_params_rescale(vc, &mfe);
    /* call PF function */
    FLT_OR_DBL en = vrna_pf(vc, prob_string);
    /* print probability string and free energy of ensemble */
    printf("%s (%.2f)\n", prob_string, en);
    /* compute centroid structure */
    double         dist;
    char           *cent = vrna_centroid(vc, &dist);
    /* print centroid structure, its free energy and mean distance to the ensemble */
    printf("%s (%.2f d=%6.2f)\n", cent, vrna_eval_structure(vc, cent), dist);
    /* free centroid structure */
    free(cent);
    /* free pseudo dot-bracket probability string */
    free(prob_string);
    /* free mfe structure */
    free(mfe_structure);
    /* free memory occupied by vrna_fold_compound */
    vrna_fold_compound_free(vc);
    return EXIT_SUCCESS;
}
```

**See also**

[examples/example1.c](#) in the source code tarball

### 8.1.6 Deprecated Examples

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "utils.h"
#include "fold_vars.h"
#include "fold.h"
#include "part_func.h"
#include "inverse.h"
#include "RNAAstruct.h"
#include "treedist.h"
#include "stringdist.h"
#include "profiledist.h"
void
main()
{
    char           *seq1 = "CGCAGGGAUACCCGCG", *seq2 = "GCGCCCAUAGGGACGC",
                 *struct1, *struct2, *xstruc;
    float          e1, e2, tree_dist, string_dist, profile_dist, kT;
    Tree           *T1, *T2;
    swString       *S1, *S2;
    float          *pf1, *pf2;
    FLT_OR_DBL    *bppm;
    /* fold at 30C instead of the default 37C */
    temperature = 30.;           /* must be set *before* initializing */
    /* allocate memory for structure and fold */
    struct1 = (char *)space(sizeof(char) * (strlen(seq1) + 1));
    e1      = fold(seq1, struct1);
    struct2 = (char *)space(sizeof(char) * (strlen(seq2) + 1));
    e2      = fold(seq2, struct2);
    free_arrays();               /* free arrays used in fold() */
    /* produce tree and string representations for comparison */
    xstruc   = expand_Full(struct1);
    T1       = make_tree(xstruc);
    S1       = Make_swString(xstruc);
    free(xstruc);
    xstruc   = expand_Full(struct2);
    T2       = make_tree(xstruc);
    S2       = Make_swString(xstruc);
    free(xstruc);
    /* calculate tree edit distance and aligned structures with gaps */
    edit_backtrack = 1;
    tree_dist   = tree_edit_distance(T1, T2);
```

```

free_tree(T1);
free_tree(T2);
unexpand_aligned_F(aligned_line);
printf("%s\n%3.2f\n", aligned_line[0], aligned_line[1], tree_dist);
/* same thing using string edit (alignment) distance */
string_dist = string_edit_distance(S1, S2);
free(S1);
free(S2);
printf("%s mfe=%5.2f\n%s mfe=%5.2f dist=%3.2f\n",
       aligned_line[0], e1, aligned_line[1], e2, string_dist);
/* for longer sequences one should also set a scaling factor for
 * partition function folding, e.g. */
kT      = (temperature + 273.15) * 1.98717 / 1000.; /* kT in kcal/mol */
pf_scale = exp(-e1 / kT / strlen(seq1));
/* calculate partition function and base pair probabilities */
e1 = pf_fold(seq1, struct1);
/* get the base pair probability matrix for the previous run of pf_fold() */
bppm = export_bppm();
pf1 = Make_bp_profile_bppm(bppm, strlen(seq1));
e2 = pf_fold(seq2, struct2);
/* get the base pair probability matrix for the previous run of pf_fold() */
bppm = export_bppm();
pf2 = Make_bp_profile_bppm(bppm, strlen(seq2));
free_pf_arrays(); /* free space allocated for pf_fold() */
profile_dist = profile_edit_distance(pf1, pf2);
printf("%s free energy=%5.2f\n%s free energy=%5.2f dist=%3.2f\n",
       aligned_line[0], e1, aligned_line[1], e2, profile_dist);
free_profile(pf1);
free_profile(pf2);
}

```

#### See also

`examples/example_old.c` in the source code tarball

## 8.2 Perl5 Examples

### Hello World Examples

#### Using the flat interface

- MFE prediction

```

use RNA;
# The RNA sequence
my $seq = "GAGUAGUGGAACCAGGCCUAUGUUUGUGACUCGCAGACUAACA";
# compute minimum free energy (MFE) and corresponding structure
my ($ss, $mfe) = RNA::fold($seq);
# print output
printf "%s\n%6.2f\n", $seq, $ss, $mfe;

```

- comparative MFE prediction for sequence alignments

```

use RNA;
# The RNA sequence alignment
my @sequences = (
    "CUGCCUCACGUUUGUGCCUCAGUUACCCGUAGAUGUAGUGAGGGU",
    "CUGCCUCACACAUUUGUGCCUCAGUUACUCAUAGAUGUAGUGAGGGU",
    "---CUCGACACCACU---GCCUCGGUUACCCAUCGGUGCAGUGCAGGGU"
);
# compute the consensus sequence
my $cons = RNA::consensus(@sequences);
# predict Minimum Free Energy and corresponding secondary structure
my ($ss, $mfe) = RNA::alifold(@sequences);
# print output
printf "%s\n%6.2f\n", $cons, $ss, $mfe;

```

#### Using the object oriented interface

- MFE prediction

```

#!/usr/bin/perl
use warnings;
use strict;
use RNA;
my $seq1 = "CGCAGGGAUACCCGCG";
# create new fold_compound object
my $fc = new RNA::fold_compound($seq1);
# compute minimum free energy (mfe) and corresponding structure
my ($ss, $mfe) = $fc->mfe();
# print output
printf "%s\n%6.2f\n", $ss, $mfe;

```

## Changing the Model Settings

### Using the object oriented interface

- MFE prediction at different temperature and dangle model

```
use RNA;
# The RNA sequence
my $seq = "GAGUAGUGGAACCAGGCCUAUGUUUGUGACUCGCAGACUAACA";
# create a new model details structure
my $md = new RNA::md();
# change temperature and dangle model
$md->{temperature} = 20.0; # 20 Deg Celcius
$md->{dangles} = 1; # Dangle Model 1
# create a fold compound
my $fc = new RNA::fold_compound($seq, $md);
# predict Minimum Free Energy and corresponding secondary structure
my ($ss, $mfe) = $fc->mfe();
# print sequence, structure and MFE
printf "%s\n%s [ %6.2f ]\n", $seq, $ss, $mfe;
```

## 8.3 Python Examples

### MFE Prediction (flat interface)

```
import RNA
# The RNA sequence
seq = "GAGUAGUGGAACCAGGCCUAUGUUUGUGACUCGCAGACUAACA"
# compute minimum free energy (MFE) and corresponding structure
(ss, mfe) = RNA.fold(seq)
# print output
print("{}\n{} [ {:.6.2f} ]".format(seq, ss, mfe))
```

### MFE Prediction (object oriented interface)

```
import RNA;
sequence = "CGCAGGGAUACCCGCG"
# create new fold_compound object
fc = RNA.fold_compound(sequence)
# compute minimum free energy (mfe) and corresponding structure
(ss, mfe) = fc.mfe()
# print output
print("{} [ {:.6.2f} ]".format(ss, mfe))
```

### Suboptimal Structure Prediction

```
import RNA
sequence = "GGGGAAAACCCC"
# Set global switch for unique ML decomposition
RNA.cvar.uniq_ML = 1
subopt_data = { 'counter' : 1, 'sequence' : sequence }
# Print a subopt result as FASTA record
def print_subopt_result(structure, energy, data):
    if not structure == None:
        print(">subopt {:.d}\n".format(data['counter']))
        print("{}\n{} [ {:.6.2f} ]\n".format(data['sequence'], structure, energy))
        # increase structure counter
        data['counter'] = data['counter'] + 1
# Create a 'fold_compound' for our sequence
a = RNA.fold_compound(sequence)
# Enumerate all structures 500 dacal/mol = 5 kcal/mol arround
# the the MFE and print each structure using the function above
a.subopt_cb(500, print_subopt_result, subopt_data);
```

### Boltzmann Sampling (a.k.a. Probabilistic Backtracing)

```
import RNA
sequence =
"UGGGAAUAGUCUUUCGAGUCUCGCGACGGCGAUCUUCGAAAGUGGAAUCGUACUUUACCGCCUGUGCGGACUACUAUCCUGACCACAUAGU"
def store_structure(s, data):
    """
    A simple callback function that stores
    a structure sample into a list
    """
    if s:
        data.append(s)
"""
First we prepare a fold_compound object
"""


```

```

# create model details
md = RNA.md()
# activate unique multibranch loop decomposition
md.uniq_ML = 1
# create fold compound object
fc = RNA.fold_compound(sequence, md)
# compute MFE
(ss, mfe) = fc.mfe()
# rescale Boltzmann factors according to MFE
fc.exp_params_rescale(mfe)
# compute partition function to fill DP matrices
fc.pf()
"""
Now we are ready to perform Boltzmann sampling
"""
# 1. backtrace a single sub-structure of length 10
print("{}".format(fc.pbacktrack5(10)))
# 2. backtrace a single sub-structure of length 50
print("{}".format(fc.pbacktrack5(50)))
# 3. backtrace multiple sub-structures of length 10 at once
for s in fc.pbacktrack5(20, 10):
    print("{}".format(s))
# 4. backtrace multiple sub-structures of length 50 at once
for s in fc.pbacktrack5(100, 50):
    print("{}".format(s))
# 5. backtrace a single structure (full length)
print("{}".format(fc.pbacktrack()))
# 6. backtrace multiple structures at once
for s in fc.pbacktrack(100):
    print("{}".format(s))
# 7. backtrace multiple structures non-redundantly
for s in fc.pbacktrack(100, RNA.PBACKTRACK_NON_REDUNDANT):
    print("{}".format(s))
# 8. backtrace multiple structures non-redundantly (with resume option)
num_samples = 500
iterations = 15
d = None # pbacktrack memory object
s_list = []
for i in range(0, iterations):
    d, ss = fc.pbacktrack(num_samples, d, RNA.PBACKTRACK_NON_REDUNDANT)
    s_list = s_list + list(ss)
for s in s_list:
    print("{}".format(s))
# 9. backtrace multiple sub-structures of length 50 in callback mode
ss = []
i = fc.pbacktrack5(100, 50, store_structure, ss)
for s in ss:
    print("{}".format(s))
# 10. backtrace multiple full-length structures in callback mode
ss = list()
i = fc.pbacktrack(100, store_structure, ss)
for s in ss:
    print("{}".format(s))
# 11. non-redundantly backtrace multiple full-length structures in callback mode
ss = list()
i = fc.pbacktrack(100, store_structure, ss, RNA.PBACKTRACK_NON_REDUNDANT)
for s in ss:
    print("{}".format(s))
# 12. non-redundantly backtrace multiple full length structures
# in callback mode with resume option
ss = []
d = None # pbacktrack memory object
for i in range(0, iterations):
    d, i = fc.pbacktrack(num_samples, store_structure, ss, d, RNA.PBACKTRACK_NON_REDUNDANT)
for s in ss:
    print("{}".format(s))
# 13. backtrace a single substructure from the sequence interval [10:50]
print("{}".format(fc.pbacktrack_sub(10, 50)))
# 14. backtrace multiple substructures from the sequence interval [10:50]
for s in fc.pbacktrack_sub(100, 10, 50):
    print("{}".format(s))
# 15. backtrace multiple substructures from the sequence interval [10:50] non-redundantly
for s in fc.pbacktrack_sub(100, 10, 50, RNA.PBACKTRACK_NON_REDUNDANT):
    print("{}".format(s))

```

## RNAfold -p –MEA equivalent

```

#!/usr/bin/python
#
import RNA
seq = "AUUUCACUAGAGAAGGUCUAGAGGUUUGUCGUUGUCAGAACGUCCUAUCCAGGUACGAACACGGUGGUAUGUUCGACGACAGGAUCGGCGCACUA"
# create fold_compound data structure (required for all subsequently applied algorithms)
fc = RNA.fold_compound(seq)
# compute MFE and MFE structure
(mfe_struct, mfe) = fc.mfe()
# rescale Boltzmann factors for partition function computation

```

```

fc.exp_params_rescale(mfe)
# compute partition function
(pp, pf) = fc.pf()
# compute centroid structure
(centroid_struct, dist) = fc.centroid()
# compute free energy of centroid structure
centroid_en = fc.eval_structure(centroid_struct)
# compute MEA structure
(MEA_struct, MEA) = fc.MEA()
# compute free energy of MEA structure
MEA_en = fc.eval_structure(MEA_struct)
# print everything like RNAfold -p --MEA
print("{}\n{} ({:.6.2f})".format(seq, mfe_struct, mfe))
print("{} [{:.6.2f}]".format(pp, pf))
print("{} {{ {:.6.2f} d={:.2f}}}".format(centroid_struct, centroid_en, dist))
print("{} {{ {:.6.2f} MEA={:.2f}}}".format(MEA_struct, MEA_en, MEA))
print(" frequency of mfe structure in ensemble {:g}; ensemble diversity
      {:.6.2f}".format(fc.pr_structure(mfe_struct), fc.mean_bp_distance()))

```

## Fun with Soft Constraints

```

import RNA
seq1 = "CUCGUCGCCUUAUCCAGUGCGGGCGCUAGACAUUAGUUAUCGCCGCAA"
# Turn-off dangles globally
RNA.cvar.dangles = 0
# Data structure that will be passed to our MaximumMatching() callback with two components:
# 1. a 'dummy' fold_compound to evaluate loop energies w/o constraints, 2. a fresh set of energy parameters
mm_data = { 'dummy': RNA.fold_compound(seq1), 'params': RNA.param() }
# Nearest Neighbor Parameter reversal functions
revert_NN = {
    RNA.DECOMP_PAIR_HP: lambda i, j, k, l, f, p: - f.eval_hp_loop(i, j) - 100,
    RNA.DECOMP_PAIR_IL: lambda i, j, k, l, f, p: - f.eval_int_loop(i, j, k, l) - 100,
    RNA.DECOMP_PAIR_ML: lambda i, j, k, l, f, p: - p.MLclosing - p.MLintern[0] - (j - i - k + l - 2) * p.MLbase - 100,
    RNA.DECOMP_ML_ML_STEM: lambda i, j, k, l, f, p: - p.MLintern[0] - (l - k - 1) * p.MLbase,
    RNA.DECOMP_ML_STEM: lambda i, j, k, l, f, p: - p.MLintern[0] - (j - i - k + 1) * p.MLbase,
    RNA.DECOMP_ML_ML: lambda i, j, k, l, f, p: - (j - i - k + 1) * p.MLbase,
    RNA.DECOMP_ML_ML_ML: lambda i, j, k, l, f, p: 0,
    RNA.DECOMP_ML_UP: lambda i, j, k, l, f, p: - (j - i + 1) * p.MLbase,
    RNA.DECOMP_EXT_STEM: lambda i, j, k, l, f, p: - f.eval_ext_stem(k, l),
    RNA.DECOMP_EXT_EXT: lambda i, j, k, l, f, p: 0,
    RNA.DECOMP_EXT_STEM_EXT: lambda i, j, k, l, f, p: - f.eval_ext_stem(i, k),
    RNA.DECOMP_EXT_EXT_STEM: lambda i, j, k, l, f, p: - f.eval_ext_stem(l, j),
}
# Maximum Matching callback function (will be called by RNALib in each decomposition step)
def MaximumMatching(i, j, k, l, d, data):
    return revert_NN[d](i, j, k, l, data['dummy'], data['params'])
# Create a 'fold_compound' for our sequence
fc = RNA.fold_compound(seq1)
# Add maximum matching soft-constraints
fc.sc_add_f(MaximumMatching)
fc.sc_add_data(mm_data, None)
# Call MFE algorithm
(s, mm) = fc.mfe()
# print result
print("{}\n{} (MM: {:d})".format(seq1, s, int(-mm)))

```



## Chapter 9

# Contributing to the ViennaRNA Package

## Contents

- General Remarks
- Reporting Bugs
- Pull Request Process
- Contributors License Agreement (CLA)

## General Remarks

The ViennaRNA Package is developed by humans and consequently may contain bugs that prevent proper operation of the implemented algorithms. If you think you have found any of those nasty animals, please help us to improve our software by reporting the bug to us.

The ViennaRNA Package also is open-source software, which means that everybody can have a closer look into our implementations to understand and potentially extend its functionality. If you implemented any novel feature into the ViennaRNA Package that might be of interest to a larger community, please don't hesitate to ask for merging of your code into our official source tree. See the Pull Request Process section below to find information on how to do that.

Please note that we have a code of conduct. Please follow it in all your interactions with this project.

If you wish to contribute to this project, please first discuss any proposed changes with the owners and main developers. You may do that either through making an issue at [our official GitHub presence](#), [by email](#), or any other personal communication with the core developer team.

More importantly, if you wish to contribute any files or software, you need to agree to our ViennaRNA Package Contributors License Agreement (CLA)! Otherwise, your contributions can't be merged into our source tree. See below for further information and the full CLA details.

## Reporting Bugs

1. Please make an issue at GitHub or notify us by emailing to [rna@tbi.univie.ac.at](mailto:rna@tbi.univie.ac.at)
2. In your report, include as much information as possible, such that we are able to reproduce it. If possible, find a minimal example that triggers the bug.
3. Include the version number for the ViennaRNA Package you experience the bug with.
4. Include at least some minimal information regarding your operating system (Linux, Mac OS X, Windows, etc.)

## Pull Request Process

1. Ensure that you have not checked-in any files that are automatically build!

2. When contributing C source code, follow our code formatting guide lines. You may use the tool `uncrustify` together with our config located in `misc/uncrustify.cfg` to accomplish that.
3. Only expose symbols (functions, variables, etc.) to the libraries interface that are absolutely necessary! Hide all other symbols in the corresponding object file(s) by declaring them as `static`.
4. Use the prefixes `vrna_` for any symbol you add to the API of our library! Preprocessor macros in header files require the prefix in capital letters, i.e. `VRNA_`.
5. Use C-style comments at any place necessary to make sure your implementation can still be understood and followed in the future.
6. Add test cases for any new implementation! The test suite is located in the `tests` directory and is split into tests for the C-library, executable programs, and the individual scripting language interfaces.
7. Run `make check` to ensure that all other test suites still run properly with your applied changes!
8. When contributing via GitHub, make a personal fork of our project and create a separate branch for your changes. Then make a pull request to our `user-contrib` branch. Pull requests to the `master` branch will be rejected to keep its history clean.
9. Pull requests that have been successfully merged into the `user-contrib` branch usually find their way into the next release of the ViennaRNA Package. However, please note that the core developers may decide to include your changes in a later version.

## Contributors License Agreement

Thank you for your interest in contributing to the ViennaRNA Package ("We" or "Us").

Before contributing, please note that we adopted a standard Contributors License Agreement (CLA) agreement provided by [Project Harmony](#), a community-centered group focused on contributor agreements for free and open source software (FOSS).

This contributor agreement ("Agreement") documents the rights granted by contributors to Us. To make this document effective, please sign it and send it to Us by email to [rna@tbi.univie.ac.at](mailto:rna@tbi.univie.ac.at).

The respective CLA PDF documents are available in the [doc/CLA directory](#) of the distribution tarball, and online at our [official ViennaRNA Website](#).

# Chapter 10

## Changelog

Below, you'll find a list of notable changes for each version of the ViennaRNA Package.

### Unreleased

#### Version 2.5.x

##### Version 2.5.1 (Release date: 2022-06-02)

###### Programs

- Refactor `ct2db` program to allow for pseudoknots in output structure

###### Library

- API: Fix MEA computation for G-quadruplex predictions
- API: Fix memory leak in hard constraints container
- API: Fix RNApuzzler edge-case that resulted in segmentation faults
- API: Fix invalid memory access in `vrna_strjoin()`
- API: Revisit generic soft constraints for sliding-window base pair probability computations
- API: Enable to overwrite automatic unpaired probability determination in MEA computation
- API: Add `#VRNA_PLIST_TYPE_UNPAIRED` and `#VRNA_PLIST_TYPE_TRIPLE` identifiers for `vrna_ep_t`
- API: Add `vrna_init_rand_seed()` to initialize RNG with seed
- API: Add `vrna_zsc_compute_raw()` to obtain mean and sd for Z-score computation
- API: Add `vrna_file_connect_read_record()` function to parse connectivity table (\*.ct) files
- API: Add `vrna_strtrim()` function
- API: Update sanity checks for input in `vrna_pbacktrack_sub*`()
- API: Allow for pseudo-knots in `vrna_db_from_ptable()`
- API: Do not use `min_loop_size = 0` for multi strand interaction prediction
- API: Remove unnecessary uses of `min_loop_size` at multiple locations
- API: Deprecate cutpoint member of `vrna_fold_compound_t` and prepare for 5'/3' encoding
- API: Refactor sequence addition/preparation for `vrna_fold_compound_t`
- DOC: Update documentation

- SWIG: Add simple dot-plot file wrapper `plot_dp_EPS()`
- SWIG: Add `sequence`, `sequence_encoding` and `sequence_encoding2` attributes to `fold_compound` objects
- SWIG: Fix RNG wrapping and initialize RNG upon module load and update associated functions
- SWIG: Add more access to member variable arrays for various objects used throughout the library
- SWIG: Add memory efficient wrapper for dynamically allocated arrays and matrices
- SWIG: Shadow pair table data structure for efficient interactions between C and target languages
- SWIG: Expose hard constraints members in `fold_compound` objects
- SWIG: Add `exp_E_ext_stem()` method (`vrna_exp_E_ext_stem()`) to `fold_compound` objects
- SWIG: Expose DP matrices within `fold_compound` objects
- SWIG: Fix memory leak in wrapper for `vrna_db_from_ptable()`

## Package

- Update dlib to version 19.23
- DOC: Update doxygen.conf for version 1.9.2
- AUTOCONF: Factor-out Naview layout algorithm to allow for deactivating the Naview layout algorithm at configure-time
- AUTOCONF: Make LaTeX checks more portable and update LaTeX package checks
- AUTOCONF: Check whether we can build the swig interface when SVM support is deactivated
- AUTOCONF: Fix condition check for CLA build

## Version 2.5.0 (Release date: 2021-11-08)

### Programs

- Add `RNAmultifold` program to compute secondary structures for multiple interacting RNAs
- Add multistrand capabilities to `RNAeval`
- Add multistrand capabilities to `RNAsubopt`
- Replace `RNACofold` with a wrapper to `RNAmultifold`
- Fix computation of BB homodimer base pair probabilities in `RNACofold`

### Library

- API: Fix use of undefined values in deprecated function `PS_dot_plot()`
- API: Fix probability computations for unstructured domains within multibranch loops
- API: Fix index error in ensemble defect computations
- API: Fix hard constraints behavior on non-specific base pairing
- API: Fix segmentation fault for short input sequences in `vrna_hx_from_ptable()`
- API: Fix memory leak in static `rna_layout()` function
- API: Fix corner-case in covariance score computation on sequence alignments that determines which alignment columns may pair and which don't
- API: Add MFE computations for multiple interacting strands

- API: Add partition function computations for multiple interacting strands
- API: Add base pair probability computations for multiple interacting strands
- API: Add suboptimal structure prediction for multiple interacting strands
- API: Add multistrand capabilities to `vrna_eval*` functions
- API: Add new function `vrna_equilibrium_conc()` for concentration dependency computations of multiple interacting strands with `dlib` backend
- API: Add `vrna_equilibrium_constants()` function to obtain equilibrium constants for different complexes of multiple interacting strands
- API: Add function `vrna_pf_add()` to add ensemble free energies of two ensembles
- API: Add function `vrna_pf_substrands()` to get ensemble free energies for complexes up to a specific number of interacting strands
- API: Add function `vrna_n_multichoose_k()` to obtain a list of k-combinations with repetition
- API: Add `vrna_cstr_discard()` function to allow for discarding char streams prior to flushing
- API: Add `vrna_bp_distance_pt()` function to allow for base pair distance computation with pseudo-knots
- API: Add functions `vrna_pbacktrack_sub*` to allow for stochastic backtracing within arbitrary sequence intervals
- API: Add functions `vrna_boustrophedon()` and `vrna_boustrophedon_pos()` to generate lists of or obtain values from sequences of Boustrophedon distributed integer numbers
- API: Add `vrna_pscore()` and `vrna_pscore_freq()` functions to obtain covariance score for particular alignment columns
- API: Rewrite Zuker suboptimals implementation
- API: Remove old cofold implementations
- API: Make `type` attribute of `vrna_mx_mfe_t` and `vrna_mx_pf_t` a constant
- API: Guard more functions in `utils/structure_utils.c` against `NULL` input
- API: Rename `vrna_E_ext_loop()` to `vrna_eval_ext_stem()`
- API: Use v3 typedefs in dot-plot function declarations
- SWIG: Fix Python 3 file handle as optional argument in `eval*` functions and methods
- SWIG: Add wrapper for `vrna_pf_add()`
- SWIG: Add wrapper for `vrna_hx_from_ptable()`
- SWIG: Add wrapper for `vrna_db_from_probs()`

## Package

- Update `libsvm` to version 3.25
- Make Python 3.x the default Python for the scripting language interfaces
- Add Python3 capability for Mac OS X installer builds
- TESTS: Create TAP driver output for all unit tests (library, executables, SWIG interfaces)
- Remove compile-time switch to deactivate Boustrophedon backtracing scheme (this is the status quo now)
- Add Contributors License Agreement (CLA) to the Package in `doc/CLA/`

## Version 2.4.x

### Version 2.4.18 (Release date: 2021-04-22)

#### Programs

- Fix and refactor RNAPkplex program
- Fix occasional backtracing errors in RNALalifold
- Restrict available dangling end models in RNALalifold to 0 and 2
- Prevent segmentation faults upon bogus input data in RNAfold, RNAalifold, RNACofold, RNAheat, and RNAeval
- Free MFE DP matrices in RNAsubopt Boltzmann sampling when not required anymore

#### Library

- API: Add `vrna_abstract_shapes()` and `vrna_abstract_shapes_pt()` functions to convert secondary structures into their respective abstract shape notation ala Giegerich et al. 2004
- API: Add functions `vrna_seq_reverse()` and `vrna_DNA_complement()` to create reverse complements of a sequence
- API: Add more soft constraint handling to comparative structure prediction
- API: Add generic soft constraints for sliding window comparative MFE backtracing
- API: Add `vrna_ensemble_defect_pt()` that accepts pair table input instead of dot-bracket string to allow for non-nested reference structures
- API: Add failure/success return values to generic soft constraints application functions
- API: Refactor RNAPKplex implementation by better using constraints framework and moving out many parts from RNAPKplex.c into RNAlib as separate re-usable functions
- API: Fix energy contributions used in RNAPKplex implementations
- API: Fix energy evaluation for cofolding with dangle model 1
- API: Fix wrong arithmetic usage for PF variant of combined generic and simple soft constraints applied to external loops
- API: Fix memory size in `vrna_fold_compound_t` initialization
- API: Fix bogus memory access for comparative prediction when preparing hard constraints
- API: Fix wrong index usage in hard constraints for comparative base pair probability computations of internal loops
- API: Fix G-Quadruplex contributions as part of multibranch loops in single sequence base pair probability computations
- API: Fix multibranch loop MFE decomposition step for multiple strand cases
- API: Fix external loop generic hard constraint index updating for partition function computations
- API: Fix memory allocation for auxiliary grammar data structure
- API: Fix incorporation of auxiliary grammar contrib for closing pairs in sliding-window MFE computation
- API: Fix DP matrix initialization in sliding window MFE computations (fixes occasional backtracing issues in comparative sliding-window MFE computations)
- API: Make `vrna_sc_t.type` attribute a constant
- API: Remove upper-triangular hard constraint matrix in favor of full matrix

- API: Always ensure sane base pair span settings after `vrna_fold_compound_prepare()`
- API: Return INF on predictions of `vrna_mfe_dimer()` that fail due to unsatisfiable constraints
- API: Rename internally used hard and soft constraints API symbols
- API: Fix header file inclusions to prevent #include cycles
- SWIG: Add wrapper for `vrna_file_fasta_read_record()`
- SWIG: Fix memory leak in wrapper for `vrna_probs_window()`
- SWIG: Refactor and therefore fix soft constraint binding functions for use in comparative structure predictions
- SWIG: Fix typo that prevented properly wrapping `vrna_params_load_RNA_Andronescu2007()`
- SWIG: Unify wrappers for `vrna_ptable()` and `vrna_ptable_from_string()`

## Package

- REFMAN: Refactored structure annotation documentation
- REFMAN: Update Mac OS X install section
- Replace DEF placeholders in energy parameter files with their value of -50
- Update RNALocmin subpackage to properly compile with more stringent C++ compilers
- Update RNAforester subpackage to properly compile with more stringent C++ compilers
- Update autotools framework, e.g. checks for pthreads
- Update universal binary build instructions for Mac OS X builds to enable ARM compilation for M1 CPUs

## Version 2.4.17 (Release date: 2020-11-25)

### Programs

- Fix RNAUp -b mode with shorter sequence first
- Add --backtrack-global option to RNALfold (currently only available for dangles == 2 | 0)
- Add --zscores-pre-filter and --zscores-report-subsumed options to RNALfold

### Library

- API: Fix multiloop backtracing with soft constraints for unpaired positions in `vrna_subopt()` and `vrna_subopt_cb()`
- API: Fix parameter parse in `vrna_params_load_from_string()`
- API: Add `vrna_heat_capacity()` and `vrna_head_capacity_cb()` functions to RNAlib
- API: Add backtracing function `vrna_backtrack_window()` for global MFE structure to sliding-window predictions
- API: Add SVG support for RNApuzzler structure layouts
- API: Make `vrna_md_t` argument to `vrna_fold_compound()` a constant pointer
- API: Remove missing symbols from header file `ViennaRNA/params/default.h`
- API: Refactor z-score threshold filter handling for sliding-window MFE prediction
- SWIG: Fix typo in interface functions to load DNA parameters
- SWIG: Add python-3.9 autoconf checks

- SWIG: Add `vrna_head_capacity()` wrappers
- SWIG: Add access to raw energy parameters
- SWIG: Add alias and pair attribute to objects of type `md`
- SWIG: Add out/varout typemaps for 2-dimensional int-like arrays
- SWIG: Add all data fields to objects of type 'param' and 'exp\_param'

## Package

- Fix Debian and Windows installer files

## Version 2.4.16 (Release date: 2020-10-09)

### Programs

- Fix backtracing errors in `RNALalifold` for alignments with more than 32768 columns
- Fix backtracing errors in `RNAalifold` and `RNALalifold` for rare cases when two alignment columns may pair due to covariance score threshold but still yield infinite energies due to energy model
- Refactored manpages/help options for `RNAlpfold`, `RNAplot`, `RNApvmin`, `RNAsubopt`, and `RNAup`

### Library

- API: Fix undefined behavior due to short int overflows when accessing alignment lengths with alignments larger than 32768 columns. This fixes occasional backtracing errors in `RNALalifold` and `vrna_mfe_window()`
- API: Fix adding pscore to base pairs that yield INF energy in comparative global and local MFE prediction
- API: Add `vrna_convert_kcal_to_dcal()` and vice-versa function for safely converting integer to float representations of energy values
- SWIG: Add a reasonable Python interface for objects of type `vrna_path_t`
- SWIG: Add a wrapper for `vrna_seq_encode()`

## Package

- Move `units.h` include file to [ViennaRNA/utils/units.h](#)

## Version 2.4.15 (Release date: 2020-08-18)

### Programs

- Fix compilation of Kinfold with GCC 10
- Add `--en-only` flag to `RNAsubopt` to allow for sorting by energy only
- Prevent `RNAcofold` to process input with more than two strands
- Add cutpoint marker to dot-plots created with `RNAcofold -a`
- Update Kinfold to version 1.4

## Library

- API: Fix removal of strand delimiter in `vrna_plot_dp_PS_list()`
- API: Fix `vrna_enumerate_necklaces()`
- API: Fix bogus backtracing for co-folded structures in `vrna_subopt()` and `vrna_subopt_cb()`
- API: Fix storing co-folded structures for sorted output in `vrna_subopt()`
- API: Fix multibranch loop component hard constraints for multi-strand cases
- API: Prevent adding internal loop energy contributions to enclosed parts with `energy=INF`
- API: Adapt `vrna_db_pack()` / `vrna_db_unpack()` functions to produce comparable strings
- API: Add sorting modes `VRNA_UNSORTED`, `VRNA_SORT_BY_ENERGY_LEXICOGRAPHIC_ASC`, and `VRNA_SORT_BY_ENERGY_ASC` to `vrna_subopt()`
- API: Add `vrna_strjoin()` function
- API: Add missing case to external loop hard constraints
- API: Make hard constraints strand-aware
- SWIG: Fix invalid memory access when using `MEA_from plist()` in Perl 5 or Python
- SWIG: Enable keyword argument features in Python interface of constructors for `fold_compound`, `md`, `move`, `param`, and `exp_param` objects
- SWIG: Enable autodoc feature for Python interface of constructors for `fold_compound`, `md`, and `move` objects
- SWIG: Enable `toString` conversion for Python interface for objects of type `fold_compound`, `md`, `move`, `params`, `exp_params`, and `subopt_solution`
- SWIG: Add (read-only) attributes `type`, `length`, `strands`, `params`, and `exp_params` to objects of type `fold_compound`
- SWIG: Make attributes of objects of type `param` and `exp_param` read-only
- Add array of strand nicks to EPS dot plot files instead of single cutpoint
- Draw separator line for each strand nick in EPS dot-plots
- Update `libsvm` to version 3.24

## Package

- Disable Link-Time-Optimization (LTO) for third-party programs linking against RNAlib using `pkg-config`
- TESTS: Fix results dir path for out-of-tree builds
- TESTS: Set default timeout for library tests to 20s

## Version 2.4.14 (Release date: 2019-08-13)

### Programs

- Fix `RNAPvmin` perturbation vector computation
- Add non-redundant sampling option to `RNAPvmin`
- Add `RNAdos` program to compute density of states
- Add `-P DNA` convenience command line parameter to most programs to quickly load DNA parameters without any input file
- MAN: Add example section to man-page of `RNAalifold`

## Library

- API: Fix memory leak in `vrna_path_gradient()`
- API: Fix release of memory for `vrna_sequence_remove_all()`
- API: Fix soft-constraints application in `vrna_sc_minimize_perturbation()` that prevented proper computation of the perturbation vector
- API: Add 5' and 3' neighbor nucleotide encoding arrays and name string to `vrna_seq_t`
- API: Add new data structure for multiple sequence alignments
- API: Add `vrna_sequence_order_update()` function
- API: Add non-redundant sampling mode to `vrna_sc_minimize_perturbation()` through passing negative sample-sizes
- API: Add v3.0 API functions for maximum expected accuracy (MEA) computation
- API: Include energy parameter sets into RNALib and provide functions to load them at runtime
- API: Prepare sequence data in `vrna_fold_compound_t` with `vrna_sequence_add()`
- API: Use `vrna_pbacktrack_num()` instead of `vrna_pbacktrack()` in `vrna_sc_minimize_perturbation()` to speed-up sample generation
- Reduce use of global variable `cut_point` in RNALib
- SWIG: Use `importlib` in favor of `imp` to determine Python 3 tag extension
- SWIG: Update various wrapper functions
- SWIG: Add wrappers for MEA computation with `vrna_MEAs()` and `vrna_MEAs_from plist`
- SWIG: Add wrappers for `vrna_pr_structure()` and `vrna_pr_energy()`

## Package

- REFMAN: Fix LaTeX code in `units.h` that prevented proper compilation with `pdflatex`
- Add an R script to create 2D landscape plots from RNA2Dfold output
- Add `gengetopt` to configure-time requirements to build man-pages
- Add new energy parameter file `rna_misc_special_hairpins.par` with additional UV-melting derived parameters for Tri- and Tetra-loops
- Update RNA Tutorial
- Colorize final configure script message
- REFMAN: Always use `pdflatex` to compile reference manual and tutorial
- EXAMPLES: Add Python script that performs computations equivalent to `RNAfold -p --MEA`

## Version 2.4.13 (Release date: 2019-05-30)

### Programs

- Fix centroid structure prediction for RNACofold
- Fix `--noLP` option for RNALalifold

---

## Library

- API: Refactor and fix collision handling in `vrna_hash_table_t`
- API: Fix one access using wrong index for odd dangles in `loops/external.c`
- API: Add two missing MLbase contributions for MFE prediction in `loops/multibranch.c`
- API: Refactor multiloop MFE backtracking for odd dangles
- API: Add function `vrna_backtrack5()` to allow for MFE backtracking of sub-sequences starting at the 5'-end
- API: Reduce usage of global macro `TURN` by replacing it with `min_loop_size` field of `vrna_md_t`
- API: Add functions `vrna_path_direct()` and `vrna_path_direct_ub()` that may also return move lists instead of dot-bracket lists
- API: Add functions `vrna_pt_pk_remove()` and `vrna_db_pk_remove()` that remove pseudoknots from an input structure
- API: Fix invalid memory access for lonely pair mode (`--noLP`) in comparative sliding-window MFE prediction
- SWIG: Fix access to global variable `pf_smooth` and `pf_smooth` attribute in `model_details` object
- SWIG: Fix Python reference counting for `Py_None` in `interfaces/findpath.i` wrapper
- SWIG: Refactor reference counting for all Python2 and Python3 wrappers
- REFMAN: Larger updates and restructuring of reference manual

## Package

- Install example scripts and source code files, e.g. to `$prefix/share/ViennaRNA/examples`
- Properly pass GSL, PTHREADS, and MPFR flags to sub-projects
- Fix `RNApuzzler` header file installation
- SWIG: Include Python 3.7 and 3.8 in list of autoconf-probed python interpreters
- SWIG: Fix wrapper building for `swig >= 4.0.0`

## Version 2.4.12 (Release date: 2019-04-16)

### Programs

- Add non-redundant stochastic backtracing option for `RNAalifold`
- Add `--noDP` option to suppress dot-plot output in `RNAfold` and `RNAalifold`
- Add `RNApuzzler` (4) and `RNAturtle` (3) secondary structure layout algorithm options to `RNAfold` and `RNAplot`
- Update help/man page of `RNALfold`
- Allow for multiple input files and parallel input processing in `RNAheat`

## Library

- API: Fix declaration of `vrna_move_apply_db()`
- API: Fix `vrna_path()` lexicographical ordering in gradient walks
- API: Enable non-redundant stochastic backtracing for comparative structure prediction
- API: Enable stochastic backtracing for circular comparative structure prediction
- API: Enable stochastic backtracing of subsequences (5' prefixes) for comparative structure prediction
- API: Add `pf_smooth` attribute to `vrna_md_t` data structure to allow for disabling Boltzmann factor energy smoothing
- API: Add functions to allow for resuming non-redundant stochastic backtracing
- API: Add functions to retrieve multiple stochastically backtraced structures (list and callback variants)
- API: Add `vrna_positional_entropy` to compute vector of positional entropies
- API: Add `RNApuzzler` and `RNAturtle` secondary structure layout algorithm (Wiegreffe et al. 2018)
- API: Add v3.0 API for secondary structure layout/coordinate algorithms
- API: Add more helper/utility functions for `vrna_move_t` data structures
- API: Add callback-based neighborhood update function for (subsequent) `vrna_move_t` application
- API: Add abstract heap data structure available as <[ViennaRNA/datastructures/heap.h](#)>
- API: Refactor and speed-up gradient walk implementation available as `vrna_path_gradient()`
- API: Substitute `vrna_file_PS_aln_sub()` alignment plot function by `vrna_file_PS_aln_slice()` that actually slices out a sub-alignment
- API: Rename `vrna_annotate_covar_struct()` to `vrna_annotate_covar_db()` and add new function `vrna_annotate_covar_db_extended()` to support more bracket types
- API: Calling `vrna_params_reset()` now implies a call to `vrna_exp_params_reset()` as well
- API: Move landscape implementations into separate directory, thus headers should be included as <[ViennaRNA/landscape/move.h](#)>, <[ViennaRNA/landscape/neighbor.h](#)>, etc.
- Ensure proper rescaling of energy parameters upon temperature changes
- Refactor soft constraints implementation in stochastic backtracing
- SWIG: Wrap all non-redundant stochastic backtracing functions to scripting language interface(s)
- SWIG: Refactor stochastic backtracing interface(s)
- SWIG: Add proper constructor for objects of type `vrna_ep_t`
- SWIG: Sanitize alignment plot function interface(s)

## Package

- Update Ubuntu/Debian and OpenSUSE build instructions
- Reduce intra-package dependency on non-v3.0 API

## Version 2.4.11 (Release date: 2018-12-17)

### Programs

- Add --commands option to RNAsubopt
- Add non-redundant Boltzmann sampling mode for RNAsubopt

## Library

- API: Fix wrong access to base pair soft constraints in equilibrium probability computations
- API: Fix behavior of `vrna_nucleotide_encode()` with lowercase characters in sequence
- API: Fix behavior of `encode_char()` with lowercase characters in sequence
- API: Fix forbidden GU pairs behavior in pscore computation for comparative folding
- API: Fix potential errors due to uninitialized next pointers in `vrna_move_t` of `vrna_eval_move_<shift_pt`
- API: Add AVX 512 optimized version of MFE multibranch loop decomposition
- API: Add functions for CPU SIMD feature detection
- API: Add dispatcher to automatically delegate exterior-/multibranch loop MFE decomposition to supported SIMD optimized implementation
- API: Add function `vrna_dist_mountain()` to compute mountain distance between two structures
- API: Add function `vrna_ensemble_defect()` to compute ensemble defect given a target structure
- API: Add non-redundant Boltzmann sampling
- API: Change behavior of `vrna_cstr_free()` and `vrna_cstr_close()` to always flush output before unregistering the stream
- SWIG: Add interface for `vrna_loopidx_from_ptable()`

## Package

- Activate compilation for compile-time supported SIMD optimized implementations by default
- Replace `--enable-sse` configure script option with `--disable-simd`

## Version 2.4.10 (Release date: 2018-09-26)

### Programs

- Fix wrong output filename for binary opening energies in `RNAplfold`
- Enable G-Quadruplex support for partition function computation in `RNAalifold`

## Library

- Fix broken SSE4.1 support for multibranch loop MFE computation that resulted in increased run times
- Fix redundant output issue in subopt backtracking with unusually high delta energies ( $>=\text{INF}$ )
- Restore default behavior of '`|`' symbol in dot-bracket hard constraint strings that got lost with version 2.2.0
- Add faster (cache-optimized) version of Nussinov Maximum Matching algorithm
- Change default linker- and loop length computations for G-Quadruplex predictions in comparative prediction modes
- Add hard constraints warning for base pairs that violate the `min_loop_size` of the model
- Update `libsvm` to version 3.23
- API: Add functions to set auxiliary grammar extension rules
- API: Replace upper-triangular hard constraints matrix with full matrix for cache-optimized access
- API: Add G-Quadruplex prediction support for comparative partition function
- API: Remove `VRNA_GQUAD_MISMATCH_PENALTY` and `VRNA_GQUAD_MISMATCH_NUM_ALI` macros
- SWIG: Fix invalid memory access in `subopt()` method of `fold_compound` object when writing to file
- SWIG: Add wrapper for Nussinov Maximum Matching algorithm

## Package

- Add `-ftree-vectorize` compile flag by default if supported

## Version 2.4.9 (Release date: 2018-07-11)

### Programs

- Fix interactive mode behavior for multiple sequence alignment input in RNAalifold, RNALalifold
- Allow for Stockholm formatted multiple sequence alignment input in RNAeval and RNAPlot
- Allow for multiple input files in RNAeval and RNAPlot
- Allow for parallel processing of input batch jobs in RNAeval and RNAPlot
- Add `-g` option to activate G-Quadruplex support in RNAheat
- Warn on unsatisfiable hard constraints from dot-bracket string input in RNAfold, RNACofold, and RNAalifold

### Library

- Fix parameter order bug in `vrna_path_findpath*` functions that resulted in too large search widths
- Fix wrong application of base pair soft constraints in partition function computations
- Fix position ruler string in EPS alignment output files
- Fix MFE backtracking errors that might appear under specific hard constrained base pair patterns
- Refrain from reading anything other than `#=GC SS_cons` to retrieve structures when parsing Stockholm 1.0 format
- Complete soft constraints additions to Boltzmann sampling implementation for single sequences
- Allow for disabling alignment wrapping in `vrna_file_PS_aln*` functions
- Do not remove G-Quadruplex annotation from WUSS formatted structure strings upon calls to `vrna_db_<-from_WUSS`
- Enable G-Quadruplex related average loop energy correction terms in verbose output of `vrna_eval_*` functions
- Speed-up backward compatibility layer for energy evaluation functions that unnecessarily slowed down third-party tools using the old API
- Allow for passing dot-bracket strings with “`&strand-end` identifier to `simplevrna_eval_*` functions
- Remove `implicitexit()` calls from global MFE backtracking implementation.

## Version 2.4.8 (Release date: 2018-06-23)

### Programs

- Fix compilation of RNAforester with C++17 standard
- Fix tty input detection in RNACofold
- Fix bad memory access with RNACofold -p

---

## Library

- API: Fix incorrect unpaired probability computations in [vrna\\_probs\\_window\(\)](#)
- API: Fix potential out-of-bounds access situations (for circular RNA folding) in eval.c
- API: Fix comparative exterior internal loop partition function computation for circfold
- SWIG: Fix false-positive use of uninitialized value in Python3/file\_py3.i

## Package

- TESTS: Add tests for special features in RNAalifold
- TESTS: Add test case for RNACofold -p

## Version 2.4.7 (Release date: 2018-06-13)

- Allow for parallel processing across multiple input files in RNAfold
- Allow for arbitrary number of input files in RNAalifold
- Allow for parallel processing of input data in RNAalifold
- Allow for arbitrary number of input files in RNACofold
- Allow for parallel processing of input data in RNACofold
- Enable parallel processing in RNAfold, RNACofold, RNAalifold for MS Windows build
- Add centroid and MEA structure computation to RNACofold
- Add configure time check for LTO capabilities of the linker
- Include ligand binding energies in centroid and MEA structure output of RNAfold
- Refactor ct2db program to process multiple structures from single .ct file
- API: Enable processing of comparative fold\_compound with [vrna\\_pr\\_\\*](#)() functions
- API: Refactor vrna\_oiostream\_t to enable NULL input in [vrna\\_oiostream\\_provide\(\)](#)
- API: Major refactoring in loop energy evaluations (MFE and PF)
- API: Make vrna\_mx\_pf\_aux\_el\_t and vrna\_mx\_pf\_aux\_ml\_s opaque pointers
- API: Make fold\_compound field type a const attribute
- API: Refactor MFE post-processing for circular RNAs
- API: Add motif name/id support for unstructured domains
- API: Remove major part of implicit exit() calls in RNAlib
- API: Add implementations of Boyer-Moore-Horspool search algorithm
- API: Add implementations to determine number of rotational symmetry for strings (of objects)
- API: Make vrna\_cmd\_t an opaque pointer
- API: Move headers for constraints, datastructures, io, loop energy evaluation, energy parameters, plotting, search, and utilities into separate subdirectories (backward compatibility is maintained)
- API: Add hash table data structure
- API: Fix discrepancy between comparative and single sequence –noLP predictions
- API: Add functions to replace 'old API' interface of [RNAstruct.h](#)

- API: Add functions to replace 'old API' interface of `aln_util.h`
- API: Add generic soft constraints support to suboptimal structure prediction sensu Wuchty et al.
- SWIG: Refactor callback execution for Python 2 / 3 interface to reduce overhead
- SWIG: Fix configure-time check for Python 3 interface build
- SWIG: Fix Python 3 IO file stream to C FILE \* conversion
- Cosmetic changes in final configure notice
- Major changes in source tree structure of the library
- Add autoconf checks for maintainer tools
- Generate C strings from static PostScript files at configure time (for structure- and dot plots)
- REFMAN: Large updates in API documentation and structure of reference manual

### **Version 2.4.6 (Release date: 2018-04-19)**

- Stabilize rounding of free energy output in RNAalifold
- API: Fix potential rounding errors for comparative free energies in eval.c and mfe.c
- API: Fix regression in exterior loop dangling end contributions for comparative base pair probabilities and Boltzmann sampling (introduced with v2.4.4)
- API: Fix regression with hard constrained base pairs for comparative structure prediction (introduced with v2.4.4)
- TESTS: Add basic tests for RNAalifold executable
- TESTS: Ignore 'frequency of MFE structure' in RNACofold partition function checks

### **Version 2.4.5 (Release date: 2018-04-17)**

- Allow for arbitrary number of input files in RNAfold
- Allow for parallel processing of input data in RNAfold (UNIX only, no Windows support yet)
- Add SHAPE reactivity support through commandline options for RNAlfold
- Fix unstructured domain motif detection in MFE, centroid, and MEA structures computed by RNAfold
- Limit allowed set of commands in command file for RNACofold to hard and soft constraints
- API: Add functions to compute equilibrium probability of particular secondary structures
- API: Add dynamic string stream data type and associated functions
- API: Add priority-queue like data structure with unordered fill capability and ordered output callback execution
- API: Add functions to detect unstructured domain motifs in MFE, centroid, and MEA structures
- API: Fix bug in sliding-window partition function computation with SHAPE reactivity and Deigan et al. conversion method
- API: Fix application of '<' and '>' constraint symbols in dot-bracket provided constraints (was broken since v2.4.2)
- API: Fix MEA structure computation in the presence of unstructured domains
- API: Stabilize order of probability entries in EPS dot-plot files
- Fix compiler warnings on wrong type of printf() in naview.c

- Define VRNA\_VERSION macro as string literal and add macros for major, minor, and patch numbers
- Stabilize parallel make of Mac OS X installer
- Add energy parameter set from Langdon et al. 2018
- Add autoconf checks for POSIX threads compiler/linker support
- SWIG: Fix 'next' is a perl keyword warnings for Perl5 wrapper
- SWIG: Catch errors and throw exceptions whenever scripting language provided callback functions are not applicable or fail
- SWIG: Add keyword arguments and autodoc feature for Python/Python3 wrappers

### **Version 2.4.4 (Release date: 2018-03-06)**

- Change verbose output for soft-constraints derived ligand binding motifs in RNAfold
- Allow for lowercase letters in ct2db input
- Fix bug in interior-like G-Quadruplex MFE computation for single sequences
- Fix autoconf switch to enable deprecation warnings
- Fix bug in eval\_int\_loop() that prevented propagation of energy evaluation for loops with nick in strands
- Fix several bugs for SHAPE reactivity related comparative partition function computations
- Fix annotation of PostScript output for soft-constraint derived ligand binding motifs in RNAfold
- Fix constraint indices for multibranch loops in unpaired probability computations of LPfold.c
- Fix dangling end contributions in comparative partition function for exterior loops
- API: Add simplified interface for [vrna\\_pf\\_dimer\(\)](#)
- API: Move concentration dependent implementation for co-folding to separate compile unit
- API: Add new API functions for exterior loop evaluations
- API: Add simplified interfaces for energy evaluation with G-Quadruplexes and circular RNAs
- API: Add findpath functions that allow for specification of an upper bound for the saddle point
- Add configure-time linker check for Python3 interface
- Add automatic CPP suggestions for deprecated function substitutes
- Major restructuring and constraints feature additions in loop type dependent energy evaluation functions
- Major restructuring in MFE implementations
- Major restructuring in PF implementations
- Minor fixes in Boltzmann sampling implementation
- SWIG: Fix wrappers for findpath() implementation
- SWIG: Add tons of energy evaluation wrappers
- SWIG: Fix configure-time check of Perl5 interface build capabilities
- SWIG: Wrap functions from walk.c and neighbor.c
- DOC: Add some missing references to manpages of executable programs
- REFMAN: Heavy re-ordering of the RNAlib reference manual

**Version 2.4.3 (Release date: 2017-11-14)**

- Fix handling of dangling end contribution at sequence boundaries for sliding window base pair probability computations
- Fix handling of base pair hard constraints in sliding-window implementations
- Fix sliding-window pair probability computations with multibranch-loop unpaired constraints
- Fix sliding-window non-specific base pair hard constraint implementation
- Fix probability computation for stochastic backtracking in RNAsubopt –stochBT\_en output
- Fix regression in comparative structure prediction for circular RNAs
- Fix LDFLAGS for scripting language interfaces in corresponding Makefiles
- Stabilize partition function scaling by always using sfact scaling factor from model details
- Add –pf\_scale commandline parameter to RNAPlfold
- Add constraint framework for single sequence circular RNA structure prediction
- Add RNAfold test suite to check for working implementation of constraints for circular RNAs
- Add a brief contribution guideline CONTRIBUTING.md
- Prevent RNAPlfold from creating inf/-inf output when solution set is empty with particular hard constraints
- Include RNAforester v2.0.1

**Version 2.4.2 (Release date: 2017-10-13)**

- Fix G-Quadruplex energy corrections in comparative structure energy evaluations
- Fix discrepancy in comparative exterior loop dangling end contribution of eval vs. MFE predictions
- Fix regression in RNAUp unstructuredness and interaction energy computations
- Fix sequence length confusions when FASTA input contains carriage returns
- Fix build problems of RNAlocmin with older compilers
- Fix sliding-window hard constraints where single nucleotides are prohibited from pairing
- Fix dot-bracket output string length in sliding-window MFE with G-Quadruplexes
- Fix unpaired probability computations for separate individual loop types in LPfold.c
- Fix bad memory access in RNAsubopt with dot-bracket constraint
- Add full WUSS support for –SS\_cons constraint option in RNAAlifold
- Add commandline option to RNALalifold that enables splitting of energy contributions into separate parts
- Add missing hard constraint cases to sliding-window partition function implementation
- Add CSV output option to RNAcofold
- Use the same model details for SCI computations in RNAAlifold
- Abort computations in `vrna_eval_structure_v()` if structure has unexpected length
- Use original MSA in all output generated by RNAAlifold and RNALalifold
- API: Add new functions to convert dot-bracket like structure annotations
- API: Add various new utility functions for alignment handling and comparative structure predictions
- API: Add function `vrna_strsplit()` to split string into tokens

- API: Do not convert sequences of input MSA to uppercase letters in `vrna_file_msa_read_record()`
- API: Rename `vrna_annotate_bp_covar()` and `vrna_annotate_pr_covar()`
- API: Add new noLP neighbor generation
- SWIG: Add wrapper for functions in `file_utils_msa.h`
- SWIG: Add wrappers for `vrna_pbacktrack()` and `vrna_pbacktrack5()`
- SWIG: Add `vrna_db_to_element_string()` to scripting language interface
- REFMAN: Fix formula to image conversion in HTML output

### **Version 2.4.1 (Release date: 2017-08-23)**

- Fix memory leak in `fold_compound` methods of SWIG interface
- Fix memory leaks in double \*\* returning functions of SWIG Perl5 interface
- Fix memory leak in `vrna_ep_t` `to-string()` function of SWIG interface
- Regression: Fix reverting `pf_scale` to defaults after `vrna_exp_params_rescale()`
- Regression: Fix homo-dimer partition function computation in RNACofold
- Add unit tests for RNACofold executable
- Add SHAPE reactivity support to RNACofold
- Add SHAPE reactivity support to RNALalifold

### **Version 2.4.0 (Release date: 2017-08-01)**

- Bump libsvm to version 3.22
- Print G-Quadruplex corrections in verbose mode of RNAeval
- Change behavior of RNAfold –outfile option to something more predictable
- Unify `max_bp_span` usage among sliding window prediction algorithms: RNAPlfold, RNALfold, and RNALalifold now consider any base pair (i,j) with  $(j - i + 1) \leq \text{max_bp_span}$
- Add SHAPE reactivity data support to RNALfold
- Add commands-file support for RNALfold, RNAPlfold (hard/soft constraints)
- Add RNAlocmin - Calculate local minima from structures via gradient walks
- Add RNA Bioinformatics tutorial (PDF version)
- Add hard constraints to sliding-window MFE implementations (RNALfold, RNALalifold)
- Add hard constraints to sliding-window PF implementations (RNAPlfold)
- Add soft constraints to sliding-window MFE implementation for single sequences (RNALfold)
- Add soft constraints to sliding-window PF implementations (RNAPlfold)
- Add SWIG interfaces for sliding-window MFE/PF implementations
- Add proper SWIG interface for alignment and structure plotting functions
- Add proper SWIG interface for duplexfold, duplex\_subopt, and its comparative variants
- Add SWIG wrapper for `vrna_exp_params_rescale()`
- Add explicit destructor for SWIG generated `vrna_md_t` objects

- Add SWIG perl5 typemap for simple nested STL vectors
- Add dummy field in [vrna\\_structured\\_domains\\_s](#)
- Add note about SSE optimized code in reference manual
- Add SWIG interface for findpath implementation
- Add prepare() functions for ptypes-arrays and vrna\_(exp\_)param\_t
- Add warnings for ignored commands in function [vrna\\_commands\\_apply\(\)](#)
- Add callback featured functions for sliding window MFE and PF implementations
- Change default behavior of adding soft constraints to a vrna\_fold\_compound\_t (store only)
- Several fixes with respect to G-Quadruplex prediction in sliding-window MFE recursions (single sequence and comparative implementation)
- Replace comparative sliding-window MFE recursions (All hits are reported to callback and can be filtered in a post-processing step)
- API: Remove E\_mb\_loop\_stack() and introduce new function [vrna\\_E\\_mb\\_loop\\_stack\(\)](#) as a replacement
- API: change data type of all constraint bit-flags from char to unsigned char
- API: change data type of a2s array in comparative structure prediction from unsigned short to unsigned int
- API: Change function parameter order in [vrna\\_probs\\_window\(\)](#) to follow the style of other callback-aware functions in RNAlib
- Move sliding-window MFE implementations to new file mfe\_window.c
- Fix building PDF Reference manual with non-standard executable paths
- Fix redefinition of macro ON\_SAME\_STRAND() in subopt.c
- Fix dangling end issues in sliding-window MFE implementations
- Fix regression for –canonicalBPonly switch in RNAfold/RNAcofold/RNAsubopt
- Fix building sliding-window MFE implementation without SVM support
- Fix parsing of STOCKHOLM 1.0 MSA files that contain MSA spanning multiple blocks
- Fix Alidot link in RNAalifold manpage
- Fix wrong pre-processor flags when enabling single-precision PF computations
- Fix unit testing perl5 interface by including builddir/tests in PERL5LIB path
- Fix buffer overflow in hairpin loop sequence motif extraction for circular RNAs
- Fix out-of-bounds memory access in neighbor.c
- Restore capability to compile stand-alone findpath utility
- Restore capability to use non-standard alphabets for structure prediction
- Restore old-API random number functions in SWIG interface
- Allow additional control characters in MAF MSA input that do not end a block
- Improve reference manual
- Make functions in [pair\\_mat.h](#) static inline
- Prevent users from adding out-of-range base pair soft constraints

- Inline print functions in color\_output.inc
- Start documenting callback features in reference manual
- Re-write large portions of sliding-window PF implementation
- Introduce soft-constraint state flag
- Clean-up SWIG unit test framework
- Remove obsolete scripts ct2b.pl and colorrna.pl from src/Utils directory
- Remove old RNAfold tutorial

## Version 2.3.x

### **Version 2.3.5 (Release date: 2017-04-14)**

- Fix duplication of output filename prefix in RNAfold
- Add V3.0 API for sliding window partition function (a.k.a. RNAPLfold)
- Add G-Quadruplex prediction to RNALalifold
- Add SWIG wrappers for callback-based sliding window comparative MFE prediction
- Add SSE4.1 multiloop decomposition for single sequence MFE prediction
- Enable RNAfold unit tests to run in parallel
- Enable users to turn-off base pair probability computations in RNACofold with -a option
- Split move set in neighbor.c

### **Version 2.3.4 (Release date: 2017-03-10)**

- Fix G-Quadruplex probability computation for single sequences
- Fix double-free when using SHAPE reactivity data in RNAalifold
- Fix out-of-bounds access in strand\_number array
- Fix weighting of SHAPE reactivity data in consensus structure prediction when fewer data than sequences are present
- Fix z-score output in RNALfold
- Substitute field name 'A0'/'B0' in data structure `vrna_dimer_conc_s` by 'Ac\_start'/'Bc\_start' to avoid clashes with termios.h (Mac OSX Python wrapper bug)
- Minimize usage of 'unsafe' sprintf() calls
- Enhance auto-id feature in executable programs
- Always sanitize output file names to avoid problems due to strange FASTA headers
- Lift restrictions of FASTA header length in RNAfold, RNACofold, and RNAeval
- Add ViennaRNA/config.h with pre-processor definitions of configure time choices
- Add test-suite for RNAfold
- Add functions to produce colored EPS structure alignments
- Add function to write Stockholm 1.0 formatted alignments
- Add function to sanitize file names

- Add callback based implementation for sliding-window MFE prediction (single sequences, comparative structure prediction)
- Add fast API 3.0 implementations to generate structural neighbors and perform steepest descent / random walks (Thanks to Gregor!)
- Add parameter option to RNALalifold for colored EPS structure alignment and structure plot output
- Add parameter option to RNALalifold to write hits into Stockholm file
- Add parameter option to RNAalifold to write Stockholm 1.0 formatted output
- Add parameter option to RNAalifold to suppress stderr spam
- Add auto-id feature to RNAPlot, RNAlfold, RNAsubopt, RNAPlfold, RNAheat
- Add SHAPE reactivity derived pseudo-energies as separate output in RNAalifold
- Add colored output to RNA2Dfold, RNALalifold, RNAlfold, RNAduplex, RNAheat, RNAinverse, RNAPlfold, and RNAsubopt
- Add command line parameters to RNAsubopt to allow for specification of input/output files

### **Version 2.3.3 (Release date: 2017-01-24)**

- Fix multiloop contributions for comparative partition function
- Fix building python2 extension module for OSX

### **Version 2.3.2 (Release date: 2017-01-18)**

- Fix pair probability plist creation with G-Quadruplexes
- Allow for specification of python2/3-config at configure time
- Fix init of vrna\_md\_t data structure after call to [set\\_model\\_details\(\)](#)
- Fix bug in consensus partition function with hard constraints that force nucleotides to be paired
- Fix compilation of functions that use ellipsis/va\_list
- Enable generic hard constraints by default
- Fix init of partition function DP matrices for unusually short RNAs
- Fix behavior of RNAPlfold for unusually short RNAs
- Report SCI of 0 in RNAalifold when sum of single sequence MFEs is 0
- Avoid multiple includes of [pair\\_mat.h](#)
- Add configure flag to build entirely static executables

### **Version 2.3.1 (Release date: 2016-11-15)**

- Add description for how to use unstructured domains through command files to reference manual and RNAfold manpage
- Fix compilation issue for Windows platforms with MingW
- Add missing newline in non-TTY-color output of [vrna\\_message\\_info\(\)](#)
- Fix regression in [vrna\\_md\\_update\(\)](#) that resulted in incomplete init of reverse-basepair type array
- Extend coverage of generic hard constraints for partition function computations
- Fix scaling of secondary structure in EPS plot such that it always fits into bounding box
- Several fixes and improvements for SWIG generated scripting language interface(s)

### **Version 2.3.0 (Release date: 2016-11-01)**

- Add grammar extension with structured and unstructured domains
- Add default implementation for unstructured domains to allow for ligand/protein binding to unpaired structure segments (MFE and PF for single sequences)
- Introduced command files that subsume constraint definition files (currently used in RNAfold and RNACofold)
- Replace explicit calls to asprintf() with portable equivalent functions in the library
- Fix configure script to deal with situations where Perl module can't be build
- Fix bug in doc/Makefile.am that prevented HTML installation due to long argument list
- Added utility functions that deal with conversion between different units
- Bugfix in SWIG wrapped generic soft constraint feature
- Add [subopt\(\)](#) and subopt\_zuker() methods to SWIG wrapped fold\_compound objects
- Bugfix multiloop decomposition in MFE for circular RNAs
- Add separate function to compute pscore for alignments
- Renamed VRNA\_VC\_TYPE\_\* macros to VRNA\_FC\_TYPE\_\*
- Bugfix regression that prevented programs to fail on too long input sequences
- Extend EPS dot-plot in RNAfold to include motif/binding probabilities from unstructured domains
- Add variadic functions for error/warning/info message
- Add ID manipulation feature to RNAeval
- Extend API for soft constraint feature for more fine-grained control
- Add section on SWIG wrapped functions in reference manual
- Fix bug in interior loop computations when hard constraints result in non-canonical base pairs

## **Version 2.2.x**

### **Version 2.2.10 (Release date: 2016-09-06)**

- Do not 'forget' subopt results when output is not written to file handle and sorting is switched off
- Fix bad memory access in [vrna\\_subopt\(\)](#) with sorted output
- Add SWIG wrappers for [vrna\\_subopt\\_cb\(\)](#)
- Correctly show if C11 features are activated in configure status
- Fix autoconf checks to allow for cross compilation again

### **Version 2.2.9 (Release date: 2016-09-01)**

- Fix bug in partition function scaling for backward compatibility of ali\_pf\_fold()
- Stabilize v3.0 API when building RNAlib and third party program linking against it with compilers that use different C/C++ standards
- Add details on how to link against RNAlib to the reference manual
- Fix RNAlib2.pc
- Fix bug for temperature setting in RNAlifold

- Use -fflat-lto-objects for static RNAlib library to allow linking without LTO
- Fix interpretation of 'P' hard constraint for single nucleotides in constraint definition files
- Add 'A' command for hard constraints
- Fix several hard constraint corner-cases in MFE and partition function computation when nucleotides must not be unpaired
- Fix order of hard constraints when read from input file
- Allow for non-canonical base pairs in MFE and partition function computations if hard constraints demand it
- Fix behavior of –without-swig configure script option
- Fix bug in hard constraints usage of exterior loop MFE prediction with odd dangles
- Add parsers for Clustal, Stockholm, FASTA, and MAF formatted alignment files
- Enable RNAalifold to use Clustal, Stockholm, FASTA, or MAF alignments as input
- Lift restriction of sequence number in alignments for RNAalifold
- Enable ANSI colors for TTY output in RNAfold, RNACofold, RNAalifold, RNAsubopt, and warnings/errors issued by RNAlib
- Add various new commandline options to manipulate sequence/alignment IDs in RNAfold, RNACofold and RNAalifold

### **Version 2.2.8 (Release date: 2016-08-01)**

- Fix bad memory access in RNAalifold
- Fix regression in RNAalifold to restore covariance contribution ratio determination for circular RNA alignments
- Changed output of RNAsubopt in energy-band enumeration mode to print MFE and energy range in kcal/mol instead of 10cal/mol
- Include latest Kinfold sources that make use of v3.0 API, therefore speeding up runtime substantially
- Re-activate warnings in RNAeval when non-canonical base pairs are encountered
- Fix syntactic incompatibilities that potentially prevented compilation with compilers other than gcc
- dd function to compare nucleotides encoded in IUPAC format
- Fix regression in energy evaluation for circular RNA sequences
- Fix regression in suboptimal structure enumeration for circular RNAs
- Allow for P i-j k-l commands in constraint definition files
- Make free energy evaluation functions polymorphic
- Add free energy evaluation functions that allow for specifying verbosity level
- Secure functions in alphabet.c against NULL pointer arguments
- Fix incompatibility with swig >= 3.0.9
- Fix memory leak in swig-generated scripting language interface(s) for user-provided target language soft-constraint callbacks
- Expose additional functions to swig-generated scripting language interface(s)
- Build Python3 interface by default
- Start of more comprehensive scripting language interface documentation

- Fix linking of python2/python3 interfaces when libpython is in non-standard directory
- Restructured viennarna.spec for RPM based distributions
- Several syntactic changes in the implementation to minimize compiler warnings
- Fix –with-\* / –without-\* and –enable-\* / –disable-\* configure script behavior

### **Version 2.2.7 (Release date: 2016-06-30)**

- Fix partition function scaling for long sequences in RNAfold, RNAalifold, and RNAup
- Fix backtracking issue in RNACofold when –noLP option is activated
- Fix hard constraints issue for circular RNAs in generating suboptimal structures
- Rebuild reference manual only when actually required

### **Version 2.2.6 (Release date: 2016-06-19)**

- Plugged memory leak in RNACofold
- Fixed partition function rescaling bug in RNAup
- Fixed bug in RNALfold with window sizes larger than sequence length
- Re-added SCI parameter for RNAalifold
- Fixed backtracking issue for large G-quadruplexes in RNAalifold
- Fixed missing FASTA id in RNAeval output
- Added option to RNAalifold that allows to specify prefix for output files
- Several fixes and additional functions/methods in scripting language interface(s)
- Added version information for scripting language interface(s)
- Some changes to allow for compilation with newer compilers, such as gcc 6.1

### **Version 2.2.5 (Release date: 2016-04-09)**

- Fixed regression in RNACofold that prohibited output of concentration computations
- Fixed behavior of RNAfold and RNACofold when hard constraints create empty solution set (programs now abort with error message)
- Added optional Python 3 interface
- Added RNA::Params Perl 5 sub-package
- Update RNA::Design Perl 5 sub-package
- Simplified usage of v3.0 API with default options
- Wrap more functions of v3.0 API in SWIG generated scripting language interfaces
- Plugged some memory leaks in SWIG generated scripting language interfaces
- Changed parameters of recursion status callback in vrna\_fold\_compound\_t
- Enable definition and binding of callback functions from within SWIG target language
- Added optional subpackage Kinwalker
- Added several configure options to ease building and packaging under MacOS X
- Added new utility script RNAdesign.pl

**Version 2.2.4 (Release date: 2016-02-19)**

- Fixed bug in RNAsubopt that occasionally produced cofolded structures twice
- Removed debugging output in preparations of consensus structure prediction datastructures

**Version 2.2.3 (Release date: 2016-02-13)**

- Added postscript annotations for found ligand motifs in RNAfold
- Added more documentation for the constraints features in RNAfold and RNAalifold
- Restore backward compatibility of `get_alipf_arrays()`

**Version 2.2.2 (Release date: 2016-02-08)**

- Fix regression bug that occasionally prevented backtracking with RNAcofold –noLP

**Version 2.2.1 (Release date: 2016-02-06)**

- Fix regression bug that made RNAcofold -a unusable
- Fix regression bug that prohibited RNAfold to compute the MEA structure when G-Quadruplex support was switched on
- Fix bug in Kinfold to enable loading energy parameters from file
- Fix potential use of uninitialized value in RNApdist
- Add manpage for ct2db
- Fix MEA computation when G-Quadruplex support is activated
- Allow for vendor installation of the perl interface using INSTALLDIRS=vendor at configure time
- Install architecture dependent and independent files of the perl and python interface to their correct file system locations

**Version 2.2.0 (Release date: 2016-01-25)**

- RNAforester is now of version 2.0
- New program RNApmin to compute pseudo-energy perturbation vector that minimizes discrepancy between observed and predicted pairing probabilities
- SHAPE reactivity support for RNAfold, RNAsubopt, and RNAalifold
- Ligand binding to hairpin- and interior-loop motif support in RNAfold
- New commandline option to limit maximum base pair span for RNAfold, RNAsubopt, RNAcofold, and RNAalifold
- Bugfix in RNAheat to remove numerical instabilities
- Bugfix in RNAPlex to allow for computation of interactions without length limitation
- Bugfix in RNAPlot for simple layouts and hairpins of size 0
- (generic) hard- and soft-constraints for MFE, partition function, base pair probabilities, stochastic backtracking, and suboptimal secondary structures of single sequences, sequence alignments, and sequence dimers
- libsvm version as required for z-scoring in RNALfold is now 3.20
- Stochastic backtracking for single sequences is faster due to usage of Boustrophedon scheme
- First polymorphic functions `vrna_mfe()`, `vrna_pf()`, and `vrna_pbacktrack()`.

- The FLT\_OR\_DBL macro is now a typedef
- New functions to convert between different secondary structure representations, such as helix lists, and RNAshapes abstractions
- First object-oriented interface for new API functions in the scripting language interfaces
- new ViennaRNA-perl submodule that augments the Perl interface to RNAlib
- Ligand binding to hairpin- and interior-loop motif support in C-library and scripting language interfaces.
- Libraries are generated using libtool
- Linking of libraries and executables defaults to use Link Time Optimization (LTO)
- Large changes in directory structure of the source code files

## Version 2.1.x

### Version 2.1.9

- Fixed integer underflow bug in RNALfold
- Added Sequence Conservation index (SCI) option to RNAalifold
- Fixed bug in energy evaluation of dangling ends / terminal mismatches of exterior loops and multibranch loops
- Fixed bug in alifold partition function for circular RNAs
- Fixed bug in alifold that scrambled backtracing with activated G-Quadruplex support
- Fixed bug in alifold backtracking for larger G-Quadruplexes

### Version 2.1.8

- Repaired incorporation of RNAinverse user provided alphabet
- Fix missing FASTA ID in RNAeval output
- prevent race condition in parallel calls of [Lfold\(\)](#)
- Fixed memory bug in [Lfold\(\)](#) that occurred using long sequences and activated G-Quad support
- Added latest version of switch.pl

### Version 2.1.7

- Fixed bug in RNALfold -z
- Python and Perl interface are compiling again under MacOSX
- Fixed handling of C arrays in Python interface
- Added latest version of switch.pl
- Make relplot.pl work with RNAcofold output

### Version 2.1.6

- New commandline switches allow for elimination of non-canonical base pairs from constraint structures in RNAfold, RNAalifold and RNAsubopt
- updated moveset functions
- final fix for discrepancy of tri-loop evaluation between partition function and mfe
- pkg-config file now includes the OpenMP linker flag if necessary
- New program ct2db allows for conversion of .ct files into dot-bracket notation (incl. pseudo-knot removal)

### Version 2.1.5

- Fix for discrepancy between special hairpin loop evaluation in partition functions and MFE

### Version 2.1.4

- Fix of G-quadruplex support in [subopt\(\)](#)
- Fix for discrepancy between special hairpin loop evaluation in partition functions and MFE

### Version 2.1.3

- RNAfold: Bugfix for ignoring user specified energy parameter files
- RNACofold: Bugfix for crashing upon constrained folding without specifying a constraint structure
- RNAsubopt: Added G-quadruplex support
- RNAalifold: Added parameter option to specify base pair probability threshold in dotplot
- Fix of several G-quadruplex related bugs
- Added G-quadruplex support in [subopt\(\)](#)

### Version 2.1.2

- RNAfold: Bugfix for randomly missing probabilities in dot-plot during batch job execution
- RNAeval: Bugfix for misinterpreted G-quadruplex containing sequences where the quadruplex starts at nucleotide 1
- RNAsubopt: Slight changes to the output of stochastic backtracking and zuker subopt
- Fix of some memory leaks
- Bugfixes in [zukersubopt\(\)](#), [assign\\_plist\\_from\\_pr\(\)](#)
- New threadsafe variants of [putoutpU\\_prob\\*](#)() for LPfold()
- Provision of python2 interface support.

### Version 2.1.1

- Bugfix to restore backward compatibility with ViennaRNA Package 1.8.x API (this bug also affected proper usage of the perl interface)

### Version 2.1.0

- G-Quadruplex support in RNAfold, RNACofold, RNALfold, RNAalifold, RNAeval and RNAPlot
- LPfold got a new option to output its computations in split-mode
- several G-Quadruplex related functions were introduced with this release
- several functions for moves in an RNA landscape were introduced
- new function in alifold.c now enables access to the partition function matrices of [alipf\\_fold\(\)](#)
- different numeric approach was implemented for concentration dependend co-folding to avoid instabilities which occurred under certain circumstances

## Version 2.0.x

### Version 2.0.7

- Bugfix for RNAlfold where segfault happened upon usage of -O option
- Corrected misbehavior of RNAeval and RNAPlot in tty mode

### Version 2.0.6

- Bugfix for bad type casting with gcc under MacOSX (resulted in accidental "sequence too long" errors)
- Bugfix for disappearing tri-/hexaloop contributions when read in from certain parameter files
- Bugfix for RNAlfold that segfaulted on short strange sequences like AT+ repeats
- Change of RNA2Dfold output format for stochastic backtracking

### Version 2.0.5

- Restored z-score computation capabilities in RNAlfold

### Version 2.0.4

- Bugfix for RNAcofold partition function
- Perl wrapper compatibility to changed RNAlib has been restored
- Backward compatibility for partition function calls has been restored

### Version 2.0.3

- Bugfix for RNAalifold partition function and base pair probabilities in v2.0.3b
- Added Boltzmann factor scaling in RNAsubopt, RNAalifold, RNAlfold and RNAcofold
- Bugfix for alifold() in v2.0.3b
- Restored threadsafety of folding matrix access in LPfold.c, alifold.c, part\_func.c, part\_func\_co.c and part\_func\_up.c
- Added several new functions regarding threadsafe function calls in terms of concurrently changing the model details
- Added pkg-config file in the distribution to allow easy checks for certain RNAlib2 versions, compiler flags and linker flags.

### Version 2.0.2

- added support for Boltzmann factor scaling in RNAfold
- fixed fastaheader to filename bug
- plugged some memory leaks

## Version 2.0.1

- First official release of version 2.0
- included latest bugfixes

## History

2011-03-10 Ronny Lorenz [ronny@tbi.univie.ac.at](mailto:ronny@tbi.univie.ac.at)

- new naming scheme for all shipped energy parameter files
- fixed bugs that appear while compiling with gcc under MacOS X
- fixed bug in RNAUp –interaction-first where the longer of the first two sequences was taken as target
- added full FASTA input support to RNAdfold, RNAdcfold, RNAdheat, RNAdpfold RNALfoldz, RNAsubopt and RNALfold

2010-11-24 Ronny Lorenz [ronny@tbi.univie.ac.at](mailto:ronny@tbi.univie.ac.at)

- first full pre-release of version 2.0

2009-11-03 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- Fix memory corruption in [PS\\_color\\_aln\(\)](#)

2009-09-09 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- Fix bug in RNAdpfold when -u and -L parameters are equal
- Fix double call to [free\\_arrays\(\)](#) in RNAdfold.c
- Improve drawing of cofolded structures

2009-05-14 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- Fix occasional segfault in RNAdalifold's print\_alifout()

2009-02-24 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- Add -MEA options to RNAdfold and RNAdalifold
- change [energy\\_of\\_alistuct](#) to return float not void

2009-02-24 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- RNAdfold will draw structures unless -noPS is used (no more "structure too long" messages)
- Restore the "alifold.out" output from RNAdalifold -p
- RNAdalifold -circ did not work due to wrong return type
- Accessibility calculation with RNAdpfold would give wrong results for u<=30

2008-12-03 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- Add zuker style suboptimals to RNAsubopt (-z)
- [get\\_line\(\)](#) should be much faster when reading huge sequences (e.g. whole chromosomes for RNALfold)

2008-08-12 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- Add Ribosum matrices for covariance scoring in RNAdalifold

2008-06-27 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- Change RNAdalifold to use berni's new energy evaluation w/o gaps
- Add stochastic backtracking in RNAdalifold

2008-07-04 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- 
- modify output of RNAUp (again). Program reading RNAUp output will have to be updated!

2008-07-02 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- RNAPlfold now computes accessibilities for all regions up to a max length simultaneously. Slightly slower when only 1 value is needed, but much faster if all of them are wanted. This entails a new output format. Programs reading accessibility output from RNAPlfold need to be updated!

2008-03-31 Stephan Bernhart [berni@tbi.univie.ac.at](mailto:berni@tbi.univie.ac.at)

- add cofolding to RNAsubopt

2008-01-08 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- ensure circfold works even for open chain

2007-12-13 Ulli Mueckstein [ulli@tbi.univie.ac.at](mailto:ulli@tbi.univie.ac.at)

- update RNAUp related files RNAUp can now include the intramolecular structure of both molecules and handles constraints.

2007-12-05 Ronny Lorenz [ronny@tbi.univie.ac.at](mailto:ronny@tbi.univie.ac.at)

- add circfold variants in part\_func.c alifold.c subopt.c

2007-09-19 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- compute the controid structure of the ensemble in RNAfold -p
- fix a missing factor 2 in `mean_bp_dist()`. CAUTION ensemble diversities returned by RNAfold -p are now twice as large as in earlier versions.

2007-09-04 Ivo Hofacker [ivo@blini.tbi.univie.ac.at](mailto:ivo@blini.tbi.univie.ac.at)

- fix a bug in `Lfold()` where base number n-max-4 would never pair

2007-08-26 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- add RNAAliduplex the alignment version of RNAduplex
- introduce a minimal distance between hits produced by duplex\_subopt()

2007-07-03 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- add a `loop_energy()` function to compute energy of a single loop

2007-06-23 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- add aliLfold() and RNALalifold, alignment variant of `Lfold()`

2007-04-30 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- add RNAUp to distribution

2007-04-15 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- fix segfault in colorps output (thanks to Andres Varon)

2007-03-03 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- avoid unnormalized doubles in scale[], big speedup for `pf_fold()` on very long sequences

2007-02-03 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- RNAalifold can now produce colored structure plots and alignment plots

2007-02-01 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- Fix segfault in RNAPlfold because of missing prototype

2006-12-01 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- RNAduplex would segfault when no structure base pairs are possible

2006-08-22 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- add computation stacking probabilities using RNAAfold -p2
- add -noPS option for RNAAfold to suppress drawing structures

2006-08-09 Stephan Bernhart [berni@tbi.univie.ac.at](mailto:berni@tbi.univie.ac.at)

- RNAPlfold can now compute probabilities of unpaired regions (scanning version of RNAUp)

2006-06-14 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- compile library with -fpic (if available) for use as shared library in the Perl module.
- fix another bug when calling [Lfold\(\)](#) repeatedly
- fix switch cmdline parsing in RNAAlifold (-mis implied -4)
- fix bug in [cofold\(\)](#) with dangles=0

2006-05-08 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- fix segfault in [Lfold\(\)](#) when calling repeatedly
- fix structure parsing in RNAstruct.c (thanks to Michael Pheasant for reporting both bugs)
- add duplexfold() and [alifold\(\)](#) to Perl module
- distinguish window size and max pair span in LPfold

2006-04-05 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- fix performance bug in [co\\_pf\\_fold\(\)](#)
- use relative error for termination of Newton iteration

2006-03-02 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- add circular folding in [alifold\(\)](#)

2006-01-18 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- cleanup berni partition cofold code, including several bug fixes

2006-01-16 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- update RNAPlfold to working version
- add PS\_dot\_plot\_turn() in PS\_dot.c

2005-11-07 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- add new utilities colorna and coloraln

2005-10-11 Christoph Flamm [xtof@tbi.univie.ac.at](mailto:xtof@tbi.univie.ac.at)

- adapt [PS\\_rna\\_plot\(\)](#) for drawing co-folded structures

2005-07-24 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- 
- fix a few memory problems in structure comparison routines

2005-04-30 Ivo Hofacker [ivo@blini.tbi.univie.ac.at](mailto:ivo@blini.tbi.univie.ac.at)

- add folding of circular RNAs

2005-03-11 Ivo Hofacker [ivo@blini.tbi.univie.ac.at](mailto:ivo@blini.tbi.univie.ac.at)

- add -mis option to RNAAlifold to give "most informative sequence" as consensus

2005-02-10 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- move `alifold()` into the library

2004-12-22 Stephan Bernhart [berni@tbi.univie.ac.at](mailto:berni@tbi.univie.ac.at)

- add partition function version of RNACofold

2004-12-23 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- add RNAPalm for fast structural alignments (RNAPdist improvement)

2004-08-12 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- fix constrained folding in stochastic backtracking

2004-07-21 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- add RNAduplex, to compute hybrid structures without intra-molecular pairs

2004-02-09 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- fix bug in fold that caused segfaults when using Intel compiler
- add computation of ensemble diversity to RNAfold

2003-09-10 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- add annotation options to RNAPlot

2003-08-04 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- stochastic backtracking finally works. Try e.g. RNAsubopt -p 10

2003-07-18 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- add relplot.pl and rotate\_ss.pl utilities for reliability annotation and rotation of rna structure plots

2003-01-29 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- add RNALfold program to compute locally optimal structures with maximum pair span.
- add RNACofold for computing hybrid structure

2002-11-07 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- change `Make_bp_profile()` and `profile_edit_distance()` to use simple (float \*) arrays; makes Perl access much easier. RNAPdist -B now works again

2002-10-28 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- Improved Perl module with pod documentation; allow to write things like `($structure, $energy) = RNA::fold($seq);` Compatibility warning: the `ptrvalue()` and related functions are gone, see the pod documentation for alternatives.

2002-10-29 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- added svg structure plots in PS\_dot.c and RNAPlot

2002-08-15 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- Improve reading of clustal files (alifold)
- add a sample alifold.cgi script

2001-09-18 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- moved suboptimal folding into the library, thus it's now accessible from the Perl module

2001-08-31 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- added co-folding support in `energy_of_struct()`, and thus RNAeval

2001-04-30 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- switch from handcrafted makefiles to automake and autoconf

2001-04-05 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- added PS\_rna\_plot\_a to produce structure plots with annotation

2001-03-03 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- add alifold; predict consensus structures from alignment

2000-09-28 Ivo Hofacker [ivo@tbi.univie.ac.at](mailto:ivo@tbi.univie.ac.at)

- add -d3 option to RNAfold for co-axial stacking

# Chapter 11

## Deprecated List

### Global `alifold (const char **strings, char *structure)`

Usage of this function is discouraged! Use [vrna\\_alifold\(\)](#), or [vrna\\_mfe\(\)](#) instead!

### Global `alimake_pair_table (const char *structure)`

Use [vrna\\_pt\\_ali\\_get\(\)](#) instead!

### Global `alipbacktrack (double *prob)`

Use [vrna\\_pbacktrack\(\)](#) instead!

### Global `alipf_circ_fold (const char **sequences, char *structure, vrna_ep_t **pl)`

Use [vrna\\_pf\(\)](#) instead

### Global `alipf_fold (const char **sequences, char *structure, vrna_ep_t **pl)`

Use [vrna\\_pf\(\)](#) instead

### Global `alipf_fold_par (const char **sequences, char *structure, vrna_ep_t **pl, vrna_exp_param_t *parameters, int calculate_bppm, int is_constrained, int is_circular)`

Use [vrna\\_pf\(\)](#) instead

### Global `aliPS_color_align (const char *structure, const char *filename, const char *seqs[], const char *names[])`

Use [vrna\\_file\\_PS\\_align\(\)](#) instead!

### File `aln_util.h`

Use [ViennaRNA/utils/alignments.h](#) instead

### Global `assign plist from db (vrna_ep_t **pl, const char *struc, float pr)`

Use [vrna\\_plist\(\)](#) instead

### Global `assign plist from pr (vrna_ep_t **pl, FLT_OR_DBL *probs, int length, double cutoff)`

Use [vrna\\_plist\\_from\\_probs\(\)](#) instead!

### Global `b2C (const char *structure)`

See [vrna\\_db\\_to\\_tree\\_string\(\)](#) and [VRNA\\_STRUCTURE\\_TREE\\_SHAPIRO\\_SHORT](#) for a replacement

### Global `b2HIT (const char *structure)`

See [vrna\\_db\\_to\\_tree\\_string\(\)](#) and [VRNA\\_STRUCTURE\\_TREE\\_HIT](#) for a replacement

### Global `b2Shapiro (const char *structure)`

See [vrna\\_db\\_to\\_tree\\_string\(\)](#) and [VRNA\\_STRUCTURE\\_TREE\\_SHAPIRO\\_WEIGHT](#) for a replacement

### Global `base_pair`

Do not use this variable anymore!

### Global `bondT`

Use [vrna\\_bp\\_stack\\_t](#) instead!

### Global `bp_distance (const char *str1, const char *str2)`

Use [vrna\\_bp\\_distance](#) instead

**Global `bppm_symbol` (`const float *x`)**

Use `vrna_bpp_symbol()` instead!

**Global `bppm_to_structure` (`char *structure, FLT_OR_DBL *pr, unsigned int length`)**

Use `vrna_db_from_probs()` instead!

**Global `centroid` (`int length, double *dist`)**

This function is deprecated and should not be used anymore as it is not threadsafe!

**File `char_stream.h`**

Use `ViennaRNA/datastructures/char_stream.h` instead

**Global `circalifold` (`const char **strings, char *structure`)**

Usage of this function is discouraged! Use `vrna_alicircfold()`, and `vrna_mfe()` instead!

**Global `circfold` (`const char *sequence, char *structure`)**

Use `vrna_circfold()`, or `vrna_mfe()` instead!

**Global `co_pf_fold` (`char *sequence, char *structure`)**

{Use `vrna_pf_dimer()` instead!}

**Global `co_pf_fold_par` (`char *sequence, char *structure, vrna_exp_param_t *parameters, int calculate_bppm, int is_constrained`)**

Use `vrna_pf_dimer()` instead!

**Global `cofold` (`const char *sequence, char *structure`)**

use `vrna_mfe_dimer()` instead

**Global `cofold_par` (`const char *string, char *structure, vrna_param_t *parameters, int is_constrained`)**

use `vrna_mfe_dimer()` instead

**Global `compute_BPdifferences` (`short *pt1, short *pt2, unsigned int turn`)**

Use `vrna_refBPdist_matrix()` instead

**Global `compute_probabilities` (`double FAB, double FEA, double FEB, vrna_ep_t *prAB, vrna_ep_t *prA, vrna_ep_t *prB, int Alength`)**

{ Use `vrna_pf_dimer_probs()` instead! }

**Global `constrain_ptypes` (`const char *constraint, unsigned int length, char *ptype, int *BP, int min_loop_size, unsigned int idx_type`)**

Do not use this function anymore! Structure constraints are now handled through `vrna_hc_t` and related functions.

**File `constraints.h`**

Use `ViennaRNA/constraints/basic.h` instead

**File `constraints_hard.h`**

Use `ViennaRNA/constraints/hard.h` instead

**File `constraints_ligand.h`**

Use `ViennaRNA/constraints/ligand.h` instead

**File `constraints_SHAPE.h`**

Use `ViennaRNA/constraints/SHAPE.h` instead

**File `constraints_soft.h`**

Use `ViennaRNA/constraints/soft.h` instead

**File `convert_epars.h`**

Use `ViennaRNA/params/convert.h` instead

**Global `copy_pair_table` (`const short *pt`)**

Use `vrna_ptable_copy()` instead

**Global `cpair`**

Use `vrna_cpair_t` instead!

**Global cv\_fact**

See [vrna\\_md\\_t.cv\\_fact](#), and [vrna\\_mfe\(\)](#) to avoid using global variables

**File data\_structures.h**

Use [ViennaRNA/datastructures/basic.h](#) instead

**Global destroy\_TwoDfold\_variables (TwoDfold\_vars \*our\_variables)**

Use the new API that relies on [vrna\\_fold\\_compound\\_t](#) and the corresponding functions [vrna\\_fold\\_compound\\_TwoD\(\)](#), [vrna\\_mfe\\_TwoD\(\)](#), and [vrna\\_fold\\_compound\\_free\(\)](#) instead!

**Global destroy\_TwoDpfold\_variables (TwoDpfold\_vars \*vars)**

Use the new API that relies on [vrna\\_fold\\_compound\\_t](#) and the corresponding functions [vrna\\_fold\\_compound\\_TwoD\(\)](#), [vrna\\_pf\\_TwoD\(\)](#), and [vrna\\_fold\\_compound\\_free\(\)](#) instead!

**Global E\_Step (int type, int si1, int sj1, int extLoop, vrna\_param\_t \*P)**

Please use one of the functions [vrna\\_E\\_ext\\_stem\(\)](#) and [E\\_MLstem\(\)](#) instead! Use the former for cases where `extLoop != 0` and the latter otherwise.

**File energy\_const.h**

Use [ViennaRNA/params/constants.h](#) instead

**Global energy\_of\_astruct (const char \*\*sequences, const char \*structure, int n\_seq, float \*energy)**

Usage of this function is discouraged! Use [vrna\\_eval\\_structure\(\)](#), and [vrna\\_eval\\_covar\\_structure\(\)](#) instead!

**Global energy\_of\_circ\_struct (const char \*string, const char \*structure)**

This function is deprecated and should not be used in future programs! Use [energy\\_of\\_circ\\_structure\(\)](#) instead!

**Global energy\_of\_circ\_struct\_par (const char \*string, const char \*structure, vrna\_param\_t \*parameters, int verbosity\_level)**

Use [vrna\\_eval\\_structure\(\)](#) or [vrna\\_eval\\_structure\\_verbose\(\)](#) instead!

**Global energy\_of\_circ\_structure (const char \*string, const char \*structure, int verbosity\_level)**

Use [vrna\\_eval\\_structure\(\)](#) or [vrna\\_eval\\_structure\\_verbose\(\)](#) instead!

**Global energy\_of\_move (const char \*string, const char \*structure, int m1, int m2)**

Use [vrna\\_eval\\_move\(\)](#) instead!

**Global energy\_of\_move\_pt (short \*pt, short \*s, short \*s1, int m1, int m2)**

Use [vrna\\_eval\\_move\\_pt\(\)](#) instead!

**Global energy\_of\_struct (const char \*string, const char \*structure)**

This function is deprecated and should not be used in future programs! Use [energy\\_of\\_structure\(\)](#) instead!

**Global energy\_of\_struct\_par (const char \*string, const char \*structure, vrna\_param\_t \*parameters, int verbosity\_level)**

Use [vrna\\_eval\\_structure\(\)](#) or [vrna\\_eval\\_structure\\_verbose\(\)](#) instead!

**Global energy\_of\_struct\_pt (const char \*string, short \*ptable, short \*s, short \*s1)**

This function is deprecated and should not be used in future programs! Use [energy\\_of\\_structure\\_pt\(\)](#) instead!

**Global energy\_of\_struct\_pt\_par (const char \*string, short \*ptable, short \*s, short \*s1, vrna\_param\_t \*parameters, int verbosity\_level)**

Use [vrna\\_eval\\_structure\\_pt\(\)](#) or [vrna\\_eval\\_structure\\_pt\\_verbose\(\)](#) instead!

**Global energy\_of\_structure (const char \*string, const char \*structure, int verbosity\_level)**

Use [vrna\\_eval\\_structure\(\)](#) or [vrna\\_eval\\_structure\\_verbose\(\)](#) instead!

**Global energy\_of\_structure\_pt (const char \*string, short \*ptable, short \*s, short \*s1, int verbosity\_level)**

Use [vrna\\_eval\\_structure\\_pt\(\)](#) or [vrna\\_eval\\_structure\\_pt\\_verbose\(\)](#) instead!

**File energy\_par.h**

Use [ViennaRNA/params/default.h](#) instead

**Global exp\_E\_ExtLoop (int type, int si1, int sj1, vrna\_exp\_param\_t \*P)**

Use [vrna\\_exp\\_E\\_ext\\_stem\(\)](#) instead!

**Global expHairpinEnergy (int u, int type, short si1, short sj1, const char \*string)**

Use [exp\\_E\\_Hairpin\(\)](#) from [loop\\_energies.h](#) instead

**Global expLoopEnergy (int u1, int u2, int type, int type2, short si1, short sj1, short sp1, short sq1)**

Use [exp\\_E\\_IntLoop\(\)](#) from [loop\\_energies.h](#) instead

**Global export\_ali\_bppm (void)**

Usage of this function is discouraged! The new [vrna\\_fold\\_compound\\_t](#) allows direct access to the folding matrices, including the pair probabilities! The pair probability array returned here reflects the one of the latest call to [vrna\\_pf\(\)](#), or any of the old API calls for consensus structure partition function folding.

**Global export\_circfold\_arrays (int \*Fc\_p, int \*FcH\_p, int \*FcI\_p, int \*FcM\_p, int \*\*fM2\_p, int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*indx\_p, char \*\*ptype\_p)**

See [vrna\\_mfe\(\)](#) and [vrna\\_fold\\_compound\\_t](#) for the usage of the new API!

**Global export\_circfold\_arrays\_par (int \*Fc\_p, int \*FcH\_p, int \*FcI\_p, int \*FcM\_p, int \*\*fM2\_p, int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*indx\_p, char \*\*ptype\_p, vrna\_param\_t \*P\_p)**

See [vrna\\_mfe\(\)](#) and [vrna\\_fold\\_compound\\_t](#) for the usage of the new API!

**Global export\_co\_bppm (void)**

This function is deprecated and will be removed soon! The base pair probability array is available through the [vrna\\_fold\\_compound\\_t](#) data structure, and its associated [vrna\\_mx\\_pf\\_t](#) member.

**Global export\_cofold\_arrays (int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*fc\_p, int \*\*indx\_p, char \*\*ptype\_p)**

folding matrices now reside within the [vrna\\_fold\\_compound\\_t](#). Thus, this function will only work in conjunction with a prior call to the deprecated functions [cofold\(\)](#) or [cofold\\_par\(\)](#)

**Global export\_cofold\_arrays\_gq (int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*fc\_p, int \*\*ggg\_p, int \*\*indx\_p, char \*\*ptype\_p)**

folding matrices now reside within the fold compound. Thus, this function will only work in conjunction with a prior call to [cofold\(\)](#) or [cofold\\_par\(\)](#)

**Global export\_fold\_arrays (int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*indx\_p, char \*\*ptype\_p)**

See [vrna\\_mfe\(\)](#) and [vrna\\_fold\\_compound\\_t](#) for the usage of the new API!

**Global export\_fold\_arrays\_par (int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*indx\_p, char \*\*ptype\_p, vrna\_param\_t \*P\_p)**

See [vrna\\_mfe\(\)](#) and [vrna\\_fold\\_compound\\_t](#) for the usage of the new API!

**File exterior\_loops.h**

Use [ViennaRNA/loops/external.h](#) instead

**File file\_formats.h**

Use [ViennaRNA/io/file\\_formats.h](#) instead

**File file\_formats\_msa.h**

Use [ViennaRNA/io/file\\_formats\\_msa.h](#) instead

**File file\_utils.h**

Use [ViennaRNA/io/utils.h](#) instead

**Global filecopy (FILE \*from, FILE \*to)**

Use [vrna\\_file\\_copy\(\)](#) instead!

**Global find\_saddle (const char \*seq, const char \*s1, const char \*s2, int width)**

Use [vrna\\_path\\_findpath\\_saddle\(\)](#) instead!

**File findpath.h**

Use [ViennaRNA/landscape/findpath.h](#) instead

**Global fold (const char \*sequence, char \*structure)**

use [vrna\\_fold\(\)](#), or [vrna\\_mfe\(\)](#) instead!

**Global `fold_par`** (`const char *sequence, char *structure, vrna_param_t *parameters, int is_constrained, int is_circular`)  
 use `vrna_mfe()` instead!

#### **Global `free_alifold_arrays` (`void`)**

Usage of this function is discouraged! It only affects memory being free'd that was allocated by an old API function before. Release of memory occupied by the newly introduced `vrna_fold_compound_t` is handled by `vrna_fold_compound_free()`

#### **Global `free_alipf_arrays` (`void`)**

Usage of this function is discouraged! This function only free's memory allocated by old API function calls. Memory allocated by any of the new API calls (starting with `vrna_`) will be not affected!

#### **Global `free_arrays` (`void`)**

See `vrna_fold()`, `vrna_circfold()`, or `vrna_mfe()` and `vrna_fold_compound_t` for the usage of the new API!

#### **Global `free_co_arrays` (`void`)**

This function will only free memory allocated by a prior call of `cofold()` or `cofold_par()`. See `vrna_mfe_dimer()` for how to use the new API

#### **Global `free_co_pf_arrays` (`void`)**

This function will be removed for the new API soon! See `vrna_pf_dimer()`, `vrna_fold_compound()`, and `vrna_fold_compound_free()` for an alternative

#### **Global `free_path` (`vrna_path_t *path`)**

Use `vrna_path_free()` instead!

#### **Global `free_pf_arrays` (`void`)**

See `vrna_fold_compound_t` and its related functions for how to free memory occupied by the dynamic programming matrices

**Global `get_alipf_arrays`** (`short ***S_p, short ***S5_p, short ***S3_p, unsigned short ***a2s_p, char ***Ss_p, FLT_OR_DBL **qb_p, FLT_OR_DBL **qm_p, FLT_OR_DBL **q1k_p, FLT_OR_DBL **qln_p, short **pscore)`

It is discouraged to use this function! The new `vrna_fold_compound_t` allows direct access to all necessary consensus structure prediction related variables!

#### **Global `get_boltzmann_factor_copy` (`vrna_exp_param_t *parameters`)**

Use `vrna_exp_params_copy()` instead!

#### **Global `get_boltzmann_factors` (`double temperature, double betaScale, vrna_md_t md, double pf_scale`)**

Use `vrna_exp_params()` instead!

#### **Global `get_boltzmann_factors_ali` (`unsigned int n_seq, double temperature, double betaScale, vrna_md_t md, double pf_scale`)**

Use `vrna_exp_params_comparative()` instead!

#### **Global `get_centroid_struct_gquad_pr` (`int length, double *dist`)**

This function is deprecated and should not be used anymore as it is not threadsafe!

#### **Global `get_centroid_struct_pl` (`int length, double *dist, vrna_ep_t *pl`)**

This function was renamed to `vrna_centroid_from plist()`

#### **Global `get_centroid_struct_pr` (`int length, double *dist, FLT_OR_DBL *pr`)**

This function was renamed to `vrna_centroid_from_probs()`

#### **Global `get_concentrations` (`double FEAB, double FEEA, double FEBB, double FEA, double FEB, double *startconc`)**

{ Use `vrna_pf_dimer_concentrations()` instead!}

#### **Global `get_line` (`FILE *fp`)**

Use `vrna_read_line()` as a substitute!

#### **Global `get_mpi` (`char *Alseq[], int n_seq, int length, int *mini`)**

Use `vrna_aln_mpi()` as a replacement

**Global `get_path` (`const char *seq, const char *s1, const char *s2, int width`)**

Use `vrna_path_findpath()` instead!

**Global `get plist` (`vrna_ep_t *pl, int length, double cut_off`)**

{ This function is deprecated and will be removed soon! } use `assign_plist_from_pr()` instead!

**Global `get_scaled_alipf_parameters` (`unsigned int n_seq`)**

Use `vrna_exp_params_comparative()` instead!

**Global `get_scaled_parameters` (`double temperature, vrna_md_t md`)**

Use `vrna_params()` instead!

**Global `get_scaled_pf_parameters` (`void`)**

Use `vrna_exp_params()` instead!

**Global `get_TwoDfold_variables` (`const char *seq, const char *structure1, const char *structure2, int circ`)**

Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_TwoD()`, `vrna_mfe_TwoD()`, and `vrna_fold_compound_free()` instead!

**Global `get_TwoDpfold_variables` (`const char *seq, const char *structure1, char *structure2, int circ`)**

Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_TwoD()`, `vrna_pf_TwoD()`, and `vrna_fold_compound_free()` instead!

**File `hairpin_loops.h`**

Use `ViennaRNA/loops/hairpin.h` instead

**Global `HairpinE` (`int size, int type, int si1, int sj1, const char *string`)**

{ This function is deprecated and will be removed soon. Use `E_Hairpin()` instead! }

**Global `hamming` (`const char *s1, const char *s2`)**

Use `vrna_hamming_distance()` instead!

**Global `hamming_bound` (`const char *s1, const char *s2, int n`)**

Use `vrna_hamming_distance_bound()` instead!

**Global `iindx`**

Do not use this variable anymore!

**Global `init_co_pf_fold` (`int length`)**

{ This function is deprecated and will be removed soon! }

**Global `init_pf_fold` (`int length`)**

This function is obsolete and will be removed soon!

**Global `init_rand` (`void`)**

Use `vrna_init_rand()` instead!

**Global `initialize_cofold` (`int length`)**

{ This function is obsolete and will be removed soon! }

**Global `initialize_fold` (`int length`)**

See `vrna_mfe()` and `vrna_fold_compound_t` for the usage of the new API!

**Global `int_urn` (`int from, int to`)**

Use `vrna_int_urn()` instead!

**File `interior_loops.h`**

Use `ViennaRNA/loops/internal.h` instead

**Global `Lfold` (`const char *string, const char *structure, int maxdist`)**

Use `vrna_mfe_window()` instead!

**Global `Lfoldz` (`const char *string, const char *structure, int maxdist, int zsc, double min_z`)**

Use `vrna_mfe_window_zscore()` instead!

**File [loop\\_energies.h](#)**

Use [ViennaRNA/loops/all.h](#) instead

**Global [loop\\_energy](#) (`short *ptable, short *s, short *s1, int i`)**

Use [vrna\\_eval\\_loop\\_pt\(\)](#) instead!

**Global [LoopEnergy](#) (`int n1, int n2, int type, int type_2, int si1, int sj1, int sp1, int sq1`)**

{This function is deprecated and will be removed soon. Use [E\\_IntLoop\(\)](#) instead!}

**Global [Make\\_bp\\_profile](#) (`int length`)**

This function is deprecated and will be removed soon! See [Make\\_bp\\_profile\\_bppm\(\)](#) for a replacement

**Global [make\\_pair\\_table](#) (`const char *structure`)**

Use [vrna\\_ptable\(\)](#) instead

**Global [make\\_pair\\_table\\_snoop](#) (`const char *structure`)**

Use [vrna\\_pt\\_snoop\\_get\(\)](#) instead!

**Global [make\\_referenceBP\\_array](#) (`short *reference_pt, unsigned int turn`)**

Use [vrna\\_refBPcnt\\_matrix\(\)](#) instead

**Global [MEA](#) (`plist *p, char *structure, double gamma`)**

Use [vrna\\_MEAs\(\)](#) or [vrna\\_MEAs\\_from\\_plist\(\)](#) instead!

**Global [mean\\_bp\\_dist](#) (`int length`)**

This function is not threadsafe and should not be used anymore. Use [mean\\_bp\\_distance\(\)](#) instead!

**Global [mean\\_bp\\_distance](#) (`int length`)**

Use [vrna\\_mean\\_bp\\_distance\(\)](#) or [vrna\\_mean\\_bp\\_distance\\_pr\(\)](#) instead!

**Global [mean\\_bp\\_distance\\_pr](#) (`int length, FLT_OR_DBL *pr`)**

Use [vrna\\_mean\\_bp\\_distance\(\)](#) or [vrna\\_mean\\_bp\\_distance\\_pr\(\)](#) instead!

**File [multibranch\\_loops.h](#)**

Use [ViennaRNA/loops/multibranch.h](#) instead

**File [naview.h](#)**

Use [ViennaRNA/plotting/naview/naview.h](#) instead

**Global [nc\\_fact](#)**

See [vrna\\_md\\_t.nc\\_fact](#), and [vrna\\_mfe\(\)](#) to avoid using global variables

**File [neighbor.h](#)**

Use [ViennaRNA/landscape/neighbor.h](#) instead

**Global [nrerror](#) (`const char message[]`)**

Use [vrna\\_message\\_error\(\)](#) instead!

**Global [pack\\_structure](#) (`const char *struc`)**

Use [vrna\\_db\\_pack\(\)](#) as a replacement

**Global [PAIR](#)**

Use [vrna\\_basepair\\_t](#) instead!

**Global [pair\\_info](#)**

Use [vrna\\_pinfo\\_t](#) instead!

**File [params.h](#)**

Use [ViennaRNA/params/basic.h](#) instead

**Global [paramT](#)**

Use [vrna\\_param\\_t](#) instead!

**Global [parenthesis\\_structure](#) (`char *structure, vrna_bp_stack_t *bp, int length`)**

use [vrna\\_parenthesis\\_structure\(\)](#) instead

**Global parenthesis\_zuker** (`char *structure, vrna_bp_stack_t *bp, int length`)

use `vrna_parenthesis_zuker` instead

**Global path\_t**

Use `vrna_path_t` instead!

**Global pbacktrack\_circ** (`char *sequence`)

Use `vrna_pbacktrack()` instead.

**Global pf\_circ\_fold** (`const char *sequence, char *structure`)

Use `vrna_pf()` instead!

**Global pf\_fold\_par** (`const char *sequence, char *structure, vrna_exp_param_t *parameters, int calculate_bppm, int is_constrained, int is_circular`)

Use `vrna_pf()` instead

**Global pf\_paramT**

Use `vrna_exp_param_t` instead!

**Global plist**

Use `vrna_ep_t` or `vrna_elem_prob_s` instead!

**File plot\_aln.h**

Use `ViennaRNA/plotting/alignments.h` instead

**File plot\_layouts.h**

Use `ViennaRNA/plotting/layouts.h` instead

**File plot\_structure.h**

Use `ViennaRNA/plotting/structures.h` instead

**File plot\_utils.h**

Use `ViennaRNA/plotting/utils.h` instead

**Global pr**

Do not use this variable anymore!

**Global print\_tty\_constraint** (`unsigned int option`)

Use `vrna_message_constraints()` instead!

**Global print\_tty\_constraint\_full** (`void`)

Use `vrna_message_constraint_options_all()` instead!

**Global print\_tty\_input\_seq** (`void`)

Use `vrna_message_input_seq_simple()` instead!

**Global print\_tty\_input\_seq\_str** (`const char *s`)

Use `vrna_message_input_seq()` instead!

**Global PS\_color\_aln** (`const char *structure, const char *filename, const char *seqs[], const char *names[]`)

Use `vrna_file_PS_aln()` instead!

**File PS\_dot.h**

Use `ViennaRNA/plotting/probabilities.h` instead

**Global PS\_dot\_plot** (`char *string, char *file`)

This function is deprecated and will be removed soon! Use `PS_dot_plot_list()` instead!

**Global PS\_rna\_plot** (`char *string, char *structure, char *file`)

Use `vrna_file_PS_rnplot()` instead!

**Global PS\_rna\_plot\_a** (`char *string, char *structure, char *file, char *pre, char *post`)

Use `vrna_file_PS_rnplot_a()` instead!

**Global PS\_rna\_plot\_a\_gquad** (`char *string, char *structure, char *ssfile, char *pre, char *post`)

Use `vrna_file_PS_rnplot_a()` instead!

**Global random\_string (int l, const char symbols[])**

Use [vrna\\_random\\_string\(\)](#) instead!

**File read\_epars.h**

Use [ViennaRNA/params/io.h](#) instead

**Global read\_parameter\_file (const char fname[])**

Use [vrna\\_params\\_load\(\)](#) instead!

**Global read\_record (char \*\*header, char \*\*sequence, char \*\*\*rest, unsigned int options)**

This function is deprecated! Use [vrna\\_file\\_fasta\\_read\\_record\(\)](#) as a replacement.

**Global scale\_parameters (void)**

Use [vrna\\_params\(\)](#) instead!

**Global sect**

Use [vrna\\_sect\\_t](#) instead!

**Global set\_model\_details (vrna\_md\_t \*md)**

This function will vanish as soon as backward compatibility of RNAlib is dropped (expected in version 3). Use [vrna\\_md\\_set\\_default\(\)](#) instead!

**Global simple\_circplot\_coordinates (short \*pair\_table, float \*x, float \*y)**

Consider switching to [vrna\\_plot\\_coords\\_circular\\_pt\(\)](#) instead!

**Global simple\_xy\_coordinates (short \*pair\_table, float \*X, float \*Y)**

Consider switching to [vrna\\_plot\\_coords\\_simple\\_pt\(\)](#) instead!

**Global SOLUTION**

Use [vrna\\_subopt\\_solution\\_t](#) instead!

**Global space (unsigned size)**

Use [vrna\\_alloc\(\)](#) instead!

**Global st\_back**

set the *uniq\_ML* flag in [vrna\\_md\\_t](#) before passing it to [vrna\\_fold\\_compound\(\)](#).

**Global stackProb (double cutoff)**

Use [vrna\\_stack\\_prob\(\)](#) instead!

**Global str\_DNA2RNA (char \*sequence)**

Use [vrna\\_seq\\_toRNA\(\)](#) instead!

**Global str\_uppercase (char \*sequence)**

Use [vrna\\_seq\\_toupper\(\)](#) instead!

**File stream\_output.h**

Use [ViennaRNA/datastructures/stream\\_output.h](#) instead

**File string\_utils.h**

Use [ViennaRNA/utils/strings.h](#) instead

**File structure\_utils.h**

Use [ViennaRNA/utils/structures.h](#) instead

**File svm\_utils.h**

Use [ViennaRNA/utils/svm.h](#) instead

**Global temperature**

Use [vrna\\_md\\_defaults\\_temperature\(\)](#), and [vrna\\_md\\_defaults\\_temperature\\_get\(\)](#) to change, and read the global default temperature settings

**Global time\_stamp (void)**

Use [vrna\\_time\\_stamp\(\)](#) instead!

**Global `TwoDfold_backtrack_f5` (`unsigned int j, int k, int l, TwoDfold_vars *vars`)**

Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_TwoD()`, `vrna_mfe_TwoD()`, `vrna_backtrack5_TwoD()`, and `vrna_fold_compound_free()` instead!

**Global `TwoDfold_vars`**

This data structure will be removed from the library soon! Use `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_TwoD()`, `vrna_mfe_TwoD()`, and `vrna_fold_compound_free()` instead!

**Global `TwoDfoldList` (`TwoDfold_vars *vars, int distance1, int distance2`)**

Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_TwoD()`, `vrna_mfe_TwoD()`, and `vrna_fold_compound_free()` instead!

**Global `TwoDpfold_pbacktrack` (`TwoDpfold_vars *vars, int d1, int d2`)**

Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_TwoD()`, `vrna_pf_TwoD()`, `vrna_pbacktrack_TwoD()`, and `vrna_fold_compound_free()` instead!

**Global `TwoDpfold_pbacktrack5` (`TwoDpfold_vars *vars, int d1, int d2, unsigned int length`)**

Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_TwoD()`, `vrna_pf_TwoD()`, `vrna_pbacktrack5_TwoD()`, and `vrna_fold_compound_free()` instead!

**Class `TwoDpfold_vars`**

This data structure will be removed from the library soon! Use `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_TwoD()`, `vrna_pf_TwoD()`, and `vrna_fold_compound_free()` instead!

**Global `TwoDpfoldList` (`TwoDpfold_vars *vars, int maxDistance1, int maxDistance2`)**

Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_TwoD()`, `vrna_pf_TwoD()`, and `vrna_fold_compound_free()` instead!

**File `units.h`**

Use `ViennaRNA/utils/units.h` instead

**Global `unpack_structure` (`const char *packed`)**

Use `vrna_db_unpack()` as a replacement

**Global `update_alifold_params` (`void`)**

Usage of this function is discouraged! The new API uses `vrna_fold_compound_t` to lump all folding related necessities together, including the energy parameters. Use `vrna_update_fold_params()` to update the energy parameters within a `vrna_fold_compound_t`.

**Global `update_co_pf_params` (`int length`)**

Use `vrna_exp_params_subst()` instead!

**Global `update_co_pf_params_par` (`int length, vrna_exp_param_t *parameters`)**

Use `vrna_exp_params_subst()` instead!

**Global `update_cofold_params` (`void`)**

See `vrna_params_subst()` for an alternative using the new API

**Global `update_cofold_params_par` (`vrna_param_t *parameters`)**

See `vrna_params_subst()` for an alternative using the new API

**Global `update_fold_params` (`void`)**

For non-default model settings use the new API with `vrna_params_subst()` and `vrna_mfe()` instead!

**Global `update_fold_params_par` (`vrna_param_t *parameters`)**

For non-default model settings use the new API with `vrna_params_subst()` and `vrna_mfe()` instead!

**Global `update_pf_params` (`int length`)**

Use `vrna_exp_params_subst()` instead

**Global `update_pf_params_par` (`int length, vrna_exp_param_t *parameters`)**

Use `vrna_exp_params_subst()` instead

**Global `urn` (`void`)**

Use `vrna_urn()` instead!

**File utils.h**

Use [ViennaRNA/utils/basic.h](#) instead

Use [ViennaRNA/utils/basic.h](#) instead

**Global vrna\_cofold (const char \*sequence, char \*structure)**

This function is obsolete since [vrna\\_mfe\(\)](#)/[vrna\\_fold\(\)](#) can handle complexes multiple sequences since v2.5.0.

Use [vrna\\_mfe\(\)](#)/[vrna\\_fold\(\)](#) for connected component MFE instead and compute MFEs of unconnected states separately.

**Global VRNA\_CONSTRAINT\_FILE**

Use 0 instead!

**Global VRNA\_CONSTRAINT\_MULTILINE**

see [vrna\\_extract\\_record\\_rest\\_structure\(\)](#)

**Global VRNA\_CONSTRAINT\_NO\_HEADER**

This mode is not supported anymore!

**Global VRNA\_CONSTRAINT\_SOFT\_MFE**

This flag has no meaning anymore, since constraints are now always stored!

**Global VRNA\_CONSTRAINT\_SOFT\_PF**

Use [VRNA\\_OPTION\\_PF](#) instead!

**Global vrna\_exp\_param\_s::id**

This attribute will be removed in version 3

**Global vrna\_extract\_record\_rest\_constraint (char \*\*cstruc, const char \*\*lines, unsigned int option)**

Use [vrna\\_extract\\_record\\_rest\\_structure\(\)](#) instead!

**Global vrna\_fc\_s::pscore\_pf\_compat**

This attribute will vanish in the future!

**Global vrna\_fc\_s::ptype\_pf\_compat**

This attribute will vanish in the future! It's meant for backward compatibility only!

**Global vrna\_mfe\_dimer (vrna\_fold\_compound\_t \*vc, char \*structure)**

This function is obsolete since [vrna\\_mfe\(\)](#) can handle complexes multiple sequences since v2.5.0. Use [vrna\\_mfe\(\)](#) for connected component MFE instead and compute MFEs of unconnected states separately.

**File walk.h**

Use [ViennaRNA/landscape/walk.h](#) instead

**Global warn\_user (const char message[])**

Use [vrna\\_message\\_warning\(\)](#) instead!

**Global write\_parameter\_file (const char fname[])**

Use [vrna\\_params\\_save\(\)](#) instead!

**Global xrealloc (void \*p, unsigned size)**

Use [vrna\\_realloc\(\)](#) instead!

**Global zukersubopt (const char \*string)**

use [vrna\\_zukersubopt\(\)](#) instead

**Global zukersubopt\_par (const char \*string, vrna\_param\_t \*parameters)**

use [vrna\\_zukersubopt\(\)](#) instead



# Chapter 12

## Bug List

### Module `domains_up`

Although the additional production rule(s) for unstructured domains as described in [Unstructured Domains](#) are always treated as 'segments possibly bound to one or more ligands', the current implementation requires that at least one ligand is bound. The default implementation already takes care of the required changes, however, upon using callback functions other than the default ones, one has to take care of this fact. Please also note, that this behavior might change in one of the next releases, such that the decomposition schemes as shown above comply with the actual implementation.

### Global `VRNA_PROBS_WINDOW_STACKP`

Currently, this flag is a placeholder doing nothing as the corresponding implementation for stack probability computation is missing.

### Global `vrna_subopt_zuker (vrna_fold_compound_t *fc)`

Due to resizing, any pre-existing constraints will be lost!



# Chapter 13

## Module Index

### 13.1 The RNAlib API

Our library is grouped into several modules, each addressing different aspects of RNA secondary structure related problems. You can find an overview of the different groups below.

|   |     |
|---|-----|
| Free Energy Evaluation . . . . .  | 133 |
| Energy Evaluation for Individual Loops . . . . .  | 153 |
| Exterior Loops . . . . .  | 369 |
| Hairpin Loops . . . . .   | 372 |
| Internal Loops . . . . .  | 376 |
| Multibranch Loops . . . . .   | 377 |
| Energy Evaluation for Atomic Moves . . . . .  | 155 |
| Deprecated Interface for Free Energy Evaluation . . . . .                                 | 156 |
| The RNA Folding Grammar . . . . .   | 168 |
| Fine-tuning of the Implemented Models . . . . .   | 169 |
| Energy Parameters . . . . .   | 196 |
| Reading/Writing Energy Parameter Sets from/to File . . . . .                              | 382 |
| Converting Energy Parameter Files . . . . .   | 388 |
| Extending the Folding Grammar with Additional Domains . . . . .                           | 207 |
| Unstructured Domains . . . . .  | 207 |
| Structured Domains . . . . .  | 216 |
| Constraining the RNA Folding Grammar . . . . .  | 216 |
| Hard Constraints . . . . .  | 229 |
| Soft Constraints . . . . .  | 237 |
| The RNA Secondary Structure Landscape . . . . .   | 247 |
| Neighborhood Relation and Move Sets for Secondary Structures . . . . .                    | 324 |
| (Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima . . . . .            | 335 |
| Direct Refolding Paths between two Secondary Structures . . . . .                         | 338 |
| Folding Paths that start at a single Secondary Structure . . . . .                        | 343 |
| Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers . . . . . | 574 |
| Minimum Free Energy (MFE) Algorithms . . . . .  | 247 |
| Global MFE Prediction . . . . .   | 249 |
| Computing MFE representatives of a Distance Based Partitioning . . . . .                  | 304 |
| Deprecated Interface for Global MFE Prediction . . . . .                                  | 532 |
| Local (sliding window) MFE Prediction . . . . .   | 254 |
| Deprecated Interface for Local (Sliding Window) MFE Prediction . . . . .                  | 542 |
| Backtracking MFE structures . . . . .   | 258 |
| Partition Function and Equilibrium Properties . . . . .                                   | 248 |
| Global Partition Function and Equilibrium Probabilities . . . . .                         | 260 |
| Computing Partition Functions of a Distance Based Partitioning . . . . .                  | 310 |
| Predicting various thermodynamic properties . . . . .                                     | 313 |
| Deprecated Interface for Global Partition Function Computation . . . . .                  | 543 |

|  |     |
|--|-----|
| Local (sliding window) Partition Function and Equilibrium Probabilities . . . . .                    | 267 |
| Deprecated Interface for Local (Sliding Window) Partition Function Computation . . . . .             | 557 |
| Suboptimals and Representative Structures . . . . .  | 274 |
| Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989 . . . . .                       | 274 |
| Suboptimal Structures within an Energy Band around the MFE . . . . .                                 | 276 |
| Random Structure Samples from the Ensemble . . . . .   | 279 |
| Stochastic Backtracking of Structures from Distance Based Partitioning . . . . .                     | 311 |
| Deprecated Interface for Stochastic Backtracking . . . . .   | 559 |
| Compute the Structure with Maximum Expected Accuracy (MEA) . . . . .                                 | 299 |
| Compute the Centroid Structure . . . . .   | 301 |
| RNA-RNA Interaction . . . . .  | 303 |
| Partition Function for Two Hybridized Sequences . . . . .  | 378 |
| Partition Function for two Hybridized Sequences as a Stepwise Process . . . . .                      | 380 |
| Classified Dynamic Programming Variants . . . . .  | 303 |
| Distance Based Partitioning of the Secondary Structure Space . . . . .                               | 304 |
| Computing MFE representatives of a Distance Based Partitioning . . . . .                             | 304 |
| Computing Partition Functions of a Distance Based Partitioning . . . . .                             | 310 |
| Stochastic Backtracking of Structures from Distance Based Partitioning . . . . .                     | 311 |
| Compute the Density of States . . . . .  | 322 |
| Inverse Folding (Design) . . . . .   | 323 |
| Experimental Structure Probing Data . . . . .  | 347 |
| SHAPE Reactivity Data . . . . .  | 347 |
| Generate Soft Constraints from Data . . . . .  | 350 |
| Ligands Binding to RNA Structures . . . . .  | 354 |
| Ligands Binding to Unstructured Domains . . . . .  | 354 |
| Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints . . . . . | 354 |
| Structure Modules and Pseudoknots . . . . .  | 356 |
| Pseudoknots . . . . .  | 356 |
| G-Quadruplexes . . . . .   | 361 |
| Utilities . . . . .  | 363 |
| Utilities to deal with Nucleotide Alphabets . . . . .  | 392 |
| (Nucleic Acid Sequence) String Utilitites . . . . .  | 396 |
| Secondary Structure Utilities . . . . .  | 406 |
| Dot-Bracket Notation of Secondary Structures . . . . .   | 408 |
| Washington University Secondary Structure (WUSS) notation . . . . .                                  | 414 |
| Pair Table Representation of Secondary Structures . . . . .  | 415 |
| Pair List Representation of Secondary Structures . . . . .   | 419 |
| Abstract Shapes Representation of Secondary Structures . . . . .                                     | 421 |
| Helix List Representation of Secondary Structures . . . . .  | 423 |
| Tree Representation of Secondary Structures . . . . .  | 424 |
| Distance measures between Secondary Structures . . . . .   | 428 |
| Deprecated Interface for Secondary Structure Utilities . . . . .                                     | 564 |
| Multiple Sequence Alignment Utilities . . . . .  | 429 |
| Deprecated Interface for Multiple Sequence Alignment Utilities . . . . .                             | 561 |
| Files and I/O . . . . .  | 436 |
| Nucleic Acid Sequences and Structures . . . . .  | 438 |
| Multiple Sequence Alignments . . . . .   | 444 |
| Command Files . . . . .  | 451 |
| Plotting . . . . .   | 455 |
| Layouts and Coordinates . . . . .  | 461 |
| Annotation . . . . .   | 474 |
| Alignment Plots . . . . .  | 474 |
| Deprecated Interface for Plotting Utilities . . . . .  | 572 |
| Search Algorithms . . . . .  | 476 |
| Combinatorics Algorithms . . . . .   | 479 |

|  |     |
|--|-----|
| (Abstract) Data Structures . . . . .       | 485 |
| The Fold Compound . . . . .                | 497 |
| The Dynamic Programming Matrices . . . . . | 510 |
| Hash Tables . . . . .                      | 514 |
| Heaps . . . . .                            | 521 |
| Buffers . . . . .                          | 528 |
| Messages . . . . .                         | 490 |
| Unit Conversion . . . . .                  | 494 |



# Chapter 14

## Data Structure Index

### 14.1 Data Structures

Here are the data structures with brief descriptions:

|   |  |     |
|---|--|-----|
| <code>_struct_en</code>                       | Data structure for <code>energy_of_move()</code>         | 577 |
| <code>LIST</code>                             |  | 577 |
| <code>LST_BUCKET</code>                       |  | 577 |
| <code>Postorder_list</code>                   | Postorder data structure                                 | 577 |
| <code>swString</code>                         | Some other data structure                                | 577 |
| <code>Tree</code>                             | <code>Tree</code> data structure                         | 578 |
| <code>TwoDpfold_vars</code>                   | Variables compound for 2Dfold partition function folding | 578 |
| <code>vrna_dimer_conc_s</code>                | Data structure for concentration dependency computations | 579 |
| <code>vrna_sc_bp_storage_t</code>             | A base pair constraint                                   | 579 |
| <code>vrna_structured_domains_s</code>        |  | 579 |
| <code>vrna_subopt_sol_s</code>                | Solution element from subopt.c                           | 579 |
| <code>vrna_unstructured_domain_motif_s</code> |  | 580 |



# Chapter 15

## File Index

### 15.1 File List

Here is a list of all documented files with brief descriptions:

|   |   |     |
|---|---|-----|
| ViennaRNA/ <a href="#">2Dfold.h</a>             | MFE structures for base pair distance classes . . . . .   | 581 |
| ViennaRNA/ <a href="#">2Dpfold.h</a>            | Partition function implementations for base pair distance classes . . . . .   | 583 |
| ViennaRNA/ <a href="#">ali_plex.h</a>           | . . . . .   | 589 |
| ViennaRNA/ <a href="#">alifold.h</a>            | Functions for comparative structure prediction using RNA sequence alignments . . . . .                                | 589 |
| ViennaRNA/ <a href="#">aln_util.h</a>           | Use <a href="#">ViennaRNA/utils/alignments.h</a> instead . . . . .  | 593 |
| ViennaRNA/ <a href="#">alphabet.h</a>           | Functions to process, convert, and generally handle different nucleotide and/or base pair alphabets . . . . .         | 593 |
| ViennaRNA/ <a href="#">boltzmann_sampling.h</a> | Boltzmann Sampling of secondary structures from the ensemble . . . . .  | 595 |
| ViennaRNA/ <a href="#">centroid.h</a>           | Centroid structure computation . . . . .  | 598 |
| ViennaRNA/ <a href="#">char_stream.h</a>        | Use <a href="#">ViennaRNA/datastructures/char_stream.h</a> instead . . . . .  | 600 |
| ViennaRNA/ <a href="#">cofold.h</a>             | MFE implementations for RNA-RNA interaction . . . . .   | 603 |
| ViennaRNA/ <a href="#">combinatorics.h</a>      | Various implementations that deal with combinatorial aspects of objects . . . . .                                     | 604 |
| ViennaRNA/ <a href="#">commands.h</a>           | Parse and apply different commands that alter the behavior of secondary structure prediction and evaluation . . . . . | 606 |
| ViennaRNA/ <a href="#">concentrations.h</a>     | Concentration computations for RNA-RNA interactions . . . . .   | 607 |
| ViennaRNA/ <a href="#">constraints.h</a>        | Use <a href="#">ViennaRNA/constraints/basic.h</a> instead . . . . .   | 609 |
| ViennaRNA/ <a href="#">constraints_hard.h</a>   | Use <a href="#">ViennaRNA/constraints/hard.h</a> instead . . . . .  | 623 |
| ViennaRNA/ <a href="#">constraints_ligand.h</a> | Use <a href="#">ViennaRNA/constraints/ligand.h</a> instead . . . . .  | 623 |
| ViennaRNA/ <a href="#">constraints_SHAPE.h</a>  | Use <a href="#">ViennaRNA/constraints/SHAPE.h</a> instead . . . . .   | 624 |
| ViennaRNA/ <a href="#">constraints_soft.h</a>   | Use <a href="#">ViennaRNA/constraints/soft.h</a> instead . . . . .  | 624 |
| ViennaRNA/ <a href="#">convert_epars.h</a>      | Use <a href="#">ViennaRNA/params/convert.h</a> instead . . . . .  | 625 |

|  |   |     |
|--|---|-----|
| ViennaRNA/ <a href="#">data_structures.h</a>   | Use ViennaRNA/ <a href="#">datastructures/basic.h</a> instead . . . . .   | 625 |
| ViennaRNA/ <a href="#">dist_vars.h</a>         | Global variables for Distance-Package . . . . .   | 630 |
| ViennaRNA/ <a href="#">dp_matrices.h</a>       | Functions to deal with standard dynamic programming (DP) matrices . . . . .   | 631 |
| ViennaRNA/ <a href="#">duplex.h</a>            | Functions for simple RNA-RNA duplex interactions . . . . .  | 635 |
| ViennaRNA/ <a href="#">edit_cost.h</a>         | Global variables for Edit Costs included by treedist.c and stringdist.c . . . . .                                   | 636 |
| ViennaRNA/ <a href="#">energy_const.h</a>      | Use ViennaRNA/ <a href="#">params/constants.h</a> instead . . . . .   | 637 |
| ViennaRNA/ <a href="#">energy_par.h</a>        | Use ViennaRNA/ <a href="#">params/default.h</a> instead . . . . .   | 637 |
| ViennaRNA/ <a href="#">equilibrium_probs.h</a> | Equilibrium Probability implementations . . . . .   | 638 |
| ViennaRNA/ <a href="#">eval.h</a>              | Functions and variables related to energy evaluation of sequence/structure pairs . . . . .                          | 639 |
| ViennaRNA/ <a href="#">exterior_loops.h</a>    | Use ViennaRNA/ <a href="#">loops/external.h</a> instead . . . . .   | 646 |
| ViennaRNA/ <a href="#">file_formats.h</a>      | Use ViennaRNA/ <a href="#">io/file_formats.h</a> instead . . . . .  | 647 |
| ViennaRNA/ <a href="#">file_formats_msa.h</a>  | Use ViennaRNA/ <a href="#">io/file_formats_msa.h</a> instead . . . . .  | 649 |
| ViennaRNA/ <a href="#">file_utils.h</a>        | Use ViennaRNA/ <a href="#">io/utils.h</a> instead . . . . .   | 652 |
| ViennaRNA/ <a href="#">findpath.h</a>          | Use ViennaRNA/ <a href="#">landscape/findpath.h</a> instead . . . . .   | 652 |
| ViennaRNA/ <a href="#">fold.h</a>              | MFE calculations for single RNA sequences . . . . .   | 654 |
| ViennaRNA/ <a href="#">fold_compound.h</a>     | The Basic Fold Compound API . . . . .   | 656 |
| ViennaRNA/ <a href="#">fold_vars.h</a>         | Here all all declarations of the global variables used throughout RNAlib . . . . .                                  | 660 |
| ViennaRNA/ <a href="#">gquad.h</a>             | G-quadruplexes . . . . .  | 662 |
| ViennaRNA/ <a href="#">grammar.h</a>           | Implementations for the RNA folding grammar . . . . .   | 682 |
| ViennaRNA/ <a href="#">hairpin_loops.h</a>     | Use ViennaRNA/ <a href="#">loops/hairpin.h</a> instead . . . . .  | 683 |
| ViennaRNA/ <a href="#">heat_capacity.h</a>     | Compute heat capacity for an RNA . . . . .  | 684 |
| ViennaRNA/ <a href="#">interior_loops.h</a>    | Use ViennaRNA/ <a href="#">loops/internal.h</a> instead . . . . .   | 685 |
| ViennaRNA/ <a href="#">inverse.h</a>           | Inverse folding routines . . . . .  | 685 |
| ViennaRNA/ <a href="#">Lfold.h</a>             | Functions for locally optimal MFE structure prediction . . . . .  | 690 |
| ViennaRNA/ <a href="#">loop_energies.h</a>     | Use ViennaRNA/ <a href="#">loops/all.h</a> instead . . . . .  | 690 |
| ViennaRNA/ <a href="#">LPfold.h</a>            | Partition function and equilibrium probability implementation for the sliding window algorithm . . . . .            | 707 |
| ViennaRNA/ <a href="#">MEA.h</a>               | Computes a MEA (maximum expected accuracy) structure . . . . .  | 709 |
| ViennaRNA/ <a href="#">mfe.h</a>               | Compute Minimum Free energy (MFE) and backtrace corresponding secondary structures from RNA sequence data . . . . . | 710 |

|  |  |      |
|--|--|------|
| ViennaRNA/ <a href="#">mfe_window.h</a>        | Compute local Minimum Free Energy (MFE) using a sliding window approach and backtrace corresponding secondary structures . . . . .   | 712  |
| ViennaRNA/ <a href="#">mm.h</a>                | Several Maximum Matching implementations . . . . .   | 714  |
| ViennaRNA/ <a href="#">model.h</a>             | The model details data structure and its corresponding modifiers . . . . .   | 715  |
| ViennaRNA/ <a href="#">move_set.h</a>          |  | 723  |
| ViennaRNA/ <a href="#">multibranch_loops.h</a> | Use ViennaRNA/loops/multibranch.h instead . . . . .  | 725  |
| ViennaRNA/ <a href="#">naview.h</a>            | Use ViennaRNA/plotting/naview/naview.h instead . . . . .   | 725  |
| ViennaRNA/ <a href="#">neighbor.h</a>          | Use ViennaRNA/landscape/neighbor.h instead . . . . .   | 727  |
| ViennaRNA/ <a href="#">pair_mat.h</a>          |  | 728  |
| ViennaRNA/ <a href="#">params.h</a>            | Use ViennaRNA/params/basic.h instead . . . . .   | 730  |
| ViennaRNA/ <a href="#">part_func.h</a>         | Partition function implementations . . . . .   | 1189 |
| ViennaRNA/ <a href="#">part_func_co.h</a>      | Partition function for two RNA sequences . . . . .   | 1195 |
| ViennaRNA/ <a href="#">part_func_up.h</a>      | Implementations for accessibility and RNA-RNA interaction as a stepwise process . . . . .  | 1197 |
| ViennaRNA/ <a href="#">part_func_window.h</a>  | Partition function and equilibrium probability implementation for the sliding window algorithm .   | 1198 |
| ViennaRNA/ <a href="#">perturbation_fold.h</a> | Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments . . . . . | 1200 |
| ViennaRNA/ <a href="#">pf_multifold.h</a>      |  | 1202 |
| ViennaRNA/ <a href="#">pk_plex.h</a>           | Heuristics for two-step pseudoknot forming interaction predictions . . . . .   | 1202 |
| ViennaRNA/ <a href="#">PKplex.h</a>            |  | 1203 |
| ViennaRNA/ <a href="#">plex.h</a>              |  | 1204 |
| ViennaRNA/ <a href="#">plot_aln.h</a>          | Use ViennaRNA/plotting/alignments.h instead . . . . .  | 1205 |
| ViennaRNA/ <a href="#">plot_layouts.h</a>      | Use ViennaRNA/plotting/layouts.h instead . . . . .   | 1205 |
| ViennaRNA/ <a href="#">plot_structure.h</a>    | Use ViennaRNA/plotting/structures.h instead . . . . .  | 1205 |
| ViennaRNA/ <a href="#">plot_utils.h</a>        | Use ViennaRNA/plotting/utils.h instead . . . . .   | 1206 |
| ViennaRNA/ <a href="#">ProfileAIn.h</a>        |  | 1227 |
| ViennaRNA/ <a href="#">profiledist.h</a>       |  | 1227 |
| ViennaRNA/ <a href="#">PS_dot.h</a>            | Use ViennaRNA/plotting/probabilities.h instead . . . . .   | 1229 |
| ViennaRNA/ <a href="#">read_epars.h</a>        |  | 1229 |
| ViennaRNA/ <a href="#">ribo.h</a>              | Parse RiboSum Scoring Matrices for Covariance Scoring of Alignments . . . . .  | 1230 |
| ViennaRNA/ <a href="#">RNAstruct.h</a>         | Parsing and Coarse Graining of Structures . . . . .  | 1230 |
| ViennaRNA/ <a href="#">sequence.h</a>          | Functions and data structures related to sequence representations , . . . . .  | 1233 |
| ViennaRNA/ <a href="#">snofold.h</a>           |  | 1235 |
| ViennaRNA/ <a href="#">snoop.h</a>             |  | 1236 |
| ViennaRNA/ <a href="#">special_const.h</a>     |  | 1239 |
| ViennaRNA/ <a href="#">stream_output.h</a>     | Use ViennaRNA/datastructures/stream_output.h instead . . . . .   | 1240 |

|   |      |
|---|------|
| ViennaRNA/ <a href="#">string_utils.h</a>   | 1240 |
| Use ViennaRNA/utils/strings.h instead . . . . .   |      |
| ViennaRNA/ <a href="#">stringdist.h</a>   | 1241 |
| Functions for String Alignment . . . . .  |      |
| ViennaRNA/ <a href="#">structure_utils.h</a>  | 1242 |
| Use ViennaRNA/utils/structures.h instead . . . . .  |      |
| ViennaRNA/ <a href="#">structured_domains.h</a>   | 1242 |
| This module provides interfaces that deal with additional structured domains in the folding grammar . . . . .                         |      |
| ViennaRNA/ <a href="#">subopt.h</a>   | 1243 |
| RNAsubopt and density of states declarations . . . . .  |      |
| ViennaRNA/ <a href="#">subopt_zuker.h</a>   | 1245 |
| ViennaRNA/ <a href="#">svm_utils.h</a>  | 1245 |
| Use ViennaRNA/utils/svm.h instead . . . . .   |      |
| ViennaRNA/ <a href="#">treedist.h</a>   | 1246 |
| Functions for Tree Edit Distances . . . . .   |      |
| ViennaRNA/ <a href="#">units.h</a>  | 1247 |
| Use ViennaRNA/utils/units.h instead . . . . .   |      |
| ViennaRNA/ <a href="#">unstructured_domains.h</a>   | 1249 |
| Functions to modify unstructured domains, e.g. to incorporate ligands binding to unpaired stretches . . . . .                         |      |
| ViennaRNA/ <a href="#">utils.h</a>  | 1254 |
| Use ViennaRNA/utils/basic.h instead . . . . .   |      |
| ViennaRNA/ <a href="#">vrna_config.h</a>  | 1260 |
| ViennaRNA/ <a href="#">walk.h</a>   | 1263 |
| Use ViennaRNA/landscape/walk.h instead . . . . .  |      |
| ViennaRNA/ <a href="#">wrap_dlib.h</a>  | 1263 |
| ViennaRNA/ <a href="#">zscore.h</a>   | 1263 |
| ViennaRNA/constraints/ <a href="#">basic.h</a>  | 882  |
| Functions and data structures for constraining secondary structure predictions and evaluation . . . . .                               |      |
| ViennaRNA/constraints/ <a href="#">hard.h</a>   | 610  |
| Functions and data structures for handling of secondary structure hard constraints . . . . .  |      |
| ViennaRNA/constraints/ <a href="#">ligand.h</a>   | 617  |
| Functions for incorporation of ligands binding to hairpin and interior loop motifs using the soft constraints framework . . . . .     |      |
| ViennaRNA/constraints/ <a href="#">SHAPE.h</a>  | 618  |
| This module provides function to incorporate SHAPE reactivity data into the folding recursions by means of soft constraints . . . . . |      |
| ViennaRNA/constraints/ <a href="#">soft.h</a>   | 619  |
| Functions and data structures for secondary structure soft constraints . . . . .  |      |
| ViennaRNA/datastructures/ <a href="#">basic.h</a>   | 883  |
| Various data structures and pre-processor macros . . . . .  |      |
| ViennaRNA/datastructures/ <a href="#">char_stream.h</a>   | 600  |
| Implementation of a dynamic, buffered character stream . . . . .  |      |
| ViennaRNA/datastructures/ <a href="#">hash_tables.h</a>   | 626  |
| Implementations of hash table functions . . . . .   |      |
| ViennaRNA/datastructures/ <a href="#">heap.h</a>  | 628  |
| Implementation of an abstract heap data structure . . . . .   |      |
| ViennaRNA/datastructures/ <a href="#">lists.h</a>   | 629  |
| ViennaRNA/datastructures/ <a href="#">stream_output.h</a>   | 1239 |
| An implementation of a buffered, ordered stream output data structure . . . . .   |      |
| ViennaRNA/io/ <a href="#">file_formats.h</a>  | 647  |
| Read and write different file formats for RNA sequences, structures . . . . .   |      |
| ViennaRNA/io/ <a href="#">file_formats_msa.h</a>  | 650  |
| Functions dealing with file formats for Multiple Sequence Alignments (MSA) . . . . .  |      |
| ViennaRNA/io/ <a href="#">utils.h</a>   | 1253 |
| Several utilities for file handling . . . . .   |      |

|   |   |      |
|---|---|------|
| ViennaRNA/landscape/ <a href="#">findpath.h</a>             | A breadth-first search heuristic for optimal direct folding paths . . . . .   | 653  |
| ViennaRNA/landscape/ <a href="#">move.h</a>                 | Methods to operate with structural neighbors of RNA secondary structures . . . . .  | 686  |
| ViennaRNA/landscape/ <a href="#">neighbor.h</a>             | Methods to compute the neighbors of an RNA secondary structure . . . . .  | 726  |
| ViennaRNA/landscape/ <a href="#">paths.h</a>                | API for computing (optimal) (re-)folding paths between secondary structures . . . . .   | 688  |
| ViennaRNA/landscape/ <a href="#">walk.h</a>                 | Methods to generate particular paths such as gradient or random walks through the energy landscape of an RNA sequence . . . . . | 1261 |
| ViennaRNA/loops/ <a href="#">all.h</a>                      | Energy evaluation for MFE and partition function calculations . . . . .   | 691  |
| ViennaRNA/loops/ <a href="#">external.h</a>                 | Energy evaluation of exterior loops for MFE and partition function calculations . . . . .                                       | 691  |
| ViennaRNA/loops/ <a href="#">hairpin.h</a>                  | Energy evaluation of hairpin loops for MFE and partition function calculations . . . . .  | 694  |
| ViennaRNA/loops/ <a href="#">internal.h</a>                 | Energy evaluation of interior loops for MFE and partition function calculations . . . . .                                       | 697  |
| ViennaRNA/loops/ <a href="#">multibranch.h</a>              | Energy evaluation of multibranch loops for MFE and partition function calculations . . . . .                                    | 705  |
| ViennaRNA/params/ <a href="#">1.8.4_epars.h</a>             | Free energy parameters for parameter file conversion . . . . .  | 730  |
| ViennaRNA/params/ <a href="#">1.8.4_intloops.h</a>          | Free energy parameters for interior loop contributions needed by the parameter file conversion functions . . . . .              | 734  |
| ViennaRNA/params/ <a href="#">basic.h</a>                   | Functions to deal with sets of energy parameters . . . . .  | 888  |
| ViennaRNA/params/ <a href="#">constants.h</a>               | Energy parameter constants . . . . .  | 898  |
| ViennaRNA/params/ <a href="#">convert.h</a>                 | Functions and definitions for energy parameter file format conversion . . . . .   | 900  |
| ViennaRNA/params/ <a href="#">default.h</a>                 | . . . . .   | 901  |
| ViennaRNA/params/ <a href="#">intl11.h</a>                  | . . . . .   | 902  |
| ViennaRNA/params/ <a href="#">intl11dH.h</a>                | . . . . .   | 907  |
| ViennaRNA/params/ <a href="#">intl21.h</a>                  | . . . . .   | 911  |
| ViennaRNA/params/ <a href="#">intl21dH.h</a>                | . . . . .   | 934  |
| ViennaRNA/params/ <a href="#">intl22.h</a>                  | . . . . .   | 957  |
| ViennaRNA/params/ <a href="#">intl22dH.h</a>                | . . . . .   | 1072 |
| ViennaRNA/params/ <a href="#">io.h</a>                      | Read and write energy parameter files . . . . .   | 1187 |
| ViennaRNA/plotting/ <a href="#">alignments.h</a>            | Various functions for plotting Sequence / Structure Alignments . . . . .  | 1206 |
| ViennaRNA/plotting/ <a href="#">layouts.h</a>               | Secondary structure plot layout algorithms . . . . .  | 1211 |
| ViennaRNA/plotting/ <a href="#">probabilities.h</a>         | Various functions for plotting RNA secondary structures, dot-plots and other visualizations . . . . .                           | 1214 |
| ViennaRNA/plotting/ <a href="#">structures.h</a>            | Various functions for plotting RNA secondary structures . . . . .   | 1218 |
| ViennaRNA/plotting/ <a href="#">utils.h</a>                 | Various utilities to assist in plotting secondary structures and consensus structures . . . . .                                 | 1254 |
| ViennaRNA/plotting/RNApuzzler/ <a href="#">RNApuzzler.h</a> | Implementation of the RNApuzzler RNA secondary structure layout algorithm [24] . . . . .  | 1216 |
| ViennaRNA/plotting/RNApuzzler/ <a href="#">RNAturtle.h</a>  | Implementation of the RNAturtle RNA secondary structure layout algorithm [24] . . . . .   | 1217 |
| ViennaRNA/search/ <a href="#">BoyerMoore.h</a>              | Variants of the Boyer-Moore string search algorithm . . . . .   | 1232 |

|  |      |
|--|------|
| ViennaRNA/utils/ <a href="#">alignments.h</a>  |      |
| Various utility- and helper-functions for sequence alignments and comparative structure prediction . . . . .                 | 1207 |
| ViennaRNA/utils/ <a href="#">basic.h</a>   |      |
| General utility- and helper-functions used throughout the <i>ViennaRNA Package</i> . . . . .                                 | 892  |
| ViennaRNA/utils/ <a href="#">cpu.h</a>   |      |
| ViennaRNA/utils/ <a href="#">higher_order_functions.h</a>  |      |
| ViennaRNA/utils/ <a href="#">strings.h</a>   |      |
| General utility- and helper-functions for RNA sequence and structure strings used throughout the ViennaRNA Package . . . . . | 1255 |
| ViennaRNA/utils/ <a href="#">structures.h</a>  |      |
| Various utility- and helper-functions for secondary structure parsing, converting, etc . . . . .                             | 1219 |
| ViennaRNA/utils/ <a href="#">svm.h</a>   |      |
| ViennaRNA/utils/ <a href="#">units.h</a>   |      |
| Physical Units and Functions to convert them into each other . . . . .   | 1248 |

# Chapter 16

## Module Documentation

### 16.1 Free Energy Evaluation

Functions and variables related to free energy evaluation of sequence/structure pairs.

#### 16.1.1 Detailed Description

Functions and variables related to free energy evaluation of sequence/structure pairs.

Several different functions to evaluate the free energy of a particular secondary structure under a particular set of parameters and the Nearest Neighbor Energy model are available. For most of them, two different forms of representations for the secondary structure may be used:

- The Dot-Bracket string
- A pair table representation

Furthermore, the evaluation functions are divided into `basic` and `simplified` variants, where `basic` functions require the use of a `vrna_fold_compound_t` data structure holding the sequence string, and model configuration (settings and parameters). The `simplified` functions, on the other hand, provide often used default model settings that may be called directly with only sequence and structure data.

Finally, `verbose` options exist for some functions that allow one to print the (individual) free energy contributions to some FILE stream. Collaboration diagram for Free Energy Evaluation:

### Modules

- [Energy Evaluation for Individual Loops](#)  
*Functions to evaluate the free energy of particular types of loops.*
- [Energy Evaluation for Atomic Moves](#)  
*Functions to evaluate the free energy change of a structure after application of (a set of) atomic moves.*
- [Deprecated Interface for Free Energy Evaluation](#)  
*Deprecated Energy Evaluation functions.*

### Files

- file [eval.h](#)  
*Functions and variables related to energy evaluation of sequence/structure pairs.*
- file [all.h](#)  
*Energy evaluation for MFE and partition function calculations.*
- file [external.h](#)  
*Energy evaluation of exterior loops for MFE and partition function calculations.*
- file [hairpin.h](#)  
*Energy evaluation of hairpin loops for MFE and partition function calculations.*
- file [internal.h](#)

*Energy evaluation of interior loops for MFE and partition function calculations.*

- file [multibranch.h](#)

*Energy evaluation of multibranch loops for MFE and partition function calculations.*

## Macros

- `#define VRNA_VERTBOSITY_QUIET -1`  
*Quiet level verbosity setting.*
- `#define VRNA_VERTBOSITY_DEFAULT 1`  
*Default level verbosity setting.*

## Basic Energy Evaluation Interface with Dot-Bracket Structure String

- float [vrna\\_eval\\_structure](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, const char \*structure)  
*Calculate the free energy of an already folded RNA.*
- float [vrna\\_eval\\_covar\\_structure](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, const char \*structure)  
*Calculate the pseudo energy derived by the covariance scores of a set of aligned sequences.*
- float [vrna\\_eval\\_structure\\_verbose](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, const char \*structure, FILE \*file)  
*Calculate the free energy of an already folded RNA and print contributions on a per-loop base.*
- float [vrna\\_eval\\_structure\\_v](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, const char \*structure, int verbosity\_level, FILE \*file)  
*Calculate the free energy of an already folded RNA and print contributions on a per-loop base.*
- float [vrna\\_eval\\_structure\\_cstr](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, const char \*structure, int verbosity\_level, vrna\_cstr\_t output\_stream)  
*Calculate the free energy of an already folded RNA and print contributions on a per-loop base.*

## Basic Energy Evaluation Interface with Structure Pair Table

- int [vrna\\_eval\\_structure\\_pt](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, const short \*pt)  
*Calculate the free energy of an already folded RNA.*
- int [vrna\\_eval\\_structure\\_pt\\_verbose](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, const short \*pt, FILE \*file)  
*Calculate the free energy of an already folded RNA.*
- int [vrna\\_eval\\_structure\\_pt\\_v](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, const short \*pt, int verbosity\_level, FILE \*file)  
*Calculate the free energy of an already folded RNA.*

## Simplified Energy Evaluation with Sequence and Dot-Bracket Strings

- float [vrna\\_eval\\_structure\\_simple](#) (const char \*string, const char \*structure)  
*Calculate the free energy of an already folded RNA.*
- float [vrna\\_eval\\_circ\\_structure](#) (const char \*string, const char \*structure)  
*Evaluate the free energy of a sequence/structure pair where the sequence is circular.*
- float [vrna\\_eval\\_gquad\\_structure](#) (const char \*string, const char \*structure)  
*Evaluate the free energy of a sequence/structure pair where the structure may contain G-Quadruplexes.*
- float [vrna\\_eval\\_circ\\_gquad\\_structure](#) (const char \*string, const char \*structure)  
*Evaluate the free energy of a sequence/structure pair where the sequence is circular and the structure may contain G-Quadruplexes.*
- float [vrna\\_eval\\_structure\\_simple\\_verbose](#) (const char \*string, const char \*structure, FILE \*file)  
*Calculate the free energy of an already folded RNA and print contributions per loop.*
- float [vrna\\_eval\\_structure\\_simple\\_v](#) (const char \*string, const char \*structure, int verbosity\_level, FILE \*file)  
*Calculate the free energy of an already folded RNA and print contributions per loop.*
- float [vrna\\_eval\\_circ\\_structure\\_v](#) (const char \*string, const char \*structure, int verbosity\_level, FILE \*file)  
*Evaluate free energy of a sequence/structure pair, assume sequence to be circular and print contributions per loop.*
- float [vrna\\_eval\\_gquad\\_structure\\_v](#) (const char \*string, const char \*structure, int verbosity\_level, FILE \*file)  
*Evaluate free energy of a sequence/structure pair, allow for G-Quadruplexes in the structure and print contributions per loop.*

- float `vrna_eval_circ_gquad_structure_v` (const char \*string, const char \*structure, int verbosity\_level, FILE \*file)

*Evaluate free energy of a sequence/structure pair, assume sequence to be circular, allow for G-Quadruplexes in the structure, and print contributions per loop.*

## Simplified Energy Evaluation with Sequence Alignments and Consensus Structure Dot-Bracket String

- float `vrna_eval_consensus_structure_simple` (const char \*\*alignment, const char \*structure)  
*Calculate the free energy of an already folded RNA sequence alignment.*
- float `vrna_eval_circ_consensus_structure` (const char \*\*alignment, const char \*structure)  
*Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the sequences are circular.*
- float `vrna_eval_gquad_consensus_structure` (const char \*\*alignment, const char \*structure)  
*Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the structure may contain G-Quadruplexes.*
- float `vrna_eval_circ_gquad_consensus_structure` (const char \*\*alignment, const char \*structure)  
*Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the sequence is circular and the structure may contain G-Quadruplexes.*
- float `vrna_eval_consensus_structure_simple_verbose` (const char \*\*alignment, const char \*structure, FILE \*file)  
*Evaluate the free energy of a consensus structure for an RNA sequence alignment and print contributions per loop.*
- float `vrna_eval_consensus_structure_simple_v` (const char \*\*alignment, const char \*structure, int verbosity\_level, FILE \*file)  
*Evaluate the free energy of a consensus structure for an RNA sequence alignment and print contributions per loop.*
- float `vrna_eval_circ_consensus_structure_v` (const char \*\*alignment, const char \*structure, int verbosity\_level, FILE \*file)  
*Evaluate the free energy of a consensus structure for an alignment of circular RNA sequences and print contributions per loop.*
- float `vrna_eval_gquad_consensus_structure_v` (const char \*\*alignment, const char \*structure, int verbosity\_level, FILE \*file)  
*Evaluate the free energy of a consensus structure for an RNA sequence alignment, allow for annotated G-Quadruplexes in the structure and print contributions per loop.*
- float `vrna_eval_circ_gquad_consensus_structure_v` (const char \*\*alignment, const char \*structure, int verbosity\_level, FILE \*file)  
*Evaluate the free energy of a consensus structure for an alignment of circular RNA sequences, allow for annotated G-Quadruplexes in the structure and print contributions per loop.*

## Simplified Energy Evaluation with Sequence String and Structure Pair Table

- int `vrna_eval_structure_pt_simple` (const char \*string, const short \*pt)  
*Calculate the free energy of an already folded RNA.*
- int `vrna_eval_structure_pt_simple_verbose` (const char \*string, const short \*pt, FILE \*file)  
*Calculate the free energy of an already folded RNA.*
- int `vrna_eval_structure_pt_simple_v` (const char \*string, const short \*pt, int verbosity\_level, FILE \*file)  
*Calculate the free energy of an already folded RNA.*

## Simplified Energy Evaluation with Sequence Alignment and Consensus Structure Pair Table

- int `vrna_eval_consensus_structure_pt_simple` (const char \*\*alignment, const short \*pt)  
*Evaluate the Free Energy of a Consensus Secondary Structure given a Sequence Alignment.*
- int `vrna_eval_consensus_structure_pt_simple_verbose` (const char \*\*alignment, const short \*pt, FILE \*file)
- int `vrna_eval_consensus_structure_pt_simple_v` (const char \*\*alignment, const short \*pt, int verbosity\_level, FILE \*file)

## 16.1.2 Function Documentation

### 16.1.2.1 vrna\_eval\_structure()

```
float vrna_eval_structure (
    vrna_fold_compound_t * fc,
    const char * structure )
#include <ViennaRNA/eval.h>
Calculate the free energy of an already folded RNA.
```

This function allows for energy evaluation of a given pair of structure and sequence (alignment). Model details, energy parameters, and possibly soft constraints are used as provided via the parameter 'fc'. The `vrna_fold_compound_t` does not need to contain any DP matrices, but requires all most basic init values as one would get from a call like this:

```
fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_EVAL_ONLY);
```

#### Note

Accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE` and `VRNA_FC_TYPE_COMPARATIVE`

#### See also

[vrna\\_eval\\_structure\\_pt\(\)](#), [vrna\\_eval\\_structure\\_verbose\(\)](#), [vrna\\_eval\\_structure\\_pt\\_verbose\(\)](#), [vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\\_compound\\_comparative\(\)](#), [vrna\\_eval\\_covar\\_structure\(\)](#)

#### Parameters

|                        |  |
|------------------------|--|
| <code>fc</code>        | A <code>vrna_fold_compound_t</code> containing the energy parameters and model details |
| <code>structure</code> | Secondary structure in dot-bracket notation  |

#### Returns

The free energy of the input structure given the input sequence in kcal/mol

**SWIG Wrapper Notes** This function is attached as method `eval_structure()` to objects of type `fold_compound`

### 16.1.2.2 vrna\_eval\_covar\_structure()

```
float vrna_eval_covar_structure (
    vrna_fold_compound_t * fc,
    const char * structure )
#include <ViennaRNA/eval.h>
Calculate the pseudo energy derived by the covariance scores of a set of aligned sequences.
```

Consensus structure prediction is driven by covariance scores of base pairs in rows of the provided alignment. This function allows one to retrieve the total amount of this covariance pseudo energy scores. The `vrna_fold_compound_t` does not need to contain any DP matrices, but requires all most basic init values as one would get from a call like this:

```
fc = vrna_fold_compound_comparative(alignment, NULL, VRNA_OPTION_EVAL_ONLY);
```

#### Note

Accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_COMPARATIVE` only!

#### See also

[vrna\\_fold\\_compound\\_comparative\(\)](#), [vrna\\_eval\\_structure\(\)](#)

## Parameters

|                  |   |
|------------------|---|
| <i>fc</i>        | A vrna_fold_compound_t containing the energy parameters and model details |
| <i>structure</i> | Secondary (consensus) structure in dot-bracket notation                   |

## Returns

The covariance pseudo energy score of the input structure given the input sequence alignment in kcal/mol

**SWIG Wrapper Notes** This function is attached as method **eval\_covar\_structure()** to objects of type *fold\_*←  
*compound*

**16.1.2.3 vrna\_eval\_structure\_verbose()**

```
float vrna_eval_structure_verbose (
    vrna_fold_compound_t * fc,
    const char * structure,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA and print contributions on a per-loop base.

This function is a simplified version of [vrna\\_eval\\_structure\\_v\(\)](#) that uses the *default* verbosity level.

## See also

[vrna\\_eval\\_structure\\_pt\(\)](#), [vrna\\_eval\\_structure\\_verbose\(\)](#), [vrna\\_eval\\_structure\\_pt\\_verbose\(\)](#),

## Parameters

|                  |   |
|------------------|---|
| <i>fc</i>        | A vrna_fold_compound_t containing the energy parameters and model details |
| <i>structure</i> | Secondary structure in dot-bracket notation                               |
| <i>file</i>      | A file handle where this function should print to (may be NULL).          |

## Returns

The free energy of the input structure given the input sequence in kcal/mol

**SWIG Wrapper Notes** This function is attached as method **eval\_structure\_verbose()** to objects of type *fold\_*←  
*compound*

**16.1.2.4 vrna\_eval\_structure\_v()**

```
float vrna_eval_structure_v (
    vrna_fold_compound_t * fc,
    const char * structure,
    int verbosity_level,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA and print contributions on a per-loop base.

This function allows for detailed energy evaluation of a given sequence/structure pair. In contrast to [vrna\\_eval\\_structure\(\)](#) this function prints detailed energy contributions based on individual loops to a file handle. If NULL is passed as file handle, this function defaults to print to stdout. Any positive *verbosity\_level* activates potential warning message of the energy evaluating functions, while values  $\geq 1$  allow for detailed control of what data is printed. A negative parameter *verbosity\_level* turns off printing all together.

Model details, energy parameters, and possibly soft constraints are used as provided via the parameter 'fc'. The fold\_compound does not need to contain any DP matrices, but all the most basic init values as one would get from a call like this:

```
fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_EVAL_ONLY);
```

See also

[vrna\\_eval\\_structure\\_pt\(\)](#), [vrna\\_eval\\_structure\\_verbose\(\)](#), [vrna\\_eval\\_structure\\_pt\\_verbose\(\)](#),

#### Parameters

|                        |   |
|------------------------|---|
| <i>fc</i>              | A vrna_fold_compound_t containing the energy parameters and model details |
| <i>structure</i>       | Secondary structure in dot-bracket notation                               |
| <i>verbosity_level</i> | The level of verbosity of this function                                   |
| <i>file</i>            | A file handle where this function should print to (may be NULL).          |

#### Returns

The free energy of the input structure given the input sequence in kcal/mol

#### 16.1.2.5 vrna\_eval\_structure\_pt()

```
int vrna_eval_structure_pt (
    vrna_fold_compound_t * fc,
    const short * pt )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

This function allows for energy evaluation of a given sequence/structure pair where the structure is provided in pair\_table format as obtained from [vrna\\_ptable\(\)](#). Model details, energy parameters, and possibly soft constraints are used as provided via the parameter 'fc'. The fold\_compound does not need to contain any DP matrices, but all the most basic init values as one would get from a call like this:

```
fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_EVAL_ONLY);
```

See also

[vrna\\_ptable\(\)](#), [vrna\\_eval\\_structure\(\)](#), [vrna\\_eval\\_structure\\_pt\\_verbose\(\)](#)

#### Parameters

|           |   |
|-----------|---|
| <i>fc</i> | A vrna_fold_compound_t containing the energy parameters and model details |
| <i>pt</i> | Secondary structure as pair_table   |

#### Returns

The free energy of the input structure given the input sequence in 10cal/mol

**SWIG Wrapper Notes** This function is attached as method **eval\_structure\_pt()** to objects of type *fold\_compound*

#### 16.1.2.6 vrna\_eval\_structure\_pt\_verbose()

```
int vrna_eval_structure_pt_verbose (
    vrna_fold_compound_t * fc,
    const short * pt,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

This function is a simplified version of [vrna\\_eval\\_structure\\_simple\\_v\(\)](#) that uses the *default* verbosity level.

#### See also

[vrna\\_eval\\_structure\\_pt\\_v\(\)](#), [vrna\\_ptable\(\)](#), [vrna\\_eval\\_structure\\_pt\(\)](#), [vrna\\_eval\\_structure\\_verbose\(\)](#)

#### Parameters

|             |   |
|-------------|---|
| <i>fc</i>   | A vrna_fold_compound_t containing the energy parameters and model details |
| <i>pt</i>   | Secondary structure as pair_table   |
| <i>file</i> | A file handle where this function should print to (may be NULL).          |

#### Returns

The free energy of the input structure given the input sequence in 10cal/mol

**SWIG Wrapper Notes** This function is attached as method **eval\_structure\_pt\_verbose()** to objects of type *fold\_compound*

#### 16.1.2.7 vrna\_eval\_structure\_pt\_v()

```
int vrna_eval_structure_pt_v (
    vrna_fold_compound_t * fc,
    const short * pt,
    int verbosity_level,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

This function allows for energy evaluation of a given sequence/structure pair where the structure is provided in pair\_table format as obtained from [vrna\\_ptable\(\)](#). Model details, energy parameters, and possibly soft constraints are used as provided via the parameter 'fc'. The fold\_compound does not need to contain any DP matrices, but all the most basic init values as one would get from a call like this:

```
fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_EVAL_ONLY);
```

In contrast to [vrna\\_eval\\_structure\\_pt\(\)](#) this function prints detailed energy contributions based on individual loops to a file handle. If NULL is passed as file handle, this function defaults to print to stdout. Any positive *verbosity\_level* activates potential warning message of the energy evaluating functions, while values  $\geq 1$  allow for detailed control of what data is printed. A negative parameter *verbosity\_level* turns off printing all together.

#### See also

[vrna\\_ptable\(\)](#), [vrna\\_eval\\_structure\\_pt\(\)](#), [vrna\\_eval\\_structure\\_verbose\(\)](#)

#### Parameters

|                        |   |
|------------------------|---|
| <i>fc</i>              | A vrna_fold_compound_t containing the energy parameters and model details |
| <i>pt</i>              | Secondary structure as pair_table   |
| <i>verbosity_level</i> | The level of verbosity of this function                                   |
| <i>file</i>            | A file handle where this function should print to (may be NULL).          |

#### Returns

The free energy of the input structure given the input sequence in 10cal/mol

### 16.1.2.8 vrna\_eval\_structure\_simple()

```
int vrna_eval_structure_simple (
    const char * string,
    const char * structure )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

This function allows for energy evaluation of a given sequence/structure pair. In contrast to [vrna\\_eval\\_structure\(\)](#) this function assumes default model details and default energy parameters in order to evaluate the free energy of the secondary structure. Therefore, it serves as a simple interface function for energy evaluation for situations where no changes on the energy model are required.

See also

[vrna\\_eval\\_structure\(\)](#), [vrna\\_eval\\_structure\\_pt\(\)](#), [vrna\\_eval\\_structure\\_verbose\(\)](#), [vrna\\_eval\\_structure\\_pt\\_verbose\(\)](#),

#### Parameters

|                  |   |
|------------------|---|
| <i>string</i>    | RNA sequence in uppercase letters           |
| <i>structure</i> | Secondary structure in dot-bracket notation |

#### Returns

The free energy of the input structure given the input sequence in kcal/mol

**SWIG Wrapper Notes** In the target scripting language, this function serves as a wrapper for [vrna\\_eval\\_structure\\_simple\\_v\(\)](#) and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to [VRNA\\_VERBOSITY QUIET](#) and NULL, respectively.

### 16.1.2.9 vrna\_eval\_circ\_structure()

```
int vrna_eval_circ_structure (
    const char * string,
    const char * structure )
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a sequence/structure pair where the sequence is circular.

See also

[vrna\\_eval\\_structure\\_simple\(\)](#), [vrna\\_eval\\_gquad\\_structure\(\)](#), [vrna\\_eval\\_circ\\_consensus\\_structure\(\)](#), [vrna\\_eval\\_circ\\_structure\\_v\(\)](#), [vrna\\_eval\\_structure\(\)](#)

#### Parameters

|                  |   |
|------------------|---|
| <i>string</i>    | RNA sequence in uppercase letters           |
| <i>structure</i> | Secondary structure in dot-bracket notation |

#### Returns

The free energy of the structure given the circular input sequence in kcal/mol

**SWIG Wrapper Notes** In the target scripting language, this function serves as a wrapper for [vrna\\_eval\\_circ\\_structure\\_v\(\)](#) and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to [VRNA\\_VERBOSITY QUIET](#) and NULL, respectively.

### 16.1.2.10 vrna\_eval\_gquad\_structure()

```
int vrna_eval_gquad_structure (
    const char * string,
    const char * structure )
#include <ViennaRNA/eval.h>
Evaluate the free energy of a sequence/structure pair where the structure may contain G-Quadruplexes.
G-Quadruplexes are annotated as plus signs ('+') for each G involved in the motif. Linker sequences must be
denoted by dots ('.') as they are considered unpaired. Below is an example of a 2-layer G-quadruplex:
GGAAGGAAAGGAGG
++...++...++...++
```

#### See also

[vrna\\_eval\\_structure\\_simple\(\)](#), [vrna\\_eval\\_circ\\_structure\(\)](#), [vrna\\_eval\\_gquad\\_consensus\\_structure\(\)](#), [vrna\\_eval\\_gquad\\_structure\\_v\(\)](#), [vrna\\_eval\\_structure\(\)](#)

#### Parameters

|                  |   |
|------------------|---|
| <i>string</i>    | RNA sequence in uppercase letters           |
| <i>structure</i> | Secondary structure in dot-bracket notation |

#### Returns

The free energy of the structure including contributions of G-quadruplexes in kcal/mol

**SWIG Wrapper Notes** In the target scripting language, this function serves as a wrapper for [vrna\\_eval\\_gquad\\_structure\\_v\(\)](#) and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to [VRNA\\_VERBOSITY QUIET](#) and NULL, respectively.

### 16.1.2.11 vrna\_eval\_circ\_gquad\_structure()

```
int vrna_eval_circ_gquad_structure (
    const char * string,
    const char * structure )
#include <ViennaRNA/eval.h>
Evaluate the free energy of a sequence/structure pair where the sequence is circular and the structure may contain
G-Quadruplexes.
G-Quadruplexes are annotated as plus signs ('+') for each G involved in the motif. Linker sequences must be
denoted by dots ('.') as they are considered unpaired. Below is an example of a 2-layer G-quadruplex:
GGAAGGAAAGGAGG
++...++...++...++
```

#### See also

[vrna\\_eval\\_structure\\_simple\(\)](#), [vrna\\_eval\\_circ\\_gquad\\_consensus\\_structure\(\)](#), [vrna\\_eval\\_circ\\_gquad\\_structure\\_v\(\)](#), [vrna\\_eval\\_structure\(\)](#)

#### Parameters

|                  |   |
|------------------|---|
| <i>string</i>    | RNA sequence in uppercase letters           |
| <i>structure</i> | Secondary structure in dot-bracket notation |

**Returns**

The free energy of the structure including contributions of G-quadruplexes in kcal/mol

**SWIG Wrapper Notes** In the target scripting language, this function serves as a wrapper for `vrna_eval_circ_gquad_structure_v()` and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to `VRNA_VERBOSITY QUIET` and `NULL`, respectively.

**16.1.2.12 `vrna_eval_structure_simple_verbose()`**

```
int vrna_eval_structure_simple_verbose (
    const char * string,
    const char * structure,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA and print contributions per loop.

This function is a simplified version of `vrna_eval_structure_simple_v()` that uses the *default* verbosity level.

**See also**

`vrna_eval_structure_simple_v()`, `vrna_eval_structure_verbose()`, `vrna_eval_structure_pt()`, `vrna_eval_structure_verbose()`, `vrna_eval_structure_pt_verbose()`

**Parameters**

|                        |  |
|------------------------|--|
| <code>string</code>    | RNA sequence in uppercase letters  |
| <code>structure</code> | Secondary structure in dot-bracket notation                                    |
| <code>file</code>      | A file handle where this function should print to (may be <code>NULL</code> ). |

**Returns**

The free energy of the input structure given the input sequence in kcal/mol

**SWIG Wrapper Notes** This function is not available. Use `vrna_eval_structure_simple_v()` instead!

**16.1.2.13 `vrna_eval_structure_simple_v()`**

```
int vrna_eval_structure_simple_v (
    const char * string,
    const char * structure,
    int verbosity_level,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA and print contributions per loop.

This function allows for detailed energy evaluation of a given sequence/structure pair. In contrast to `vrna_eval_structure()` this function prints detailed energy contributions based on individual loops to a file handle. If `NULL` is passed as file handle, this function defaults to print to `stdout`. Any positive `verbosity_level` activates potential warning message of the energy evaluating functions, while values  $\geq 1$  allow for detailed control of what data is printed. A negative parameter `verbosity_level` turns off printing all together.

In contrast to `vrna_eval_structure_verbose()` this function assumes default model details and default energy parameters in order to evaluate the free energy of the secondary structure. Therefore, it serves as a simple interface function for energy evaluation for situations where no changes on the energy model are required.

**See also**

`vrna_eval_structure_verbose()`, `vrna_eval_structure_pt()`, `vrna_eval_structure_pt_verbose()`,

### Parameters

|                        |  |
|------------------------|--|
| <i>string</i>          | RNA sequence in uppercase letters                                |
| <i>structure</i>       | Secondary structure in dot-bracket notation                      |
| <i>verbosity_level</i> | The level of verbosity of this function                          |
| <i>file</i>            | A file handle where this function should print to (may be NULL). |

### Returns

The free energy of the input structure given the input sequence in kcal/mol

**SWIG Wrapper Notes** This function is available through an overloaded version of [vrna\\_eval\\_structure\\_simple\(\)](#). The last two arguments for this function are optional and default to [VRNA\\_VERBOSITY QUIET](#) and NULL, respectively.

### 16.1.2.14 vrna\_eval\_circ\_structure\_v()

```
int vrna_eval_circ_structure_v (
    const char * string,
    const char * structure,
    int verbosity_level,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Evaluate free energy of a sequence/structure pair, assume sequence to be circular and print contributions per loop. This function is the same as [vrna\\_eval\\_structure\\_simple\\_v\(\)](#) but assumes the input sequence to be circularized.

### See also

[vrna\\_eval\\_structure\\_simple\\_v\(\)](#), [vrna\\_eval\\_circ\\_structure\(\)](#), [vrna\\_eval\\_structure\\_verbose\(\)](#)

### Parameters

|                        |  |
|------------------------|--|
| <i>string</i>          | RNA sequence in uppercase letters                                |
| <i>structure</i>       | Secondary structure in dot-bracket notation                      |
| <i>verbosity_level</i> | The level of verbosity of this function                          |
| <i>file</i>            | A file handle where this function should print to (may be NULL). |

### Returns

The free energy of the input structure given the input sequence in kcal/mol

**SWIG Wrapper Notes** This function is available through an overloaded version of [vrna\\_eval\\_circ\\_structure\(\)](#). The last two arguments for this function are optional and default to [VRNA\\_VERBOSITY QUIET](#) and NULL, respectively.

### 16.1.2.15 vrna\_eval\_gquad\_structure\_v()

```
int vrna_eval_gquad_structure_v (
    const char * string,
    const char * structure,
    int verbosity_level,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Evaluate free energy of a sequence/structure pair, allow for G-Quadruplexes in the structure and print contributions per loop.

This function is the same as [vrna\\_eval\\_structure\\_simple\\_v\(\)](#) but allows for annotated G-Quadruplexes in the dot-bracket structure input.

G-Quadruplexes are annotated as plus signs ('+') for each G involved in the motif. Linker sequences must be denoted by dots ('.') as they are considered unpaired. Below is an example of a 2-layer G-quadruplex:

```
GGAGGAAAGGAGG
++...++...++...++
```

#### See also

[vrna\\_eval\\_structure\\_simple\\_v\(\)](#), [vrna\\_eval\\_gquad\\_structure\(\)](#), [vrna\\_eval\\_structure\\_verbose\(\)](#)

#### Parameters

|                        |  |
|------------------------|--|
| <i>string</i>          | RNA sequence in uppercase letters                                |
| <i>structure</i>       | Secondary structure in dot-bracket notation                      |
| <i>verbosity_level</i> | The level of verbosity of this function                          |
| <i>file</i>            | A file handle where this function should print to (may be NULL). |

#### Returns

The free energy of the input structure given the input sequence in kcal/mol

**SWIG Wrapper Notes** This function is available through an overloaded version of [vrna\\_eval\\_gquad\\_structure\(\)](#). The last two arguments for this function are optional and default to [VRNA\\_VERBOSITY QUIET](#) and NULL, respectively.

### 16.1.2.16 vrna\_eval\_circ\_gquad\_structure\_v()

```
int vrna_eval_circ_gquad_structure_v (
    const char * string,
    const char * structure,
    int verbosity_level,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Evaluate free energy of a sequence/structure pair, assume sequence to be circular, allow for G-Quadruplexes in the structure, and print contributions per loop.

This function is the same as [vrna\\_eval\\_structure\\_simple\\_v\(\)](#) but assumes the input sequence to be circular and allows for annotated G-Quadruplexes in the dot-bracket structure input.

G-Quadruplexes are annotated as plus signs ('+') for each G involved in the motif. Linker sequences must be denoted by dots ('.') as they are considered unpaired. Below is an example of a 2-layer G-quadruplex:

```
GGAGGAAAGGAGG
++...++...++...++
```

#### Parameters

|                        |  |
|------------------------|--|
| <i>string</i>          | RNA sequence in uppercase letters                                |
| <i>structure</i>       | Secondary structure in dot-bracket notation                      |
| <i>verbosity_level</i> | The level of verbosity of this function                          |
| <i>file</i>            | A file handle where this function should print to (may be NULL). |

**Returns**

The free energy of the input structure given the input sequence in kcal/mol

**SWIG Wrapper Notes** This function is available through an overloaded version of [vrna\\_eval\\_circ\\_gquad\\_structure\(\)](#). The last two arguments for this function are optional and default to [VRNA\\_VERBOSITY QUIET](#) and NULL, respectively.

**16.1.2.17 vrna\_eval\_consensus\_structure\_simple()**

```
int vrna_eval_consensus_structure_simple (
    const char ** alignment,
    const char * structure )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA sequence alignment.

This function allows for energy evaluation for a given multiple sequence alignment and consensus structure pair. In contrast to [vrna\\_eval\\_structure\(\)](#) this function assumes default model details and default energy parameters in order to evaluate the free energy of the secondary structure. Therefore, it serves as a simple interface function for energy evaluation for situations where no changes on the energy model are required.

**Note**

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

**See also**

[vrna\\_eval\\_covar\\_structure\(\)](#), [vrna\\_eval\\_structure\(\)](#), [vrna\\_eval\\_structure\\_pt\(\)](#), [vrna\\_eval\\_structure\\_verbose\(\)](#), [vrna\\_eval\\_structure\\_pt\\_verbose\(\)](#)

**Parameters**

|                  |   |
|------------------|---|
| <i>alignment</i> | RNA sequence alignment in uppercase letters and hyphen ('-') to denote gaps |
| <i>structure</i> | Consensus Secondary structure in dot-bracket notation                       |

**Returns**

The free energy of the consensus structure given the input alignment in kcal/mol

**SWIG Wrapper Notes** This function is available through an overloadeed version of [vrna\\_eval\\_structure\\_simple\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

**16.1.2.18 vrna\_eval\_circ\_consensus\_structure()**

```
int vrna_eval_circ_consensus_structure (
    const char ** alignment,
    const char * structure )
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the sequences are circular.

**Note**

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

## See also

[vrna\\_eval\\_covar\\_structure\(\)](#), [vrna\\_eval\\_consensus\\_structure\\_simple\(\)](#), [vrna\\_eval\\_gquad\\_consensus\\_structure\(\)](#),  
[vrna\\_eval\\_circ\\_structure\(\)](#), [vrna\\_eval\\_circ\\_consensus\\_structure\\_v\(\)](#), [vrna\\_eval\\_structure\(\)](#)

## Parameters

|                  |   |
|------------------|---|
| <i>alignment</i> | RNA sequence alignment in uppercase letters           |
| <i>structure</i> | Consensus secondary structure in dot-bracket notation |

## Returns

The free energy of the consensus structure given the circular input sequence in kcal/mol

**SWIG Wrapper Notes** This function is available through an overloaded version of [vrna\\_eval\\_circ\\_structure\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

**16.1.2.19 vrna\_eval\_gquad\_consensus\_structure()**

```
int vrna_eval_gquad_consensus_structure (
    const char ** alignment,
    const char * structure )
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the structure may contain G-Quadruplexes.

G-Quadruplexes are annotated as plus signs ('+') for each G involved in the motif. Linker sequences must be denoted by dots ('.') as they are considered unpaired. Below is an example of a 2-layer G-quadruplex:

```
GGAGGAAAGGAGG
++...++...++...++
```

## Note

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

## See also

[vrna\\_eval\\_covar\\_structure\(\)](#), [vrna\\_eval\\_consensus\\_structure\\_simple\(\)](#), [vrna\\_eval\\_circ\\_consensus\\_structure\(\)](#),  
[vrna\\_eval\\_gquad\\_structure\(\)](#), [vrna\\_eval\\_gquad\\_consensus\\_structure\\_v\(\)](#), [vrna\\_eval\\_structure\(\)](#)

## Parameters

|                  |   |
|------------------|---|
| <i>alignment</i> | RNA sequence alignment in uppercase letters           |
| <i>structure</i> | Consensus secondary structure in dot-bracket notation |

## Returns

The free energy of the consensus structure including contributions of G-quadruplexes in kcal/mol

**SWIG Wrapper Notes** This function is available through an overloaded version of [vrna\\_eval\\_gquad\\_structure\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

**16.1.2.20 vrna\_eval\_circ\_gquad\_consensus\_structure()**

```
int vrna_eval_circ_gquad_consensus_structure (
```

```
    const char ** alignment,
    const char * structure )
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the sequence is circular and the structure may contain G-Quadruplexes.

G-Quadruplexes are annotated as plus signs ('+') for each G involved in the motif. Linker sequences must be denoted by dots ('.') as they are considered unpaired. Below is an example of a 2-layer G-quadruplex:

```
GGAGGAAAGGAGG
++...++...++..++
```

#### Note

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

#### See also

[vrna\\_eval\\_covar\\_structure\(\)](#), [vrna\\_eval\\_consensus\\_structure\\_simple\(\)](#), [vrna\\_eval\\_circ\\_consensus\\_structure\(\)](#),  
[vrna\\_eval\\_gquad\\_structure\(\)](#), [vrna\\_eval\\_circ\\_gquad\\_consensus\\_structure\\_v\(\)](#), [vrna\\_eval\\_structure\(\)](#)

#### Parameters

|                  |   |
|------------------|---|
| <i>alignment</i> | RNA sequence alignment in uppercase letters           |
| <i>structure</i> | Consensus secondary structure in dot-bracket notation |

#### Returns

The free energy of the consensus structure including contributions of G-quadruplexes in kcal/mol

**SWIG Wrapper Notes** This function is available through an overloaded version of [vrna\\_eval\\_circ\\_gquad\\_structure\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

### 16.1.2.21 vrna\_eval\_consensus\_structure\_simple\_verbose()

```
int vrna_eval_consensus_structure_simple_verbose (
    const char ** alignment,
    const char * structure,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a consensus structure for an RNA sequence alignment and print contributions per loop. This function is a simplified version of [vrna\\_eval\\_consensus\\_structure\\_simple\\_v\(\)](#) that uses the *default* verbosity level.

#### Note

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

#### See also

[vrna\\_eval\\_consensus\\_structure\\_simple\\_v\(\)](#), [vrna\\_eval\\_structure\\_verbose\(\)](#), [vrna\\_eval\\_structure\\_pt\(\)](#),  
[vrna\\_eval\\_structure\\_pt\\_verbose\(\)](#)

#### Parameters

|                  |  |
|------------------|--|
| <i>alignment</i> | RNA sequence alignment in uppercase letters. Gaps are denoted by hyphens ('-') |
| <i>structure</i> | Consensus secondary structure in dot-bracket notation                          |
| <i>file</i>      | A file handle where this function should print to (may be NULL).               |

**Returns**

The free energy of the consensus structure given the aligned input sequences in kcal/mol

**SWIG Wrapper Notes** This function is not available. Use [vrna\\_eval\\_consensus\\_structure\\_simple\\_v\(\)](#) instead!

**16.1.2.22 vrna\_eval\_consensus\_structure\_simple\_v()**

```
int vrna_eval_consensus_structure_simple_v (
    const char ** alignment,
    const char * structure,
    int verbosity_level,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a consensus structure for an RNA sequence alignment and print contributions per loop. This function allows for detailed energy evaluation of a given sequence alignment/consensus structure pair. In contrast to [vrna\\_eval\\_consensus\\_structure\\_simple\(\)](#) this function prints detailed energy contributions based on individual loops to a file handle. If NULL is passed as file handle, this function defaults to print to stdout. Any positive `verbosity_level` activates potential warning message of the energy evaluating functions, while values  $\geq 1$  allow for detailed control of what data is printed. A negative parameter `verbosity_level` turns off printing all together.

**Note**

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

**See also**

[vrna\\_eval\\_consensus\\_structure\(\)](#), [vrna\\_eval\\_structure\(\)](#)

**Parameters**

|                              |  |
|------------------------------|--|
| <code>alignment</code>       | RNA sequence alignment in uppercase letters. Gaps are denoted by hyphens ('-') |
| <code>structure</code>       | Consensus secondary structure in dot-bracket notation                          |
| <code>verbosity_level</code> | The level of verbosity of this function  |
| <code>file</code>            | A file handle where this function should print to (may be NULL).               |

**Returns**

The free energy of the consensus structure given the sequence alignment in kcal/mol

**SWIG Wrapper Notes** This function is available through an overloaded version of [vrna\\_eval\\_structure\\_simple\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to [VRNA\\_VERBOSITY QUIET](#) and NULL, respectively.

**16.1.2.23 vrna\_eval\_circ\_consensus\_structure\_v()**

```
int vrna_eval_circ_consensus_structure_v (
    const char ** alignment,
    const char * structure,
    int verbosity_level,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a consensus structure for an alignment of circular RNA sequences and print contributions per loop.

This function is identical with [vrna\\_eval\\_consensus\\_structure\\_simple\\_v\(\)](#) but assumed the aligned sequences to be circular.

#### Note

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

#### See also

[vrna\\_eval\\_consensus\\_structure\\_simple\\_v\(\)](#), [vrna\\_eval\\_circ\\_consensus\\_structure\(\)](#), [vrna\\_eval\\_structure\(\)](#)

#### Parameters

|                        |  |
|------------------------|--|
| <i>alignment</i>       | RNA sequence alignment in uppercase letters. Gaps are denoted by hyphens ('-') |
| <i>structure</i>       | Consensus secondary structure in dot-bracket notation                          |
| <i>verbosity_level</i> | The level of verbosity of this function  |
| <i>file</i>            | A file handle where this function should print to (may be NULL).               |

#### Returns

The free energy of the consensus structure given the sequence alignment in kcal/mol

**SWIG Wrapper Notes** This function is available through an overloaded version of [vrna\\_eval\\_circ\\_structure\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to [VRNA\\_VERBOSITY QUIET](#) and NULL, respectively.

#### 16.1.2.24 vrna\_eval\_gquad\_consensus\_structure\_v()

```
int vrna_eval_gquad_consensus_structure_v (
    const char ** alignment,
    const char * structure,
    int verbosity_level,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a consensus structure for an RNA sequence alignment, allow for annotated G-Quadruplexes in the structure and print contributions per loop.

This function is identical with [vrna\\_eval\\_consensus\\_structure\\_simple\\_v\(\)](#) but allows for annotated G-Quadruplexes in the consensus structure.

G-Quadruplexes are annotated as plus signs ('+') for each G involved in the motif. Linker sequences must be denoted by dots ('.') as they are considered unpaired. Below is an example of a 2-layer G-quadruplex:

```
GGAAGGAAAGGAGG
++...++....++..++
```

#### Note

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

#### See also

[vrna\\_eval\\_consensus\\_structure\\_simple\\_v\(\)](#), [vrna\\_eval\\_gquad\\_consensus\\_structure\(\)](#), [vrna\\_eval\\_structure\(\)](#)

**Parameters**

|                        |  |
|------------------------|--|
| <i>alignment</i>       | RNA sequence alignment in uppercase letters. Gaps are denoted by hyphens ('-') |
| <i>structure</i>       | Consensus secondary structure in dot-bracket notation                          |
| <i>verbosity_level</i> | The level of verbosity of this function  |
| <i>file</i>            | A file handle where this function should print to (may be NULL).               |

**Returns**

The free energy of the consensus structure given the sequence alignment in kcal/mol

**SWIG Wrapper Notes** This function is available through an overloaded version of [vrna\\_eval\\_gquad\\_structure\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to [VRNA\\_VERBOSITY QUIET](#) and NULL, respectively.

**16.1.2.25 vrna\_eval\_circ\_gquad\_consensus\_structure\_v()**

```
int vrna_eval_circ_gquad_consensus_structure_v (
    const char ** alignment,
    const char * structure,
    int verbosity_level,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a consensus structure for an alignment of circular RNA sequences, allow for annotated G-Quadruplexes in the structure and print contributions per loop.

This function is identical with [vrna\\_eval\\_consensus\\_structure\\_simple\\_v\(\)](#) but assumes the sequences in the alignment to be circular and allows for annotated G-Quadruplexes in the consensus structure.

G-Quadruplexes are annotated as plus signs ('+') for each G involved in the motif. Linker sequences must be denoted by dots ('.') as they are considered unpaired. Below is an example of a 2-layer G-quadruplex:

```
GGAGGAAAGGAGG
++...++...++..++
```

**Note**

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

**See also**

[vrna\\_eval\\_consensus\\_structure\\_simple\\_v\(\)](#), [vrna\\_eval\\_circ\\_gquad\\_consensus\\_structure\(\)](#), [vrna\\_eval\\_structure\(\)](#)

**Parameters**

|                        |  |
|------------------------|--|
| <i>alignment</i>       | RNA sequence alignment in uppercase letters. Gaps are denoted by hyphens ('-') |
| <i>structure</i>       | Consensus secondary structure in dot-bracket notation                          |
| <i>verbosity_level</i> | The level of verbosity of this function  |
| <i>file</i>            | A file handle where this function should print to (may be NULL).               |

**Returns**

The free energy of the consensus structure given the sequence alignment in kcal/mol

**SWIG Wrapper Notes** This function is available through an overloaded version of [vrna\\_eval\\_circ\\_gquad\\_structure\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to [VRNA\\_VERBOSITY QUIET](#) and NULL, respectively.

### 16.1.2.26 vrna\_eval\_structure\_pt\_simple()

```
int vrna_eval_structure_pt_simple (
    const char * string,
    const short * pt )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

In contrast to [vrna\\_eval\\_structure\\_pt\(\)](#) this function assumes default model details and default energy parameters in order to evaluate the free energy of the secondary structure. Therefore, it serves as a simple interface function for energy evaluation for situations where no changes on the energy model are required.

See also

[vrna\\_ptable\(\)](#), [vrna\\_eval\\_structure\\_simple\(\)](#), [vrna\\_eval\\_structure\\_pt\(\)](#)

#### Parameters

|               |                                   |
|---------------|-----------------------------------|
| <i>string</i> | RNA sequence in uppercase letters |
| <i>pt</i>     | Secondary structure as pair_table |

#### Returns

The free energy of the input structure given the input sequence in 10cal/mol

**SWIG Wrapper Notes** In the target scripting language, this function serves as a wrapper for [vrna\\_eval\\_structure\\_pt\\_v\(\)](#) and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to [VRNA\\_VERBOSITY QUIET](#) and NULL, respectively.

### 16.1.2.27 vrna\_eval\_structure\_pt\_simple\_verbose()

```
int vrna_eval_structure_pt_simple_verbose (
    const char * string,
    const short * pt,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

This function is a simplified version of [vrna\\_eval\\_structure\\_pt\\_simple\\_v\(\)](#) that uses the *default* verbosity level.

See also

[vrna\\_eval\\_structure\\_pt\\_simple\\_v\(\)](#), [vrna\\_ptable\(\)](#), [vrna\\_eval\\_structure\\_pt\\_verbose\(\)](#), [vrna\\_eval\\_structure\\_simple\(\)](#)

#### Parameters

|               |  |
|---------------|--|
| <i>string</i> | RNA sequence in uppercase letters                                |
| <i>pt</i>     | Secondary structure as pair_table                                |
| <i>file</i>   | A file handle where this function should print to (may be NULL). |

#### Returns

The free energy of the input structure given the input sequence in 10cal/mol

### 16.1.2.28 vrna\_eval\_structure\_pt\_simple\_v()

```
int vrna_eval_structure_pt_simple_v (
    const char * string,
    const short * pt,
    int verbosity_level,
    FILE * file )
#include <ViennaRNA/eval.h>
Calculate the free energy of an already folded RNA.
```

This function allows for energy evaluation of a given sequence/structure pair where the structure is provided in pair\_table format as obtained from [vrna\\_ptable\(\)](#). Model details, energy parameters, and possibly soft constraints are used as provided via the parameter 'fc'. The fold\_compound does not need to contain any DP matrices, but all the most basic init values as one would get from a call like this:

```
fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_EVAL_ONLY);
```

In contrast to [vrna\\_eval\\_structure\\_pt\\_verbose\(\)](#) this function assumes default model details and default energy parameters in order to evaluate the free energy of the secondary structure. Threefore, it serves as a simple interface function for energy evaluation for situations where no changes on the energy model are required.

#### See also

[vrna\\_ptable\(\)](#), [vrna\\_eval\\_structure\\_pt\\_v\(\)](#), [vrna\\_eval\\_structure\\_simple\(\)](#)

#### Parameters

|                        |  |
|------------------------|--|
| <i>string</i>          | RNA sequence in uppercase letters                                |
| <i>pt</i>              | Secondary structure as pair_table                                |
| <i>verbosity_level</i> | The level of verbosity of this function                          |
| <i>file</i>            | A file handle where this function should print to (may be NULL). |

#### Returns

The free energy of the input structure given the input sequence in 10cal/mol

### 16.1.2.29 vrna\_eval\_consensus\_structure\_pt\_simple()

```
int vrna_eval_consensus_structure_pt_simple (
    const char ** alignment,
    const short * pt )
#include <ViennaRNA/eval.h>
```

Evaluate the Free Energy of a Consensus Secondary Structure given a Sequence Alignment.

#### Note

The free energy returned from this function already includes the covariation pseudo energies that is used fir comparative structure prediction within this library.

#### See also

[vrna\\_eval\\_consensus\\_structure\\_simple\(\)](#), [vrna\\_eval\\_structure\\_pt\(\)](#), [vrna\\_eval\\_structure\(\)](#), [vrna\\_eval\\_covar\\_structure\(\)](#)

#### Parameters

|                  |  |
|------------------|--|
| <i>alignment</i> | RNA sequence alignment in uppercase letters. Gaps are denoted by hyphens ('-') |
| <i>pt</i>        | Secondary structure in pair table format                                       |

**Returns**

Free energy of the consensus structure in 10cal/mol

**SWIG Wrapper Notes** This function is available through an overloaded version of `vrna_eval_structure_pt_simple()`. Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

**16.1.2.30 vrna\_eval\_consensus\_structure\_pt\_simple\_verbose()**

```
int vrna_eval_consensus_structure_pt_simple_verbose (
    const char ** alignment,
    const short * pt,
    FILE * file )
#include <ViennaRNA/eval.h>
```

**SWIG Wrapper Notes** This function is not available. Use `vrna_eval_consensus_structure_pt_v()` instead!

**16.1.2.31 vrna\_eval\_consensus\_structure\_pt\_simple\_v()**

```
int vrna_eval_consensus_structure_pt_simple_v (
    const char ** alignment,
    const short * pt,
    int verbosity_level,
    FILE * file )
#include <ViennaRNA/eval.h>
```

**SWIG Wrapper Notes** This function is available through an overloaded version of `vrna_eval_structure_pt_simple()`. Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to `VRNA_VERBOSITY QUIET` and `NULL`, respectively.

## 16.2 Energy Evaluation for Individual Loops

Functions to evaluate the free energy of particular types of loops.

### 16.2.1 Detailed Description

Functions to evaluate the free energy of particular types of loops.

To assess the free energy contribution of a particular loop within a secondary structure, two variants are provided:

- The bare free energy  $E$  (usually in deka-calories, i.e. multiples of 10cal/mol), and
- The Boltzmann weight  $q = \exp(-\beta E)$  of the free energy  $E$  (with  $\beta = \frac{1}{RT}$ , gas constant  $R$  and temperature  $T$ )

The latter is usually required for partition function computations. Collaboration diagram for Energy Evaluation for Individual Loops:

### Modules

- [Exterior Loops](#)
  - Functions to evaluate the free energy contributions for exterior loops.*
- [Hairpin Loops](#)
  - Functions to evaluate the free energy contributions for hairpin loops.*
- [Internal Loops](#)
  - Functions to evaluate the free energy contributions for internal loops.*
- [Multibranch Loops](#)
  - Functions to evaluate the free energy contributions for multibranch loops.*

## Files

- file [all.h](#)  
*Energy evaluation for MFE and partition function calculations.*
- file [external.h](#)  
*Energy evaluation of exterior loops for MFE and partition function calculations.*
- file [hairpin.h](#)  
*Energy evaluation of hairpin loops for MFE and partition function calculations.*
- file [internal.h](#)  
*Energy evaluation of interior loops for MFE and partition function calculations.*
- file [multibranch.h](#)  
*Energy evaluation of multibranch loops for MFE and partition function calculations.*

## Functions

- int [vrna\\_eval\\_loop\\_pt](#) ([vrna\\_fold\\_compound\\_t](#) \**fc*, int *i*, const short \**pt*)  
*Calculate energy of a loop.*
- int [vrna\\_eval\\_loop\\_pt\\_v](#) ([vrna\\_fold\\_compound\\_t](#) \**fc*, int *i*, const short \**pt*, int *verbosity\_level*)  
*Calculate energy of a loop.*

### 16.2.2 Function Documentation

#### 16.2.2.1 [vrna\\_eval\\_loop\\_pt\(\)](#)

```
int vrna_eval_loop_pt (
    vrna\_fold\_compound\_t * fc,
    int i,
    const short * pt )
#include <ViennaRNA/eval.h>
Calculate energy of a loop.
```

##### Parameters

|           |   |
|-----------|---|
| <i>fc</i> | A <a href="#">vrna_fold_compound_t</a> containing the energy parameters and model details |
| <i>i</i>  | position of covering base pair  |
| <i>pt</i> | the pair table of the secondary structure   |

##### Returns

free energy of the loop in 10cal/mol

**SWIG Wrapper Notes** This function is attached as method [eval\\_loop\\_pt\(\)](#) to objects of type *fold\_compound*

#### 16.2.2.2 [vrna\\_eval\\_loop\\_pt\\_v\(\)](#)

```
int vrna_eval_loop_pt_v (
    vrna\_fold\_compound\_t * fc,
    int i,
    const short * pt,
    int verbosity_level )
#include <ViennaRNA/eval.h>
Calculate energy of a loop.
```

**Parameters**

|                        |   |
|------------------------|---|
| <i>fc</i>              | A vrna_fold_compound_t containing the energy parameters and model details |
| <i>i</i>               | position of covering base pair  |
| <i>pt</i>              | the pair table of the secondary structure                                 |
| <i>verbosity_level</i> | The level of verbosity of this function                                   |

**Returns**

free energy of the loop in 10cal/mol

## 16.3 Energy Evaluation for Atomic Moves

Functions to evaluate the free energy change of a structure after application of (a set of) atomic moves.

### 16.3.1 Detailed Description

Functions to evaluate the free energy change of a structure after application of (a set of) atomic moves. Here, atomic moves are not to be confused with moves of actual physical atoms. Instead, an atomic move is considered the smallest conformational change a secondary structure can undergo to form another, distinguishable structure. We currently support the following moves

**Atomic Moves:**

- Opening (dissociation) of a single base pair
- Closing (formation) of a single base pair
- Shifting one pairing partner of an existing pair to a different location

Collaboration diagram for Energy Evaluation for Atomic Moves:

## Functions

- float [vrna\\_eval\\_move](#) (*vrna\_fold\_compound\_t* \**fc*, const char \**structure*, int *m1*, int *m2*)  
*Calculate energy of a move (closing or opening of a base pair)*
- int [vrna\\_eval\\_move\\_pt](#) (*vrna\_fold\_compound\_t* \**fc*, short \**pt*, int *m1*, int *m2*)  
*Calculate energy of a move (closing or opening of a base pair)*

### 16.3.2 Function Documentation

#### 16.3.2.1 [vrna\\_eval\\_move\(\)](#)

```
float vrna_eval_move (
    vrna_fold_compound_t * fc,
    const char * structure,
    int m1,
    int m2 )
#include <ViennaRNA/eval.h>
Calculate energy of a move (closing or opening of a base pair)
If the parameters m1 and m2 are negative, it is deletion (opening) of a base pair, otherwise it is insertion (opening).
```

**See also**

[vrna\\_eval\\_move\\_pt\(\)](#)

**Parameters**

|                  |   |
|------------------|---|
| <i>fc</i>        | A vrna_fold_compound_t containing the energy parameters and model details |
| <i>structure</i> | secondary structure in dot-bracket notation                               |
| <i>m1</i>        | first coordinate of base pair   |
| <i>m2</i>        | second coordinate of base pair  |

**Returns**

energy change of the move in kcal/mol ([INF](#) / 100. upon any error)

**SWIG Wrapper Notes** This function is attached as method **eval\_move()** to objects of type *fold\_compound*

**16.3.2.2 vrna\_eval\_move\_pt()**

```
int vrna_eval_move_pt (
    vrna_fold_compound_t * fc,
    short * pt,
    int m1,
    int m2 )
#include <ViennaRNA/eval.h>
Calculate energy of a move (closing or opening of a base pair)
If the parameters m1 and m2 are negative, it is deletion (opening) of a base pair, otherwise it is insertion (opening).
```

**See also**

[vrna\\_eval\\_move\(\)](#)

**Parameters**

|           |   |
|-----------|---|
| <i>fc</i> | A vrna_fold_compound_t containing the energy parameters and model details |
| <i>pt</i> | the pair table of the secondary structure                                 |
| <i>m1</i> | first coordinate of base pair   |
| <i>m2</i> | second coordinate of base pair  |

**Returns**

energy change of the move in 10cal/mol

**SWIG Wrapper Notes** This function is attached as method **eval\_move\_pt()** to objects of type *fold\_compound*

## 16.4 Deprecated Interface for Free Energy Evaluation

Deprecated Energy Evaluation functions.

### 16.4.1 Detailed Description

Deprecated Energy Evaluation functions.

Using the functions below is discouraged as they have been marked deprecated and will be removed from the library in the (near) future! Collaboration diagram for Deprecated Interface for Free Energy Evaluation:

### Functions

- float [energy\\_of\\_structure](#) (const char \*string, const char \*structure, int verbosity\_level)

*Calculate the free energy of an already folded RNA using global model detail settings.*

- float `energy_of_struct_par` (const char \*string, const char \*structure, `vrna_param_t` \*parameters, int verbosity\_level)

*Calculate the free energy of an already folded RNA.*

- float `energy_of_circ_structure` (const char \*string, const char \*structure, int verbosity\_level)

*Calculate the free energy of an already folded circular RNA.*

- float `energy_of_circ_struct_par` (const char \*string, const char \*structure, `vrna_param_t` \*parameters, int verbosity\_level)

*Calculate the free energy of an already folded circular RNA.*

- int `energy_of_structure_pt` (const char \*string, short \*ptable, short \*s, short \*s1, int verbosity\_level)

*Calculate the free energy of an already folded RNA.*

- int `energy_of_struct_pt_par` (const char \*string, short \*ptable, short \*s, short \*s1, `vrna_param_t` \*parameters, int verbosity\_level)

*Calculate the free energy of an already folded RNA.*

- float `energy_of_move` (const char \*string, const char \*structure, int m1, int m2)

*Calculate energy of a move (closing or opening of a base pair)*

- int `energy_of_move_pt` (short \*pt, short \*s, short \*s1, int m1, int m2)

*Calculate energy of a move (closing or opening of a base pair)*

- int `loop_energy` (short \*ptable, short \*s, short \*s1, int i)

*Calculate energy of a loop.*

- float `energy_of_struct` (const char \*string, const char \*structure)

- int `energy_of_struct_pt` (const char \*string, short \*ptable, short \*s, short \*s1)

- float `energy_of_circ_struct` (const char \*string, const char \*structure)

- int `E_Stem` (int type, int si1, int sj1, int extLoop, `vrna_param_t` \*P)

*Compute the energy contribution of a stem branching off a loop-region.*

- `FLT_OR_DBL exp_E_ExtLoop` (int type, int si1, int sj1, `vrna_exp_param_t` \*P)

- `FLT_OR_DBL exp_E_Stem` (int type, int si1, int sj1, int extLoop, `vrna_exp_param_t` \*P)

- PRIVATE int `E_IntLoop` (int n1, int n2, int type, int type\_2, int si1, int sj1, int sp1, int sq1, `vrna_param_t` \*P)

- PRIVATE `FLT_OR_DBL exp_E_IntLoop` (int u1, int u2, int type, int type2, short si1, short sj1, short sp1, short sq1, `vrna_exp_param_t` \*P)

## Variables

- int `cut_point`

*first pos of second seq for cofolding*

- int `eos_debug`

*verbose info from energy\_of\_struct*

### 16.4.2 Function Documentation

#### 16.4.2.1 `energy_of_structure()`

```
float energy_of_structure (
    const char * string,
    const char * structure,
    int verbosity_level )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA using global model detail settings.

If verbosity level is set to a value >0, energies of structure elements are printed to stdout

**Note**

OpenMP: This function relies on several global model settings variables and thus is not to be considered threadsafe. See [energy\\_of\\_struct\\_par\(\)](#) for a completely threadsafe implementation.

**Deprecated** Use [vrna\\_eval\\_structure\(\)](#) or [vrna\\_eval\\_structure\\_verbose\(\)](#) instead!

**See also**

[vrna\\_eval\\_structure\(\)](#)

**Parameters**

|                        |   |
|------------------------|---|
| <i>string</i>          | RNA sequence                                |
| <i>structure</i>       | secondary structure in dot-bracket notation |
| <i>verbosity_level</i> | a flag to turn verbose output on/off        |

**Returns**

the free energy of the input structure given the input sequence in kcal/mol

**16.4.2.2 energy\_of\_struct\_par()**

```
float energy_of_struct_par (
    const char * string,
    const char * structure,
    vrna_param_t * parameters,
    int verbosity_level )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

If verbosity level is set to a value >0, energies of structure elements are printed to stdout

**Deprecated** Use [vrna\\_eval\\_structure\(\)](#) or [vrna\\_eval\\_structure\\_verbose\(\)](#) instead!

**See also**

[vrna\\_eval\\_structure\(\)](#)

**Parameters**

|                        |   |
|------------------------|---|
| <i>string</i>          | RNA sequence in uppercase letters   |
| <i>structure</i>       | Secondary structure in dot-bracket notation   |
| <i>parameters</i>      | A data structure containing the prescaled energy contributions and the model details. |
| <i>verbosity_level</i> | A flag to turn verbose output on/off  |

**Returns**

The free energy of the input structure given the input sequence in kcal/mol

**16.4.2.3 energy\_of\_circ\_structure()**

```
float energy_of_circ_structure (
    const char * string,
```

```

    const char * structure,
    int verbosity_level )
#include <ViennaRNA/eval.h>
Calculate the free energy of an already folded circular RNA.
```

**Note**

OpenMP: This function relies on several global model settings variables and thus is not to be considered threadsafe. See [energy\\_of\\_circ\\_struct\\_par\(\)](#) for a completely threadsafe implementation.

If verbosity level is set to a value  $>0$ , energies of structure elements are printed to stdout

**Deprecated** Use [vrna\\_eval\\_structure\(\)](#) or [vrna\\_eval\\_structure\\_verbose\(\)](#) instead!

**See also**

[vrna\\_eval\\_structure\(\)](#)

**Parameters**

|                        |   |
|------------------------|---|
| <i>string</i>          | RNA sequence                                |
| <i>structure</i>       | Secondary structure in dot-bracket notation |
| <i>verbosity_level</i> | A flag to turn verbose output on/off        |

**Returns**

The free energy of the input structure given the input sequence in kcal/mol

**16.4.2.4 energy\_of\_circ\_struct\_par()**

```

float energy_of_circ_struct_par (
    const char * string,
    const char * structure,
    vrna_param_t * parameters,
    int verbosity_level )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded circular RNA.

If verbosity level is set to a value  $>0$ , energies of structure elements are printed to stdout

**Deprecated** Use [vrna\\_eval\\_structure\(\)](#) or [vrna\\_eval\\_structure\\_verbose\(\)](#) instead!

**See also**

[vrna\\_eval\\_structure\(\)](#)

**Parameters**

|                        |   |
|------------------------|---|
| <i>string</i>          | RNA sequence  |
| <i>structure</i>       | Secondary structure in dot-bracket notation   |
| <i>parameters</i>      | A data structure containing the prescaled energy contributions and the model details. |
| <i>verbosity_level</i> | A flag to turn verbose output on/off  |

**Returns**

The free energy of the input structure given the input sequence in kcal/mol

**16.4.2.5 energy\_of\_structure\_pt()**

```
int energy_of_structure_pt (
    const char * string,
    short * ptable,
    short * s,
    short * sl,
    int verbosity_level )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

If verbosity level is set to a value >0, energies of structure elements are printed to stdout

**Note**

OpenMP: This function relies on several global model settings variables and thus is not to be considered threadsafe. See [energy\\_of\\_struct\\_pt\\_par\(\)](#) for a completely threadsafe implementation.

**Deprecated** Use [vrna\\_eval\\_structure\\_pt\(\)](#) or [vrna\\_eval\\_structure\\_pt\\_verbose\(\)](#) instead!

**See also**

[vrna\\_eval\\_structure\\_pt\(\)](#)

**Parameters**

|                        |   |
|------------------------|---|
| <i>string</i>          | RNA sequence                              |
| <i>ptable</i>          | the pair table of the secondary structure |
| <i>s</i>               | encoded RNA sequence                      |
| <i>s1</i>              | encoded RNA sequence                      |
| <i>verbosity_level</i> | a flag to turn verbose output on/off      |

**Returns**

the free energy of the input structure given the input sequence in 10kcal/mol

**16.4.2.6 energy\_of\_struct\_pt\_par()**

```
int energy_of_struct_pt_par (
    const char * string,
    short * ptable,
    short * s,
    short * sl,
    vrna_param_t * parameters,
    int verbosity_level )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

If verbosity level is set to a value >0, energies of structure elements are printed to stdout

**Deprecated** Use [vrna\\_eval\\_structure\\_pt\(\)](#) or [vrna\\_eval\\_structure\\_pt\\_verbose\(\)](#) instead!

See also

[vrna\\_eval\\_structure\\_pt\(\)](#)

#### Parameters

|                        |   |
|------------------------|---|
| <i>string</i>          | RNA sequence in uppercase letters   |
| <i>ptable</i>          | The pair table of the secondary structure   |
| <i>s</i>               | Encoded RNA sequence  |
| <i>s1</i>              | Encoded RNA sequence  |
| <i>parameters</i>      | A data structure containing the prescaled energy contributions and the model details. |
| <i>verbosity_level</i> | A flag to turn verbose output on/off  |

#### Returns

The free energy of the input structure given the input sequence in 10kcal/mol

#### 16.4.2.7 `energy_of_move()`

```
float energy_of_move (
    const char * string,
    const char * structure,
    int m1,
    int m2 )
#include <ViennaRNA/eval.h>
Calculate energy of a move (closing or opening of a base pair)
If the parameters m1 and m2 are negative, it is deletion (opening) of a base pair, otherwise it is insertion (opening).
```

**Deprecated** Use [vrna\\_eval\\_move\(\)](#) instead!

See also

[vrna\\_eval\\_move\(\)](#)

#### Parameters

|                  |   |
|------------------|---|
| <i>string</i>    | RNA sequence                                |
| <i>structure</i> | secondary structure in dot-bracket notation |
| <i>m1</i>        | first coordinate of base pair               |
| <i>m2</i>        | second coordinate of base pair              |

#### Returns

energy change of the move in kcal/mol

#### 16.4.2.8 `energy_of_move_pt()`

```
int energy_of_move_pt (
    short * pt,
    short * s,
    short * s1,
    int m1,
    int m2 )
```

```
#include <ViennaRNA/eval.h>
Calculate energy of a move (closing or opening of a base pair)
If the parameters m1 and m2 are negative, it is deletion (opening) of a base pair, otherwise it is insertion (opening).
```

**Deprecated** Use [vrna\\_eval\\_move\\_pt\(\)](#) instead!

#### See also

[vrna\\_eval\\_move\\_pt\(\)](#)

#### Parameters

|           |   |
|-----------|---|
| <i>pt</i> | the pair table of the secondary structure |
| <i>s</i>  | encoded RNA sequence                      |
| <i>s1</i> | encoded RNA sequence                      |
| <i>m1</i> | first coordinate of base pair             |
| <i>m2</i> | second coordinate of base pair            |

#### Returns

energy change of the move in 10cal/mol

#### 16.4.2.9 loop\_energy()

```
int loop_energy (
    short * ptable,
    short * s,
    short * s1,
    int i )
#include <ViennaRNA/eval.h>
Calculate energy of a loop.
```

**Deprecated** Use [vrna\\_eval\\_loop\\_pt\(\)](#) instead!

#### See also

[vrna\\_eval\\_loop\\_pt\(\)](#)

#### Parameters

|               |   |
|---------------|---|
| <i>ptable</i> | the pair table of the secondary structure |
| <i>s</i>      | encoded RNA sequence                      |
| <i>s1</i>     | encoded RNA sequence                      |
| <i>i</i>      | position of covering base pair            |

#### Returns

free energy of the loop in 10cal/mol

#### 16.4.2.10 energy\_of\_struct()

```
float energy_of_struct (
```

```

    const char * string,
    const char * structure )
#include <ViennaRNA/eval.h>
Calculate the free energy of an already folded RNA

```

**Note**

This function is not entirely threadsafe! Depending on the state of the global variable `eos_debug` it prints energy information to stdout or not...

**Deprecated** This function is deprecated and should not be used in future programs! Use [energy\\_of\\_structure\(\)](#) instead!

**See also**

[energy\\_of\\_structure](#), [energy\\_of\\_circ\\_struct\(\)](#), [energy\\_of\\_struct\\_pt\(\)](#)

**Parameters**

|                        |   |
|------------------------|---|
| <code>string</code>    | RNA sequence                                |
| <code>structure</code> | secondary structure in dot-bracket notation |

**Returns**

the free energy of the input structure given the input sequence in kcal/mol

**16.4.2.11 energy\_of\_struct\_pt()**

```

int energy_of_struct_pt (
    const char * string,
    short * ptable,
    short * s,
    short * s1 )
#include <ViennaRNA/eval.h>
Calculate the free energy of an already folded RNA

```

**Note**

This function is not entirely threadsafe! Depending on the state of the global variable `eos_debug` it prints energy information to stdout or not...

**Deprecated** This function is deprecated and should not be used in future programs! Use [energy\\_of\\_structure\\_pt\(\)](#) instead!

**See also**

[make\\_pair\\_table\(\)](#), [energy\\_of\\_structure\(\)](#)

**Parameters**

|                     |   |
|---------------------|---|
| <code>string</code> | RNA sequence                              |
| <code>ptable</code> | the pair table of the secondary structure |
| <code>s</code>      | encoded RNA sequence                      |
| <code>s1</code>     | encoded RNA sequence                      |

**Returns**

the free energy of the input structure given the input sequence in 10kcal/mol

**16.4.2.12 energy\_of\_circ\_struct()**

```
float energy_of_circ_struct (
    const char * string,
    const char * structure )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded circular RNA

**Note**

This function is not entirely threadsafe! Depending on the state of the global variable `eos_debug` it prints energy information to stdout or not...

**Deprecated** This function is deprecated and should not be used in future programs Use [energy\\_of\\_circ\\_structure\(\)](#) instead!

**See also**

[energy\\_of\\_circ\\_structure\(\)](#), [energy\\_of\\_struct\(\)](#), [energy\\_of\\_struct\\_pt\(\)](#)

**Parameters**

|                        |   |
|------------------------|---|
| <code>string</code>    | RNA sequence                                |
| <code>structure</code> | secondary structure in dot-bracket notation |

**Returns**

the free energy of the input structure given the input sequence in kcal/mol

**16.4.2.13 E\_Stem()**

```
int E_Stem (
    int type,
    int si1,
    int sj1,
    int extLoop,
    vrna_param_t * P )
#include <ViennaRNA/loops/external.h>
```

Compute the energy contribution of a stem branching off a loop-region.

This function computes the energy contribution of a stem that branches off a loop region. This can be the case in multiloops, when a stem branching off increases the degree of the loop but also *immediately interior base pairs* of an exterior loop contribute free energy. To switch the behavior of the function according to the evaluation of a multiloop- or exterior-loop-stem, you pass the flag 'extLoop'. The returned energy contribution consists of a TerminalAU penalty if the pair type is greater than 2, dangling end contributions of mismatching nucleotides adjacent to the stem if only one of the si1, sj1 parameters is greater than 0 and mismatch energies if both mismatching nucleotides are positive values. Thus, to avoid incorporating dangling end or mismatch energies just pass a negative number, e.g. -1 to the mismatch argument.

This is an illustration of how the energy contribution is assembled:

```
3'   5'
|   |
X - Y
5'-si1      sj1-3'
```

Here, (X,Y) is the base pair that closes the stem that branches off a loop region. The nucleotides si1 and sj1 are the 5'- and 3'- mismatches, respectively. If the base pair type of (X,Y) is greater than 2 (i.e. an A-U or G-U pair, the TerminalAU penalty will be included in the energy contribution returned. If si1 and sj1 are both nonnegative numbers, mismatch energies will also be included. If one of si1 or sj1 is a negative value, only 5' or 3' dangling end contributions are taken into account. To prohibit any of these mismatch contributions to be incorporated, just pass a negative number to both, si1 and sj1. In case the argument extLoop is 0, the returned energy contribution also includes the *internal-loop-penalty* of a multiloop stem with closing pair type.

#### See also

[E\\_MLstem\(\)](#)  
[E\\_ExtLoop\(\)](#)

#### Note

This function is threadsafe

**Deprecated** Please use one of the functions [vrna\\_E\\_ext\\_stem\(\)](#) and [E\\_MLstem\(\)](#) instead! Use the former for cases where `extLoop != 0` and the latter otherwise.

#### Parameters

|                      |  |
|----------------------|--|
| <code>type</code>    | The pair type of the first base pair un the stem   |
| <code>si1</code>     | The 5'-mismatching nucleotide  |
| <code>sj1</code>     | The 3'-mismatching nucleotide  |
| <code>extLoop</code> | A flag that indicates whether the contribution reflects the one of an exterior loop or not |
| <code>P</code>       | The data structure containing scaled energy parameters                                     |

#### Returns

The Free energy of the branch off the loop in dcal/mol

#### 16.4.2.14 exp\_E\_ExtLoop()

```
FLT_OR_DBL exp_E_ExtLoop (
    int type,
    int si1,
    int sj1,
    vrna_exp_param_t * P )
#include <ViennaRNA/loops/external.h>
```

This is the partition function variant of [E\\_ExtLoop\(\)](#)

**Deprecated** Use [vrna\\_exp\\_E\\_ext\\_stem\(\)](#) instead!

#### See also

[E\\_ExtLoop\(\)](#)

#### Returns

The Boltzmann weighted energy contribution of the introduced exterior-loop stem

#### 16.4.2.15 exp\_E\_Stack()

```
FLOAT_OR_DBL exp_E_Stack (
    int type,
    int sil,
    int sjl,
    int extLoop,
    vrna_exp_param_t * P )
#include <ViennaRNA/loops/external.h>
```

**Compute the Boltzmann weighted energy contribution of a stem branching off a loop-region**

This is the partition function variant of [E\\_Stack\(\)](#)

See also

[E\\_Stack\(\)](#)

Note

This function is threadsafe

Returns

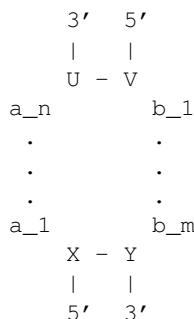
The Boltzmann weighted energy contribution of the branch off the loop

#### 16.4.2.16 E\_IntLoop()

```
PRIVATE int E_IntLoop (
    int n1,
    int n2,
    int type,
    int type_2,
    int sil,
    int sjl,
    int spl,
    int sql,
    vrna_param_t * P )
#include <ViennaRNA/loops/internal.h>
```

**Compute the Energy of an interior-loop**

This function computes the free energy  $\Delta G$  of an interior-loop with the following structure:



This general structure depicts an interior-loop that is closed by the base pair (X,Y). The enclosed base pair is (V,U) which leaves the unpaired bases a\_1-a\_n and b\_1-b\_m that constitute the loop. In this example, the length of the interior-loop is  $(n + m)$  where n or m may be 0 resulting in a bulge-loop or base pair stack. The mismatching

nucleotides for the closing pair (X,Y) are:

5'-mismatch: a\_1

3'-mismatch: b\_m

and for the enclosed base pair (V,U):

5'-mismatch: b\_1

3'-mismatch: a\_n

#### Note

Base pairs are always denoted in 5'->3' direction. Thus the enclosed base pair must be 'turned arround' when evaluating the free energy of the interior-loop

#### See also

[scale\\_parameters\(\)](#)

[vrna\\_param\\_t](#)

#### Note

This function is threadsafe

#### Parameters

|                         |   |
|-------------------------|---|
| <i>n1</i>               | The size of the 'left'-loop (number of unpaired nucleotides)  |
| <i>n2</i>               | The size of the 'right'-loop (number of unpaired nucleotides) |
| <i>type</i>             | The pair type of the base pair closing the interior loop      |
| <i>type<sub>2</sub></i> | The pair type of the enclosed base pair                       |
| <i>si1</i>              | The 5'-mismatching nucleotide of the closing pair             |
| <i>sj1</i>              | The 3'-mismatching nucleotide of the closing pair             |
| <i>sp1</i>              | The 3'-mismatching nucleotide of the enclosed pair            |
| <i>sq1</i>              | The 5'-mismatching nucleotide of the enclosed pair            |
| <i>P</i>                | The datastructure containing scaled energy parameters         |

#### Returns

The Free energy of the Interior-loop in dcal/mol

#### 16.4.2.17 exp\_E\_IntLoop()

```
PRIVATE FLT_OR_DBL exp_E_IntLoop (
    int u1,
    int u2,
    int type,
    int type2,
    short si1,
    short sj1,
    short sp1,
    short sq1,
    vrna_exp_param_t * P )
#include <ViennaRNA/loops/internal.h>
```

**Compute Boltzmann weight  $e^{-\Delta G/kT}$  of interior loop**

multiply by scale[u1+u2+2] for scaling

**See also**

[get\\_scaled\\_pf\\_parameters\(\)](#)  
[vrna\\_exp\\_param\\_t](#)  
[E\\_IntLoop\(\)](#)

**Note**

This function is threadsafe

**Parameters**

|              |  |
|--------------|--|
| <i>u1</i>    | The size of the 'left'-loop (number of unpaired nucleotides)                   |
| <i>u2</i>    | The size of the 'right'-loop (number of unpaired nucleotides)                  |
| <i>type</i>  | The pair type of the base pair closing the interior loop                       |
| <i>type2</i> | The pair type of the enclosed base pair  |
| <i>si1</i>   | The 5'-mismatching nucleotide of the closing pair                              |
| <i>sj1</i>   | The 3'-mismatching nucleotide of the closing pair                              |
| <i>sp1</i>   | The 3'-mismatching nucleotide of the enclosed pair                             |
| <i>sq1</i>   | The 5'-mismatching nucleotide of the enclosed pair                             |
| <i>P</i>     | The datastructure containing scaled Boltzmann weights of the energy parameters |

**Returns**

The Boltzmann weight of the Interior-loop

## 16.5 The RNA Folding Grammar

The RNA folding grammar as implemented in RNAlib.

### 16.5.1 Detailed Description

The RNA folding grammar as implemented in RNAlib.  
Collaboration diagram for The RNA Folding Grammar:

### Modules

- [Fine-tuning of the Implemented Models](#)

*Functions and data structures to fine-tune the implemented secondary structure evaluation model.*

- [Energy Parameters](#)

*All relevant functions to retrieve and copy pre-calculated energy parameter sets as well as reading/writing the energy parameter set from/to file(s).*

- [Extending the Folding Grammar with Additional Domains](#)

*This module covers simple and straight-forward extensions to the RNA folding grammar.*

- [Constraining the RNA Folding Grammar](#)

*This module provides general functions that allow for an easy control of constrained secondary structure prediction and evaluation.*

### Files

- file [grammar.h](#)

*Implementations for the RNA folding grammar.*

## Data Structures

- struct [vrna\\_gr\\_aux\\_s](#)

### 16.5.2 Data Structure Documentation

#### 16.5.2.1 struct vrna\_gr\_aux\_s

##### Data Fields

- vrna\_callback\_gr\_cond \* **cb\_proc**

*A callback for pre- and post-processing of auxiliary grammar rules.*

## 16.6 Fine-tuning of the Implemented Models

Functions and data structures to fine-tune the implemented secondary structure evaluation model.

### 16.6.1 Detailed Description

Functions and data structures to fine-tune the implemented secondary structure evaluation model.

Collaboration diagram for Fine-tuning of the Implemented Models:

## Files

- file [model.h](#)

*The model details data structure and its corresponding modifiers.*

## Data Structures

- struct [vrna\\_md\\_s](#)

*The data structure that contains the complete model details used throughout the calculations.* [More...](#)

## Macros

- #define [VRNA\\_MODEL\\_DEFAULT\\_TEMPERATURE](#) 37.0  
*Default temperature for structure prediction and free energy evaluation in °C*
- #define [VRNA\\_MODEL\\_DEFAULT\\_PF\\_SCALE](#) -1  
*Default scaling factor for partition function computations.*
- #define [VRNA\\_MODEL\\_DEFAULT\\_BETA\\_SCALE](#) 1.  
*Default scaling factor for absolute thermodynamic temperature in Boltzmann factors.*
- #define [VRNA\\_MODEL\\_DEFAULT\\_DANGLES](#) 2  
*Default dangling end model.*
- #define [VRNA\\_MODEL\\_DEFAULT\\_SPECIAL\\_HP](#) 1  
*Default model behavior for lookup of special tri-, tetra-, and hexa-loops.*
- #define [VRNA\\_MODEL\\_DEFAULT\\_NO\\_LP](#) 0  
*Default model behavior for so-called 'lonely pairs'.*
- #define [VRNA\\_MODEL\\_DEFAULT\\_NO\\_GU](#) 0  
*Default model behavior for G-U base pairs.*
- #define [VRNA\\_MODEL\\_DEFAULT\\_NO\\_GU\\_CLOSURE](#) 0  
*Default model behavior for G-U base pairs closing a loop.*
- #define [VRNA\\_MODEL\\_DEFAULT\\_CIRC](#) 0  
*Default model behavior to treat a molecule as a circular RNA (DNA)*
- #define [VRNA\\_MODEL\\_DEFAULT\\_GQUAD](#) 0  
*Default model behavior regarding the treatment of G-Quadruplexes.*
- #define [VRNA\\_MODEL\\_DEFAULT\\_UNIQ\\_ML](#) 0

- `#define VRNA_MODEL_DEFAULT_ENERGY_SET 0`

*Default model behavior on which energy set to use.*
- `#define VRNA_MODEL_DEFAULT_BACKTRACK 1`

*Default model behavior with regards to backtracking of structures.*
- `#define VRNA_MODEL_DEFAULT_BACKTRACK_TYPE 'F'`

*Default model behavior on what type of backtracking to perform.*
- `#define VRNA_MODEL_DEFAULT_COMPUTE_BPP 1`

*Default model behavior with regards to computing base pair probabilities.*
- `#define VRNA_MODEL_DEFAULT_MAX_BP_SPAN -1`

*Default model behavior for the allowed maximum base pair span.*
- `#define VRNA_MODEL_DEFAULT_WINDOW_SIZE -1`

*Default model behavior for the sliding window approach.*
- `#define VRNA_MODEL_DEFAULT_LOG_ML 0`

*Default model behavior on how to evaluate the energy contribution of multi-branch loops.*
- `#define VRNA_MODEL_DEFAULT_ALI_OLD_EN 0`

*Default model behavior for consensus structure energy evaluation.*
- `#define VRNA_MODEL_DEFAULT_ALI_RIBO 0`

*Default model behavior for consensus structure co-variance contribution assessment.*
- `#define VRNA_MODEL_DEFAULT_ALI_CV_FACT 1.`

*Default model behavior for weighting the co-variance score in consensus structure prediction.*
- `#define VRNA_MODEL_DEFAULT_ALI_NC_FACT 1.`

*Default model behavior for weighting the nucleotide conservation? in consensus structure prediction.*
- `#define MAXALPHA 20`

*Maximal length of alphabet.*

## Typedefs

- `typedef struct vrna_md_s vrna_md_t`

*Typename for the model details data structure `vrna_md_s`.*

## Functions

- `void vrna_md_set_default (vrna_md_t *md)`

*Apply default model details to a provided `vrna_md_t` data structure.*
- `void vrna_md_update (vrna_md_t *md)`

*Update the model details data structure.*
- `vrna_md_t * vrna_md_copy (vrna_md_t *md_to, const vrna_md_t *md_from)`

*Copy/Clone a `vrna_md_t` model.*
- `char * vrna_md_option_string (vrna_md_t *md)`

*Get a corresponding commandline parameter string of the options in a `vrna_md_t`.*
- `void vrna_md_defaults_reset (vrna_md_t *md_p)`

*Reset the global default model details to a specific set of parameters, or their initial values.*
- `void vrna_md_defaults_temperature (double T)`

*Set default temperature for energy evaluation of loops.*
- `double vrna_md_defaults_temperature_get (void)`

*Get default temperature for energy evaluation of loops.*
- `void vrna_md_defaults_betaScale (double b)`

*Set default scaling factor of thermodynamic temperature in Boltzmann factors.*
- `double vrna_md_defaults_betaScale_get (void)`

*Get default scaling factor of thermodynamic temperature in Boltzmann factors.*
- `void vrna_md_defaults_dangles (int d)`

- `int vrna_md_defaults_dangles_get (void)`  
    Set default dangle model for structure prediction.
- `void vrna_md_defaults_special_hp (int flag)`  
    Get default dangle model for structure prediction.
- `int vrna_md_defaults_special_hp_get (void)`  
    Set default behavior for lookup of tabulated free energies for special hairpin loops, such as Tri-, Tetra-, or Hexa-loops.
- `void vrna_md_defaults_noLP (int flag)`  
    Get default behavior for lookup of tabulated free energies for special hairpin loops, such as Tri-, Tetra-, or Hexa-loops.
- `int vrna_md_defaults_noLP_get (void)`  
    Set default behavior for prediction of canonical secondary structures.
- `void vrna_md_defaults_noGU (int flag)`  
    Get default behavior for prediction of canonical secondary structures.
- `int vrna_md_defaults_noGU_get (void)`  
    Set default behavior for treatment of G-U wobble pairs.
- `void vrna_md_defaults_noGUclosure (int flag)`  
    Get default behavior for treatment of G-U wobble pairs.
- `int vrna_md_defaults_noGUclosure_get (void)`  
    Set default behavior for G-U pairs as closing pair for loops.
- `void vrna_md_defaults_logML (int flag)`  
    Get default behavior for G-U pairs as closing pair for loops.
- `int vrna_md_defaults_logML_get (void)`  
    Set default behavior recomputing free energies of multi-branch loops using a logarithmic model.
- `void vrna_md_defaults_circ (int flag)`  
    Get default behavior recomputing free energies of multi-branch loops using a logarithmic model.
- `int vrna_md_defaults_circ_get (void)`  
    Set default behavior whether input sequences are circularized.
- `void vrna_md_defaults_gquad (int flag)`  
    Get default behavior whether input sequences are circularized.
- `int vrna_md_defaults_gquad_get (void)`  
    Set default behavior for treatment of G-Quadruplexes.
- `void vrna_md_defaults_uniq_ML (int flag)`  
    Get default behavior for treatment of G-Quadruplexes.
- `int vrna_md_defaults_uniq_ML_get (void)`  
    Set default behavior for creating additional matrix for unique multi-branch loop prediction.
- `void vrna_md_defaults_energy_set (int e)`  
    Get default behavior for creating additional matrix for unique multi-branch loop prediction.
- `int vrna_md_defaults_energy_set_get (void)`  
    Set default energy set.
- `void vrna_md_defaults_backtrack (int flag)`  
    Get default energy set.
- `int vrna_md_defaults_backtrack_get (void)`  
    Set default behavior for whether to backtrack secondary structures.
- `void vrna_md_defaults_backtrack_type (char t)`  
    Get default behavior for whether to backtrack secondary structures.
- `char vrna_md_defaults_backtrack_type_get (void)`  
    Set default backtrack type, i.e. which DP matrix is used.
- `void vrna_md_defaults_compute_bpp (int flag)`  
    Get default backtrack type, i.e. which DP matrix is used.
- `int vrna_md_defaults_compute_bpp_get (void)`  
    Set the default behavior for whether to compute base pair probabilities after partition function computation.
- `int vrna_md_defaults_compute_bpp_get (void)`  
    Get the default behavior for whether to compute base pair probabilities after partition function computation.

- void `vrna_md_defaults_max_bp_span` (int span)  
*Set default maximal base pair span.*
- int `vrna_md_defaults_max_bp_span_get` (void)  
*Get default maximal base pair span.*
- void `vrna_md_defaults_min_loop_size` (int size)  
*Set default minimal loop size.*
- int `vrna_md_defaults_min_loop_size_get` (void)  
*Get default minimal loop size.*
- void `vrna_md_defaults_window_size` (int size)  
*Set default window size for sliding window structure prediction approaches.*
- int `vrna_md_defaults_window_size_get` (void)  
*Get default window size for sliding window structure prediction approaches.*
- void `vrna_md_defaults_oldAliEn` (int flag)  
*Set default behavior for whether to use old energy model for comparative structure prediction.*
- int `vrna_md_defaults_oldAliEn_get` (void)  
*Get default behavior for whether to use old energy model for comparative structure prediction.*
- void `vrna_md_defaults_ribo` (int flag)  
*Set default behavior for whether to use Ribosum Scoring in comparative structure prediction.*
- int `vrna_md_defaults_ribo_get` (void)  
*Get default behavior for whether to use Ribosum Scoring in comparative structure prediction.*
- void `vrna_md_defaults_cv_fact` (double factor)  
*Set the default co-variance scaling factor used in comparative structure prediction.*
- double `vrna_md_defaults_cv_fact_get` (void)  
*Get the default co-variance scaling factor used in comparative structure prediction.*
- void `vrna_md_defaults_nc_fact` (double factor)
- double `vrna_md_defaults_nc_fact_get` (void)
- void `vrna_md_defaults_sfact` (double factor)  
*Set the default scaling factor used to avoid under-/overflows in partition function computation.*
- double `vrna_md_defaults_sfact_get` (void)  
*Get the default scaling factor used to avoid under-/overflows in partition function computation.*
- void `set_model_details` (`vrna_md_t` \*md)  
*Set default model details.*

## Variables

- double `temperature`  
*Rescale energy parameters to a temperature in degC.*
- double `pf_scale`  
*A scaling factor used by `pf_fold()` to avoid overflows.*
- int `dangles`  
*Switch the energy model for dangling end contributions (0, 1, 2, 3)*
- int `tetra_loop`  
*Include special stabilizing energies for some tri-, tetra- and hexa-loops;.*
- int `noLonelyPairs`  
*Global switch to avoid/allow helices of length 1.*
- int `noGU`  
*Global switch to forbid/allow GU base pairs at all.*
- int `no_closingGU`  
*GU allowed only inside stacks if set to 1.*
- int `circ`  
*backward compatibility variable.. this does not effect anything*

- int **gquad**  
*Allow G-quadruplex formation.*
- int **uniq\_ML**  
*do ML decomposition uniquely (for subopt)*
- int **energy\_set**  
*0 = BP; 1=any with GC; 2=any with AU-parameter*
- int **do\_backtrack**  
*do backtracking, i.e. compute secondary structures or base pair probabilities*
- char **backtrack\_type**  
*A backtrack array marker for [inverse\\_fold\(\)](#)*
- char \* **nonstandards**  
*contains allowed non standard base pairs*
- int **max\_bp\_span**  
*Maximum allowed base pair span.*
- int **oldAliEn**  
*use old alifold energies (with gaps)*
- int **ribo**  
*use ribosum matrices*
- int **logML**  
*if nonzero use logarithmic ML energy in [energy\\_of\\_struct](#)*

## 16.6.2 Data Structure Documentation

### 16.6.2.1 struct vrna\_md\_s

The data structure that contains the complete model details used throughout the calculations.

For convenience reasons, we provide the type name **vrna\_md\_t** to address this data structure without the use of the `struct` keyword

See also

[vrna\\_md\\_set\\_default\(\)](#), [set\\_model\\_details\(\)](#), [vrna\\_md\\_update\(\)](#), [vrna\\_md\\_t](#)

**SWIG Wrapper Notes** This data structure is wrapped as an object **md** with multiple related functions attached as methods.

A new set of default parameters can be obtained by calling the constructor of **md**:

- *md()* – Initialize with default settings

The resulting object has a list of attached methods which directly correspond to functions that mainly operate on the corresponding C data structure:

- *reset()* – [vrna\\_md\\_set\\_default\(\)](#)
- *set\_from\_globals()* – [set\\_model\\_details\(\)](#)
- *option\_string()* – [vrna\\_md\\_option\\_string\(\)](#)

Note, that default parameters can be modified by directly setting any of the following global variables. Internally, getting/setting default parameters using their global variable representative translates into calls of the following functions, therefore these wrappers for these functions do not exist in the scripting language interface(s):

| global variable | C getter  | C setter  |
|-----------------|---|---|
| temperature     | <a href="#">vrna_md_defaults_temperature_get()</a>    | <a href="#">vrna_md_defaults_temperature()</a>    |
| dangles         | <a href="#">vrna_md_defaults_dangles_get()</a>        | <a href="#">vrna_md_defaults_dangles()</a>        |
| betaScale       | <a href="#">vrna_md_defaults_betaScale_get()</a>      | <a href="#">vrna_md_defaults_betaScale()</a>      |
| tetra_loop      | this is an alias of <i>special_hp</i>                 |   |
| special_hp      | <a href="#">vrna_md_defaults_special_hp_get()</a>     | <a href="#">vrna_md_defaults_special_hp()</a>     |
| noLonelyPairs   | this is an alias of <i>noLP</i>                       |   |
| noLP            | <a href="#">vrna_md_defaults_noLP_get()</a>           | <a href="#">vrna_md_defaults_noLP()</a>           |
| noGU            | <a href="#">vrna_md_defaults_noGU_get()</a>           | <a href="#">vrna_md_defaults_noGU()</a>           |
| no_closingGU    | this is an alias of <i>noGUclosure</i>                |   |
| noGUclosure     | <a href="#">vrna_md_defaults_noGUclosure_get()</a>    | <a href="#">vrna_md_defaults_noGUclosure()</a>    |
| logML           | <a href="#">vrna_md_defaults_logML_get()</a>          | <a href="#">vrna_md_defaults_logML()</a>          |
| circ            | <a href="#">vrna_md_defaults_circ_get()</a>           | <a href="#">vrna_md_defaults_circ()</a>           |
| gquad           | <a href="#">vrna_md_defaults_gquad_get()</a>          | <a href="#">vrna_md_defaults_gquad()</a>          |
| uniq_ML         | <a href="#">vrna_md_defaults_uniq_ML_get()</a>        | <a href="#">vrna_md_defaults_uniq_ML()</a>        |
| energy_set      | <a href="#">vrna_md_defaults_energy_set_get()</a>     | <a href="#">vrna_md_defaults_energy_set()</a>     |
| backtrack       | <a href="#">vrna_md_defaults_backtrack_get()</a>      | <a href="#">vrna_md_defaults_backtrack()</a>      |
| backtrack_type  | <a href="#">vrna_md_defaults_backtrack_type_get()</a> | <a href="#">vrna_md_defaults_backtrack_type()</a> |
| do_backtrack    | this is an alias of <i>compute_bpp</i>                |   |
| compute_bpp     | <a href="#">vrna_md_defaults_compute_bpp_get()</a>    | <a href="#">vrna_md_defaults_compute_bpp()</a>    |
| max_bp_span     | <a href="#">vrna_md_defaults_max_bp_span_get()</a>    | <a href="#">vrna_md_defaults_max_bp_span()</a>    |
| min_loop_size   | <a href="#">vrna_md_defaults_min_loop_size_get()</a>  | <a href="#">vrna_md_defaults_min_loop_size()</a>  |
| window_size     | <a href="#">vrna_md_defaults_window_size_get()</a>    | <a href="#">vrna_md_defaults_window_size()</a>    |
| oldAliEn        | <a href="#">vrna_md_defaults_oldAliEn_get()</a>       | <a href="#">vrna_md_defaults_oldAliEn()</a>       |
| ribo            | <a href="#">vrna_md_defaults_ribo_get()</a>           | <a href="#">vrna_md_defaults_ribo()</a>           |
| cv_fact         | <a href="#">vrna_md_defaults_cv_fact_get()</a>        | <a href="#">vrna_md_defaults_cv_fact()</a>        |
| nc_fact         | <a href="#">vrna_md_defaults_nc_fact_get()</a>        | <a href="#">vrna_md_defaults_nc_fact()</a>        |
| sfact           | <a href="#">vrna_md_defaults_sfact_get()</a>          | <a href="#">vrna_md_defaults_sfact()</a>          |

## Data Fields

- double **temperature**

*The temperature used to scale the thermodynamic parameters.*
- double **betaScale**

*A scaling factor for the thermodynamic temperature of the Boltzmann factors.*
- int **pf\_smooth**

*A flat specifying whether energies in Boltzmann factors need to be smoothed.*
- int **dangles**

*Specifies the dangle model used in any energy evaluation (0,1,2 or 3)*
- int **special\_hp**

*Include special hairpin contributions for tri, tetra and hexaloops.*
- int **noLP**

*Only consider canonical structures, i.e. no 'lonely' base pairs.*
- int **noGU**

*Do not allow GU pairs.*
- int **noGUclosure**

*Do not allow loops to be closed by GU pair.*

- int **logML**  
*Use logarithmic scaling for multiloops.*
- int **circ**  
*Assume RNA to be circular instead of linear.*
- int **gquad**  
*Include G-quadruplexes in structure prediction.*
- int **uniq\_ML**  
*Flag to ensure unique multi-branch loop decomposition during folding.*
- int **energy\_set**  
*Specifies the energy set that defines set of compatible base pairs.*
- int **backtrack**  
*Specifies whether or not secondary structures should be backtraced.*
- char **backtrack\_type**  
*Specifies in which matrix to backtrack.*
- int **compute\_bpp**  
*Specifies whether or not backward recursions for base pair probability (bpp) computation will be performed.*
- char **nonstandards** [64]  
*contains allowed non standard bases*
- int **max\_bp\_span**  
*maximum allowed base pair span*
- int **min\_loop\_size**  
*Minimum size of hairpin loops.*
- int **window\_size**  
*Size of the sliding window for locally optimal structure prediction.*
- int **oldAliEn**  
*Use old alifold energy model.*
- int **ribo**  
*Use ribosum scoring table in alifold energy model.*
- double **cv\_fact**  
*Co-variance scaling factor for consensus structure prediction.*
- double **nc\_fact**  
*Scaling factor to weight co-variance contributions of non-canonical pairs.*
- double **sfact**  
*Scaling factor for partition function scaling.*
- int **rtype** [8]  
*Reverse base pair type array.*
- short **alias** [MAXALPHA+1]  
*alias of an integer nucleotide representation*
- int **pair** [MAXALPHA+1][MAXALPHA+1]  
*Integer representation of a base pair.*
- float **pair\_dist** [7][7]  
*Base pair dissimilarity, a.k.a. distance matrix.*

#### 16.6.2.1.1 Field Documentation

**16.6.2.1.1.1 dangles int vrna\_md\_s::dangles**

Specifies the dangle model used in any energy evaluation (0,1,2 or 3)

If set to 0 no stabilizing energies are assigned to bases adjacent to helices in free ends and multiloops (so called dangling ends). Normally (dangles = 1) dangling end energies are assigned only to unpaired bases and a base cannot participate simultaneously in two dangling ends. In the partition function algorithm [vrna\\_pf\(\)](#) these checks are neglected. To provide comparability between free energy minimization and partition function algorithms, the default setting is 2. This treatment of dangling ends gives more favorable energies to helices directly adjacent to one another, which can be beneficial since such helices often do engage in stabilizing interactions through co-axial stacking.

If set to 3 co-axial stacking is explicitly included for adjacent helices in multiloops. The option affects only mfe folding and energy evaluation ([vrna\\_mfe\(\)](#) and [vrna\\_eval\\_structure\(\)](#)), as well as suboptimal folding ([vrna\\_subopt\(\)](#)) via re-evaluation of energies. Co-axial stacking with one intervening mismatch is not considered so far.

**Note**

Some functions do not implement all dangle model but only a subset of (0,1,2,3). In particular, partition function algorithms can only handle 0 and 2. Read the documentation of the particular recurrences or energy evaluation function for information about the provided dangle model.

**16.6.2.1.1.2 min\_loop\_size int vrna\_md\_s::min\_loop\_size**

Minimum size of hairpin loops.

**Note**

The default value for this field is [TURN](#), however, it may be 0 in cofolding context.

## 16.6.3 Macro Definition Documentation

### 16.6.3.1 VRNA\_MODEL\_DEFAULT\_TEMPERATURE

```
#define VRNA_MODEL_DEFAULT_TEMPERATURE 37.0
#include <ViennaRNA/model.h>
```

Default temperature for structure prediction and free energy evaluation in  $^{\circ}\text{C}$

**See also**

[vrna\\_md\\_t.temperature](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

### 16.6.3.2 VRNA\_MODEL\_DEFAULT\_PF\_SCALE

```
#define VRNA_MODEL_DEFAULT_PF_SCALE -1
#include <ViennaRNA/model.h>
```

Default scaling factor for partition function computations.

**See also**

[vrna\\_exp\\_param\\_t.pf\\_scale](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

### 16.6.3.3 VRNA\_MODEL\_DEFAULT\_BETA\_SCALE

```
#define VRNA_MODEL_DEFAULT_BETA_SCALE 1.
#include <ViennaRNA/model.h>
```

Default scaling factor for absolute thermodynamic temperature in Boltzmann factors.

**See also**

[vrna\\_exp\\_param\\_t.alpha](#), [vrna\\_md\\_t.betaScale](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

#### 16.6.3.4 VRNA\_MODEL\_DEFAULT\_DANGLES

```
#define VRNA_MODEL_DEFAULT_DANGLES 2
#include <ViennaRNA/model.h>
Default dangling end model.
```

See also

[vrna\\_md\\_t.dangles](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

#### 16.6.3.5 VRNA\_MODEL\_DEFAULT\_SPECIAL\_HP

```
#define VRNA_MODEL_DEFAULT_SPECIAL_HP 1
#include <ViennaRNA/model.h>
Default model behavior for lookup of special tri-, tetra-, and hexa-loops.
```

See also

[vrna\\_md\\_t.special\\_hp](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

#### 16.6.3.6 VRNA\_MODEL\_DEFAULT\_NO\_LP

```
#define VRNA_MODEL_DEFAULT_NO_LP 0
#include <ViennaRNA/model.h>
Default model behavior for so-called 'lonely pairs'.
```

See also

[vrna\\_md\\_t.noLP](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

#### 16.6.3.7 VRNA\_MODEL\_DEFAULT\_NO\_GU

```
#define VRNA_MODEL_DEFAULT_NO_GU 0
#include <ViennaRNA/model.h>
Default model behavior for G-U base pairs.
```

See also

[vrna\\_md\\_t.noGU](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

#### 16.6.3.8 VRNA\_MODEL\_DEFAULT\_NO\_GU\_CLOSURE

```
#define VRNA_MODEL_DEFAULT_NO_GU_CLOSURE 0
#include <ViennaRNA/model.h>
Default model behavior for G-U base pairs closing a loop.
```

See also

[vrna\\_md\\_t.noGUclosure](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

#### 16.6.3.9 VRNA\_MODEL\_DEFAULT\_CIRC

```
#define VRNA_MODEL_DEFAULT_CIRC 0
#include <ViennaRNA/model.h>
Default model behavior to treat a molecule as a circular RNA (DNA)
```

See also

[vrna\\_md\\_t.circ](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

### 16.6.3.10 VRNA\_MODEL\_DEFAULT\_GQUAD

```
#define VRNA_MODEL_DEFAULT_GQUAD 0
#include <ViennaRNA/model.h>
```

Default model behavior regarding the treatment of G-Quadruplexes.

See also

[vrna\\_md\\_t.gquad](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

### 16.6.3.11 VRNA\_MODEL\_DEFAULT\_UNIQ\_ML

```
#define VRNA_MODEL_DEFAULT_UNIQ_ML 0
#include <ViennaRNA/model.h>
```

Default behavior of the model regarding unique multi-branch loop decomposition.

See also

[vrna\\_md\\_t.uniq\\_ML](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

### 16.6.3.12 VRNA\_MODEL\_DEFAULT\_ENERGY\_SET

```
#define VRNA_MODEL_DEFAULT_ENERGY_SET 0
#include <ViennaRNA/model.h>
```

Default model behavior on which energy set to use.

See also

[vrna\\_md\\_t.energy\\_set](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

### 16.6.3.13 VRNA\_MODEL\_DEFAULT\_BACKTRACK

```
#define VRNA_MODEL_DEFAULT_BACKTRACK 1
#include <ViennaRNA/model.h>
```

Default model behavior with regards to backtracking of structures.

See also

[vrna\\_md\\_t.backtrack](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

### 16.6.3.14 VRNA\_MODEL\_DEFAULT\_BACKTRACK\_TYPE

```
#define VRNA_MODEL_DEFAULT_BACKTRACK_TYPE 'F'
#include <ViennaRNA/model.h>
```

Default model behavior on what type of backtracking to perform.

See also

[vrna\\_md\\_t.backtrack\\_type](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

### 16.6.3.15 VRNA\_MODEL\_DEFAULT\_COMPUTE\_BPP

```
#define VRNA_MODEL_DEFAULT_COMPUTE_BPP 1
#include <ViennaRNA/model.h>
```

Default model behavior with regards to computing base pair probabilities.

See also

[vrna\\_md\\_t.compute\\_bpp](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

### 16.6.3.16 VRNA\_MODEL\_DEFAULT\_MAX\_BP\_SPAN

```
#define VRNA_MODEL_DEFAULT_MAX_BP_SPAN -1
#include <ViennaRNA/model.h>
```

Default model behavior for the allowed maximum base pair span.

See also

[vrna\\_md\\_t.max\\_bp\\_span](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

### 16.6.3.17 VRNA\_MODEL\_DEFAULT\_WINDOW\_SIZE

```
#define VRNA_MODEL_DEFAULT_WINDOW_SIZE -1
#include <ViennaRNA/model.h>
```

Default model behavior for the sliding window approach.

See also

[vrna\\_md\\_t.window\\_size](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

### 16.6.3.18 VRNA\_MODEL\_DEFAULT\_LOG\_ML

```
#define VRNA_MODEL_DEFAULT_LOG_ML 0
#include <ViennaRNA/model.h>
```

Default model behavior on how to evaluate the energy contribution of multi-branch loops.

See also

[vrna\\_md\\_t.logML](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

### 16.6.3.19 VRNA\_MODEL\_DEFAULT\_ALI\_OLD\_EN

```
#define VRNA_MODEL_DEFAULT_ALI_OLD_EN 0
#include <ViennaRNA/model.h>
```

Default model behavior for consensus structure energy evaluation.

See also

[vrna\\_md\\_t.oldAliEn](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

### 16.6.3.20 VRNA\_MODEL\_DEFAULT\_ALI\_RIBO

```
#define VRNA_MODEL_DEFAULT_ALI_RIBO 0
#include <ViennaRNA/model.h>
```

Default model behavior for consensus structure co-variance contribution assessment.

See also

[vrna\\_md\\_t.ribo](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

### 16.6.3.21 VRNA\_MODEL\_DEFAULT\_ALI\_CV\_FACT

```
#define VRNA_MODEL_DEFAULT_ALI_CV_FACT 1.
#include <ViennaRNA/model.h>
```

Default model behavior for weighting the co-variance score in consensus structure prediction.

See also

[vrna\\_md\\_t.cv\\_fact](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

### 16.6.3.22 VRNA\_MODEL\_DEFAULT\_ALI\_NC\_FACT

```
#define VRNA_MODEL_DEFAULT_ALI_NC_FACT 1.
#include <ViennaRNA/model.h>
```

Default model behavior for weighting the nucleotide conservation? in consensus structure prediction.

See also

[vrna\\_md\\_t.nc\\_fact](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#)

## 16.6.4 Function Documentation

### 16.6.4.1 vrna\_md\_set\_default()

```
void vrna_md_set_default (
    vrna_md_t * md )
```

```
#include <ViennaRNA/model.h>
```

Apply default model details to a provided [vrna\\_md\\_t](#) data structure.

Use this function to initialize a [vrna\\_md\\_t](#) data structure with its default values

Parameters

|           |   |
|-----------|---|
| <i>md</i> | A pointer to the data structure that is about to be initialized |
|-----------|---|

### 16.6.4.2 vrna\_md\_update()

```
void vrna_md_update (
    vrna_md_t * md )
```

```
#include <ViennaRNA/model.h>
```

Update the model details data structure.

This function should be called after changing the [vrna\\_md\\_t.energy\\_set](#) attribute since it re-initializes base pairing related arrays within the [vrna\\_md\\_t](#) data structure. In particular, [vrna\\_md\\_t.pair](#), [vrna\\_md\\_t.alias](#), and [vrna\\_md\\_t.rtype](#) are set to the values that correspond to the specified [vrna\\_md\\_t.energy\\_set](#) option

See also

[vrna\\_md\\_t](#), [vrna\\_md\\_t.energy\\_set](#), [vrna\\_md\\_t.pair](#), [vrna\\_md\\_t.rtype](#), [vrna\\_md\\_t.alias](#), [vrna\\_md\\_set\\_default\(\)](#)

### 16.6.4.3 vrna\_md\_copy()

```
vrna_md_t * vrna_md_copy (
    vrna_md_t * md_to,
    const vrna_md_t * md_from )
```

```
#include <ViennaRNA/model.h>
```

Copy/Clone a [vrna\\_md\\_t](#) model.

Use this function to clone a given model either inplace (target container *md\_to* given) or create a copy by cloning the source model and returning it (*md\_to* == NULL).

Parameters

|                |   |
|----------------|---|
| <i>md_to</i>   | The model to be overwritten (if non-NULL and <i>md_to</i> != <i>md_from</i> ) |
| <i>md_from</i> | The model to copy (if non-NULL)   |

**Returns**

A pointer to the copy model (or NULL if `md_from == NULL`)

**16.6.4.4 `vrna_md_option_string()`**

```
char * vrna_md_option_string (
    vrna_md_t * md )
#include <ViennaRNA/model.h>
```

Get a corresponding commandline parameter string of the options in a `vrna_md_t`.

**Note**

This function is not threadsafe!

**16.6.4.5 `vrna_md_defaults_reset()`**

```
void vrna_md_defaults_reset (
    vrna_md_t * md_p )
#include <ViennaRNA/model.h>
```

Reset the global default model details to a specific set of parameters, or their initial values.

This function resets the global default model details to their initial values, i.e. as specified by the ViennaRNA Package release, upon passing NULL as argument. Alternatively it resets them according to a set of provided parameters.

**Note**

The global default parameters affect all function calls of RNAlib where model details are not explicitly provided. Hence, any change of them is not considered threadsafe

**Warning**

This function first resets the global default settings to factory defaults, and only then applies user provided settings (if any). User settings that do not meet specifications are skipped.

**See also**

[vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#)

**Parameters**

|                   |  |
|-------------------|--|
| <code>md_p</code> | A set of model details to use as global default (if NULL is passed, factory defaults are restored) |
|-------------------|--|

**16.6.4.6 `vrna_md_defaults_temperature()`**

```
void vrna_md_defaults_temperature (
    double T )
#include <ViennaRNA/model.h>
```

Set default temperature for energy evaluation of loops.

**See also**

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_TEMPERATURE](#)

### Parameters

|     |                           |
|-----|---------------------------|
| $T$ | Temperature in centigrade |
|-----|---------------------------|

#### 16.6.4.7 vrna\_md\_defaults\_temperature\_get()

```
double vrna_md_defaults_temperature_get (
    void )
```

```
#include <ViennaRNA/model.h>
```

Get default temperature for energy evaluation of loops.

### See also

[vrna\\_md\\_defaults\\_temperature\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_TEMPERATURE](#)

### Returns

The global default settings for temperature in centigrade

#### 16.6.4.8 vrna\_md\_defaults\_betaScale()

```
void vrna_md_defaults_betaScale (
    double b )
```

```
#include <ViennaRNA/model.h>
```

Set default scaling factor of thermodynamic temperature in Boltzmann factors.

Boltzmann factors are then computed as  $\exp(-E/(b \cdot kT))$ .

### See also

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_BETA\\_SCALE](#)

### Parameters

|     |                                       |
|-----|---------------------------------------|
| $b$ | The scaling factor, default is<br>1.0 |
|-----|---------------------------------------|

#### 16.6.4.9 vrna\_md\_defaults\_betaScale\_get()

```
double vrna_md_defaults_betaScale_get (
    void )
```

```
#include <ViennaRNA/model.h>
```

Get default scaling factor of thermodynamic temperature in Boltzmann factors.

### See also

[vrna\\_md\\_defaults\\_betaScale\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_BETA\\_SCALE](#)

### Returns

The global default thermodynamic temperature scaling factor

**16.6.4.10 vrna\_md\_defaults\_dangles()**

```
void vrna_md_defaults_dangles (
    int d )
#include <ViennaRNA/model.h>
Set default dangle model for structure prediction.
```

**See also**

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_DANGLES](#)

**Parameters**

|          |                  |
|----------|------------------|
| <i>d</i> | The dangle model |
|----------|------------------|

**16.6.4.11 vrna\_md\_defaults\_dangles\_get()**

```
int vrna_md_defaults_dangles_get (
    void )
#include <ViennaRNA/model.h>
Get default dangle model for structure prediction.
```

**See also**

[vrna\\_md\\_defaults\\_dangles\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_DANGLES](#)

**Returns**

The global default settings for the dangle model

**16.6.4.12 vrna\_md\_defaults\_special\_hp()**

```
void vrna_md_defaults_special_hp (
    int flag )
#include <ViennaRNA/model.h>
Set default behavior for lookup of tabulated free energies for special hairpin loops, such as Tri-, Tetra-, or Hexa-loops.
```

**See also**

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_SPECIAL\\_HP](#)

**Parameters**

|             |                                    |
|-------------|------------------------------------|
| <i>flag</i> | On/Off switch (0 = OFF, else = ON) |
|-------------|------------------------------------|

**16.6.4.13 vrna\_md\_defaults\_special\_hp\_get()**

```
int vrna_md_defaults_special_hp_get (
    void )
#include <ViennaRNA/model.h>
Get default behavior for lookup of tabulated free energies for special hairpin loops, such as Tri-, Tetra-, or Hexa-loops.
```

**See also**

[vrna\\_md\\_defaults\\_special\\_hp\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_SPECIAL\\_HP](#)

**Returns**

The global default settings for the treatment of special hairpin loops

**16.6.4.14 vrna\_md\_defaults\_noLP()**

```
void vrna_md_defaults_noLP (
    int flag )
```

```
#include <ViennaRNA/model.h>
```

Set default behavior for prediction of canonical secondary structures.

**See also**

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_NO\\_LP](#)

**Parameters**

|             |                                    |
|-------------|------------------------------------|
| <i>flag</i> | On/Off switch (0 = OFF, else = ON) |
|-------------|------------------------------------|

**16.6.4.15 vrna\_md\_defaults\_noLP\_get()**

```
int vrna_md_defaults_noLP_get (
    void )
```

```
#include <ViennaRNA/model.h>
```

Get default behavior for prediction of canonical secondary structures.

**See also**

[vrna\\_md\\_defaults\\_noLP\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_NO\\_LP](#)

**Returns**

The global default settings for predicting canonical secondary structures

**16.6.4.16 vrna\_md\_defaults\_noGU()**

```
void vrna_md_defaults_noGU (
    int flag )
```

```
#include <ViennaRNA/model.h>
```

Set default behavior for treatment of G-U wobble pairs.

**See also**

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_NO\\_GU](#)

**Parameters**

|             |                                    |
|-------------|------------------------------------|
| <i>flag</i> | On/Off switch (0 = OFF, else = ON) |
|-------------|------------------------------------|

**16.6.4.17 vrna\_md\_defaults\_noGU\_get()**

```
int vrna_md_defaults_noGU_get (
    void )
#include <ViennaRNA/model.h>
Get default behavior for treatment of G-U wobble pairs.
```

**See also**

[vrna\\_md\\_defaults\\_noGU\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_NO\\_GU](#)

**Returns**

The global default settings for treatment of G-U wobble pairs

**16.6.4.18 vrna\_md\_defaults\_noGUclosure()**

```
void vrna_md_defaults_noGUclosure (
    int flag )
#include <ViennaRNA/model.h>
Set default behavior for G-U pairs as closing pair for loops.
```

**See also**

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_NO\\_GU\\_CLOSURE](#)

**Parameters**

|             |                                    |
|-------------|------------------------------------|
| <i>flag</i> | On/Off switch (0 = OFF, else = ON) |
|-------------|------------------------------------|

**16.6.4.19 vrna\_md\_defaults\_noGUclosure\_get()**

```
int vrna_md_defaults_noGUclosure_get (
    void )
#include <ViennaRNA/model.h>
Get default behavior for G-U pairs as closing pair for loops.
```

**See also**

[vrna\\_md\\_defaults\\_noGUclosure\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_NO\\_GU\\_CLOSURE](#)

**Returns**

The global default settings for treatment of G-U pairs closing a loop

**16.6.4.20 vrna\_md\_defaults\_logML()**

```
void vrna_md_defaults_logML (
    int flag )
#include <ViennaRNA/model.h>
Set default behavior recomputing free energies of multi-branch loops using a logarithmic model.
```

**See also**

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_LOG\\_ML](#)

**Parameters**

|             |                                    |
|-------------|------------------------------------|
| <i>flag</i> | On/Off switch (0 = OFF, else = ON) |
|-------------|------------------------------------|

**16.6.4.21 vrna\_md\_defaults\_logML\_get()**

```
int vrna_md_defaults_logML_get (
    void )
#include <ViennaRNA/model.h>
```

Get default behavior recomputing free energies of multi-branch loops using a logarithmic model.

**See also**

[vrna\\_md\\_defaults\\_logML\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_LOG\\_ML](#)

**Returns**

The global default settings for logarithmic model in multi-branch loop free energy evaluation

**16.6.4.22 vrna\_md\_defaults\_circ()**

```
void vrna_md_defaults_circ (
    int flag )
#include <ViennaRNA/model.h>
```

Set default behavior whether input sequences are circularized.

**See also**

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_CIRC](#)

**Parameters**

|             |                                    |
|-------------|------------------------------------|
| <i>flag</i> | On/Off switch (0 = OFF, else = ON) |
|-------------|------------------------------------|

**16.6.4.23 vrna\_md\_defaults\_circ\_get()**

```
int vrna_md_defaults_circ_get (
    void )
#include <ViennaRNA/model.h>
```

Get default behavior whether input sequences are circularized.

**See also**

[vrna\\_md\\_defaults\\_circ\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_CIRC](#)

**Returns**

The global default settings for treating input sequences as circular

**16.6.4.24 vrna\_md\_defaults\_gquad()**

```
void vrna_md_defaults_gquad (
    int flag )
```

```
#include <ViennaRNA/model.h>
Set default behavior for treatment of G-Quadruplexes.
```

#### See also

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_GQUAD](#)

#### Parameters

|             |                                    |
|-------------|------------------------------------|
| <i>flag</i> | On/Off switch (0 = OFF, else = ON) |
|-------------|------------------------------------|

#### 16.6.4.25 vrna\_md\_defaults\_gquad\_get()

```
int vrna_md_defaults_gquad_get (
    void )
#include <ViennaRNA/model.h>
Get default behavior for treatment of G-Quadruplexes.
```

#### See also

[vrna\\_md\\_defaults\\_gquad\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_GQUAD](#)

#### Returns

The global default settings for treatment of G-Quadruplexes

#### 16.6.4.26 vrna\_md\_defaults\_uniq\_ML()

```
void vrna_md_defaults_uniq_ML (
    int flag )
#include <ViennaRNA/model.h>
Set default behavior for creating additional matrix for unique multi-branch loop prediction.
```

#### Note

Activating this option usually results in higher memory consumption!

#### See also

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_UNIQ\\_ML](#)

#### Parameters

|             |                                    |
|-------------|------------------------------------|
| <i>flag</i> | On/Off switch (0 = OFF, else = ON) |
|-------------|------------------------------------|

#### 16.6.4.27 vrna\_md\_defaults\_uniq\_ML\_get()

```
int vrna_md_defaults_uniq_ML_get (
    void )
#include <ViennaRNA/model.h>
Get default behavior for creating additional matrix for unique multi-branch loop prediction.
```

**See also**

[vrna\\_md\\_defaults\\_uniq\\_ML\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_UNIQ](#)

**Returns**

The global default settings for creating additional matrices for unique multi-branch loop prediction

**16.6.4.28 vrna\_md\_defaults\_energy\_set()**

```
void vrna_md_defaults_energy_set (
    int e )
#include <ViennaRNA/model.h>
Set default energy set.
```

**See also**

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_ENERGY\\_SET](#)

**Parameters**

|   |                         |
|---|-------------------------|
| e | Energy set (0, 1, 2, 3) |
|---|-------------------------|

**16.6.4.29 vrna\_md\_defaults\_energy\_set\_get()**

```
int vrna_md_defaults_energy_set_get (
    void )
#include <ViennaRNA/model.h>
Get default energy set.
```

**See also**

[vrna\\_md\\_defaults\\_energy\\_set\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_EN](#)

**Returns**

The global default settings for the energy set

**16.6.4.30 vrna\_md\_defaults\_backtrack()**

```
void vrna_md_defaults_backtrack (
    int flag )
#include <ViennaRNA/model.h>
Set default behavior for whether to backtrack secondary structures.
```

**See also**

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_BACKTRACK](#)

**Parameters**

|      |                                    |
|------|------------------------------------|
| flag | On/Off switch (0 = OFF, else = ON) |
|------|------------------------------------|

#### 16.6.4.31 vrna\_md\_defaults\_backtrack\_get()

```
int vrna_md_defaults_backtrack_get (
    void )
#include <ViennaRNA/model.h>
Get default behavior for whether to backtrack secondary structures.
```

See also

[vrna\\_md\\_defaults\\_backtrack\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_BACKTRACK\\_TYPE](#)

Returns

The global default settings for backtracking structures

#### 16.6.4.32 vrna\_md\_defaults\_backtrack\_type()

```
void vrna_md_defaults_backtrack_type (
    char t )
#include <ViennaRNA/model.h>
Set default backtrack type, i.e. which DP matrix is used.
```

See also

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_BACKTRACK\\_TYPE](#)

Parameters

|          |                             |
|----------|-----------------------------|
| <i>t</i> | The type ('F', 'C', or 'M') |
|----------|-----------------------------|

#### 16.6.4.33 vrna\_md\_defaults\_backtrack\_type\_get()

```
char vrna_md_defaults_backtrack_type_get (
    void )
#include <ViennaRNA/model.h>
Get default backtrack type, i.e. which DP matrix is used.
```

See also

[vrna\\_md\\_defaults\\_backtrack\\_type\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_BACKTRACK\\_TYPE](#)

Returns

The global default settings that specify which DP matrix is used for backtracking

#### 16.6.4.34 vrna\_md\_defaults\_compute\_bpp()

```
void vrna_md_defaults_compute_bpp (
    int flag )
#include <ViennaRNA/model.h>
Set the default behavior for whether to compute base pair probabilities after partition function computation.
```

See also

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_COMPUTE\\_BPP](#)

### Parameters

|             |                                    |
|-------------|------------------------------------|
| <i>flag</i> | On/Off switch (0 = OFF, else = ON) |
|-------------|------------------------------------|

### 16.6.4.35 vrna\_md\_defaults\_compute\_bpp\_get()

```
int vrna_md_defaults_compute_bpp_get (
    void )
```

```
#include <ViennaRNA/model.h>
```

Get the default behavior for whether to compute base pair probabilities after partition function computation.

### See also

[vrna\\_md\\_defaults\\_compute\\_bpp\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_BPP](#)

### Returns

The global default settings that specify whether base pair probabilities are computed together with partition function

### 16.6.4.36 vrna\_md\_defaults\_max\_bp\_span()

```
void vrna_md_defaults_max_bp_span (
    int span )
```

```
#include <ViennaRNA/model.h>
```

Set default maximal base pair span.

### See also

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_MAX\\_BP\\_SPAN](#)

### Parameters

|             |                        |
|-------------|------------------------|
| <i>span</i> | Maximal base pair span |
|-------------|------------------------|

### 16.6.4.37 vrna\_md\_defaults\_max\_bp\_span\_get()

```
int vrna_md_defaults_max_bp_span_get (
    void )
```

```
#include <ViennaRNA/model.h>
```

Get default maximal base pair span.

### See also

[vrna\\_md\\_defaults\\_max\\_bp\\_span\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_MAX\\_BP\\_SPAN](#)

### Returns

The global default settings for maximum base pair span

### 16.6.4.38 vrna\_md\_defaults\_min\_loop\_size()

```
void vrna_md_defaults_min_loop_size (
    int size )
```

```
#include <ViennaRNA/model.h>
Set default minimal loop size.
```

See also

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [TURN](#)

#### Parameters

|             |  |
|-------------|--|
| <i>size</i> | Minimal size, i.e. number of unpaired nucleotides for a hairpin loop |
|-------------|--|

#### 16.6.4.39 [vrna\\_md\\_defaults\\_min\\_loop\\_size\\_get\(\)](#)

```
int vrna_md_defaults_min_loop_size_get (
    void )
#include <ViennaRNA/model.h>
Get default minimal loop size.
```

See also

[vrna\\_md\\_defaults\\_min\\_loop\\_size\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [TURN](#)

#### Returns

The global default settings for minimal size of hairpin loops

#### 16.6.4.40 [vrna\\_md\\_defaults\\_window\\_size\(\)](#)

```
void vrna_md_defaults_window_size (
    int size )
#include <ViennaRNA/model.h>
Set default window size for sliding window structure prediction approaches.
```

See also

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_WINDOW\\_SIZE](#)

#### Parameters

|             |                                |
|-------------|--------------------------------|
| <i>size</i> | The size of the sliding window |
|-------------|--------------------------------|

#### 16.6.4.41 [vrna\\_md\\_defaults\\_window\\_size\\_get\(\)](#)

```
int vrna_md_defaults_window_size_get (
    void )
#include <ViennaRNA/model.h>
Get default window size for sliding window structure prediction approaches.
```

See also

[vrna\\_md\\_defaults\\_window\\_size\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_WINDOW\\_SIZE](#)

**Returns**

The global default settings for the size of the sliding window

**16.6.4.42 vrna\_md\_defaults\_oldAliEn()**

```
void vrna_md_defaults_oldAliEn (
    int flag )
#include <ViennaRNA/model.h>
```

Set default behavior for whether to use old energy model for comparative structure prediction.

**Note**

This option is outdated. Activating the old energy model usually results in worse consensus structure predictions.

**See also**

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_ALI\\_OLD\\_EN](#)

**Parameters**

|             |                                    |
|-------------|------------------------------------|
| <i>flag</i> | On/Off switch (0 = OFF, else = ON) |
|-------------|------------------------------------|

**16.6.4.43 vrna\_md\_defaults\_oldAliEn\_get()**

```
int vrna_md_defaults_oldAliEn_get (
    void )
#include <ViennaRNA/model.h>
```

Get default behavior for whether to use old energy model for comparative structure prediction.

**See also**

[vrna\\_md\\_defaults\\_oldAliEn\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_ALI\\_O](#)

**Returns**

The global default settings for using old energy model for comparative structure prediction

**16.6.4.44 vrna\_md\_defaults\_ribo()**

```
void vrna_md_defaults_ribo (
    int flag )
#include <ViennaRNA/model.h>
```

Set default behavior for whether to use Ribosum Scoring in comparative structure prediction.

**See also**

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_ALI\\_RIBO](#)

**Parameters**

|             |                                    |
|-------------|------------------------------------|
| <i>flag</i> | On/Off switch (0 = OFF, else = ON) |
|-------------|------------------------------------|

**16.6.4.45 vrna\_md\_defaults\_ribo\_get()**

```
int vrna_md_defaults_ribo_get (
    void )
#include <ViennaRNA/model.h>
```

Get default behavior for whether to use Ribosum Scoring in comparative structure prediction.

**See also**

[vrna\\_md\\_defaults\\_ribo\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_ALI\\_RIBO](#)

**Returns**

The global default settings for using Ribosum scoring in comparative structure prediction

**16.6.4.46 vrna\_md\_defaults\_cv\_fact()**

```
void vrna_md_defaults_cv_fact (
    double factor )
#include <ViennaRNA/model.h>
```

Set the default co-variance scaling factor used in comparative structure prediction.

**See also**

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_ALI\\_CV\\_FACT](#)

**Parameters**

|               |                        |
|---------------|------------------------|
| <i>factor</i> | The co-variance factor |
|---------------|------------------------|

**16.6.4.47 vrna\_md\_defaults\_cv\_fact\_get()**

```
double vrna_md_defaults_cv_fact_get (
    void )
#include <ViennaRNA/model.h>
```

Get the default co-variance scaling factor used in comparative structure prediction.

**See also**

[vrna\\_md\\_defaults\\_cv\\_fact\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_ALI\\_CV\\_FACT](#)

**Returns**

The global default settings for the co-variance factor

**16.6.4.48 vrna\_md\_defaults\_nc\_fact()**

```
void vrna_md_defaults_nc_fact (
    double factor )
#include <ViennaRNA/model.h>
```

**See also**

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_ALI\\_NC\\_FACT](#)

## Parameters

|               |  |
|---------------|--|
| <i>factor</i> |  |
|---------------|--|

**16.6.4.49 vrna\_md\_defaults\_nc\_fact\_get()**

```
double vrna_md_defaults_nc_fact_get (
    void )
#include <ViennaRNA/model.h>
```

## See also

[vrna\\_md\\_defaults\\_nc\\_fact\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_MODEL\\_DEFAULT\\_ALI\\_NC](#)

## Returns

**16.6.4.50 vrna\_md\_defaults\_sfact()**

```
void vrna_md_defaults_sfact (
    double factor )
#include <ViennaRNA/model.h>
```

Set the default scaling factor used to avoid under-/overflows in partition function computation.

## See also

[vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#)

## Parameters

|               |                                    |
|---------------|------------------------------------|
| <i>factor</i> | The scaling factor (default: 1.07) |
|---------------|------------------------------------|

**16.6.4.51 vrna\_md\_defaults\_sfact\_get()**

```
double vrna_md_defaults_sfact_get (
    void )
#include <ViennaRNA/model.h>
```

Get the default scaling factor used to avoid under-/overflows in partition function computation.

## See also

[vrna\\_md\\_defaults\\_sfact\(\)](#), [vrna\\_md\\_defaults\\_reset\(\)](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_md\\_t](#)

## Returns

The global default settings of the scaling factor

**16.6.4.52 set\_model\_details()**

```
void set_model_details (
    vrna_md_t * md )
#include <ViennaRNA/model.h>
```

Set default model details.

Use this function if you wish to initialize a `vrna_md_t` data structure with its default values, i.e. the global model settings as provided by the deprecated global variables.

**Deprecated** This function will vanish as soon as backward compatibility of RNAlib is dropped (expected in version 3). Use `vrna_md_set_default()` instead!

#### Parameters

|                 |   |
|-----------------|---|
| <code>md</code> | A pointer to the data structure that is about to be initialized |
|-----------------|---|

## 16.6.5 Variable Documentation

### 16.6.5.1 temperature

```
double temperature [extern]
#include <ViennaRNA/model.h>
```

Rescale energy parameters to a temperature in degC.

Default is 37C. You have to call the `update_..._params()` functions after changing this parameter.

**Deprecated** Use `vrna_md_defaults_temperature()`, and `vrna_md_defaults_temperature_get()` to change, and read the global default temperature settings

#### See also

`vrna_md_defaults_temperature()`, `vrna_md_defaults_temperature_get()`, `vrna_md_defaults_reset()`

### 16.6.5.2 pf\_scale

```
double pf_scale [extern]
#include <ViennaRNA/model.h>
```

A scaling factor used by `pf_fold()` to avoid overflows.

Should be set to approximately  $\exp((-F/kT)/length)$ , where  $F$  is an estimate for the ensemble free energy, for example the minimum free energy. You must call `update_pf_params()` after changing this parameter.

If `pf_scale` is -1 (the default), an estimate will be provided automatically when computing partition functions, e.g. `pf_fold()`. The automatic estimate is usually insufficient for sequences more than a few hundred bases long.

### 16.6.5.3 dangles

```
int dangles [extern]
#include <ViennaRNA/model.h>
```

Switch the energy model for dangling end contributions (0, 1, 2, 3)

If set to 0 no stabilizing energies are assigned to bases adjacent to helices in free ends and multiloops (so called dangling ends). Normally (dangles = 1) dangling end energies are assigned only to unpaired bases and a base cannot participate simultaneously in two dangling ends. In the partition function algorithm `pf_fold()` these checks are neglected. If `dangles` is set to 2, all folding routines will follow this convention. This treatment of dangling ends gives more favorable energies to helices directly adjacent to one another, which can be beneficial since such helices often do engage in stabilizing interactions through co-axial stacking.

If `dangles` = 3 co-axial stacking is explicitly included for adjacent helices in multiloops. The option affects only mfe folding and energy evaluation (`fold()` and `energy_of_structure()`), as well as suboptimal folding (`subopt()`) via re-evaluation of energies. Co-axial stacking with one intervening mismatch is not considered so far.

Default is 2 in most algorithms, partition function algorithms can only handle 0 and 2

#### 16.6.5.4 tetra\_loop

```
int tetra_loop [extern]
#include <ViennaRNA/model.h>
Include special stabilizing energies for some tri-, tetra- and hexa-loops;
default is 1.
```

#### 16.6.5.5 noLonelyPairs

```
int noLonelyPairs [extern]
#include <ViennaRNA/model.h>
Global switch to avoid/allow helices of length 1.
Disallow all pairs which can only occur as lonely pairs (i.e. as helix of length 1). This avoids lonely base pairs in the predicted structures in most cases.
```

#### 16.6.5.6 energy\_set

```
int energy_set [extern]
#include <ViennaRNA/model.h>
0 = BP; 1=any with GC; 2=any with AU-parameter
If set to 1 or 2: fold sequences from an artificial alphabet ABCD..., where A pairs B, C pairs D, etc. using either GC (1) or AU parameters (2); default is 0, you probably don't want to change it.
```

#### 16.6.5.7 do\_backtrack

```
int do_backtrack [extern]
#include <ViennaRNA/model.h>
do backtracking, i.e. compute secondary structures or base pair probabilities
If 0, do not calculate pair probabilities in pf\_fold\(\); this is about twice as fast. Default is 1.
```

#### 16.6.5.8 backtrack\_type

```
char backtrack_type [extern]
#include <ViennaRNA/model.h>
A backtrack array marker for inverse\_fold\(\)
If set to 'C': force (1,N) to be paired, 'M' fold as if the sequence were inside a multiloop. Otherwise ('F') the usual mfe structure is computed.
```

#### 16.6.5.9 nonstandards

```
char* nonstandards [extern]
#include <ViennaRNA/model.h>
contains allowed non standard base pairs
Lists additional base pairs that will be allowed to form in addition to GC, CG, AU, UA, GU and UG. Nonstandard base pairs are given a stacking energy of 0.
```

#### 16.6.5.10 max\_bp\_span

```
int max_bp_span [extern]
#include <ViennaRNA/model.h>
Maximum allowed base pair span.
A value of -1 indicates no restriction for distant base pairs.
```

## 16.7 Energy Parameters

All relevant functions to retrieve and copy pre-calculated energy parameter sets as well as reading/writing the energy parameter set from/to file(s).

### 16.7.1 Detailed Description

All relevant functions to retrieve and copy pre-calculated energy parameter sets as well as reading/writing the energy parameter set from/to file(s).

This module covers all relevant functions for pre-calculation of the energy parameters necessary for the folding routines provided by RNAlib. Furthermore, the energy parameter set in the RNAlib can be easily exchanged by a user-defined one. It is also possible to write the current energy parameter set into a text file. Collaboration diagram for Energy Parameters:

## Modules

- [Reading/Writing Energy Parameter Sets from/to File](#)

*Read and Write energy parameter sets from and to files or strings.*

## Files

- file [basic.h](#)  
*Functions to deal with sets of energy parameters.*
- file [constants.h](#)  
*Energy parameter constants.*
- file [convert.h](#)  
*Functions and definitions for energy parameter file format conversion.*
- file [io.h](#)  
*Read and write energy parameter files.*

## Data Structures

- struct [vrna\\_param\\_s](#)  
*The datastructure that contains temperature scaled energy parameters. [More...](#)*
- struct [vrna\\_exp\\_param\\_s](#)  
*The data structure that contains temperature scaled Boltzmann weights of the energy parameters. [More...](#)*

## Typedefs

- typedef struct [vrna\\_param\\_s](#) [vrna\\_param\\_t](#)  
*Typename for the free energy parameter data structure [vrna\\_params](#).*
- typedef struct [vrna\\_exp\\_param\\_s](#) [vrna\\_exp\\_param\\_t](#)  
*Typename for the Boltzmann factor data structure [vrna\\_exp\\_params](#).*
- typedef struct [vrna\\_param\\_s](#) [paramT](#)  
*Old typename of [vrna\\_param\\_s](#).*
- typedef struct [vrna\\_exp\\_param\\_s](#) [pf\\_paramT](#)  
*Old typename of [vrna\\_exp\\_param\\_s](#).*

## Functions

- [vrna\\_param\\_t \\* vrna\\_params \(vrna\\_md\\_t \\*md\)](#)  
*Get a data structure containing prescaled free energy parameters.*
- [vrna\\_param\\_t \\* vrna\\_params\\_copy \(vrna\\_param\\_t \\*par\)](#)  
*Get a copy of the provided free energy parameters.*
- [vrna\\_exp\\_param\\_t \\* vrna\\_exp\\_params \(vrna\\_md\\_t \\*md\)](#)  
*Get a data structure containing prescaled free energy parameters already transformed to Boltzmann factors.*
- [vrna\\_exp\\_param\\_t \\* vrna\\_exp\\_params\\_comparative \(unsigned int n\\_seq, vrna\\_md\\_t \\*md\)](#)  
*Get a data structure containing prescaled free energy parameters already transformed to Boltzmann factors (alifold version)*

- `vrna_exp_param_t * vrna_exp_params_copy (vrna_exp_param_t *par)`  
*Get a copy of the provided free energy parameters (provided as Boltzmann factors)*
- `void vrna_params_subst (vrna_fold_compound_t *vc, vrna_param_t *par)`  
*Update/Reset energy parameters data structure within a `vrna_fold_compound_t`.*
- `void vrna_exp_params_subst (vrna_fold_compound_t *vc, vrna_exp_param_t *params)`  
*Update the energy parameters for subsequent partition function computations.*
- `void vrna_exp_params_rescale (vrna_fold_compound_t *vc, double *mfe)`  
*Rescale Boltzmann factors for partition function computations.*
- `void vrna_params_reset (vrna_fold_compound_t *vc, vrna_md_t *md_p)`  
*Reset free energy parameters within a `vrna_fold_compound_t` according to provided, or default model details.*
- `void vrna_exp_params_reset (vrna_fold_compound_t *vc, vrna_md_t *md_p)`  
*Reset Boltzmann factors for partition function computations within a `vrna_fold_compound_t` according to provided, or default model details.*
- `vrna_exp_param_t * get_scaled_pf_parameters (void)`
- `vrna_exp_param_t * get_boltzmann_factors (double temperature, double betaScale, vrna_md_t md, double pf_scale)`  
*Get precomputed Boltzmann factors of the loop type dependent energy contributions with independent thermodynamic temperature.*
- `vrna_exp_param_t * get_boltzmann_factor_copy (vrna_exp_param_t *parameters)`  
*Get a copy of already precomputed Boltzmann factors.*
- `vrna_exp_param_t * get_scaled_alipf_parameters (unsigned int n_seq)`  
*Get precomputed Boltzmann factors of the loop type dependent energy contributions (alifold variant)*
- `vrna_exp_param_t * get_boltzmann_factors_ali (unsigned int n_seq, double temperature, double betaScale, vrna_md_t md, double pf_scale)`  
*Get precomputed Boltzmann factors of the loop type dependent energy contributions (alifold variant) with independent thermodynamic temperature.*
- `vrna_param_t * scale_parameters (void)`  
*Get precomputed energy contributions for all the known loop types.*
- `vrna_param_t * get_scaled_parameters (double temperature, vrna_md_t md)`  
*Get precomputed energy contributions for all the known loop types.*

## 16.7.2 Data Structure Documentation

### 16.7.2.1 struct vrna\_param\_s

The datastructure that contains temperature scaled energy parameters.

Collaboration diagram for `vrna_param_s`:

#### Data Fields

- `double temperature`  
*Temperature used for loop contribution scaling.*
- `vrna_md_t model_details`  
*Model details to be used in the recursions.*
- `char param_file [256]`  
*The filename the parameters were derived from, or empty string if they represent the default.*

### 16.7.2.2 struct vrna\_exp\_param\_s

The data structure that contains temperature scaled Boltzmann weights of the energy parameters.

Collaboration diagram for `vrna_exp_param_s`:

## Data Fields

- int **id**  
*An identifier for the data structure.*
- double **pf\_scale**  
*Scaling factor to avoid over-/underflows.*
- double **temperature**  
*Temperature used for loop contribution scaling.*
- double **alpha**  
*Scaling factor for the thermodynamic temperature.*
- **vrna\_md\_t model\_details**  
*Model details to be used in the recursions.*
- char **param\_file** [256]  
*The filename the parameters were derived from, or empty string if they represent the default.*

### 16.7.2.2.1 Field Documentation

#### 16.7.2.2.1.1 **id** int vrna\_exp\_param\_s::id

An identifier for the data structure.

**Deprecated** This attribute will be removed in version 3

#### 16.7.2.2.1.2 **alpha** double vrna\_exp\_param\_s::alpha

Scaling factor for the thermodynamic temperature.

This allows for temperature scaling in Boltzmann factors independently from the energy contributions. The resulting Boltzmann factors are then computed by  $e^{-E}/(\alpha \cdot K \cdot T)$

## 16.7.3 Typedef Documentation

### 16.7.3.1 **paramT**

```
typedef struct vrna_param_s paramT
#include <ViennaRNA/params/basic.h>
Old typename of vrna_param_s.
```

**Deprecated** Use [vrna\\_param\\_t](#) instead!

### 16.7.3.2 **pf\_paramT**

```
typedef struct vrna_exp_param_s pf_paramT
#include <ViennaRNA/params/basic.h>
Old typename of vrna_exp_param_s.
```

**Deprecated** Use [vrna\\_exp\\_param\\_t](#) instead!

## 16.7.4 Function Documentation

#### 16.7.4.1 vrna\_params()

```
vrna_param_t * vrna_params (
    vrna_md_t * md )
#include <ViennaRNA/params/basic.h>
```

Get a data structure containing prescaled free energy parameters.

If a NULL pointer is passed for the model details parameter, the default model parameters are stored within the requested `vrna_param_t` structure.

See also

[vrna\\_md\\_t](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_exp\\_params\(\)](#)

#### Parameters

|                 |   |
|-----------------|---|
| <code>md</code> | A pointer to the model details to store inside the structure (Maybe NULL) |
|-----------------|---|

#### Returns

A pointer to the memory location where the requested parameters are stored

#### 16.7.4.2 vrna\_params\_copy()

```
vrna_param_t * vrna_params_copy (
    vrna_param_t * par )
#include <ViennaRNA/params/basic.h>
```

Get a copy of the provided free energy parameters.

If NULL is passed as parameter, a default set of energy parameters is created and returned.

See also

[vrna\\_params\(\)](#), [vrna\\_param\\_t](#)

#### Parameters

|                  |   |
|------------------|---|
| <code>par</code> | The free energy parameters that are to be copied (Maybe NULL) |
|------------------|---|

#### Returns

A copy or a default set of the (provided) parameters

#### 16.7.4.3 vrna\_exp\_params()

```
vrna_exp_param_t * vrna_exp_params (
    vrna_md_t * md )
#include <ViennaRNA/params/basic.h>
```

Get a data structure containing prescaled free energy parameters already transformed to Boltzmann factors.

This function returns a data structure that contains all necessary precomputed energy contributions for each type of loop.

In contrast to [vrna\\_params\(\)](#), the free energies within this data structure are stored as their Boltzmann factors, i.e.  $\exp(-E/kT)$

where  $E$  is the free energy.

If a NULL pointer is passed for the model details parameter, the default model parameters are stored within the requested `vrna_exp_param_t` structure.

## See also

[vrna\\_md\\_t](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_params\(\)](#), [vrna\\_rescale\\_pf\\_params\(\)](#)

## Parameters

|           |   |
|-----------|---|
| <i>md</i> | A pointer to the model details to store inside the structure (Maybe NULL) |
|-----------|---|

## Returns

A pointer to the memory location where the requested parameters are stored

**16.7.4.4 vrna\_exp\_params\_comparative()**

```
vrna_exp_param_t * vrna_exp_params_comparative (
    unsigned int n_seq,
    vrna_md_t * md )
```

```
#include <ViennaRNA/params/basic.h>
```

Get a data structure containing prescaled free energy parameters already transformed to Boltzmann factors (alifold version)

If a NULL pointer is passed for the model details parameter, the default model parameters are stored within the requested [vrna\\_exp\\_param\\_t](#) structure.

## See also

[vrna\\_md\\_t](#), [vrna\\_md\\_set\\_default\(\)](#), [vrna\\_exp\\_params\(\)](#), [vrna\\_params\(\)](#)

## Parameters

|              |   |
|--------------|---|
| <i>n_seq</i> | The number of sequences in the alignment                                  |
| <i>md</i>    | A pointer to the model details to store inside the structure (Maybe NULL) |

## Returns

A pointer to the memory location where the requested parameters are stored

**16.7.4.5 vrna\_exp\_params\_copy()**

```
vrna_exp_param_t * vrna_exp_params_copy (
    vrna_exp_param_t * par )
```

```
#include <ViennaRNA/params/basic.h>
```

Get a copy of the provided free energy parameters (provided as Boltzmann factors)

If NULL is passed as parameter, a default set of energy parameters is created and returned.

## See also

[vrna\\_exp\\_params\(\)](#), [vrna\\_exp\\_param\\_t](#)

## Parameters

|            |   |
|------------|---|
| <i>par</i> | The free energy parameters that are to be copied (Maybe NULL) |
|------------|---|

**Returns**

A copy or a default set of the (provided) parameters

**16.7.4.6 vrna\_params\_subst()**

```
void vrna_params_subst (
    vrna_fold_compound_t * vc,
    vrna_param_t * par )
#include <ViennaRNA/params/basic.h>
Update/Reset energy parameters data structure within a vrna_fold_compound_t.
Passing NULL as second argument leads to a reset of the energy parameters within vc to their default values.
Otherwise, the energy parameters provided will be copied over into vc.
```

**See also**

[vrna\\_params\\_reset\(\)](#), [vrna\\_param\\_t](#), [vrna\\_md\\_t](#), [vrna\\_params\(\)](#)

**Parameters**

|            |   |
|------------|---|
| <i>vc</i>  | The <a href="#">vrna_fold_compound_t</a> that is about to receive updated energy parameters |
| <i>par</i> | The energy parameters used to substitute those within vc (Maybe NULL)                       |

**SWIG Wrapper Notes** This function is attached to [vrna\\_fc\\_s](#) objects as overloaded **params\_subst()** method.

When no parameter is passed, the resulting action is the same as passing *NULL* as second parameter to [vrna\\_params\\_subst\(\)](#), i.e. resetting the parameters to the global defaults.

**16.7.4.7 vrna\_exp\_params\_subst()**

```
void vrna_exp_params_subst (
    vrna_fold_compound_t * vc,
    vrna_exp_param_t * params )
#include <ViennaRNA/params/basic.h>
Update the energy parameters for subsequent partition function computations.
This function can be used to properly assign new energy parameters for partition function computations to a vrna\_fold\_compound\_t. For this purpose, the data of the provided pointer params will be copied into vc and a recomputation of the partition function scaling factor is issued, if the pf_scale attribute of params is less than 1.0.
Passing NULL as second argument leads to a reset of the energy parameters within vc to their default values
```

**See also**

[vrna\\_exp\\_params\\_reset\(\)](#), [vrna\\_exp\\_params\\_rescale\(\)](#), [vrna\\_exp\\_param\\_t](#), [vrna\\_md\\_t](#), [vrna\\_exp\\_params\(\)](#)

**Parameters**

|               |  |
|---------------|--|
| <i>vc</i>     | The fold compound data structure       |
| <i>params</i> | A pointer to the new energy parameters |

**SWIG Wrapper Notes** This function is attached to [vrna\\_fc\\_s](#) objects as overloaded **exp\_params\_subst()** method.

When no parameter is passed, the resulting action is the same as passing *NULL* as second parameter to [vrna\\_exp\\_params\\_subst\(\)](#), i.e. resetting the parameters to the global defaults.

#### 16.7.4.8 vrna\_exp\_params\_rescale()

```
void vrna_exp_params_rescale (
    vrna_fold_compound_t * vc,
    double * mfe )
#include <ViennaRNA/params/basic.h>
```

Rescale Boltzmann factors for partition function computations.

This function may be used to (automatically) rescale the Boltzmann factors used in partition function computations. Since partition functions over subsequences can easily become extremely large, the RNAlib internally rescales them to avoid numerical over- and/or underflow. Therefore, a proper scaling factor  $s$  needs to be chosen that in turn is then used to normalize the corresponding partition functions  $\hat{q}[i,j] = q[i,j]/s^{(j-i+1)}$ .

This function provides two ways to automatically adjust the scaling factor.

1. Automatic guess
2. Automatic adjustment according to MFE

Passing `NULL` as second parameter activates the *automatic guess mode*. Here, the scaling factor is recomputed according to a mean free energy of  $184.3 \cdot \text{length} \cdot \text{cal}$  for random sequences.

##### Note

This recomputation only takes place if the `pf_scale` attribute of the `exp_params` data structure contained in `vc` has a value below `1.0`.

On the other hand, if the MFE for a sequence is known, it can be used to recompute a more robust scaling factor, since it represents the lowest free energy of the entire ensemble of structures, i.e. the highest Boltzmann factor. To activate this second mode of *automatic adjustment according to MFE*, a pointer to the MFE value needs to be passed as second argument. This value is then taken to compute the scaling factor as  $s = \exp((sfact * MFE)/kT/\text{length})$ , where `sfact` is an additional scaling weight located in the `vrna_md_t` data structure of `exp_params` in `vc`.

The computed scaling factor  $s$  will be stored as `pf_scale` attribute of the `exp_params` data structure in `vc`.

##### See also

[vrna\\_exp\\_params\\_subst\(\)](#), [vrna\\_md\\_t](#), [vrna\\_exp\\_param\\_t](#), [vrna\\_fold\\_compound\\_t](#)

##### Parameters

|                  |   |
|------------------|---|
| <code>vc</code>  | The fold compound data structure                        |
| <code>mfe</code> | A pointer to the MFE (in kcal/mol) or <code>NULL</code> |

**SWIG Wrapper Notes** This function is attached to `vrna_fc_s` objects as overloaded `exp_params_rescale()` method.

When no parameter is passed to this method, the resulting action is the same as passing `NULL` as second parameter to [vrna\\_exp\\_params\\_rescale\(\)](#), i.e. default scaling of the partition function. Passing an energy in kcal/mol, e.g. as retrieved by a previous call to the `mfe()` method, instructs all subsequent calls to scale the partition function accordingly.

#### 16.7.4.9 vrna\_params\_reset()

```
void vrna_params_reset (
    vrna_fold_compound_t * vc,
    vrna_md_t * md_p )
#include <ViennaRNA/params/basic.h>
```

Reset free energy parameters within a `vrna_fold_compound_t` according to provided, or default model details.

This function allows one to rescale free energy parameters for subsequent structure prediction or evaluation according to a set of model details, e.g. temperature values. To do so, the caller provides either a pointer to a set of model details to be used for rescaling, or `NULL` if global default setting should be used.

## See also

[vrna\\_exp\\_params\\_reset\(\)](#), [vrna\\_params\\_sub\(\)](#)

## Parameters

|             |  |
|-------------|--|
| <i>vc</i>   | The fold compound data structure                                   |
| <i>md_p</i> | A pointer to the new model details (or NULL for reset to defaults) |

**SWIG Wrapper Notes** This function is attached to [vrna\\_fc\\_s](#) objects as overloaded **params\_reset()** method.

When no parameter is passed to this method, the resulting action is the same as passing *NULL* as second parameter to [vrna\\_params\\_reset\(\)](#), i.e. global default model settings are used. Passing an object of type [vrna\\_md\\_s](#) resets the fold compound according to the specifications stored within the [vrna\\_md\\_s](#) object.

**16.7.4.10 vrna\_exp\_params\_reset()**

```
vrna_exp_params_reset (
    vrna_fold_compound_t * vc,
    vrna_md_t * md_p )
#include <ViennaRNA/params/basic.h>
```

Reset Boltzmann factors for partition function computations within a [vrna\\_fold\\_compound\\_t](#) according to provided, or default model details.

This function allows one to rescale Boltzmann factors for subsequent partition function computations according to a set of model details, e.g. temperature values. To do so, the caller provides either a pointer to a set of model details to be used for rescaling, or *NULL* if global default setting should be used.

## See also

[vrna\\_params\\_reset\(\)](#), [vrna\\_exp\\_params\\_subst\(\)](#), [vrna\\_exp\\_params\\_rescale\(\)](#)

## Parameters

|             |  |
|-------------|--|
| <i>vc</i>   | The fold compound data structure                                   |
| <i>md_p</i> | A pointer to the new model details (or NULL for reset to defaults) |

**SWIG Wrapper Notes** This function is attached to [vrna\\_fc\\_s](#) objects as overloaded **exp\_params\_reset()** method.

When no parameter is passed to this method, the resulting action is the same as passing *NULL* as second parameter to [vrna\\_exp\\_params\\_reset\(\)](#), i.e. global default model settings are used. Passing an object of type [vrna\\_md\\_s](#) resets the fold compound according to the specifications stored within the [vrna\\_md\\_s](#) object.

**16.7.4.11 get\_scaled\_pf\_parameters()**

```
vrna_exp_param_t * get_scaled_pf_parameters (
    void )
```

```
#include <ViennaRNA/params/basic.h>
```

get a data structure of type [vrna\\_exp\\_param\\_t](#) which contains the Boltzmann weights of several energy parameters scaled according to the current temperature

**Deprecated** Use [vrna\\_exp\\_params\(\)](#) instead!

#### Returns

The data structure containing Boltzmann weights for use in partition function calculations

#### 16.7.4.12 get\_boltzmann\_factors()

```
vrna_exp_param_t * get_boltzmann_factors (
    double temperature,
    double betaScale,
    vrna_md_t md,
    double pf_scale )
#include <ViennaRNA/params/basic.h>
Get precomputed Boltzmann factors of the loop type dependent energy contributions with independent thermodynamic temperature.
This function returns a data structure that contains all necessary precalculated Boltzmann factors for each loop type contribution.
In contrast to get\_scaled\_pf\_parameters\(\), this function enables setting of independent temperatures for both, the individual energy contributions as well as the thermodynamic temperature used in  $\exp(-\Delta G/kT)$ 
```

**Deprecated** Use [vrna\\_exp\\_params\(\)](#) instead!

#### See also

[get\\_scaled\\_pf\\_parameters\(\)](#), [get\\_boltzmann\\_factor\\_copy\(\)](#)

#### Parameters

|                    |  |
|--------------------|--|
| <i>temperature</i> | The temperature in degrees Celcius used for (re-)scaling the energy contributions                  |
| <i>betaScale</i>   | A scaling value that is used as a multiplication factor for the absolute temperature of the system |
| <i>md</i>          | The model details to be used   |
| <i>pf_scale</i>    | The scaling factor for the Boltzmann factors   |

#### Returns

A set of precomputed Boltzmann factors

#### 16.7.4.13 get\_boltzmann\_factor\_copy()

```
vrna_exp_param_t * get_boltzmann_factor_copy (
    vrna_exp_param_t * parameters )
#include <ViennaRNA/params/basic.h>
Get a copy of already precomputed Boltzmann factors.
```

**Deprecated** Use [vrna\\_exp\\_params\\_copy\(\)](#) instead!

#### See also

[get\\_boltzmann\\_factors\(\)](#), [get\\_scaled\\_pf\\_parameters\(\)](#)

#### Parameters

|                   |   |
|-------------------|---|
| <i>parameters</i> | The input data structure that shall be copied |
|-------------------|---|

**Returns**

A copy of the provided Boltzmann factor data set

**16.7.4.14 get\_scaled\_alipf\_parameters()**

```
vrna_exp_param_t * get_scaled_alipf_parameters (
    unsigned int n_seq )
#include <ViennaRNA/params/basic.h>
Get precomputed Boltzmann factors of the loop type dependent energy contributions (alifold variant)
```

**Deprecated** Use [vrna\\_exp\\_params\\_comparative\(\)](#) instead!

**16.7.4.15 get\_boltzmann\_factors\_ali()**

```
vrna_exp_param_t * get_boltzmann_factors_ali (
    unsigned int n_seq,
    double temperature,
    double betaScale,
    vrna_md_t md,
    double pf_scale )
#include <ViennaRNA/params/basic.h>
Get precomputed Boltzmann factors of the loop type dependent energy contributions (alifold variant) with independent thermodynamic temperature.
```

**Deprecated** Use [vrna\\_exp\\_params\\_comparative\(\)](#) instead!

**16.7.4.16 scale\_parameters()**

```
vrna_param_t * scale_parameters (
    void )
#include <ViennaRNA/params/basic.h>
Get precomputed energy contributions for all the known loop types.
```

**Note**

OpenMP: This function relies on several global model settings variables and thus is not to be considered threadsafe. See [get\\_scaled\\_parameters\(\)](#) for a completely threadsafe implementation.

**Deprecated** Use [vrna\\_params\(\)](#) instead!

**Returns**

A set of precomputed energy contributions

**16.7.4.17 get\_scaled\_parameters()**

```
vrna_param_t * get_scaled_parameters (
    double temperature,
    vrna_md_t md )
#include <ViennaRNA/params/basic.h>
Get precomputed energy contributions for all the known loop types.
Call this function to retrieve precomputed energy contributions, i.e. scaled according to the temperature passed.
Furthermore, this function assumes a data structure that contains the model details as well, such that subsequent folding recursions are able to retrieve the correct model settings
```

**Deprecated** Use [vrna\\_params\(\)](#) instead!

See also

[vrna\\_md\\_t, set\\_model\\_details\(\)](#)

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <i>temperature</i> | The temperature in degrees Celcius |
| <i>md</i>          | The model details                  |

#### Returns

precomputed energy contributions and model settings

## 16.8 Extending the Folding Grammar with Additional Domains

This module covers simple and straight-forward extensions to the RNA folding grammar.

### 16.8.1 Detailed Description

This module covers simple and straight-forward extensions to the RNA folding grammar.

Collaboration diagram for Extending the Folding Grammar with Additional Domains:

#### Modules

- [Unstructured Domains](#)

*Add and modify unstructured domains to the RNA folding grammar.*

- [Structured Domains](#)

*Add and modify structured domains to the RNA folding grammar.*

## 16.9 Unstructured Domains

Add and modify unstructured domains to the RNA folding grammar.

### 16.9.1 Detailed Description

Add and modify unstructured domains to the RNA folding grammar.

This module provides the tools to add and modify unstructured domains to the production rules of the RNA folding grammar. Usually this functionality is utilized for incorporating ligand binding to unpaired stretches of an RNA.

**Bug** Although the additional production rule(s) for unstructured domains as described in [Unstructured Domains](#) are always treated as 'segments possibly bound to one or more ligands', the current implementation requires that at least one ligand is bound. The default implementation already takes care of the required changes, however, upon using callback functions other than the default ones, one has to take care of this fact. Please also note, that this behavior might change in one of the next releases, such that the decomposition schemes as shown above comply with the actual implementation.

A default implementation allows one to readily use this feature by simply adding sequence motifs and corresponding binding free energies with the function [vrna\\_ud\\_add\\_motif\(\)](#) (see also [Ligands Binding to Unstructured Domains](#)). The grammar extension is realized using a callback function that

- evaluates the binding free energy of a ligand to its target sequence segment (white boxes in the figures above), or
- returns the free energy of an unpaired stretch possibly bound by a ligand, stored in the additional *UDP* matrix.

The callback is passed the segment positions, the loop context, and which of the two above mentioned evaluations are required. A second callback implements the pre-processing step that prepares the *UDP* matrix by evaluating all possible cases of the additional production rule. Both callbacks have a default implementation in *RNAlib*, but may be over-written by a user-implementation, making it fully user-customizable.

For equilibrium probability computations, two additional callbacks exist. One to store/add and one to retrieve the probability of unstructured domains at particular positions. Our implementation already takes care of computing the probabilities, but users of the unstructured domain feature are required to provide a mechanism to efficiently store/add the corresponding values into some external data structure. Collaboration diagram for Unstructured Domains:

## Files

- file [unstructured\\_domains.h](#)

*Functions to modify unstructured domains, e.g. to incorporate ligands binding to unpaired stretches.*

## Data Structures

- struct [vrna\\_unstructured\\_domain\\_s](#)

*Data structure to store all functionality for ligand binding. [More...](#)*

## Macros

- #define **VRNA\_UNSTRUCTURED\_DOMAIN\_EXT\_LOOP** 1U  
*Flag to indicate ligand bound to unpaired stretch in the exterior loop.*
- #define **VRNA\_UNSTRUCTURED\_DOMAIN\_HP\_LOOP** 2U  
*Flag to indicate ligand bound to unpaired stretch in a hairpin loop.*
- #define **VRNA\_UNSTRUCTURED\_DOMAIN\_INT\_LOOP** 4U  
*Flag to indicate ligand bound to unpaired stretch in an interior loop.*
- #define **VRNA\_UNSTRUCTURED\_DOMAIN\_MB\_LOOP** 8U  
*Flag to indicate ligand bound to unpaired stretch in a multibranch loop.*
- #define **VRNA\_UNSTRUCTURED\_DOMAIN\_MOTIF** 16U  
*Flag to indicate ligand binding without additional unbound nucleotides (motif-only)*
- #define **VRNA\_UNSTRUCTURED\_DOMAIN\_ALL\_LOOPS**  
*Flag to indicate ligand bound to unpaired stretch in any loop (convenience macro)*

## Typedefs

- typedef struct [vrna\\_unstructured\\_domain\\_s](#) **vrna\_ud\_t**  
*Typename for the ligand binding extension data structure [vrna\\_unstructured\\_domain\\_s](#).*
- typedef int() [vrna\\_callback\\_ud\\_energy](#)([vrna\\_fold\\_compound\\_t](#) \*vc, int i, int j, unsigned int loop\_type, void \*data)  
*Callback to retrieve binding free energy of a ligand bound to an unpaired sequence segment.*
- typedef **FLT\_OR\_DBL()** [vrna\\_callback\\_ud\\_exp\\_energy](#)([vrna\\_fold\\_compound\\_t](#) \*vc, int i, int j, unsigned int loop\_type, void \*data)  
*Callback to retrieve Boltzmann factor of the binding free energy of a ligand bound to an unpaired sequence segment.*
- typedef void() [vrna\\_callback\\_ud\\_production](#)([vrna\\_fold\\_compound\\_t](#) \*vc, void \*data)  
*Callback for pre-processing the production rule of the ligand binding to unpaired stretches feature.*
- typedef void() [vrna\\_callback\\_ud\\_exp\\_production](#)([vrna\\_fold\\_compound\\_t](#) \*vc, void \*data)  
*Callback for pre-processing the production rule of the ligand binding to unpaired stretches feature (partition function variant)*
- typedef void() [vrna\\_callback\\_ud\\_probs\\_add](#)([vrna\\_fold\\_compound\\_t](#) \*vc, int i, int j, unsigned int loop\_type, **FLT\_OR\_DBL** exp\_energy, void \*data)  
*Callback to store/add equilibrium probability for a ligand bound to an unpaired sequence segment.*
- typedef **FLT\_OR\_DBL()** [vrna\\_callback\\_ud\\_probs\\_get](#)([vrna\\_fold\\_compound\\_t](#) \*vc, int i, int j, unsigned int loop\_type, int motif, void \*data)  
*Callback to retrieve equilibrium probability for a ligand bound to an unpaired sequence segment.*

## Functions

- `vrna_ud_motif_t * vrna_ud_motifs_centroid (vrna_fold_compound_t *fc, const char *structure)`  
*Detect unstructured domains in centroid structure.*
- `vrna_ud_motif_t * vrna_ud_motifs_MEA (vrna_fold_compound_t *fc, const char *structure, vrna_ep_t *probability_list)`  
*Detect unstructured domains in MEA structure.*
- `vrna_ud_motif_t * vrna_ud_motifs_MFE (vrna_fold_compound_t *fc, const char *structure)`  
*Detect unstructured domains in MFE structure.*
- `void vrna_ud_add_motif (vrna_fold_compound_t *vc, const char *motif, double motif_en, const char *motif_name, unsigned int loop_type)`  
*Add an unstructured domain motif, e.g. for ligand binding.*
- `void vrna_ud_remove (vrna_fold_compound_t *vc)`  
*Remove ligand binding to unpaired stretches.*
- `void vrna_ud_set_data (vrna_fold_compound_t *vc, void *data, vrna_callback_free_auxdata *free_cb)`  
*Attach an auxiliary data structure.*
- `void vrna_ud_set_prod_rule_cb (vrna_fold_compound_t *vc, vrna_callback_ud_production *pre_cb, vrna_callback_ud_energy *e_cb)`  
*Attach production rule callbacks for free energies computations.*
- `void vrna_ud_set_exp_prod_rule_cb (vrna_fold_compound_t *vc, vrna_callback_ud_exp_production *pre_cb, vrna_callback_ud_exp_energy *exp_e_cb)`  
*Attach production rule for partition function.*

### 16.9.2 Data Structure Documentation

#### 16.9.2.1 struct vrna\_unstructured\_domain\_s

Data structure to store all functionality for ligand binding.

#### Data Fields

- `int uniq_motif_count`  
*The unique number of motifs of different lengths.*
- `unsigned int * uniq_motif_size`  
*An array storing a unique list of motif lengths.*
- `int motif_count`  
*Total number of distinguished motifs.*
- `char ** motif`  
*Motif sequences.*
- `char ** motif_name`  
*Motif identifier/name.*
- `unsigned int * motif_size`  
*Motif lengths.*
- `double * motif_en`  
*Ligand binding free energy contribution.*
- `unsigned int * motif_type`  
*Type of motif, i.e. loop type the ligand binds to.*
- `vrna_callback_ud_production * prod_cb`  
*Callback to ligand binding production rule, i.e. create/fill DP free energy matrices.*
- `vrna_callback_ud_exp_production * exp_prod_cb`  
*Callback to ligand binding production rule, i.e. create/fill DP partition function matrices.*
- `vrna_callback_ud_energy * energy_cb`  
*Callback to evaluate free energy of ligand binding to a particular unpaired stretch.*

- **`vrna_callback_ud_exp_energy`** \* **`exp_energy_cb`**  
*Callback to evaluate Boltzmann factor of ligand binding to a particular unpaired stretch.*
- **`void *`** **`data`**  
*Auxiliary data structure passed to energy evaluation callbacks.*
- **`vrna_callback_free_auxdata`** \* **`free_data`**  
*Callback to free auxiliary data structure.*
- **`vrna_callback_ud_probs_add`** \* **`probs_add`**  
*Callback to store/add outside partition function.*
- **`vrna_callback_ud_probs_get`** \* **`probs_get`**  
*Callback to retrieve outside partition function.*

#### 16.9.2.1.1 Field Documentation

##### 16.9.2.1.1.1 **`prod_cb`** `vrna_callback_ud_production`\* `vrna_unstructured_domain_s::prod_cb`

Callback to ligand binding production rule, i.e. create/fill DP free energy matrices.

This callback will be executed right before the actual secondary structure decompositions, and, therefore, any implementation must not interleave with the regular DP matrices.

#### 16.9.3 Typedef Documentation

##### 16.9.3.1 **`vrna_callback_ud_energy`**

```
typedef int() vrna_callback_ud_energy(vrna_fold_compound_t *vc, int i, int j, unsigned int
loop_type, void *data)
#include <ViennaRNA/unstructured_domains.h>
Callback to retrieve binding free energy of a ligand bound to an unpaired sequence segment.
```

**Notes on Callback Functions** This function will be called to determine the additional energy contribution of a specific unstructured domain, e.g. the binding free energy of some ligand.

##### Parameters

|                  |   |
|------------------|---|
| <i>vc</i>        | The current <code>vrna_fold_compound_t</code> |
| <i>i</i>         | The start of the unstructured domain (5' end) |
| <i>j</i>         | The end of the unstructured domain (3' end)   |
| <i>loop_type</i> | The loop context of the unstructured domain   |
| <i>data</i>      | Auxiliary data                                |

##### Returns

The auxiliary energy contribution in deka-cal/mol

##### 16.9.3.2 **`vrna_callback_ud_exp_energy`**

```
typedef FLT_OR_DBL() vrna_callback_ud_exp_energy(vrna_fold_compound_t *vc, int i, int j, unsigned
int loop_type, void *data)
#include <ViennaRNA/unstructured_domains.h>
Callback to retrieve Boltzmann factor of the binding free energy of a ligand bound to an unpaired sequence segment.
```

**Notes on Callback Functions** This function will be called to determine the additional energy contribution of a specific unstructured domain, e.g. the binding free energy of some ligand (Partition function variant, i.e. the Boltzmann factors instead of actual free energies).

**Parameters**

|                  |   |
|------------------|---|
| <i>vc</i>        | The current <code>vrna_fold_compound_t</code> |
| <i>i</i>         | The start of the unstructured domain (5' end) |
| <i>j</i>         | The end of the unstructured domain (3' end)   |
| <i>loop_type</i> | The loop context of the unstructured domain   |
| <i>data</i>      | Auxiliary data                                |

**Returns**

The auxiliary energy contribution as Boltzmann factor

**16.9.3.3 vrna\_callback\_ud\_production**

```
typedef void() vrna_callback_ud_production(vrna_fold_compound_t *vc, void *data)
```

```
#include <ViennaRNA/unstructured_domains.h>
```

Callback for pre-processing the production rule of the ligand binding to unpaired stretches feature.

**Notes on Callback Functions** The production rule for the unstructured domain grammar extension

**16.9.3.4 vrna\_callback\_ud\_exp\_production**

```
typedef void() vrna_callback_ud_exp_production(vrna_fold_compound_t *vc, void *data)
```

```
#include <ViennaRNA/unstructured_domains.h>
```

Callback for pre-processing the production rule of the ligand binding to unpaired stretches feature (partition function variant)

**Notes on Callback Functions** The production rule for the unstructured domain grammar extension (Partition function variant)

**16.9.3.5 vrna\_callback\_ud\_probs\_add**

```
typedef void() vrna_callback_ud_probs_add(vrna_fold_compound_t *vc, int i, int j, unsigned int
loop_type, FLT_OR_DBL exp_energy, void *data)
```

```
#include <ViennaRNA/unstructured_domains.h>
```

Callback to store/add equilibrium probability for a ligand bound to an unpaired sequence segment.

**Notes on Callback Functions** A callback function to store equilibrium probabilities for the unstructured domain feature

**16.9.3.6 vrna\_callback\_ud\_probs\_get**

```
typedef FLT_OR_DBL() vrna_callback_ud_probs_get(vrna_fold_compound_t *vc, int i, int j, unsigned
int loop_type, int motif, void *data)
```

```
#include <ViennaRNA/unstructured_domains.h>
```

Callback to retrieve equilibrium probability for a ligand bound to an unpaired sequence segment.

**Notes on Callback Functions** A callback function to retrieve equilibrium probabilities for the unstructured domain feature

**16.9.4 Function Documentation**

#### 16.9.4.1 vrna\_ud\_motifs\_centroid()

```
vrna_ud_motif_t * vrna_ud_motifs_centroid (
    vrna_fold_compound_t * fc,
    const char * structure )
#include <ViennaRNA/unstructured_domains.h>
```

Detect unstructured domains in centroid structure.

Given a centroid structure and a set of unstructured domains compute the list of unstructured domain motifs present in the centroid. Since we do not explicitly annotate unstructured domain motifs in dot-bracket strings, this function can be used to check for the presence and location of unstructured domain motifs under the assumption that the dot-bracket string is the centroid structure of the equilibrium ensemble.

See also

[vrna\\_centroid\(\)](#)

#### Parameters

|                  |   |
|------------------|---|
| <i>fc</i>        | The fold_compound data structure with pre-computed equilibrium probabilities and model settings |
| <i>structure</i> | The centroid structure in dot-bracket notation  |

#### Returns

A list of unstructured domain motifs (possibly NULL). The last element terminates the list with `start=0, number=-1`

#### 16.9.4.2 vrna\_ud\_motifs\_MEA()

```
vrna_ud_motif_t * vrna_ud_motifs_MEA (
    vrna_fold_compound_t * fc,
    const char * structure,
    vrna_ep_t * probability_list )
#include <ViennaRNA/unstructured_domains.h>
```

Detect unstructured domains in MEA structure.

Given an MEA structure and a set of unstructured domains compute the list of unstructured domain motifs present in the MEA structure. Since we do not explicitly annotate unstructured domain motifs in dot-bracket strings, this function can be used to check for the presence and location of unstructured domain motifs under the assumption that the dot-bracket string is the MEA structure of the equilibrium ensemble.

See also

[MEA\(\)](#)

#### Parameters

|                         |   |
|-------------------------|---|
| <i>fc</i>               | The fold_compound data structure with pre-computed equilibrium probabilities and model settings |
| <i>structure</i>        | The MEA structure in dot-bracket notation   |
| <i>probability_list</i> | The list of probabilities to extract the MEA structure from                                     |

#### Returns

A list of unstructured domain motifs (possibly NULL). The last element terminates the list with `start=0, number=-1`

#### 16.9.4.3 vrna\_ud\_motifs\_MFE()

```
vrna_ud_motif_t * vrna_ud_motifs_MFE (
    vrna_fold_compound_t * fc,
    const char * structure )
#include <ViennaRNA/unstructured_domains.h>
```

Detect unstructured domains in MFE structure.

Given an MFE structure and a set of unstructured domains compute the list of unstructured domain motifs present in the MFE structure. Since we do not explicitly annotate unstructured domain motifs in dot-bracket strings, this function can be used to check for the presence and location of unstructured domain motifs under the assumption that the dot-bracket string is the MFE structure of the equilibrium ensemble.

See also

[vrna\\_mfe\(\)](#)

#### Parameters

|                  |  |
|------------------|--|
| <i>fc</i>        | The fold_compound data structure with model settings |
| <i>structure</i> | The MFE structure in dot-bracket notation            |

#### Returns

A list of unstructured domain motifs (possibly NULL). The last element terminates the list with `start=0, number=-1`

#### 16.9.4.4 vrna\_ud\_add\_motif()

```
void vrna_ud_add_motif (
    vrna_fold_compound_t * vc,
    const char * motif,
    double motif_en,
    const char * motif_name,
    unsigned int loop_type )
#include <ViennaRNA/unstructured_domains.h>
```

Add an unstructured domain motif, e.g. for ligand binding.

This function adds a ligand binding motif and the associated binding free energy to the `vrna_ud_t` attribute of a `vrna_fold_compound_t`. The motif data will then be used in subsequent secondary structure predictions. Multiple calls to this function with different motifs append all additional data to a list of ligands, which all will be evaluated. Ligand motif data can be removed from the `vrna_fold_compound_t` again using the `vrna_ud_remove()` function. The loop type parameter allows one to limit the ligand binding to particular loop type, such as the exterior loop, hairpin loops, interior loops, or multibranch loops.

See also

[VRNA\\_UNSTRUCTURED\\_DOMAIN\\_EXT\\_LOOP](#), [VRNA\\_UNSTRUCTURED\\_DOMAIN\\_HP\\_LOOP](#), [VRNA\\_UNSTRUCTURED\\_DOMAIN\\_MB\\_LOOP](#), [VRNA\\_UNSTRUCTURED\\_DOMAIN\\_ALL\\_LOOPS](#), [vrna\\_ud\\_remove\(\)](#)

#### Parameters

|                   |  |
|-------------------|--|
| <i>vc</i>         | The <code>vrna_fold_compound_t</code> data structure the ligand motif should be bound to |
| <i>motif</i>      | The sequence motif the ligand binds to   |
| <i>motif_en</i>   | The binding free energy of the ligand in kcal/mol  |
| <i>motif_name</i> | The name/id of the motif (may be NULL)   |
| <i>loop_type</i>  | The loop type the ligand binds to  |

#### 16.9.4.5 vrna\_ud\_remove()

```
void vrna_ud_remove (
    vrna_fold_compound_t * vc )
#include <ViennaRNA/unstructured_domains.h>
Remove ligand binding to unpaired stretches.
This function removes all ligand motifs that were bound to a vrna_fold_compound_t using the vrna_ud_add_motif() function.
```

##### Parameters

|                 |   |
|-----------------|---|
| <code>vc</code> | The <code>vrna_fold_compound_t</code> data structure the ligand motif data should be removed from |
|-----------------|---|

**SWIG Wrapper Notes** This function is attached as method `ud_remove()` to objects of type `fold_compound`

#### 16.9.4.6 vrna\_ud\_set\_data()

```
void vrna_ud_set_data (
    vrna_fold_compound_t * vc,
    void * data,
    vrna_callback_free_auxdata * free_cb )
#include <ViennaRNA/unstructured_domains.h>
Attach an auxiliary data structure.
This function binds an arbitrary, auxiliary data structure for user-implemented ligand binding. The optional callback free_cb will be passed the bound data structure whenever the vrna_fold_compound_t is removed from memory to avoid memory leaks.
```

##### See also

`vrna_ud_set_prod_rule_cb()`, `vrna_ud_set_exp_prod_rule_cb()`, `vrna_ud_remove()`

##### Parameters

|                      |  |
|----------------------|--|
| <code>vc</code>      | The <code>vrna_fold_compound_t</code> data structure the auxiliary data structure should be bound to |
| <code>data</code>    | A pointer to the auxiliary data structure  |
| <code>free_cb</code> | A pointer to a callback function that free's memory occupied by <code>data</code>                    |

**SWIG Wrapper Notes** This function is attached as method `ud_set_data()` to objects of type `fold_compound`

#### 16.9.4.7 vrna\_ud\_set\_prod\_rule\_cb()

```
void vrna_ud_set_prod_rule_cb (
    vrna_fold_compound_t * vc,
    vrna_callback_ud_production * pre_cb,
    vrna_callback_ud_energy * e_cb )
#include <ViennaRNA/unstructured_domains.h>
```

Attach production rule callbacks for free energies computations.

Use this function to bind a user-implemented grammar extension for unstructured domains.

The callback `e_cb` needs to evaluate the free energy contribution  $f(i,j)$  of the unpaired segment  $[i,j]$ . It will be executed in each of the regular secondary structure production rules. Whenever the callback is passed the `VRNA_UNSTRUCTURED_DOMAIN_MOTIF` flag via its `loop_type` parameter the contribution of

any ligand that consecutively binds from position  $i$  to  $j$  (the white box) is requested. Otherwise, the callback usually performs a lookup in the precomputed  $B$  matrices. Which  $B$  matrix is addressed will be indicated by the flags `VRNA_UNSTRUCTURED_DOMAIN_EXT_LOOP`, `VRNA_UNSTRUCTURED_DOMAIN_HP_LOOP`, `VRNA_UNSTRUCTURED_DOMAIN_INT_LOOP`, and `VRNA_UNSTRUCTURED_DOMAIN_MB_LOOP`. As their names already imply, they specify exterior loops ( $F$  production rule), hairpin loops and interior loops ( $C$  production rule), and multibranch loops ( $M$  and  $M1$  production rule).

$$f(i,j) = \boxed{\quad} \mid \boxed{B} \\ i \quad j \quad i \quad j$$

The `pre_cb` callback will be executed as a pre-processing step right before the regular secondary structure rules. Usually one would use this callback to fill the dynamic programming matrices  $U$  and preparations of the auxiliary data structure `vrna_unstructured_domain_s.data`

$$\boxed{B} = \boxed{B} \bullet \mid \boxed{B} \quad \boxed{\quad} \\ i \quad j \quad i \quad j-1 \quad j \quad i \quad u \quad u+1 \quad j$$

#### Parameters

|                     |  |
|---------------------|--|
| <code>vc</code>     | The <code>vrna_fold_compound_t</code> data structure the callback will be bound to |
| <code>pre_cb</code> | A pointer to a callback function for the $B$ production rule                       |
| <code>e_cb</code>   | A pointer to a callback function for free energy evaluation                        |

**SWIG Wrapper Notes** This function is attached as method `ud_set_prod_rule_cb()` to objects of type `fold_<compound>`

#### 16.9.4.8 `vrna_ud_set_exp_prod_rule_cb()`

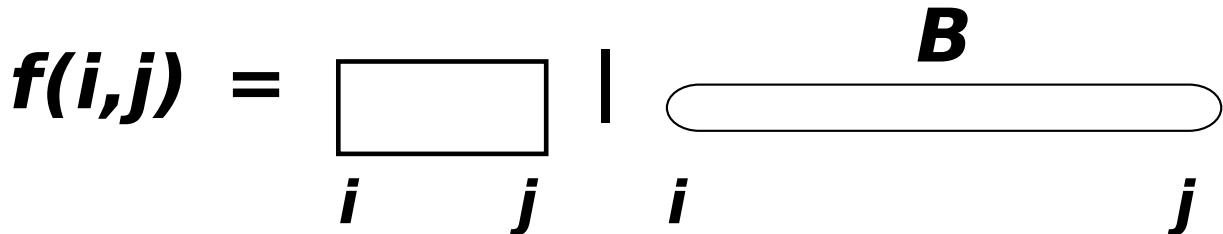
```
void vrna_ud_set_exp_prod_rule_cb (
    vrna_fold_compound_t * vc,
    vrna_callback_ud_exp_production * pre_cb,
    vrna_callback_ud_exp_energy * exp_e_cb )
#include <ViennaRNA/unstructured_domains.h>
```

Attach production rule for partition function.

This function is the partition function companion of `vrna_ud_set_prod_rule_cb()`.

Use it to bind callbacks to (i) fill the  $U$  production rule dynamic programming matrices and/or prepare the `vrna_unstructured_domain_s.data`, and (ii) provide a callback to retrieve partition functions for subsegments  $[i, j]$ .

$$\boxed{B} = \boxed{B} \bullet \mid \boxed{B} \quad \boxed{\quad} \\ i \quad j \quad i \quad j-1 \quad j \quad i \quad u \quad u+1 \quad j$$



See also

[vrna\\_ud\\_set\\_prod\\_rule\\_cb\(\)](#)

#### Parameters

|                       |   |
|-----------------------|---|
| <code>vc</code>       | The <code>vrna_fold_compound_t</code> data structure the callback will be bound to  |
| <code>pre_cb</code>   | A pointer to a callback function for the <code>B</code> production rule   |
| <code>exp_e_cb</code> | A pointer to a callback function that retrieves the partition function for a segment $[i, j]$ that may be bound by one or more ligands. |

**SWIG Wrapper Notes** This function is attached as method `ud_set_exp_prod_rule_cb()` to objects of type `fold_compound`

## 16.10 Structured Domains

Add and modify structured domains to the RNA folding grammar.

### 16.10.1 Detailed Description

Add and modify structured domains to the RNA folding grammar.

This module provides the tools to add and modify structured domains to the production rules of the RNA folding grammar. Usually this functionality is utilized for incorporating self-enclosed structural modules that exhibit a more or less complex base pairing pattern. Collaboration diagram for Structured Domains:

#### Files

- file `structured_domains.h`

*This module provides interfaces that deal with additional structured domains in the folding grammar.*

## 16.11 Constraining the RNA Folding Grammar

This module provides general functions that allow for an easy control of constrained secondary structure prediction and evaluation.

### 16.11.1 Detailed Description

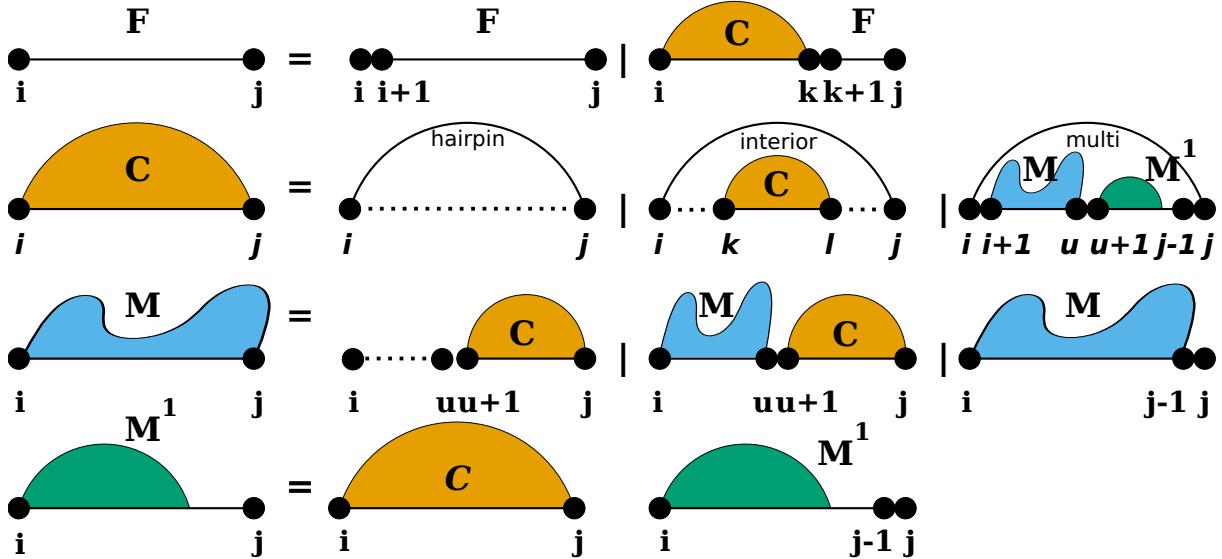
This module provides general functions that allow for an easy control of constrained secondary structure prediction and evaluation.

Secondary Structure constraints can be subdivided into two groups:

- [Hard Constraints](#), and
- [Soft Constraints](#).

While Hard-Constraints directly influence the production rules used in the folding recursions by allowing, disallowing, or enforcing certain decomposition steps, Soft-constraints on the other hand are used to change position specific contributions in the recursions by adding bonuses/penalties in form of pseudo free energies to certain loop configurations.

Secondary structure constraints are always applied at decomposition level, i.e. in each step of the recursive structure decomposition, for instance during MFE prediction. Below is a visualization of the decomposition scheme



For **Hard Constraints** the following option flags may be used to constrain the pairing behavior of single, or pairs of nucleotides:

- `VRNA_CONSTRAINT_CONTEXT_EXT_LOOP` - Hard constraints flag, base pair in the exterior loop.
- `VRNA_CONSTRAINT_CONTEXT_HP_LOOP` - Hard constraints flag, base pair encloses hairpin loop.
- `VRNA_CONSTRAINT_CONTEXT_INT_LOOP` - Hard constraints flag, base pair encloses an interior loop.
- `VRNA_CONSTRAINT_CONTEXT_INT_LOOP_ENC` - Hard constraints flag, base pair encloses a multi branch loop.
- `VRNA_CONSTRAINT_CONTEXT_MB_LOOP` - Hard constraints flag, base pair is enclosed in an interior loop.
- `VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC` - Hard constraints flag, base pair is enclosed in a multi branch loop.
- `VRNA_CONSTRAINT_CONTEXT_ENFORCE` - Hard constraint flag to indicate enforcement of constraints.
- `VRNA_CONSTRAINT_CONTEXT_NO_REMOVE` - Hard constraint flag to indicate not to remove base pairs that conflict with a given constraint.
- `VRNA_CONSTRAINT_CONTEXT_ALL_LOOPS` - Constraint context flag indicating any loop context.

However, for **Soft Constraints** we do not allow for simple loop type dependent constraining. But soft constraints are equipped with generic constraint support. This enables the user to pass arbitrary callback functions that return auxiliary energy contributions for evaluation the evaluation of any decomposition.

The callback will then always be notified about the type of decomposition that is happening, and the corresponding delimiting sequence positions. The following decomposition steps are distinguished, and should be captured by the user's implementation of the callback:

- `VRNA_DECOMP_PAIR_HP` - Flag passed to generic soft constraints callback to indicate hairpin loop decomposition step.
- `VRNA_DECOMP_PAIR_IL` - Indicator for interior loop decomposition step.

- `VRNA_DECOMP_PAIR_ML` - Indicator for multibranch loop decomposition step.
- `VRNA_DECOMP_ML_ML_ML` - Indicator for decomposition of multibranch loop part.
- `VRNA_DECOMP_ML_STEM` - Indicator for decomposition of multibranch loop part.
- `VRNA_DECOMP_ML_ML` - Indicator for decomposition of multibranch loop part.
- `VRNA_DECOMP_ML_UP` - Indicator for decomposition of multibranch loop part.
- `VRNA_DECOMP_ML_ML_STEM` - Indicator for decomposition of multibranch loop part.
- `VRNA_DECOMP_ML_COAXIAL` - Indicator for decomposition of multibranch loop part.
- `VRNA_DECOMP_EXT_EXT` - Indicator for decomposition of exterior loop part.
- `VRNA_DECOMP_EXT_UP` - Indicator for decomposition of exterior loop part.
- `VRNA_DECOMP_EXT_STEM` - Indicator for decomposition of exterior loop part.
- `VRNA_DECOMP_EXT_EXT_EXT` - Indicator for decomposition of exterior loop part.
- `VRNA_DECOMP_EXT_STEM_EXT` - Indicator for decomposition of exterior loop part.
- `VRNA_DECOMP_EXT_STEM_OUTSIDE` - Indicator for decomposition of exterior loop part.
- `VRNA_DECOMP_EXT_EXT_STEM` - Indicator for decomposition of exterior loop part.
- `VRNA_DECOMP_EXT_EXT_STEM1` - Indicator for decomposition of exterior loop part.

Simplified interfaces to the soft constraints framework can be obtained by the implementations in the submodules

- [SHAPE Reactivity Data](#) and
- [Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints](#).

An implementation that generates soft constraints for unpaired nucleotides by minimizing the discrepancy between their predicted and expected pairing probability is available in submodule [Generate Soft Constraints from Data](#). Collaboration diagram for Constraining the RNA Folding Grammar:

## Modules

- [Hard Constraints](#)  
*This module covers all functionality for hard constraints in secondary structure prediction.*
- [Soft Constraints](#)  
*Functions and data structures for secondary structure soft constraints.*

## Files

- file `basic.h`  
*Functions and data structures for constraining secondary structure predictions and evaluation.*

## Macros

- `#define VRNA_CONSTRAINT_FILE 0`  
*Flag for `vrna_constraints_add()` to indicate that constraints are present in a text file.*
- `#define VRNA_CONSTRAINT_SOFT_MFE 0`  
*Indicate generation of constraints for MFE folding.*
- `#define VRNA_CONSTRAINT_SOFT_PF VRNA_OPTION_PF`  
*Indicate generation of constraints for partition function computation.*
- `#define VRNA_DECOMP_PAIR_HP (unsigned char)1`  
*Flag passed to generic softt constraints callback to indicate hairpin loop decomposition step.*

- `#define VRNA_DECOMP_PAIR_IL` (unsigned char)2  
*Indicator for interior loop decomposition step.*
- `#define VRNA_DECOMP_PAIR_ML` (unsigned char)3  
*Indicator for multibranch loop decomposition step.*
- `#define VRNA_DECOMP_ML_ML_ML` (unsigned char)5  
*Indicator for decomposition of multibranch loop part.*
- `#define VRNA_DECOMP_ML_STEM` (unsigned char)6  
*Indicator for decomposition of multibranch loop part.*
- `#define VRNA_DECOMP_ML_ML` (unsigned char)7  
*Indicator for decomposition of multibranch loop part.*
- `#define VRNA_DECOMP_ML_UP` (unsigned char)8  
*Indicator for decomposition of multibranch loop part.*
- `#define VRNA_DECOMP_ML_ML_STEM` (unsigned char)9  
*Indicator for decomposition of multibranch loop part.*
- `#define VRNA_DECOMP_ML_COAXIAL` (unsigned char)10  
*Indicator for decomposition of multibranch loop part.*
- `#define VRNA_DECOMP_ML_COAXIAL_ENC` (unsigned char)11  
*Indicator for decomposition of multibranch loop part.*
- `#define VRNA_DECOMP_EXT_EXT` (unsigned char)12  
*Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_UP` (unsigned char)13  
*Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_STEM` (unsigned char)14  
*Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_EXT_EXT` (unsigned char)15  
*Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_STEM_EXT` (unsigned char)16  
*Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_STEM_OUTSIDE` (unsigned char)17  
*Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_EXT_STEM` (unsigned char)18  
*Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_EXT_STEM1` (unsigned char)19  
*Indicator for decomposition of exterior loop part.*

## Functions

- `void vrna_constraints_add (vrna_fold_compound_t *vc, const char *constraint, unsigned int options)`  
*Add constraints to a `vrna_fold_compound_t` data structure.*
- `void vrna_message_constraint_options (unsigned int option)`  
*Print a help message for pseudo dot-bracket structure constraint characters to stdout. (constraint support is specified by option parameter)*
- `void vrna_message_constraint_options_all (void)`  
*Print structure constraint characters to stdout (full constraint support)*

### 16.11.2 Macro Definition Documentation

### 16.11.2.1 VRNA\_CONSTRAINT\_FILE

```
#define VRNA_CONSTRAINT_FILE 0
#include <ViennaRNA/constraints/basic.h>
Flag for vrna\_constraints\_add\(\) to indicate that constraints are present in a text file.
```

See also

[vrna\\_constraints\\_add\(\)](#)

**Deprecated** Use 0 instead!

### 16.11.2.2 VRNA\_CONSTRAINT\_SOFT\_MFE

```
#define VRNA_CONSTRAINT_SOFT_MFE 0
#include <ViennaRNA/constraints/basic.h>
Indicate generation of constraints for MFE folding.
```

**Deprecated** This flag has no meaning anymore, since constraints are now always stored!

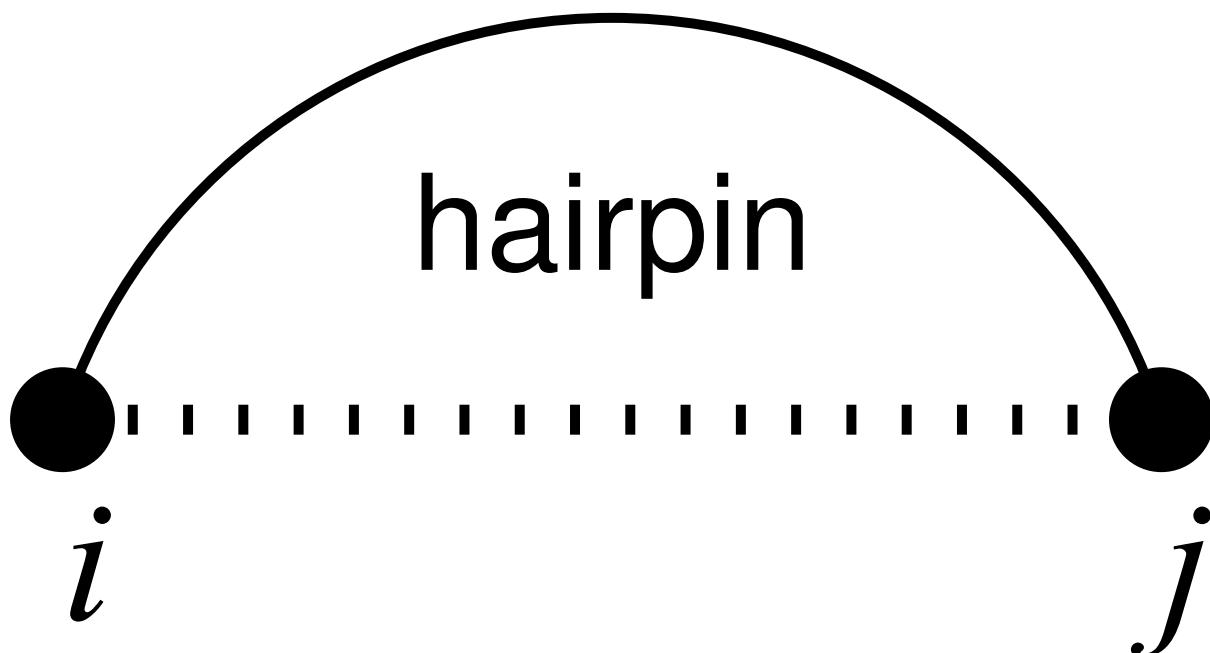
### 16.11.2.3 VRNA\_CONSTRAINT\_SOFT\_PF

```
#define VRNA_CONSTRAINT_SOFT_PF VRNA_OPTION_PF
#include <ViennaRNA/constraints/basic.h>
Indicate generation of constraints for partition function computation.
```

**Deprecated** Use [VRNA\\_OPTION\\_PF](#) instead!

### 16.11.2.4 VRNA\_DECOMP\_PAIR\_HP

```
#define VRNA_DECOMP_PAIR_HP (unsigned char)1
#include <ViennaRNA/constraints/basic.h>
Flag passed to generic softt constraints callback to indicate hairpin loop decomposition step.
This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a hairpin
loop enclosed by the base pair  $(i, j)$ .
```

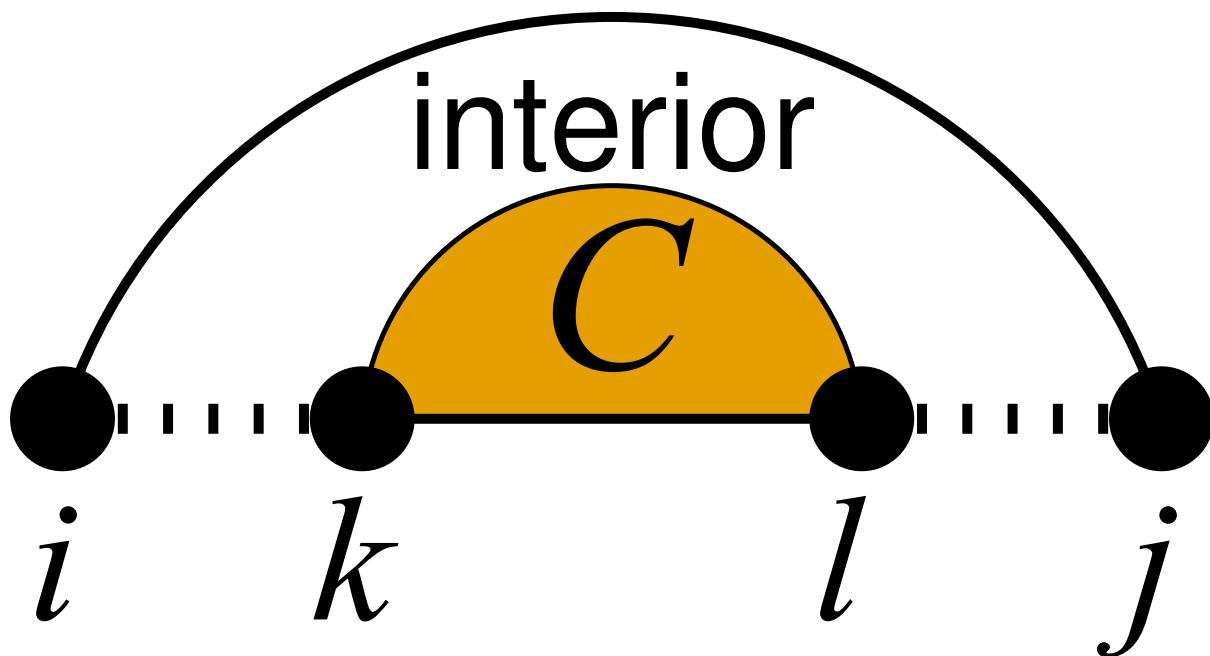


### 16.11.2.5 VRNA\_DECOMP\_PAIR\_IL

```
#define VRNA_DECOMP_PAIR_IL (unsigned char)2
#include <ViennaRNA/constraints/basic.h>
```

Indicator for interior loop decomposition step.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates an interior loop enclosed by the base pair  $(i, j)$ , and enclosing the base pair  $(k, l)$ .

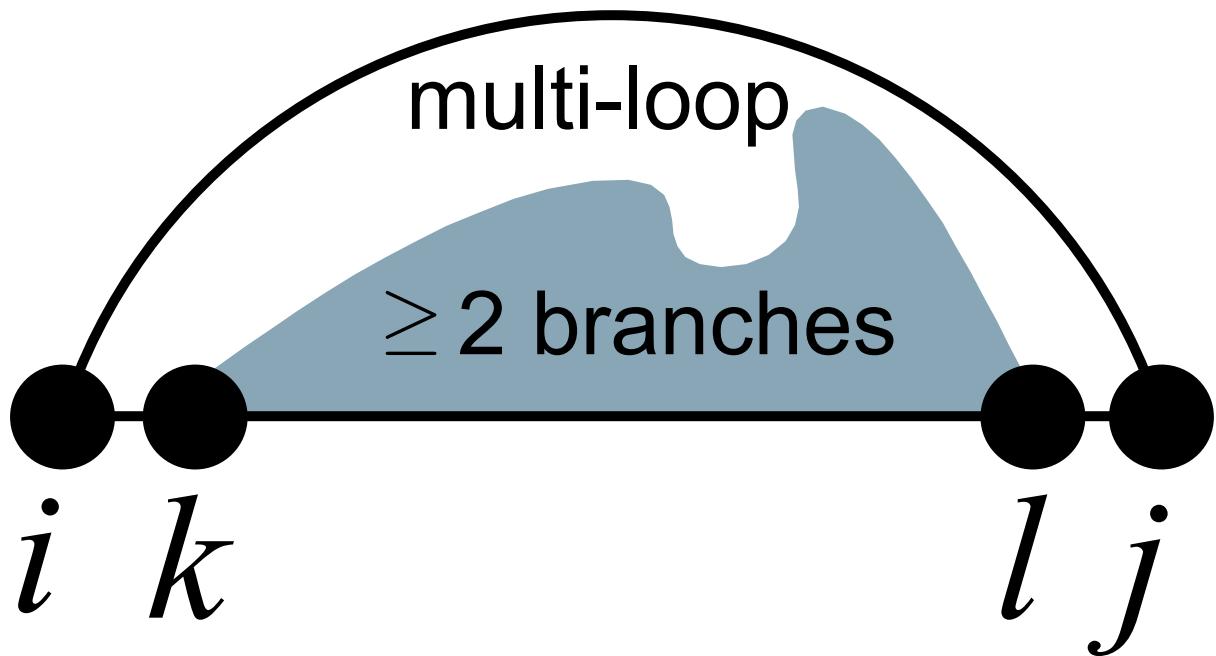


### 16.11.2.6 VRNA\_DECOMP\_PAIR\_ML

```
#define VRNA_DECOMP_PAIR_ML (unsigned char)3
#include <ViennaRNA/constraints/basic.h>
```

Indicator for multibranch loop decomposition step.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a multi-branch loop enclosed by the base pair  $(i, j)$ , and consisting of some enclosed multi loop content from  $k$  to  $l$ .

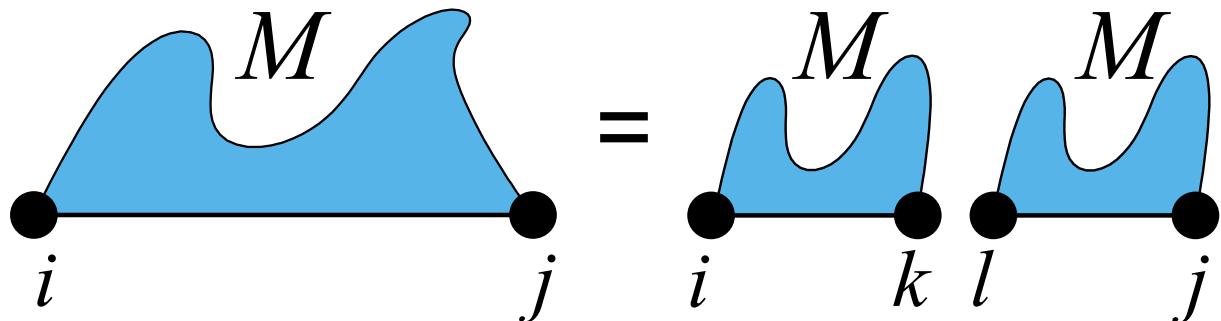


#### 16.11.2.7 VRNA\_DECOMP\_ML\_ML\_ML

```
#define VRNA_DECOMP_ML_ML_ML (unsigned char)5
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of multibranch loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a multibranch loop part in the interval  $[i : j]$ , which will be decomposed into two multibranch loop parts  $[i : k]$ , and  $[l : j]$ .

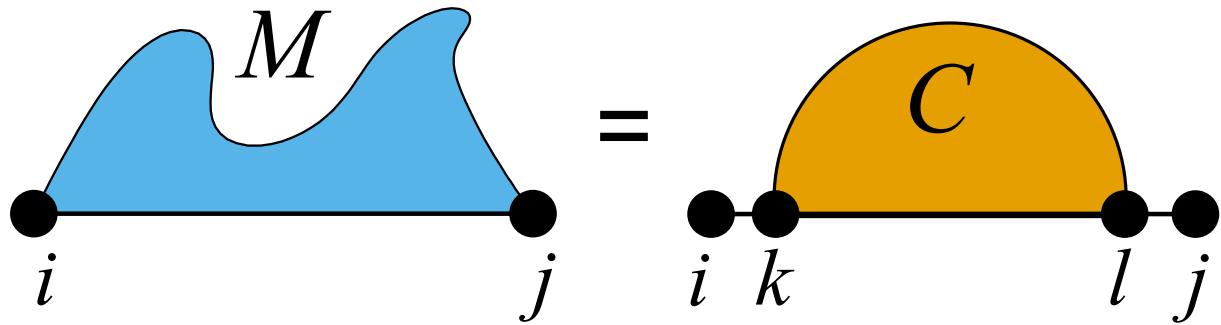


#### 16.11.2.8 VRNA\_DECOMP\_ML\_STEM

```
#define VRNA_DECOMP_ML_STEM (unsigned char)6
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of multibranch loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a multibranch loop part in the interval  $[i : j]$ , which will be considered a single stem branching off with base pair  $(k, l)$ .

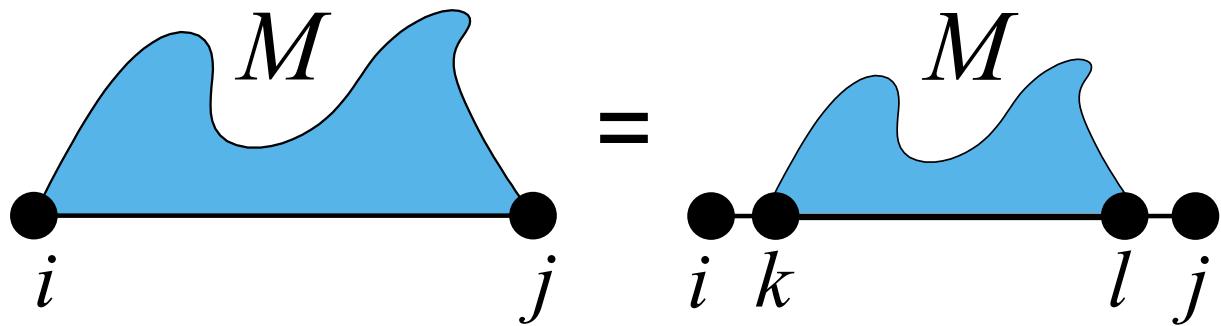


#### 16.11.2.9 VRNA\_DECOMP\_ML\_ML

```
#define VRNA_DECOMP_ML_ML (unsigned char) 7
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of multibranch loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a multibranch loop part in the interval  $[i : j]$ , which will be decomposed into a (usually) smaller multibranch loop part  $[k : l]$ .

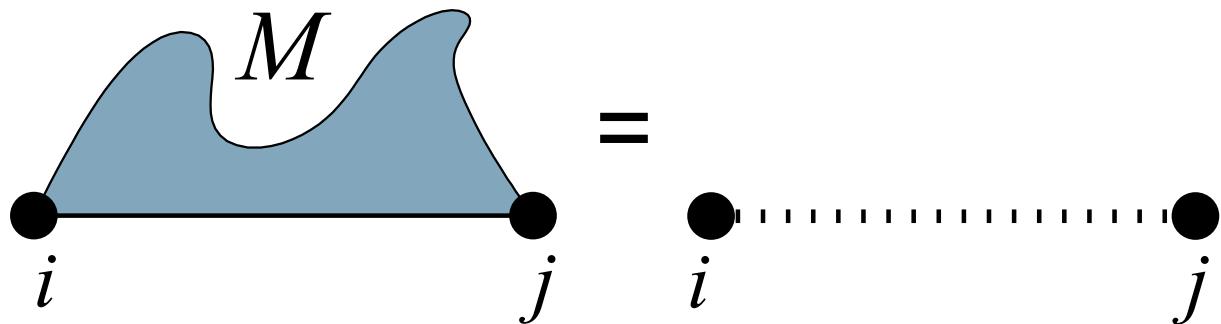


#### 16.11.2.10 VRNA\_DECOMP\_ML\_UP

```
#define VRNA_DECOMP_ML_UP (unsigned char) 8
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of multibranch loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a multibranch loop part in the interval  $[i : j]$ , which will be considered a multibranch loop part that only consists of unpaired nucleotides.

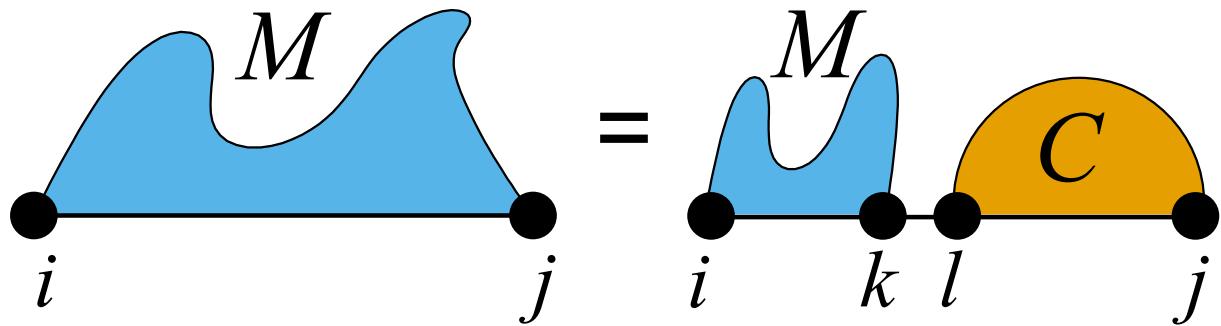


### 16.11.2.11 VRNA\_DECOMP\_ML\_ML\_STEM

```
#define VRNA_DECOMP_ML_ML_STEM (unsigned char)9
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of multibranch loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a multi-branch loop part in the interval  $[i : j]$ , which will decomposed into a multibranch loop part  $[i : k]$ , and a stem with enclosing base pair  $(l, j)$ .

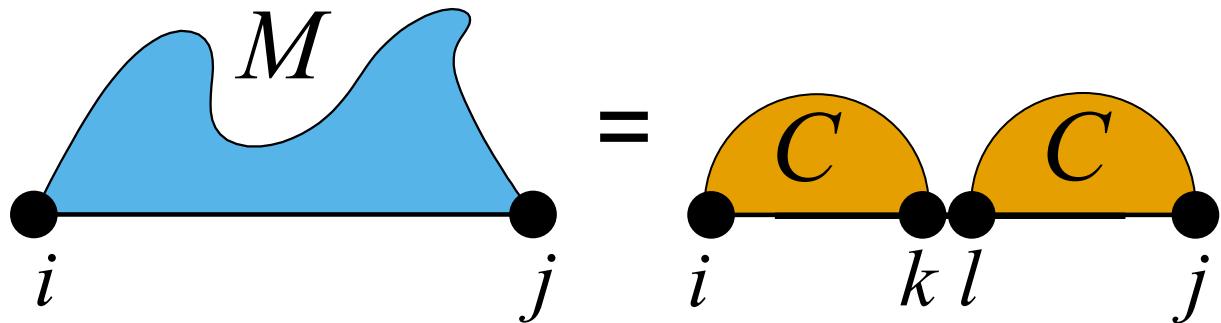


### 16.11.2.12 VRNA\_DECOMP\_ML\_COAXIAL

```
#define VRNA_DECOMP_ML_COAXIAL (unsigned char)10
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of multibranch loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a multi-branch loop part in the interval  $[i : j]$ , where two stems with enclosing pairs  $(i, k)$  and  $(l, j)$  are coaxially stacking onto each other.

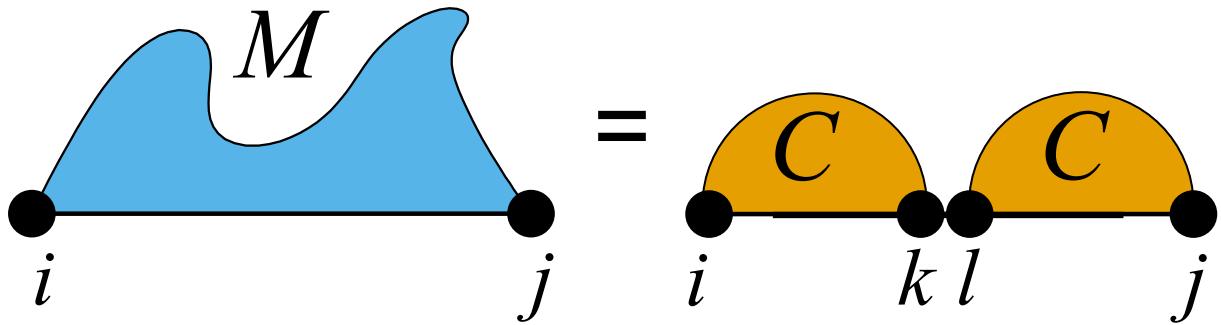


### 16.11.2.13 VRNA\_DECOMP\_ML\_COAXIAL\_ENC

```
#define VRNA_DECOMP_ML_COAXIAL_ENC (unsigned char)11
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of multibranch loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a multi-branch loop part in the interval  $[i : j]$ , where two stems with enclosing pairs  $(i, k)$  and  $(l, j)$  are coaxially stacking onto each other.

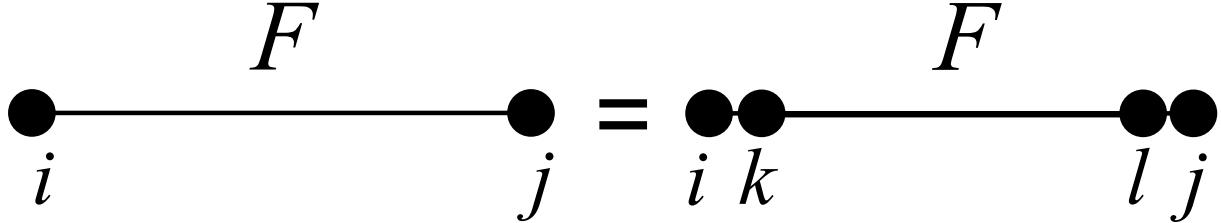


#### 16.11.2.14 VRNA\_DECOMP\_EXT\_EXT

```
#define VRNA_DECOMP_EXT_EXT (unsigned char)12
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of exterior loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates an exterior loop part in the interval  $[i : j]$ , which will be decomposed into a (usually) smaller exterior loop part  $[k : l]$ .

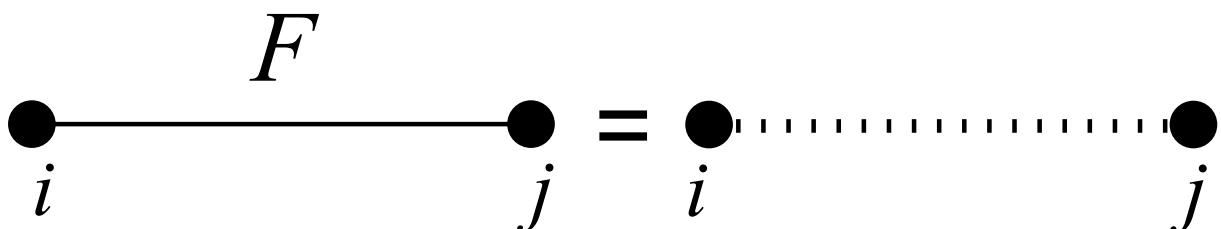


#### 16.11.2.15 VRNA\_DECOMP\_EXT\_UP

```
#define VRNA_DECOMP_EXT_UP (unsigned char)13
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of exterior loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates an exterior loop part in the interval  $[i : j]$ , which will be considered as an exterior loop component consisting of only unpaired nucleotides.

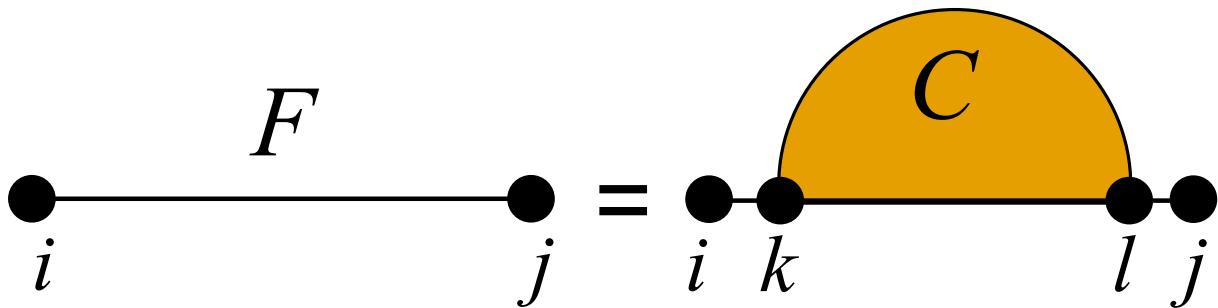


#### 16.11.2.16 VRNA\_DECOMP\_EXT\_STEM

```
#define VRNA_DECOMP_EXT_STEM (unsigned char)14
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of exterior loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates an exterior loop part in the interval  $[i : j]$ , which will be considered a stem with enclosing pair  $(k, l)$ .

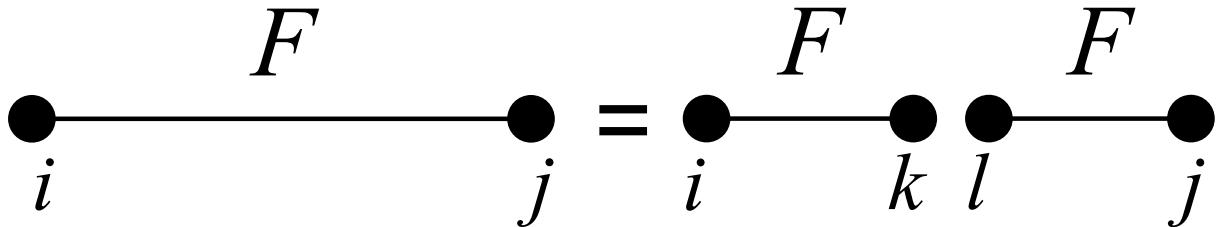


#### 16.11.2.17 VRNA\_DECOMP\_EXT\_EXT\_EXT

```
#define VRNA_DECOMP_EXT_EXT_EXT (unsigned char)15
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of exterior loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates an exterior loop part in the interval  $[i : j]$ , which will be decomposed into two exterior loop parts  $[i : k]$  and  $[l : j]$ .

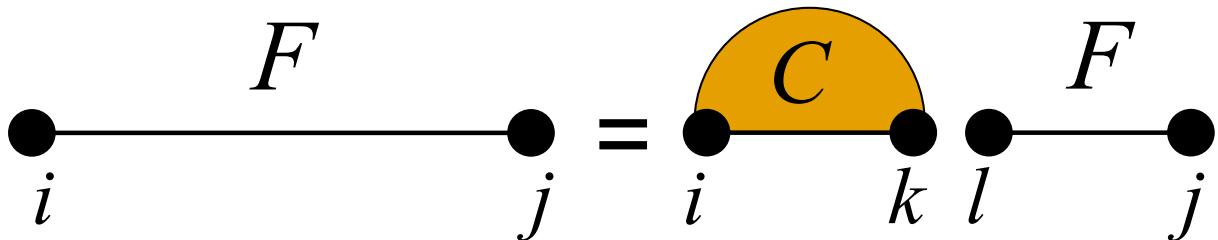


#### 16.11.2.18 VRNA\_DECOMP\_EXT\_STEM\_EXT

```
#define VRNA_DECOMP_EXT_STEM_EXT (unsigned char)16
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of exterior loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates an exterior loop part in the interval  $[i : j]$ , which will be decomposed into a stem branching off with base pair  $(i, k)$ , and an exterior loop part  $[l : j]$ .

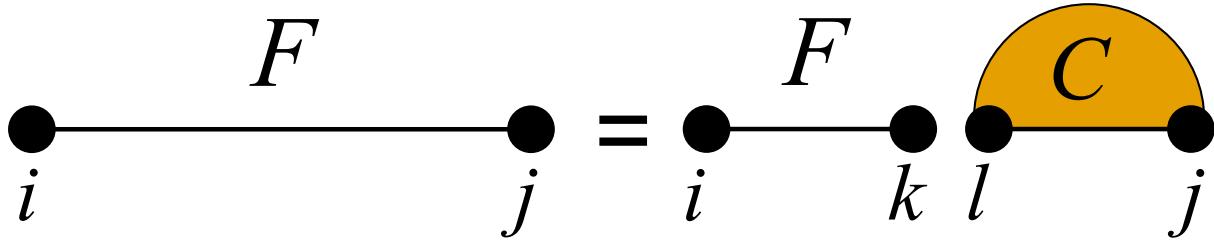


#### 16.11.2.19 VRNA\_DECOMP\_EXT\_EXT\_STEM

```
#define VRNA_DECOMP_EXT_EXT_STEM (unsigned char)18
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of exterior loop part.

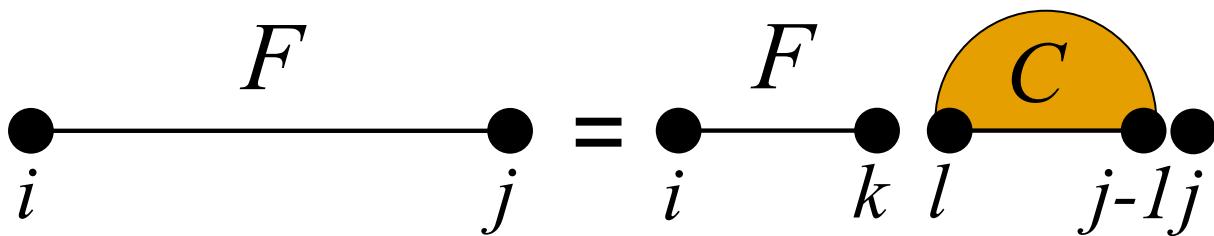
This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates an exterior loop part in the interval  $[i : j]$ , which will be decomposed into an exterior loop part  $[i : k]$ , and a stem branching off with base pair  $(l, j)$ .



### 16.11.2.20 VRNA\_DECOMP\_EXT\_EXT\_STEM1

```
#define VRNA_DECOMP_EXT_EXT_STEM1 (unsigned char)19
#include <ViennaRNA/constraints/basic.h>
Indicator for decomposition of exterior loop part.
```

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates an exterior loop part in the interval  $[i : j]$ , which will be decomposed into an exterior loop part  $[i : k]$ , and a stem branching off with base pair  $(l, j - 1)$ .



### 16.11.3 Function Documentation

#### 16.11.3.1 vrna\_constraints\_add()

```
void vrna_constraints_add (
    vrna_fold_compound_t * vc,
    const char * constraint,
    unsigned int options )
#include <ViennaRNA/constraints/basic.h>
```

Add constraints to a `vrna_fold_compound_t` data structure.

Use this function to add/update the hard/soft constraints. The function allows for passing a string 'constraint' that can either be a filename that points to a constraints definition file or it may be a pseudo dot-bracket notation indicating hard constraints. For the latter, the user has to pass the `VRNA_CONSTRAINT_DB` option. Also, the user has to specify, which characters are allowed to be interpreted as constraints by passing the corresponding options via the third parameter.

See also

[vrna\\_hc\\_init\(\)](#), [vrna\\_hc\\_add\\_up\(\)](#), [vrna\\_hc\\_add\\_up\\_batch\(\)](#), [vrna\\_hc\\_add\\_bp\(\)](#), [vrna\\_sc\\_init\(\)](#), [vrna\\_sc\\_set\\_up\(\)](#),  
[vrna\\_sc\\_set\\_bp\(\)](#), [vrna\\_sc\\_add\\_SHAPE\\_deigan\(\)](#), [vrna\\_sc\\_add\\_SHAPE\\_zarringhalam\(\)](#), [vrna\\_hc\\_free\(\)](#),  
[vrna\\_sc\\_free\(\)](#), [VRNA\\_CONSTRAINT\\_DB](#), [VRNA\\_CONSTRAINT\\_DB\\_DEFAULT](#), [VRNA\\_CONSTRAINT\\_DB\\_PIPE](#),  
[VRNA\\_CONSTRAINT\\_DB\\_DOT](#), [VRNA\\_CONSTRAINT\\_DB\\_X](#), [VRNA\\_CONSTRAINT\\_DB\\_ANG\\_BRACK](#),  
[VRNA\\_CONSTRAINT\\_DB\\_RND\\_BRACK](#), [VRNA\\_CONSTRAINT\\_DB\\_INTRAMOL](#), [VRNA\\_CONSTRAINT\\_DB\\_INTERMOL](#),  
[VRNA\\_CONSTRAINT\\_DB\\_GQUAD](#)

The following is an example for adding hard constraints given in pseudo dot-bracket notation. Here, `vc` is the `vrna_fold_compound_t` object, `structure` is a char array with the hard constraint in dot-bracket notation, and `enforceConstraints` is a flag indicating whether or not constraints for base pairs should be enforced instead of just doing a removal of base pair that conflict with the constraint.

```

unsigned int constraint_options = VRNA_CONSTRAINT_DB_DEFAULT;
if (enforceConstraints)
    constraint_options |= VRNA_CONSTRAINT_DB_ENFORCE_BP;
if (canonicalBPonly)
    constraint_options |= VRNA_CONSTRAINT_DB_CANONICAL_BP;
vrna_constraints_add(fc, (const char *)cstruc, constraint_options);

```

In contrast to the above, constraints may also be read from file:

```
vrna_constraints_add(fc, constraints_file, VRNA_OPTION_DEFAULT);
```

#### See also

[vrna\\_hc\\_add\\_from\\_db\(\)](#), [vrna\\_hc\\_add\\_up\(\)](#), [vrna\\_hc\\_add\\_up\\_batch\(\)](#) [vrna\\_hc\\_add\\_bp\\_unspecific\(\)](#), [vrna\\_hc\\_add\\_bp\(\)](#)

#### Parameters

|                   |   |
|-------------------|---|
| <i>vc</i>         | The fold compound   |
| <i>constraint</i> | A string with either the filename of the constraint definitions or a pseudo dot-bracket notation of the hard constraint. May be NULL. |
| <i>options</i>    | The option flags  |

### 16.11.3.2 vrna\_message\_constraint\_options()

```

void vrna_message_constraint_options (
    unsigned int option )
#include <ViennaRNA/constraints/hard.h>

```

Print a help message for pseudo dot-bracket structure constraint characters to stdout. (constraint support is specified by option parameter)

Currently available options are:

VRNA\_CONSTRAINT\_DB\_PIPE (paired with another base)  
VRNA\_CONSTRAINT\_DB\_DOT (no constraint at all)  
VRNA\_CONSTRAINT\_DB\_X (base must not pair)  
VRNA\_CONSTRAINT\_DB\_ANG\_BRACK (paired downstream/upstream)  
VRNA\_CONSTRAINT\_DB\_RND\_BRACK (base i pairs base j)

pass a collection of options as one value like this:

```
vrna_message_constraints(option_1 | option_2 | option_n)
```

#### See also

[vrna\\_message\\_constraint\\_options\\_all\(\)](#), [vrna\\_constraints\\_add\(\)](#), VRNA\_CONSTRAINT\_DB, VRNA\_CONSTRAINT\_DB\_PIPE, VRNA\_CONSTRAINT\_DB\_DOT, VRNA\_CONSTRAINT\_DB\_X, VRNA\_CONSTRAINT\_DB\_ANG\_BRACK, VRNA\_CONSTRAINT\_DB\_RND\_BRACK, VRNA\_CONSTRAINT\_DB\_INTERMOL, VRNA\_CONSTRAINT\_DB\_INTRAMOL

#### Parameters

|               |  |
|---------------|--|
| <i>option</i> | Option switch that tells which constraint help will be printed |
|---------------|--|

### 16.11.3.3 vrna\_message\_constraint\_options\_all()

```

void vrna_message_constraint_options_all (
    void )
#include <ViennaRNA/constraints/hard.h>

```

Print structure constraint characters to stdout (full constraint support)

See also

[vrna\\_message\\_constraint\\_options\(\)](#), [vrna\\_constraints\\_add\(\)](#), [VRNA\\_CONSTRAINT\\_DB](#), [VRNA\\_CONSTRAINT\\_DB\\_PIPE](#), [VRNA\\_CONSTRAINT\\_DB\\_DOT](#), [VRNA\\_CONSTRAINT\\_DB\\_X](#), [VRNA\\_CONSTRAINT\\_DB\\_ANG\\_BRACK](#), [VRNA\\_CONSTRAINT\\_DB\\_RND\\_BRACK](#), [VRNA\\_CONSTRAINT\\_DB\\_INTERMOL](#), [VRNA\\_CONSTRAINT\\_DB\\_INTRAMOL](#)

## 16.12 Hard Constraints

This module covers all functionality for hard constraints in secondary structure prediction.

### 16.12.1 Detailed Description

This module covers all functionality for hard constraints in secondary structure prediction.  
Collaboration diagram for Hard Constraints:

#### Files

- file [hard.h](#)  
*Functions and data structures for handling of secondary structure hard constraints.*

#### Data Structures

- struct [vrna\\_hc\\_s](#)  
*The hard constraints data structure. [More...](#)*
- struct [vrna\\_hc\\_up\\_s](#)  
*A single hard constraint for a single nucleotide. [More...](#)*

#### Macros

- #define [VRNA\\_CONSTRAINT\\_DB](#) 16384U  
*Flag for [vrna\\_constraints\\_add\(\)](#) to indicate that constraint is passed in pseudo dot-bracket notation.*
- #define [VRNA\\_CONSTRAINT\\_DB\\_ENFORCE\\_BP](#) 32768U  
*Switch for dot-bracket structure constraint to enforce base pairs.*
- #define [VRNA\\_CONSTRAINT\\_DB\\_PIPE](#) 65536U  
*Flag that is used to indicate the pipe '|' sign in pseudo dot-bracket notation of hard constraints.*
- #define [VRNA\\_CONSTRAINT\\_DB\\_DOT](#) 131072U  
*dot '.' switch for structure constraints (no constraint at all)*
- #define [VRNA\\_CONSTRAINT\\_DB\\_X](#) 262144U  
*'x' switch for structure constraint (base must not pair)*
- #define [VRNA\\_CONSTRAINT\\_DB\\_RND\\_BRACK](#) 1048576U  
*round brackets '()' switch for structure constraint (base i pairs base j)*
- #define [VRNA\\_CONSTRAINT\\_DB\\_INTRAMOL](#) 2097152U  
*Flag that is used to indicate the character 'l' in pseudo dot-bracket notation of hard constraints.*
- #define [VRNA\\_CONSTRAINT\\_DB\\_INTERMOL](#) 4194304U  
*Flag that is used to indicate the character 'e' in pseudo dot-bracket notation of hard constraints.*
- #define [VRNA\\_CONSTRAINT\\_DB\\_GQUAD](#) 8388608U  
 *'+' switch for structure constraint (base is involved in a gquad)*
- #define [VRNA\\_CONSTRAINT\\_DB\\_WUSS](#) 33554432U  
*Flag to indicate Washington University Secondary Structure (WUSS) notation of the hard constraint string.*
- #define [VRNA\\_CONSTRAINT\\_DB\\_DEFAULT](#)  
*Switch for dot-bracket structure constraint with default symbols.*
- #define [VRNA\\_CONSTRAINT\\_CONTEXT\\_EXT\\_LOOP](#) (unsigned char)0x01  
*Hard constraints flag, base pair in the exterior loop.*
- #define [VRNA\\_CONSTRAINT\\_CONTEXT\\_HP\\_LOOP](#) (unsigned char)0x02

- Hard constraints flag, base pair encloses hairpin loop.
- #define **VRNA\_CONSTRAINT\_CONTEXT\_INT\_LOOP** (unsigned char)0x04  
Hard constraints flag, base pair encloses an interior loop.
- #define **VRNA\_CONSTRAINT\_CONTEXT\_INT\_LOOP\_ENC** (unsigned char)0x08  
Hard constraints flag, base pair encloses a multi branch loop.
- #define **VRNA\_CONSTRAINT\_CONTEXT\_MB\_LOOP** (unsigned char)0x10  
Hard constraints flag, base pair is enclosed in an interior loop.
- #define **VRNA\_CONSTRAINT\_CONTEXT\_MB\_LOOP\_ENC** (unsigned char)0x20  
Hard constraints flag, base pair is enclosed in a multi branch loop.
- #define **VRNA\_CONSTRAINT\_CONTEXT\_ALL\_LOOPS**  
Constraint context flag indicating any loop context.

## Typedefs

- typedef struct [vrna\\_hc\\_s](#) **vrna\_hc\_t**  
Typename for the hard constraints data structure [vrna\\_hc\\_s](#).
- typedef struct [vrna\\_hc\\_up\\_s](#) **vrna\_hc\_up\_t**  
Typename for the single nucleotide hard constraint data structure [vrna\\_hc\\_up\\_s](#).
- typedef unsigned char() **vrna\_callback\_hc\_evaluate**(int i, int j, int k, int l, unsigned char d, void \*data)  
Callback to evaluate whether or not a particular decomposition step is contributing to the solution space.

## Functions

- void **vrna\_hc\_init** ([vrna\\_fold\\_compound\\_t](#) \*vc)  
Initialize/Reset hard constraints to default values.
- void **vrna\_hc\_add\_up** ([vrna\\_fold\\_compound\\_t](#) \*vc, int i, unsigned char option)  
Make a certain nucleotide unpaired.
- int **vrna\_hc\_add\_up\_batch** ([vrna\\_fold\\_compound\\_t](#) \*vc, [vrna\\_hc\\_up\\_t](#) \*constraints)  
Apply a list of hard constraints for single nucleotides.
- int **vrna\_hc\_add\_bp** ([vrna\\_fold\\_compound\\_t](#) \*vc, int i, int j, unsigned char option)  
Favorize/Enforce a certain base pair (i,j)
- void **vrna\_hc\_add\_bp\_nonspecific** ([vrna\\_fold\\_compound\\_t](#) \*vc, int i, int d, unsigned char option)  
Enforce a nucleotide to be paired (upstream/downstream)
- void **vrna\_hc\_free** ([vrna\\_hc\\_t](#) \*hc)  
Free the memory allocated by a [vrna\\_hc\\_t](#) data structure.
- int **vrna\_hc\_add\_from\_db** ([vrna\\_fold\\_compound\\_t](#) \*vc, const char \*constraint, unsigned int options)  
Add hard constraints from pseudo dot-bracket notation.

### 16.12.2 Data Structure Documentation

#### 16.12.2.1 struct [vrna\\_hc\\_s](#)

The hard constraints data structure.

The content of this data structure determines the decomposition pattern used in the folding recursions. Attribute 'matrix' is used as source for the branching pattern of the decompositions during all folding recursions. Any entry in matrix[i,j] consists of the 6 LSB that allows one to distinguish the following types of base pairs:

- in the exterior loop ([VRNA\\_CONSTRAINT\\_CONTEXT\\_EXT\\_LOOP](#))
- enclosing a hairpin ([VRNA\\_CONSTRAINT\\_CONTEXT\\_HP\\_LOOP](#))
- enclosing an interior loop ([VRNA\\_CONSTRAINT\\_CONTEXT\\_INT\\_LOOP](#))
- enclosed by an exterior loop ([VRNA\\_CONSTRAINT\\_CONTEXT\\_INT\\_LOOP\\_ENC](#))
- enclosing a multi branch loop ([VRNA\\_CONSTRAINT\\_CONTEXT\\_MB\\_LOOP](#))

- enclosed by a multi branch loop ([VRNA\\_CONSTRAINT\\_CONTEXT\\_MB\\_LOOP\\_ENC](#))

The four linear arrays 'up\_xxx' provide the number of available unpaired nucleotides (including position i) 3' of each position in the sequence.

See also

[vrna\\_hc\\_init\(\)](#), [vrna\\_hc\\_free\(\)](#), [VRNA\\_CONSTRAINT\\_CONTEXT\\_EXT\\_LOOP](#), [VRNA\\_CONSTRAINT\\_CONTEXT\\_HP\\_LOOP](#),  
[VRNA\\_CONSTRAINT\\_CONTEXT\\_INT\\_LOOP](#), [VRNA\\_CONSTRAINT\\_CONTEXT\\_MB\\_LOOP](#), [VRNA\\_CONSTRAINT\\_CONTEXT\\_MB\\_LOOP\\_ENC](#)

## Data Fields

- **int \* up\_ext**  
*A linear array that holds the number of allowed unpaired nucleotides in an exterior loop.*
- **int \* up\_hp**  
*A linear array that holds the number of allowed unpaired nucleotides in a hairpin loop.*
- **int \* up\_int**  
*A linear array that holds the number of allowed unpaired nucleotides in an interior loop.*
- **int \* up\_ml**  
*A linear array that holds the number of allowed unpaired nucleotides in a multi branched loop.*
- **vrna\_callback\_hc\_evaluate \* f**  
*A function pointer that returns whether or not a certain decomposition may be evaluated.*
- **void \* data**  
*A pointer to some structure where the user may store necessary data to evaluate its generic hard constraint function.*
- **vrna\_callback\_free\_auxdata \* free\_data**  
*A pointer to a function to free memory occupied by auxiliary data.*

### 16.12.2.1.1 Field Documentation

#### 16.12.2.1.1 **free\_data** [vrna\\_callback\\_free\\_auxdata](#)\* vrna\_hc\_s::free\_data

A pointer to a function to free memory occupied by auxiliary data.

The function this pointer is pointing to will be called upon destruction of the [vrna\\_hc\\_s](#), and provided with the [vrna\\_hc\\_s.data](#) pointer that may hold auxiliary data. Hence, to avoid leaking memory, the user may use this pointer to free memory occupied by auxiliary data.

### 16.12.2.2 struct vrna\_hc\_up\_s

A single hard constraint for a single nucleotide.

## Data Fields

- **int position**  
*The sequence position (1-based)*
- **unsigned char options**  
*The hard constraint option*

### 16.12.3 Macro Definition Documentation

### 16.12.3.1 VRNA\_CONSTRAINT\_DB

```
#define VRNA_CONSTRAINT_DB 16384U
#include <ViennaRNA/constraints/hard.h>
Flag for vrna\_constraints\_add\(\) to indicate that constraint is passed in pseudo dot-bracket notation.
```

See also

[vrna\\_constraints\\_add\(\)](#), [vrna\\_message\\_constraint\\_options\(\)](#), [vrna\\_message\\_constraint\\_options\\_all\(\)](#)

### 16.12.3.2 VRNA\_CONSTRAINT\_DB\_ENFORCE\_BP

```
#define VRNA_CONSTRAINT_DB_ENFORCE_BP 32768U
#include <ViennaRNA/constraints/hard.h>
Switch for dot-bracket structure constraint to enforce base pairs.  
This flag should be used to really enforce base pairs given in dot-bracket constraint rather than just weakly-enforcing them.
```

See also

[vrna\\_hc\\_add\\_from\\_db\(\)](#), [vrna\\_constraints\\_add\(\)](#), [vrna\\_message\\_constraint\\_options\(\)](#), [vrna\\_message\\_constraint\\_options\\_all\(\)](#)

### 16.12.3.3 VRNA\_CONSTRAINT\_DB\_PIPE

```
#define VRNA_CONSTRAINT_DB_PIPE 65536U
#include <ViennaRNA/constraints/hard.h>
Flag that is used to indicate the pipe '|' sign in pseudo dot-bracket notation of hard constraints.  
Use this definition to indicate the pipe sign '|' (paired with another base)
```

See also

[vrna\\_hc\\_add\\_from\\_db\(\)](#), [vrna\\_constraints\\_add\(\)](#), [vrna\\_message\\_constraint\\_options\(\)](#), [vrna\\_message\\_constraint\\_options\\_all\(\)](#)

### 16.12.3.4 VRNA\_CONSTRAINT\_DB\_DOT

```
#define VRNA_CONSTRAINT_DB_DOT 131072U
#include <ViennaRNA/constraints/hard.h>
dot '.' switch for structure constraints (no constraint at all)
```

See also

[vrna\\_hc\\_add\\_from\\_db\(\)](#), [vrna\\_constraints\\_add\(\)](#), [vrna\\_message\\_constraint\\_options\(\)](#), [vrna\\_message\\_constraint\\_options\\_all\(\)](#)

### 16.12.3.5 VRNA\_CONSTRAINT\_DB\_X

```
#define VRNA_CONSTRAINT_DB_X 262144U
#include <ViennaRNA/constraints/hard.h>
'X' switch for structure constraint (base must not pair)
```

See also

[vrna\\_hc\\_add\\_from\\_db\(\)](#), [vrna\\_constraints\\_add\(\)](#), [vrna\\_message\\_constraint\\_options\(\)](#), [vrna\\_message\\_constraint\\_options\\_all\(\)](#)

### 16.12.3.6 VRNA\_CONSTRAINT\_DB\_RND\_BRACK

```
#define VRNA_CONSTRAINT_DB_RND_BRACK 1048576U
#include <ViennaRNA/constraints/hard.h>
round brackets '(),' switch for structure constraint (base i pairs base j)
```

See also

[vrna\\_hc\\_add\\_from\\_db\(\)](#), [vrna\\_constraints\\_add\(\)](#), [vrna\\_message\\_constraint\\_options\(\)](#), [vrna\\_message\\_constraint\\_options\\_all\(\)](#)

### 16.12.3.7 VRNA\_CONSTRAINT\_DB\_INTRAMOL

```
#define VRNA_CONSTRAINT_DB_INTRAMOL 2097152U
#include <ViennaRNA/constraints/hard.h>
Flag that is used to indicate the character 'l' in pseudo dot-bracket notation of hard constraints.
Use this definition to indicate the usage of 'l' character (intramolecular pairs only)
```

See also

[vrna\\_hc\\_add\\_from\\_db\(\)](#), [vrna\\_constraints\\_add\(\)](#), [vrna\\_message\\_constraint\\_options\(\)](#), [vrna\\_message\\_constraint\\_options\\_all\(\)](#)

### 16.12.3.8 VRNA\_CONSTRAINT\_DB\_INTERMOL

```
#define VRNA_CONSTRAINT_DB_INTERMOL 4194304U
#include <ViennaRNA/constraints/hard.h>
Flag that is used to indicate the character 'e' in pseudo dot-bracket notation of hard constraints.
Use this definition to indicate the usage of 'e' character (intermolecular pairs only)
```

See also

[vrna\\_hc\\_add\\_from\\_db\(\)](#), [vrna\\_constraints\\_add\(\)](#), [vrna\\_message\\_constraint\\_options\(\)](#), [vrna\\_message\\_constraint\\_options\\_all\(\)](#)

### 16.12.3.9 VRNA\_CONSTRAINT\_DB\_GQUAD

```
#define VRNA_CONSTRAINT_DB_GQUAD 8388608U
#include <ViennaRNA/constraints/hard.h>
'+' switch for structure constraint (base is involved in a gquad)
```

See also

[vrna\\_hc\\_add\\_from\\_db\(\)](#), [vrna\\_constraints\\_add\(\)](#), [vrna\\_message\\_constraint\\_options\(\)](#), [vrna\\_message\\_constraint\\_options\\_all\(\)](#)

#### Warning

This flag is for future purposes only! No implementation recognizes it yet.

### 16.12.3.10 VRNA\_CONSTRAINT\_DB\_WUSS

```
#define VRNA_CONSTRAINT_DB_WUSS 33554432U
#include <ViennaRNA/constraints/hard.h>
Flag to indicate Washington University Secondary Structure (WUSS) notation of the hard constraint string.
This secondary structure notation for RNAs is usually used as consensus secondary structure (SS_cons) entry in
Stockholm formatted files
```

### 16.12.3.11 VRNA\_CONSTRAINT\_DB\_DEFAULT

```
#define VRNA_CONSTRAINT_DB_DEFAULT
#include <ViennaRNA/constraints/hard.h>
Value:
(VRNA_CONSTRAINT_DB \
| VRNA_CONSTRAINT_DB_PIPE \
| VRNA_CONSTRAINT_DB_DOT \
| VRNA_CONSTRAINT_DB_X \
| VRNA_CONSTRAINT_DB_ANG_BRACK \
| VRNA_CONSTRAINT_DB_RND_BRACK \
| VRNA_CONSTRAINT_DB_INTRAMOL \
| VRNA_CONSTRAINT_DB_INTERMOL \
| VRNA_CONSTRAINT_DB_GQUAD \
)
```

Switch for dot-bracket structure constraint with default symbols.

This flag conveniently combines all possible symbols in dot-bracket notation for hard constraints and [VRNA\\_CONSTRAINT\\_DB](#)

See also

[vrna\\_hc\\_add\\_from\\_db\(\)](#), [vrna\\_constraints\\_add\(\)](#), [vrna\\_message\\_constraint\\_options\(\)](#), [vrna\\_message\\_constraint\\_options\\_all\(\)](#)

## 16.12.4 Typedef Documentation

### 16.12.4.1 vrna\_callback\_hc\_evaluate

```
typedef unsigned char() vrna_callback_hc_evaluate(int i, int j, int k, int l, unsigned char d,
void *data)
#include <ViennaRNA/constraints/hard.h>
```

Callback to evaluate whether or not a particular decomposition step is contributing to the solution space.

This is the prototype for callback functions used by the folding recursions to evaluate generic hard constraints. The first four parameters passed indicate the delimiting nucleotide positions of the decomposition, and the parameter *d* denotes the decomposition step. The last parameter *data* is the auxiliary data structure associated to the hard constraints via [vrna\\_hc\\_add\\_data\(\)](#), or NULL if no auxiliary data was added.

**Notes on Callback Functions** This callback enables one to over-rule default hard constraints in secondary structure decompositions.

See also

[VRNA\\_DECOMP\\_PAIR\\_HP](#), [VRNA\\_DECOMP\\_PAIR\\_IL](#), [VRNA\\_DECOMP\\_PAIR\\_ML](#), [VRNA\\_DECOMP\\_ML\\_ML\\_ML](#),  
[VRNA\\_DECOMP\\_ML\\_STEM](#), [VRNA\\_DECOMP\\_ML\\_ML](#), [VRNA\\_DECOMP\\_ML\\_UP](#), [VRNA\\_DECOMP\\_ML\\_ML\\_STEM](#),  
[VRNA\\_DECOMP\\_ML\\_COAXIAL](#), [VRNA\\_DECOMP\\_EXT\\_EXT](#), [VRNA\\_DECOMP\\_EXT\\_UP](#), [VRNA\\_DECOMP\\_EXT\\_STEM](#),  
[VRNA\\_DECOMP\\_EXT\\_EXT\\_EXT](#), [VRNA\\_DECOMP\\_EXT\\_STEM\\_EXT](#), [VRNA\\_DECOMP\\_EXT\\_EXT\\_STEM](#),  
[VRNA\\_DECOMP\\_EXT\\_EXT\\_STEM1](#), [vrna\\_hc\\_add\\_f\(\)](#), [vrna\\_hc\\_add\\_data\(\)](#)

#### Parameters

|             |   |
|-------------|---|
| <i>i</i>    | Left (5') delimiter position of substructure  |
| <i>j</i>    | Right (3') delimiter position of substructure |
| <i>k</i>    | Left delimiter of decomposition               |
| <i>l</i>    | Right delimiter of decomposition              |
| <i>d</i>    | Decomposition step indicator                  |
| <i>data</i> | Auxiliary data                                |

**Returns**

A non-zero value if the decomposition is valid, 0 otherwise

### 16.12.5 Function Documentation

#### 16.12.5.1 vrna\_hc\_init()

```
void vrna_hc_init (
    vrna_fold_compound_t * vc )
#include <ViennaRNA/constraints/hard.h>
```

Initialize/Reset hard constraints to default values.

This function resets the hard constraints to their default values, i.e. all positions may be unpaired in all contexts, and base pairs are allowed in all contexts, if they resemble canonical pairs. Previously set hard constraints will be removed before initialization.

**See also**

[vrna\\_hc\\_add\\_bp\(\)](#), [vrna\\_hc\\_add\\_bp\\_nonspecific\(\)](#), [vrna\\_hc\\_add\\_up\(\)](#)

**Parameters**

|    |                   |
|----|-------------------|
| vc | The fold compound |
|----|-------------------|

**SWIG Wrapper Notes** This function is attached as method **hc\_init()** to objects of type *fold\_compound*

#### 16.12.5.2 vrna\_hc\_add\_up()

```
void vrna_hc_add_up (
    vrna_fold_compound_t * vc,
    int i,
    unsigned char option )
#include <ViennaRNA/constraints/hard.h>
```

Make a certain nucleotide unpaired.

**See also**

[vrna\\_hc\\_add\\_bp\(\)](#), [vrna\\_hc\\_add\\_bp\\_nonspecific\(\)](#), [vrna\\_hc\\_init\(\)](#), VRNA\_CONSTRAINT\_CONTEXT\_EXT\_LOOP, VRNA\_CONSTRAINT\_CONTEXT\_HP\_LOOP, VRNA\_CONSTRAINT\_CONTEXT\_INT\_LOOP, VRNA\_CONSTRAINT\_CONTEXT\_ALL\_LOOPS

**Parameters**

|        |   |
|--------|---|
| vc     | The <a href="#">vrna_fold_compound_t</a> the hard constraints are associated with |
| i      | The position that needs to stay unpaired (1-based)                                |
| option | The options flag indicating how/where to store the hard constraints               |

#### 16.12.5.3 vrna\_hc\_add\_up\_batch()

```
int vrna_hc_add_up_batch (
    vrna_fold_compound_t * vc,
    vrna_hc_up_t * constraints )
```

```
#include <ViennaRNA/constraints/hard.h>
Apply a list of hard constraints for single nucleotides.
```

#### Parameters

|                    |   |
|--------------------|---|
| <i>vc</i>          | The <code>vrna_fold_compound_t</code> the hard constraints are associated with      |
| <i>constraints</i> | The list off constraints to apply, last entry must have position attribute set to 0 |

#### 16.12.5.4 `vrna_hc_add_bp()`

```
int vrna_hc_add_bp (
    vrna_fold_compound_t * vc,
    int i,
    int j,
    unsigned char option )
#include <ViennaRNA/constraints/hard.h>
Favorize/Enforce a certain base pair (i,j)
```

#### See also

```
vrna_hc_add_bp_nonspecific(), vrna_hc_add_up(), vrna_hc_init(), VRNA_CONSTRAINT_CONTEXT_EXT_LOOP,
VRNA_CONSTRAINT_CONTEXT_HP_LOOP, VRNA_CONSTRAINT_CONTEXT_INT_LOOP, VRNA_CONSTRAINT_CONTEXT_MB_LOOP,
VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC, VRNA_CONSTRAINT_CONTEXT_ALL_LOOPS
```

#### Parameters

|               |  |
|---------------|--|
| <i>vc</i>     | The <code>vrna_fold_compound_t</code> the hard constraints are associated with |
| <i>i</i>      | The 5' located nucleotide position of the base pair (1-based)                  |
| <i>j</i>      | The 3' located nucleotide position of the base pair (1-based)                  |
| <i>option</i> | The options flag indicating how/where to store the hard constraints            |

#### 16.12.5.5 `vrna_hc_add_bp_nonspecific()`

```
void vrna_hc_add_bp_nonspecific (
    vrna_fold_compound_t * vc,
    int i,
    int d,
    unsigned char option )
#include <ViennaRNA/constraints/hard.h>
Enforce a nucleotide to be paired (upstream/downstream)
```

#### See also

```
vrna_hc_add_bp(), vrna_hc_add_up(), vrna_hc_init(), VRNA_CONSTRAINT_CONTEXT_EXT_LOOP,
VRNA_CONSTRAINT_CONTEXT_HP_LOOP, VRNA_CONSTRAINT_CONTEXT_INT_LOOP, VRNA_CONSTRAINT_CONTEXT_MB_LOOP,
VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC, VRNA_CONSTRAINT_CONTEXT_ALL_LOOPS
```

#### Parameters

|           |  |
|-----------|--|
| <i>vc</i> | The <code>vrna_fold_compound_t</code> the hard constraints are associated with |
| <i>i</i>  | The position that needs to stay unpaired (1-based)                             |

## Parameters

|               |  |
|---------------|--|
| <i>d</i>      | The direction of base pairing ( <i>d</i> < 0: pairs upstream, <i>d</i> > 0: pairs downstream, <i>d</i> == 0: no direction) |
| <i>option</i> | The options flag indicating in which loop type context the pairs may appear  |

**16.12.5.6 vrna\_hc\_free()**

```
void vrna_hc_free (
    vrna_hc_t * hc )
#include <ViennaRNA/constraints/hard.h>
Free the memory allocated by a vrna_hc_t data structure.
Use this function to free all memory that was allocated for a data structure of type vrna_hc_t .
```

## See also

[get\\_hard\\_constraints\(\)](#), [vrna\\_hc\\_t](#)

**16.12.5.7 vrna\_hc\_add\_from\_db()**

```
int vrna_hc_add_from_db (
    vrna_fold_compound_t * vc,
    const char * constraint,
    unsigned int options )
#include <ViennaRNA/constraints/hard.h>
```

Add hard constraints from pseudo dot-bracket notation.

This function allows one to apply hard constraints from a pseudo dot-bracket notation. The *options* parameter controls, which characters are recognized by the parser. Use the [VRNA\\_CONSTRAINT\\_DB\\_DEFAULT](#) convenience macro, if you want to allow all known characters

## See also

[VRNA\\_CONSTRAINT\\_DB\\_PIPE](#), [VRNA\\_CONSTRAINT\\_DB\\_DOT](#), [VRNA\\_CONSTRAINT\\_DB\\_X](#), [VRNA\\_CONSTRAINT\\_DB\\_RND\\_BRACK](#), [VRNA\\_CONSTRAINT\\_DB\\_INTRAMOL](#), [VRNA\\_CONSTRAINT\\_DB\\_INTERMOL](#), [VRNA\\_CONSTRAINT\\_DB\\_GQUAD](#)

## Parameters

|                   |   |
|-------------------|---|
| <i>vc</i>         | The fold compound                                     |
| <i>constraint</i> | A pseudo dot-bracket notation of the hard constraint. |
| <i>options</i>    | The option flags                                      |

**SWIG Wrapper Notes** This function is attached as method [hc\\_add\\_from\\_db\(\)](#) to objects of type *fold\_compound*

**16.13 Soft Constraints**

Functions and data structures for secondary structure soft constraints.

**16.13.1 Detailed Description**

Functions and data structures for secondary structure soft constraints.

Soft-constraints are used to change position specific contributions in the recursions by adding bonuses/penalties in form of pseudo free energies to certain loop configurations. Collaboration diagram for Soft Constraints:

## Files

- file [soft.h](#)  
*Functions and data structures for secondary structure soft constraints.*

## Data Structures

- struct [vrna\\_sc\\_s](#)  
*The soft constraints data structure. [More...](#)*

## TypeDefs

- [typedef struct vrna\\_sc\\_s vrna\\_sc\\_t](#)  
*Typename for the soft constraints data structure [vrna\\_sc\\_s](#).*
- [typedef int\(\) vrna\\_callback\\_sc\\_energy](#)(int i, int j, int k, int l, unsigned char d, void \*data)  
*Callback to retrieve pseudo energy contribution for soft constraint feature.*
- [typedef FLT\\_OR\\_DBL\(\) vrna\\_callback\\_sc\\_exp\\_energy](#)(int i, int j, int k, int l, unsigned char d, void \*data)  
*Callback to retrieve pseudo energy contribution as Boltzmann Factors for soft constraint feature.*
- [typedef vrna\\_basepair\\_t \\*\(\) vrna\\_callback\\_sc\\_backtrack](#)(int i, int j, int k, int l, unsigned char d, void \*data)  
*Callback to retrieve auxiliary base pairs for soft constraint feature.*

## Functions

- [void vrna\\_sc\\_init \(vrna\\_fold\\_compound\\_t \\*vc\)](#)  
*Initialize an empty soft constraints data structure within a [vrna\\_fold\\_compound\\_t](#).*
- [int vrna\\_sc\\_set\\_bp \(vrna\\_fold\\_compound\\_t \\*vc, const FLT\\_OR\\_DBL \\*\\*constraints, unsigned int options\)](#)  
*Set soft constraints for paired nucleotides.*
- [int vrna\\_sc\\_add\\_bp \(vrna\\_fold\\_compound\\_t \\*vc, int i, int j, FLT\\_OR\\_DBL energy, unsigned int options\)](#)  
*Add soft constraints for paired nucleotides.*
- [int vrna\\_sc\\_set\\_up \(vrna\\_fold\\_compound\\_t \\*vc, const FLT\\_OR\\_DBL \\*constraints, unsigned int options\)](#)  
*Set soft constraints for unpaired nucleotides.*
- [int vrna\\_sc\\_add\\_up \(vrna\\_fold\\_compound\\_t \\*vc, int i, FLT\\_OR\\_DBL energy, unsigned int options\)](#)  
*Add soft constraints for unpaired nucleotides.*
- [void vrna\\_sc\\_remove \(vrna\\_fold\\_compound\\_t \\*vc\)](#)  
*Remove soft constraints from [vrna\\_fold\\_compound\\_t](#).*
- [void vrna\\_sc\\_free \(vrna\\_sc\\_t \\*sc\)](#)  
*Free memory occupied by a [vrna\\_sc\\_t](#) data structure.*
- [int vrna\\_sc\\_add\\_data \(vrna\\_fold\\_compound\\_t \\*vc, void \\*data, vrna\\_callback\\_free\\_auxdata \\*free\\_data\)](#)  
*Add an auxiliary data structure for the generic soft constraints callback function.*
- [int vrna\\_sc\\_add\\_f \(vrna\\_fold\\_compound\\_t \\*vc, vrna\\_callback\\_sc\\_energy \\*f\)](#)  
*Bind a function pointer for generic soft constraint feature (MFE version)*
- [int vrna\\_sc\\_add\\_bt \(vrna\\_fold\\_compound\\_t \\*vc, vrna\\_callback\\_sc\\_backtrack \\*f\)](#)  
*Bind a backtracking function pointer for generic soft constraint feature.*
- [int vrna\\_sc\\_add\\_exp\\_f \(vrna\\_fold\\_compound\\_t \\*vc, vrna\\_callback\\_sc\\_exp\\_energy \\*exp\\_f\)](#)  
*Bind a function pointer for generic soft constraint feature (PF version)*

### 16.13.2 Data Structure Documentation

#### 16.13.2.1 struct vrna\_sc\_s

The soft constraints data structure.

Collaboration diagram for [vrna\\_sc\\_s](#):

## Data Fields

- **int \*\* energy\_up**  
*Energy contribution for stretches of unpaired nucleotides.*
- **FLT\_OR\_DBL \*\* exp\_energy\_up**  
*Boltzmann Factors of the energy contributions for unpaired sequence stretches.*
- **int \* up\_storage**  
*Storage container for energy contributions per unpaired nucleotide.*
- **vrna\_sc\_bp\_storage\_t \*\* bp\_storage**  
*Storage container for energy contributions per base pair.*
- **int \* energy\_stack**  
*Pseudo Energy contribution per base pair involved in a stack.*
- **FLT\_OR\_DBL \* exp\_energy\_stack**  
*Boltzmann weighted pseudo energy contribution per nucleotide involved in a stack.*
- **vrna\_callback\_sc\_energy \* f**  
*A function pointer used for pseudo energy contribution in MFE calculations.*
- **vrna\_callback\_sc\_backtrack \* bt**  
*A function pointer used to obtain backtraced base pairs in loop regions that were altered by soft constrained pseudo energy contributions.*
- **vrna\_callback\_sc\_exp\_energy \* exp\_f**  
*A function pointer used for pseudo energy contribution boltzmann factors in PF calculations.*
- **void \* data**  
*A pointer to the data object provided for for pseudo energy contribution functions of the generic soft constraints feature.*
- **int \* energy\_bp**  
*Energy contribution for base pairs.*
- **FLT\_OR\_DBL \* exp\_energy\_bp**  
*Boltzmann Factors of the energy contribution for base pairs.*
- **int \*\* energy\_bp\_local**  
*Energy contribution for base pairs (sliding window approach)*
- **FLT\_OR\_DBL \*\* exp\_energy\_bp\_local**  
*Boltzmann Factors of the energy contribution for base pairs (sliding window approach)*

### 16.13.2.1.1 Field Documentation

#### 16.13.2.1.1 **f vrna\_callback\_sc\_energy\* vrna\_sc\_s::f**

A function pointer used for pseudo energy contribution in MFE calculations.

See also

[vrna\\_sc\\_add\\_f\(\)](#)

#### 16.13.2.1.2 **bt vrna\_callback\_sc\_backtrack\* vrna\_sc\_s::bt**

A function pointer used to obtain backtraced base pairs in loop regions that were altered by soft constrained pseudo energy contributions.

See also

[vrna\\_sc\\_add\\_bt\(\)](#)

**16.13.2.1.1.3 `exp_f`** `vrna_callback_sc_exp_energy*` `vrna_sc_s::exp_f`

A function pointer used for pseudo energy contribution boltzmann factors in PF calculations.

See also

[vrna\\_sc\\_add\\_exp\\_f\(\)](#)

## 16.13.3 Typedef Documentation

### 16.13.3.1 `vrna_callback_sc_energy`

```
typedef int() vrna_callback_sc_energy(int i, int j, int k, int l, unsigned char d, void *data)
#include <ViennaRNA/constraints/soft.h>
```

Callback to retrieve pseudo energy contribution for soft constraint feature.

This is the prototype for callback functions used by the folding recursions to evaluate generic soft constraints. The first four parameters passed indicate the delimiting nucleotide positions of the decomposition, and the parameter denotes the decomposition step. The last parameter data is the auxiliary data structure associated to the hard constraints via [vrna\\_sc\\_add\\_data\(\)](#), or NULL if no auxiliary data was added.

**Notes on Callback Functions** This callback enables one to add (pseudo-)energy contributions to individual decompositions of the secondary structure.

See also

```
VRNA_DECOMP_PAIR_HP, VRNA_DECOMP_PAIR_IL, VRNA_DECOMP_PAIR_ML, VRNA_DECOMP_ML_ML_ML,
VRNA_DECOMP_ML_STEM, VRNA_DECOMP_ML_ML, VRNA_DECOMP_ML_UP, VRNA_DECOMP_ML_ML_STEM,
VRNA_DECOMP_ML_COAXIAL, VRNA_DECOMP_EXT_EXT, VRNA_DECOMP_EXT_UP, VRNA_DECOMP_EXT_STEM,
VRNA_DECOMP_EXT_EXT_EXT, VRNA_DECOMP_EXT_STEM_EXT, VRNA_DECOMP_EXT_EXT_STEM,
VRNA_DECOMP_EXT_EXT_STEM1, vrna\_sc\_add\_f\(\), vrna\_sc\_add\_exp\_f\(\), vrna\_sc\_add\_bt\(\), vrna\_sc\_add\_data\(\)
```

#### Parameters

|             |   |
|-------------|---|
| <i>i</i>    | Left (5') delimiter position of substructure  |
| <i>j</i>    | Right (3') delimiter position of substructure |
| <i>k</i>    | Left delimiter of decomposition               |
| <i>l</i>    | Right delimiter of decomposition              |
| <i>d</i>    | Decomposition step indicator                  |
| <i>data</i> | Auxiliary data                                |

#### Returns

Pseudo energy contribution in deka-kalories per mol

### 16.13.3.2 `vrna_callback_sc_exp_energy`

```
typedef FLT\_OR\_DBL\(\) vrna_callback_sc_exp_energy(int i, int j, int k, int l, unsigned char d,
void *data)
#include <ViennaRNA/constraints/soft.h>
```

Callback to retrieve pseudo energy contribution as Boltzmann Factors for soft constraint feature.

This is the prototype for callback functions used by the partition function recursions to evaluate generic soft constraints. The first four parameters passed indicate the delimiting nucleotide positions of the decomposition, and the parameter denotes the decomposition step. The last parameter data is the auxiliary data structure associated to the hard constraints via [vrna\\_sc\\_add\\_data\(\)](#), or NULL if no auxiliary data was added.

**Notes on Callback Functions** This callback enables one to add (pseudo-)energy contributions to individual decompositions of the secondary structure (Partition function variant, i.e. contributions must be returned as Boltzmann factors).

See also

[VRNA\\_DECOMP\\_PAIR\\_HP](#), [VRNA\\_DECOMP\\_PAIR\\_IL](#), [VRNA\\_DECOMP\\_PAIR\\_ML](#), [VRNA\\_DECOMP\\_ML\\_ML\\_ML](#),  
[VRNA\\_DECOMP\\_ML\\_STEM](#), [VRNA\\_DECOMP\\_ML\\_ML](#), [VRNA\\_DECOMP\\_ML\\_UP](#), [VRNA\\_DECOMP\\_ML\\_ML\\_STEM](#),  
[VRNA\\_DECOMP\\_ML\\_COAXIAL](#), [VRNA\\_DECOMP\\_EXT\\_EXT](#), [VRNA\\_DECOMP\\_EXT\\_UP](#), [VRNA\\_DECOMP\\_EXT\\_STEM](#),  
[VRNA\\_DECOMP\\_EXT\\_EXT\\_EXT](#), [VRNA\\_DECOMP\\_EXT\\_STEM\\_EXT](#), [VRNA\\_DECOMP\\_EXT\\_EXT\\_STEM](#),  
[VRNA\\_DECOMP\\_EXT\\_EXT\\_STEM1](#), [vrna\\_sc\\_add\\_exp\\_f\(\)](#), [vrna\\_sc\\_add\\_f\(\)](#), [vrna\\_sc\\_add\\_bt\(\)](#), [vrna\\_sc\\_add\\_data\(\)](#)

Parameters

|             |   |
|-------------|---|
| <i>i</i>    | Left (5') delimiter position of substructure  |
| <i>j</i>    | Right (3') delimiter position of substructure |
| <i>k</i>    | Left delimiter of decomposition               |
| <i>l</i>    | Right delimiter of decomposition              |
| <i>d</i>    | Decomposition step indicator                  |
| <i>data</i> | Auxiliary data                                |

Returns

Pseudo energy contribution in deka-kalories per mol

### 16.13.3.3 vrna\_callback\_sc\_backtrack

```
typedef vrna_basepair_t *() vrna_callback_sc_backtrack(int i, int j, int k, int l, unsigned
char d, void *data)
#include <ViennaRNA/constraints/soft.h>
Callback to retrieve auxiliary base pairs for soft constraint feature.
```

**Notes on Callback Functions** This callback enables one to add auxiliary base pairs in the backtracking steps of hairpin- and interior loops.

See also

[VRNA\\_DECOMP\\_PAIR\\_HP](#), [VRNA\\_DECOMP\\_PAIR\\_IL](#), [VRNA\\_DECOMP\\_PAIR\\_ML](#), [VRNA\\_DECOMP\\_ML\\_ML\\_ML](#),  
[VRNA\\_DECOMP\\_ML\\_STEM](#), [VRNA\\_DECOMP\\_ML\\_ML](#), [VRNA\\_DECOMP\\_ML\\_UP](#), [VRNA\\_DECOMP\\_ML\\_ML\\_STEM](#),  
[VRNA\\_DECOMP\\_ML\\_COAXIAL](#), [VRNA\\_DECOMP\\_EXT\\_EXT](#), [VRNA\\_DECOMP\\_EXT\\_UP](#), [VRNA\\_DECOMP\\_EXT\\_STEM](#),  
[VRNA\\_DECOMP\\_EXT\\_EXT\\_EXT](#), [VRNA\\_DECOMP\\_EXT\\_STEM\\_EXT](#), [VRNA\\_DECOMP\\_EXT\\_EXT\\_STEM](#),  
[VRNA\\_DECOMP\\_EXT\\_EXT\\_STEM1](#), [vrna\\_sc\\_add\\_bt\(\)](#), [vrna\\_sc\\_add\\_f\(\)](#), [vrna\\_sc\\_add\\_exp\\_f\(\)](#), [vrna\\_sc\\_add\\_data\(\)](#)

Parameters

|             |   |
|-------------|---|
| <i>i</i>    | Left (5') delimiter position of substructure  |
| <i>j</i>    | Right (3') delimiter position of substructure |
| <i>k</i>    | Left delimiter of decomposition               |
| <i>l</i>    | Right delimiter of decomposition              |
| <i>d</i>    | Decomposition step indicator                  |
| <i>data</i> | Auxiliary data                                |

**Returns**

List of additional base pairs

### 16.13.4 Function Documentation

#### 16.13.4.1 vrna\_sc\_init()

```
void vrna_sc_init (
    vrna_fold_compound_t * vc )
#include <ViennaRNA/constraints/soft.h>
```

Initialize an empty soft constraints data structure within a `vrna_fold_compound_t`.

This function adds a proper soft constraints data structure to the `vrna_fold_compound_t` data structure. If soft constraints already exist within the fold compound, they are removed.

**Note**

Accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE` and `VRNA_FC_TYPE_COMPARATIVE`

**See also**

[vrna\\_sc\\_set\\_bp\(\)](#), [vrna\\_sc\\_set\\_up\(\)](#), [vrna\\_sc\\_add\\_SHAPE\\_deigan\(\)](#), [vrna\\_sc\\_add\\_SHAPE\\_zarringhalam\(\)](#),  
[vrna\\_sc\\_remove\(\)](#), [vrna\\_sc\\_add\\_f\(\)](#), [vrna\\_sc\\_add\\_exp\\_f\(\)](#), [vrna\\_sc\\_add\\_pre\(\)](#), [vrna\\_sc\\_add\\_post\(\)](#)

**Parameters**

|                 |  |
|-----------------|--|
| <code>vc</code> | The <code>vrna_fold_compound_t</code> where an empty soft constraint feature is to be added to |
|-----------------|--|

**SWIG Wrapper Notes** This function is attached as method `sc_init()` to objects of type `fold_compound`

#### 16.13.4.2 vrna\_sc\_set\_bp()

```
int vrna_sc_set_bp (
    vrna_fold_compound_t * vc,
    const FLT_OR_DBL ** constraints,
    unsigned int options )
#include <ViennaRNA/constraints/soft.h>
```

Set soft constraints for paired nucleotides.

**Note**

This function replaces any pre-existing soft constraints with the ones supplied in `constraints`.

**See also**

[vrna\\_sc\\_add\\_bp\(\)](#), [vrna\\_sc\\_set\\_up\(\)](#), [vrna\\_sc\\_add\\_up\(\)](#)

**Parameters**

|                          |  |
|--------------------------|--|
| <code>vc</code>          | The <code>vrna_fold_compound_t</code> the soft constraints are associated with |
| <code>constraints</code> | A two-dimensional array of pseudo free energies in <i>kcal/mol</i>             |
| <code>options</code>     | The options flag indicating how/where to store the soft constraints            |

**Returns**

Non-zero on successful application of the constraint, 0 otherwise.

**SWIG Wrapper Notes** This function is attached as method **sc\_set\_bp()** to objects of type *fold\_compound*

**16.13.4.3 vrna\_sc\_add\_bp()**

```
int vrna_sc_add_bp (
    vrna_fold_compound_t * vc,
    int i,
    int j,
    FLT_OR_DBL energy,
    unsigned int options )
#include <ViennaRNA/constraints/soft.h>
```

Add soft constraints for paired nucleotides.

**See also**

[vrna\\_sc\\_set\\_bp\(\)](#), [vrna\\_sc\\_set\\_up\(\)](#), [vrna\\_sc\\_add\\_up\(\)](#)

**Parameters**

|                |   |
|----------------|---|
| <i>vc</i>      | The <a href="#">vrna_fold_compound_t</a> the soft constraints are associated with |
| <i>i</i>       | The 5' position of the base pair the soft constraint is added for                 |
| <i>j</i>       | The 3' position of the base pair the soft constraint is added for                 |
| <i>energy</i>  | The free energy (soft-constraint) in <i>kcal/mol</i>                              |
| <i>options</i> | The options flag indicating how/where to store the soft constraints               |

**Returns**

Non-zero on successful application of the constraint, 0 otherwise.

**SWIG Wrapper Notes** This function is attached as an overloaded method **sc\_add\_bp()** to objects of type *fold\_compound*. The method either takes arguments for a single base pair (i,j) with the corresponding energy value:

```
fold_compound.sc_add_bp(i, j, energy, options)
or an entire 2-dimensional matrix with dimensions n x n that stores free energy contributions for any base pair (i,j) with  $1 \leq i < j \leq n$ :
fold_compound.sc_add_bp(matrix, options)
```

In both variants, the `options` argument is optional can may be omitted.

**16.13.4.4 vrna\_sc\_set\_up()**

```
int vrna_sc_set_up (
    vrna_fold_compound_t * vc,
    const FLT_OR_DBL * constraints,
    unsigned int options )
#include <ViennaRNA/constraints/soft.h>
```

Set soft constraints for unpaired nucleotides.

**Note**

This function replaces any pre-existing soft constraints with the ones supplied in `constraints`.

**See also**

[vrna\\_sc\\_add\\_up\(\)](#), [vrna\\_sc\\_set\\_bp\(\)](#), [vrna\\_sc\\_add\\_bp\(\)](#)

**Parameters**

|                    |  |
|--------------------|--|
| <i>vc</i>          | The <code>vrna_fold_compound_t</code> the soft constraints are associated with |
| <i>constraints</i> | A vector of pseudo free energies in <i>kcal/mol</i>                            |
| <i>options</i>     | The options flag indicating how/where to store the soft constraints            |

**Returns**

Non-zero on successful application of the constraint, 0 otherwise.

**SWIG Wrapper Notes** This function is attached as method `sc_set_up()` to objects of type *fold\_compound*

**16.13.4.5 vrna\_sc\_add\_up()**

```
int vrna_sc_add_up (
    vrna_fold_compound_t * vc,
    int i,
    FLT_OR_DBL energy,
    unsigned int options )
#include <ViennaRNA/constraints/soft.h>
Add soft constraints for unpaired nucleotides.
```

**See also**

[vrna\\_sc\\_set\\_up\(\)](#), [vrna\\_sc\\_add\\_bp\(\)](#), [vrna\\_sc\\_set\\_bp\(\)](#)

**Parameters**

|                |  |
|----------------|--|
| <i>vc</i>      | The <code>vrna_fold_compound_t</code> the soft constraints are associated with |
| <i>i</i>       | The nucleotide position the soft constraint is added for                       |
| <i>energy</i>  | The free energy (soft-constraint) in <i>kcal/mol</i>                           |
| <i>options</i> | The options flag indicating how/where to store the soft constraints            |

**Returns**

Non-zero on successful application of the constraint, 0 otherwise.

**SWIG Wrapper Notes** This function is attached as an overloaded method `sc_add_up()` to objects of type *fold\_compound*. The method either takes arguments for a single nucleotide *i* with the corresponding energy value:

```
fold_compound.sc_add_up(i, energy, options)
or an entire vector that stores free energy contributions for each nucleotide i with  $1 \leq i \leq n$ :
fold_compound.sc_add_bp(vector, options)
```

In both variants, the `options` argument is optional can may be omitted.

**16.13.4.6 vrna\_sc\_remove()**

```
void vrna_sc_remove (
    vrna_fold_compound_t * vc )
#include <ViennaRNA/constraints/soft.h>
Remove soft constraints from vrna_fold_compound_t.
```

**Note**

Accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE` and `VRNA_FC_TYPE_COMPARATIVE`

## Parameters

|                 |  |
|-----------------|--|
| <code>vc</code> | The <code>vrna_fold_compound_t</code> possibly containing soft constraints |
|-----------------|--|

**SWIG Wrapper Notes** This function is attached as method `sc_remove()` to objects of type `fold_compound`

**16.13.4.7 vrna\_sc\_free()**

```
void vrna_sc_free (
    vrna_sc_t * sc )
#include <ViennaRNA/constraints/soft.h>
Free memory occupied by a vrna_sc_t data structure.
```

## Parameters

|                 |  |
|-----------------|--|
| <code>sc</code> | The data structure to free from memory |
|-----------------|--|

**16.13.4.8 vrna\_sc\_add\_data()**

```
int vrna_sc_add_data (
    vrna_fold_compound_t * vc,
    void * data,
    vrna_callback_free_auxdata * free_data )
#include <ViennaRNA/constraints/soft.h>
Add an auxiliary data structure for the generic soft constraints callback function.
```

## See also

`vrna_sc_add_f()`, `vrna_sc_add_exp_f()`, `vrna_sc_add_bt()`

## Parameters

|                        |   |
|------------------------|---|
| <code>vc</code>        | The fold compound the generic soft constraint function should be bound to                 |
| <code>data</code>      | A pointer to the data structure that holds required data for function 'f'                 |
| <code>free_data</code> | A pointer to a function that free's the memory occupied by <code>data</code> (Maybe NULL) |

## Returns

Non-zero on successful binding the data (and free-function), 0 otherwise

**SWIG Wrapper Notes** This function is attached as method `sc_add_data()` to objects of type `fold_compound`

**16.13.4.9 vrna\_sc\_add\_f()**

```
int vrna_sc_add_f (
    vrna_fold_compound_t * vc,
    vrna_callback_sc_energy * f )
#include <ViennaRNA/constraints/soft.h>
```

Bind a function pointer for generic soft constraint feature (MFE version)

This function allows one to easily bind a function pointer and corresponding data structure to the soft constraint part `vrna_sc_t` of the `vrna_fold_compound_t`. The function for evaluating the generic soft constraint feature has to return a pseudo free energy  $\hat{E}$  in `dacal/mol`, where  $1dacal/mol = 10cal/mol$ .

## See also

[vrna\\_sc\\_add\\_data\(\)](#), [vrna\\_sc\\_add\\_bt\(\)](#), [vrna\\_sc\\_add\\_exp\\_f\(\)](#)

## Parameters

|                 |  |
|-----------------|--|
| <code>vc</code> | The fold compound the generic soft constraint function should be bound to    |
| <code>f</code>  | A pointer to the function that evaluates the generic soft constraint feature |

## Returns

Non-zero on successful binding the callback function, 0 otherwise

**SWIG Wrapper Notes** This function is attached as method `sc_add_f()` to objects of type `fold_compound`

**16.13.4.10 vrna\_sc\_add\_bt()**

```
int vrna_sc_add_bt (
    vrna_fold_compound_t * vc,
    vrna_callback_sc_backtrack * f )
#include <ViennaRNA/constraints/soft.h>
```

Bind a backtracking function pointer for generic soft constraint feature.

This function allows one to easily bind a function pointer to the soft constraint part `vrna_sc_t` of the `vrna_fold_compound_t`. The provided function should be used for backtracking purposes in loop regions that were altered via the generic soft constraint feature. It has to return an array of `vrna_basepair_t` data structures, were the last element in the list is indicated by a value of -1 in it's `i` position.

## See also

[vrna\\_sc\\_add\\_data\(\)](#), [vrna\\_sc\\_add\\_f\(\)](#), [vrna\\_sc\\_add\\_exp\\_f\(\)](#)

## Parameters

|                 |   |
|-----------------|---|
| <code>vc</code> | The fold compound the generic soft constraint function should be bound to |
| <code>f</code>  | A pointer to the function that returns additional base pairs              |

## Returns

Non-zero on successful binding the callback function, 0 otherwise

**SWIG Wrapper Notes** This function is attached as method `sc_add_bt()` to objects of type `fold_compound`

**16.13.4.11 vrna\_sc\_add\_exp\_f()**

```
int vrna_sc_add_exp_f (
    vrna_fold_compound_t * vc,
    vrna_callback_sc_exp_energy * exp_f )
#include <ViennaRNA/constraints/soft.h>
```

Bind a function pointer for generic soft constraint feature (PF version)

This function allows one to easily bind a function pointer and corresponding data structure to the soft constraint part `vrna_sc_t` of the `vrna_fold_compound_t`. The function for evaluating the generic soft constraint feature has to return a pseudo free energy  $\hat{E}$  as Boltzmann factor, i.e.  $\exp(-\hat{E}/kT)$ . The required unit for  $E$  is *cal/mol*.

See also

[vrna\\_sc\\_add\\_bt\(\)](#), [vrna\\_sc\\_add\\_f\(\)](#), [vrna\\_sc\\_add\\_data\(\)](#)

#### Parameters

|                    |  |
|--------------------|--|
| <code>vc</code>    | The fold compound the generic soft constraint function should be bound to    |
| <code>exp_f</code> | A pointer to the function that evaluates the generic soft constraint feature |

#### Returns

Non-zero on successful binding the callback function, 0 otherwise

**SWIG Wrapper Notes** This function is attached as method `sc_add_exp_f()` to objects of type `fold_compound`

## 16.14 The RNA Secondary Structure Landscape

### 16.14.1 Detailed Description

Collaboration diagram for The RNA Secondary Structure Landscape:

#### Modules

- [Neighborhood Relation and Move Sets for Secondary Structures](#)

*Different functions to generate structural neighbors of a secondary structure according to a particular Move Set.*

- [\(Re-\)folding Paths, Saddle Points, Energy Barriers, and Local Minima](#)

*API for various RNA folding path algorithms.*

## 16.15 Minimum Free Energy (MFE) Algorithms

Predicting the Minimum Free Energy (MFE) and a corresponding (consensus) secondary structure.

### 16.15.1 Detailed Description

Predicting the Minimum Free Energy (MFE) and a corresponding (consensus) secondary structure.

In a nutshell we provide two different flavors for MFE prediction:

- [Global MFE Prediction](#) - to compute the MFE for the entire sequence
- [Local \(sliding window\) MFE Prediction](#) - to compute MFEs for each window using a sliding window approach

Each of these flavors, again, provides two implementations to either compute the MFE based on

- single RNA (DNA) sequence(s), or
- a comparative approach using multiple sequence alignments (MSA).

For the latter, a consensus secondary structure is predicted and our implementations compute an average of free energies for each sequence in the MSA plus an additional covariance pseudo-energy term.

The implementations for [Backtracking MFE structures](#) are generally agnostic with respect to whether local or global structure prediction is in place. Collaboration diagram for Minimum Free Energy (MFE) Algorithms:

#### Modules

- [Global MFE Prediction](#)

*Variations of the global Minimum Free Energy (MFE) prediction algorithm.*

- [Local \(sliding window\) MFE Prediction](#)

*Variations of the local (sliding window) Minimum Free Energy (MFE) prediction algorithm.*

- [Backtracking MFE structures](#)

*Backtracking related interfaces.*

## Files

- file [mfe.h](#)  
*Compute Minimum Free energy (MFE) and backtrace corresponding secondary structures from RNA sequence data.*
- file [mfe\\_window.h](#)  
*Compute local Minimum Free Energy (MFE) using a sliding window approach and backtrace corresponding secondary structures.*

## 16.16 Partition Function and Equilibrium Properties

Compute the partition function to assess various equilibrium properties.

### 16.16.1 Detailed Description

Compute the partition function to assess various equilibrium properties.

Similar to our [Minimum Free Energy \(MFE\) Algorithms](#), we provide two different flavors for partition function computations:

- [Global Partition Function and Equilibrium Probabilities](#) - to compute the partition function for a full length sequence
- [Local \(sliding window\) Partition Function and Equilibrium Probabilities](#) - to compute the partition function of each window using a sliding window approach

While the global partition function approach supports predictions using single sequences as well as consensus partition functions for multiple sequence alignments (MSA), we currently do not support MSA input for the local variant.

Comparative prediction computes an average of the free energy contributions plus an additional covariance pseudo-energy term, exactly as we do for the [Minimum Free Energy \(MFE\) Algorithms](#) implementation.

Boltzmann weights for the free energy contributions of individual loops can be found in [Energy Evaluation for Individual Loops](#). Our implementations also provide a stochastic backtracking procedure to draw [Random Structure Samples from the Ensemble](#) according to their equilibrium probability. Collaboration diagram for Partition Function and Equilibrium Properties:

## Modules

- [Global Partition Function and Equilibrium Probabilities](#)  
*Variations of the global partition function algorithm.*
- [Local \(sliding window\) Partition Function and Equilibrium Probabilities](#)  
*Scanning version using a sliding window approach to compute equilibrium probabilities.*

## Files

- file [concentrations.h](#)  
*Concentration computations for RNA-RNA interactions.*
- file [part\\_func.h](#)  
*Partition function implementations.*
- file [part\\_func\\_window.h](#)  
*Partition function and equilibrium probability implementation for the sliding window algorithm.*

## Functions

- int [vrna\\_pf\\_float\\_precision](#) (void)  
*Find out whether partition function computations are using single precision floating points.*

### 16.16.2 Function Documentation

### 16.16.2.1 vrna\_pf\_float\_precision()

```
int vrna_pf_float_precision (
    void )
#include <ViennaRNA/part_func.h>
```

Find out whether partition function computations are using single precision floating points.

See also

[FLT\\_OR\\_DBL](#)

Returns

1 if single precision is used, 0 otherwise

## 16.17 Global MFE Prediction

Variations of the global Minimum Free Energy (MFE) prediction algorithm.

### 16.17.1 Detailed Description

Variations of the global Minimum Free Energy (MFE) prediction algorithm.

We provide implementations of the global MFE prediction algorithm for

- Single sequences,
- Multiple sequence alignments (MSA), and
- RNA-RNA hybrids

Collaboration diagram for Global MFE Prediction:

### Modules

- [Computing MFE representatives of a Distance Based Partitioning](#)

*Compute the minimum free energy (MFE) and secondary structures for a partitioning of the secondary structure space according to the base pair distance to two fixed reference structures basepair distance to two fixed reference structures.*

- [Deprecated Interface for Global MFE Prediction](#)

### Files

- file [mfe.h](#)

*Compute Minimum Free energy (MFE) and backtrace corresponding secondary structures from RNA sequence data.*

### Basic global MFE prediction interface

- float [vrna\\_mfe](#) ([vrna\\_fold\\_compound\\_t](#) \*vc, char \*structure)

*Compute minimum free energy and an appropriate secondary structure of an RNA sequence, or RNA sequence alignment.*

- float [vrna\\_mfe\\_dimer](#) ([vrna\\_fold\\_compound\\_t](#) \*vc, char \*structure)

*Compute the minimum free energy of two interacting RNA molecules.*

### Simplified global MFE prediction using sequence(s) or multiple sequence alignment(s)

- float [vrna\\_fold](#) (const char \*sequence, char \*structure)

*Compute Minimum Free Energy (MFE), and a corresponding secondary structure for an RNA sequence.*

- float [vrna\\_circfold](#) (const char \*sequence, char \*structure)

*Compute Minimum Free Energy (MFE), and a corresponding secondary structure for a circular RNA sequence.*

- float [vrna\\_alifold](#) (const char \*\*sequences, char \*structure)  
*Compute Minimum Free Energy (MFE), and a corresponding consensus secondary structure for an RNA sequence alignment using a comparative method.*
- float [vrna\\_circalifold](#) (const char \*\*sequences, char \*structure)  
*Compute Minimum Free Energy (MFE), and a corresponding consensus secondary structure for a sequence alignment of circular RNAs using a comparative method.*
- float [vrna\\_cofold](#) (const char \*sequence, char \*structure)  
*Compute Minimum Free Energy (MFE), and a corresponding secondary structure for two dimerized RNA sequences.*

## 16.17.2 Function Documentation

### 16.17.2.1 [vrna\\_mfe\(\)](#)

```
float vrna_mfe (
    vrna_fold_compound_t * vc,
    char * structure )
#include <ViennaRNA/mfe.h>
```

Compute minimum free energy and an appropriate secondary structure of an RNA sequence, or RNA sequence alignment.

Depending on the type of the provided [vrna\\_fold\\_compound\\_t](#), this function predicts the MFE for a single sequence (or connected component of multiple sequences), or an averaged MFE for a sequence alignment. If backtracking is activated, it also constructs the corresponding secondary structure, or consensus structure. Therefore, the second parameter, *structure*, has to point to an allocated block of memory with a size of at least `strlen(sequence) + 1` to store the backtracked MFE structure. (For consensus structures, this is the length of the alignment + 1. If `NULL` is passed, no backtracking will be performed.

#### Note

This function is polymorphic. It accepts [vrna\\_fold\\_compound\\_t](#) of type [VRNA\\_FC\\_TYPE\\_SINGLE](#), and [VRNA\\_FC\\_TYPE\\_COMPARATIVE](#).

#### See also

[vrna\\_fold\\_compound\\_t](#), [vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\(\)](#), [vrna\\_circfold\(\)](#), [vrna\\_fold\\_compound\\_comparative\(\)](#), [vrna\\_alifold\(\)](#), [vrna\\_circalifold\(\)](#)

#### Parameters

|                  |  |
|------------------|--|
| <i>vc</i>        | fold compound  |
| <i>structure</i> | A pointer to the character array where the secondary structure in dot-bracket notation will be written to (Maybe <code>NULL</code> ) |

#### Returns

the minimum free energy (MFE) in kcal/mol

**SWIG Wrapper Notes** This function is attached as method `mfe()` to objects of type `fold_compound`

### 16.17.2.2 [vrna\\_mfe\\_dimer\(\)](#)

```
float vrna_mfe_dimer (
    vrna_fold_compound_t * vc,
    char * structure )
#include <ViennaRNA/mfe.h>
```

Compute the minimum free energy of two interacting RNA molecules.  
The code is analog to the [vrna\\_mfe\(\)](#) function.

**Deprecated** This function is obsolete since [vrna\\_mfe\(\)](#) can handle complexes multiple sequences since v2.5.0.  
Use [vrna\\_mfe\(\)](#) for connected component MFE instead and compute MFEs of unconnected states separately.

#### See also

[vrna\\_mfe\(\)](#)

#### Parameters

|                  |   |
|------------------|---|
| <i>vc</i>        | fold compound   |
| <i>structure</i> | Will hold the bracket dot structure of the dimer molecule |

#### Returns

minimum free energy of the structure

**SWIG Wrapper Notes** This function is attached as method [mfe\\_dimer\(\)](#) to objects of type *fold\_compound*

### 16.17.2.3 vrna\_fold()

```
float vrna_fold (
    const char * sequence,
    char * structure )
#include <ViennaRNA/mfe.h>
```

Compute Minimum Free Energy (MFE), and a corresponding secondary structure for an RNA sequence.  
This simplified interface to [vrna\\_mfe\(\)](#) computes the MFE and, if required, a secondary structure for an RNA sequence using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing, e.g. suboptimal backtracking, etc.

#### Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna\\_mfe\(\)](#), and the data structure [vrna\\_fold\\_compound\\_t](#) instead.

#### See also

[vrna\\_circfold\(\)](#), [vrna\\_mfe\(\)](#)

#### Parameters

|                  |   |
|------------------|---|
| <i>sequence</i>  | RNA sequence  |
| <i>structure</i> | A pointer to the character array where the secondary structure in dot-bracket notation will be written to |

**Returns**

the minimum free energy (MFE) in kcal/mol

**16.17.2.4 vrna\_circfold()**

```
float vrna_circfold (
    const char * sequence,
    char * structure )
#include <ViennaRNA/mfe.h>
```

Compute Minimum Free Energy (MFE), and a corresponding secondary structure for a circular RNA sequence. This simplified interface to [vrna\\_mfe\(\)](#) computes the MFE and, if required, a secondary structure for a circular RNA sequence using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing, e.g. suboptimal backtracking, etc. Folding of circular RNA sequences is handled as a post-processing step of the forward recursions. See [13] for further details.

**Note**

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna\\_mfe\(\)](#), and the data structure [vrna\\_fold\\_compound\\_t](#) instead.

**See also**

[vrna\\_fold\(\)](#), [vrna\\_mfe\(\)](#)

**Parameters**

|                  |   |
|------------------|---|
| <i>sequence</i>  | RNA sequence  |
| <i>structure</i> | A pointer to the character array where the secondary structure in dot-bracket notation will be written to |

**Returns**

the minimum free energy (MFE) in kcal/mol

**16.17.2.5 vrna\_alifold()**

```
float vrna_alifold (
    const char ** sequences,
    char * structure )
#include <ViennaRNA/mfe.h>
```

Compute Minimum Free Energy (MFE), and a corresponding consensus secondary structure for an RNA sequence alignment using a comparative method.

This simplified interface to [vrna\\_mfe\(\)](#) computes the MFE and, if required, a consensus secondary structure for an RNA sequence alignment using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing, e.g. suboptimal backtracking, etc.

**Note**

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna\\_mfe\(\)](#), and the data structure [vrna\\_fold\\_compound\\_t](#) instead.

See also

[vrna\\_circalifold\(\)](#), [vrna\\_mfe\(\)](#)

Parameters

|                  |   |
|------------------|---|
| <i>sequences</i> | RNA sequence alignment  |
| <i>structure</i> | A pointer to the character array where the secondary structure in dot-bracket notation will be written to |

Returns

the minimum free energy (MFE) in kcal/mol

#### 16.17.2.6 vrna\_circalifold()

```
float vrna_circalifold (
    const char ** sequences,
    char * structure )
#include <ViennaRNA/mfe.h>
```

Compute Minimum Free Energy (MFE), and a corresponding consensus secondary structure for a sequence alignment of circular RNAs using a comparative method.

This simplified interface to [vrna\\_mfe\(\)](#) computes the MFE and, if required, a consensus secondary structure for an RNA sequence alignment using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing, e.g. suboptimal backtracking, etc.

Folding of circular RNA sequences is handled as a post-processing step of the forward recursions. See [13] for further details.

Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna\\_mfe\(\)](#), and the data structure [vrna\\_fold\\_compound\\_t](#) instead.

See also

[vrna\\_alifold\(\)](#), [vrna\\_mfe\(\)](#)

Parameters

|                  |   |
|------------------|---|
| <i>sequences</i> | Sequence alignment of circular RNAs   |
| <i>structure</i> | A pointer to the character array where the secondary structure in dot-bracket notation will be written to |

Returns

the minimum free energy (MFE) in kcal/mol

#### 16.17.2.7 vrna\_cofold()

```
float vrna_cofold (
    const char * sequence,
    char * structure )
#include <ViennaRNA/mfe.h>
```

Compute Minimum Free Energy (MFE), and a corresponding secondary structure for two dimerized RNA sequences.

This simplified interface to [vrna\\_mfe\(\)](#) computes the MFE and, if required, a secondary structure for two RNA sequences upon dimerization using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing, e.g. suboptimal backtracking, etc.

#### Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna\\_mfe\(\)](#), and the data structure [vrna\\_fold\\_compound\\_t](#) instead.

**Deprecated** This function is obsolete since [vrna\\_mfe\(\)](#)/[vrna\\_fold\(\)](#) can handle complexes multiple sequences since v2.5.0. Use [vrna\\_mfe\(\)](#)/[vrna\\_fold\(\)](#) for connected component MFE instead and compute MFEs of unconnected states separately.

#### See also

[vrna\\_fold\(\)](#), [vrna\\_mfe\(\)](#), [vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\\_compound\\_t](#), [vrna\\_cut\\_point\\_insert\(\)](#)

#### Parameters

|                        |   |
|------------------------|---|
| <code>sequence</code>  | two RNA sequences separated by the '&' character  |
| <code>structure</code> | A pointer to the character array where the secondary structure in dot-bracket notation will be written to |

#### Returns

the minimum free energy (MFE) in kcal/mol

## 16.18 Local (sliding window) MFE Prediction

Variations of the local (sliding window) Minimum Free Energy (MFE) prediction algorithm.

### 16.18.1 Detailed Description

Variations of the local (sliding window) Minimum Free Energy (MFE) prediction algorithm.  
We provide implementations for the local (sliding window) MFE prediction algorithm for

- Single sequences,
- Multiple sequence alignments (MSA), and

Note, that our implementation scans an RNA sequence (or MSA) from the 3' to the 5' end, and reports back locally optimal (consensus) structures, the corresponding free energy, and the position of the sliding window in global coordinates.

For any particular RNA sequence (or MSA) multiple locally optimal (consensus) secondary structures may be predicted. Thus, we tried to implement an interface that allows for an effortless conversion of the corresponding hits into any target data structure. As a consequence, we provide two distinct ways to retrieve the corresponding predictions, either

- through directly writing to an open FILE stream on-the-fly, or
- through a callback function mechanism.

The latter allows one to store the results in any possible target data structure. Our implementations then pass the results through the user-implemented callback as soon as the prediction for a particular window is finished. Collaboration diagram for Local (sliding window) MFE Prediction:

## Modules

- Deprecated Interface for Local (Sliding Window) MFE Prediction

## Files

- file [mfe\\_window.h](#)

*Compute local Minimum Free Energy (MFE) using a sliding window approach and backtrace corresponding secondary structures.*

## Typedefs

- `typedef void() vrna_mfe_window_callback(int start, int end, const char *structure, float en, void *data)`  
*The default callback for sliding window MFE structure predictions.*

## Basic local (sliding window) MFE prediction interface

- `float vrna_mfe_window (vrna_fold_compound_t *vc, FILE *file)`  
*Local MFE prediction using a sliding window approach.*
- `float vrna_mfe_window_cb (vrna_fold_compound_t *vc, vrna_mfe_window_callback *cb, void *data)`
- `float vrna_mfe_window_zscore (vrna_fold_compound_t *vc, double min_z, FILE *file)`  
*Local MFE prediction using a sliding window approach (with z-score cut-off)*
- `float vrna_mfe_window_zscore_cb (vrna_fold_compound_t *vc, double min_z, vrna_mfe_window_zscore_callback *cb, void *data)`

## Simplified local MFE prediction using sequence(s) or multiple sequence alignment(s)

- `float vrna_Lfold (const char *string, int window_size, FILE *file)`  
*Local MFE prediction using a sliding window approach (simplified interface)*
- `float vrna_Lfold_cb (const char *string, int window_size, vrna_mfe_window_callback *cb, void *data)`
- `float vrna_Lfoldz (const char *string, int window_size, double min_z, FILE *file)`  
*Local MFE prediction using a sliding window approach with z-score cut-off (simplified interface)*
- `float vrna_Lfoldz_cb (const char *string, int window_size, double min_z, vrna_mfe_window_zscore_callback *cb, void *data)`
- `float vrna_aliLfold (const char **alignment, int maxdist, FILE *fp)`
- `float vrna_aliLfold_cb (const char **alignment, int maxdist, vrna_mfe_window_callback *cb, void *data)`

## 16.18.2 Typedef Documentation

### 16.18.2.1 [vrna\\_mfe\\_window\\_callback](#)

```
typedef void() vrna_mfe_window_callback(int start, int end, const char *structure, float en,
void *data)
#include <ViennaRNA/mfe_window.h>
The default callback for sliding window MFE structure predictions.
```

**Notes on Callback Functions** This function will be called for each hit in a sliding window MFE prediction.

**Parameters****See also**

[vrna\\_mfe\\_window\(\)](#)

**Parameters**

|                  |  |
|------------------|--|
| <i>start</i>     | provides the first position of the hit (1-based, relative to entire sequence/alignment)    |
| <i>end</i>       | provides the last position of the hit (1-based, relative to the entire sequence/alignment) |
| <i>structure</i> | provides the (sub)structure in dot-bracket notation  |
| <i>en</i>        | is the free energy of the structure hit in kcal/mol  |
| <i>data</i>      | is some arbitrary data pointer passed through by the function executing the callback       |

**16.18.3 Function Documentation****16.18.3.1 vrna\_mfe\_window()**

```
float vrna_mfe_window (
    vrna_fold_compound_t * vc,
    FILE * file )
#include <ViennaRNA/mfe_window.h>
```

Local MFE prediction using a sliding window approach.

Computes minimum free energy structures using a sliding window approach, where base pairs may not span outside the window. In contrast to [vrna\\_mfe\(\)](#), where a maximum base pair span may be set using the [vrna\\_md\\_t.max\\_bp\\_span](#) attribute and one globally optimal structure is predicted, this function uses a sliding window to retrieve all locally optimal structures within each window. The size of the sliding window is set in the [vrna\\_md\\_t.window\\_size](#) attribute, prior to the retrieval of the [vrna\\_fold\\_compound\\_t](#) using [vrna\\_fold\\_compound\(\)](#) with option [VRNA\\_OPTION\\_WINDOW](#)

The predicted structures are written on-the-fly, either to stdout, if a NULL pointer is passed as file parameter, or to the corresponding filehandle.

**See also**

[vrna\\_fold\\_compound\(\)](#), [vrna\\_mfe\\_window\\_zscore\(\)](#), [vrna\\_mfe\(\)](#), [vrna\\_Lfold\(\)](#), [vrna\\_Lfoldz\(\)](#), [VRNA\\_OPTION\\_WINDOW](#), [vrna\\_md\\_t.max\\_bp\\_span](#), [vrna\\_md\\_t.window\\_size](#)

**Parameters**

|             |   |
|-------------|---|
| <i>vc</i>   | The <a href="#">vrna_fold_compound_t</a> with preallocated memory for the DP matrices |
| <i>file</i> | The output file handle where predictions are written to (maybe NULL)                  |

**SWIG Wrapper Notes** This function is attached as method [mfe\\_window\(\)](#) to objects of type *fold\_compound*

**16.18.3.2 vrna\_mfe\_window\_zscore()**

```
float vrna_mfe_window_zscore (
    vrna_fold_compound_t * vc,
    double min_z,
    FILE * file )
#include <ViennaRNA/mfe_window.h>
```

Local MFE prediction using a sliding window approach (with z-score cut-off)

Computes minimum free energy structures using a sliding window approach, where base pairs may not span outside the window. This function is the z-score version of [vrna\\_mfe\\_window\(\)](#), i.e. only predictions above a certain z-score cut-off value are printed. As for [vrna\\_mfe\\_window\(\)](#), the size of the sliding window is set in the [vrna\\_md\\_t.window\\_size](#) attribute, prior to the retrieval of the [vrna\\_fold\\_compound\\_t](#) using [vrna\\_fold\\_compound\(\)](#) with option [VRNA\\_OPTION\\_WINDOW](#).

The predicted structures are written on-the-fly, either to stdout, if a NULL pointer is passed as file parameter, or to the corresponding filehandle.

#### See also

[vrna\\_fold\\_compound\(\)](#), [vrna\\_mfe\\_window\\_zscore\(\)](#), [vrna\\_mfe\(\)](#), [vrna\\_Lfold\(\)](#), [vrna\\_Lfoldz\(\)](#), [VRNA\\_OPTION\\_WINDOW](#), [vrna\\_md\\_t.max\\_bp\\_span](#), [vrna\\_md\\_t.window\\_size](#)

#### Parameters

|              |   |
|--------------|---|
| <i>vc</i>    | The <a href="#">vrna_fold_compound_t</a> with preallocated memory for the DP matrices |
| <i>min_z</i> | The minimal z-score for a predicted structure to appear in the output                 |
| <i>file</i>  | The output file handle where predictions are written to (maybe NULL)                  |

#### 16.18.3.3 [vrna\\_Lfold\(\)](#)

```
float vrna_Lfold (
    const char * string,
    int window_size,
    FILE * file )
#include <ViennaRNA/mfe_window.h>
```

Local MFE prediction using a sliding window approach (simplified interface)

This simplified interface to [vrna\\_mfe\\_window\(\)](#) computes the MFE and locally optimal secondary structure using default options. Structures are predicted using a sliding window approach, where base pairs may not span outside the window. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing.

#### Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna\\_mfe\\_window\(\)](#), and the data structure [vrna\\_fold\\_compound\\_t](#) instead.

#### See also

[vrna\\_mfe\\_window\(\)](#), [vrna\\_Lfoldz\(\)](#), [vrna\\_mfe\\_window\\_zscore\(\)](#)

#### Parameters

|                    |  |
|--------------------|--|
| <i>string</i>      | The nucleic acid sequence  |
| <i>window_size</i> | The window size for locally optimal structures   |
| <i>file</i>        | The output file handle where predictions are written to (if NULL, output is written to stdout) |

#### 16.18.3.4 [vrna\\_Lfoldz\(\)](#)

```
float vrna_Lfoldz (
```

```

    const char * string,
    int window_size,
    double min_z,
    FILE * file )
#include <ViennaRNA/mfe_window.h>

```

Local MFE prediction using a sliding window approach with z-score cut-off (simplified interface)

This simplified interface to [vrna\\_mfe\\_window\\_zscore\(\)](#) computes the MFE and locally optimal secondary structure using default options. Structures are predicted using a sliding window approach, where base pairs may not span outside the window. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing. This function is the z-score version of [vrna\\_Lfold\(\)](#), i.e. only predictions above a certain z-score cut-off value are printed.

#### Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna\\_mfe\\_window\(\)](#), and the data structure [vrna\\_fold\\_compound\\_t](#) instead.

#### See also

[vrna\\_mfe\\_window\\_zscore\(\)](#), [vrna\\_Lfold\(\)](#), [vrna\\_mfe\\_window\(\)](#)

#### Parameters

|                    |  |
|--------------------|--|
| <i>string</i>      | The nucleic acid sequence  |
| <i>window_size</i> | The window size for locally optimal structures   |
| <i>min_z</i>       | The minimal z-score for a predicted structure to appear in the output                          |
| <i>file</i>        | The output file handle where predictions are written to (if NULL, output is written to stdout) |

## 16.19 Backtracking MFE structures

Backtracking related interfaces.

### 16.19.1 Detailed Description

Backtracking related interfaces.

Collaboration diagram for Backtracking MFE structures:

#### Functions

- float [vrna\\_backtrack5](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, unsigned int length, char \*structure)  
*Backtrack an MFE (sub)structure.*
- int [vrna\\_BT\\_hp\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j, int en, [vrna\\_bp\\_stack\\_t](#) \*bp\_stack, int \*stack←\_count)  
*Backtrack a hairpin loop closed by (i, j).*
- int [vrna\\_BT\\_stack](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int \*i, int \*j, int \*en, [vrna\\_bp\\_stack\\_t](#) \*bp\_stack, int \*stack←\_count)  
*Backtrack a stacked pair closed by (i, j).*
- int [vrna\\_BT\\_int\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int \*i, int \*j, int en, [vrna\\_bp\\_stack\\_t](#) \*bp\_stack, int \*stack←\_count)  
*Backtrack an interior loop closed by (i, j).*
- int [vrna\\_BT\\_mb\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int \*i, int \*j, int \*k, int en, int \*component1, int \*component2)  
*Backtrack the decomposition of a multi branch loop closed by (i, j).*

## 16.19.2 Function Documentation

### 16.19.2.1 vrna\_backtrack5()

```
float vrna_backtrack5 (
    vrna_fold_compound_t * fc,
    unsigned int length,
    char * structure )
#include <ViennaRNA/mfe.h>
Backtrack an MFE (sub)structure.
```

This function allows one to backtrack the MFE structure for a (sub)sequence

#### Note

On error, the function returns **INF** / 100. and stores the empty string in `structure`.

#### Precondition

Requires pre-filled MFE dynamic programming matrices, i.e. one has to call [vrna\\_mfe\(\)](#) prior to calling this function

#### See also

[vrna\\_mfe\(\)](#), [vrna\\_pbacktrack5\(\)](#)

#### Parameters

|                        |  |
|------------------------|--|
| <code>fc</code>        | fold compound  |
| <code>length</code>    | The length of the subsequence, starting from the 5' end  |
| <code>structure</code> | A pointer to the character array where the secondary structure in dot-bracket notation will be written to. (Must have size of at least \$p length + 1) |

#### Returns

The minimum free energy (MFE) for the specified `length` in kcal/mol and a corresponding secondary structure in dot-bracket notation (stored in `structure`)

**SWIG Wrapper Notes** This function is attached as overloaded method **backtrack()** to objects of type `fold_<-compound` with default parameter `length` equal to the total length of the RNA.

### 16.19.2.2 vrna\_BT\_hp\_loop()

```
int vrna_BT_hp_loop (
    vrna_fold_compound_t * fc,
    int i,
    int j,
    int en,
    vrna_bp_stack_t * bp_stack,
    int * stack_count )
#include <ViennaRNA/loops/hairpin.h>
Backtrack a hairpin loop closed by (i,j).
```

#### Note

This function is polymorphic! The provided `vrna_fold_compound_t` may be of type `VRNA_FC_TYPE_SINGLE` or `VRNA_FC_TYPE_COMPARATIVE`

### 16.19.2.3 vrna\_BT\_mb\_loop()

```
int vrna_BT_mb_loop (
    vrna_fold_compound_t * fc,
    int * i,
    int * j,
    int * k,
    int en,
    int * component1,
    int * component2 )
#include <ViennaRNA/loops/multibranch.h>
```

Backtrack the decomposition of a multi branch loop closed by  $(i, j)$ .

#### Parameters

|                   |  |
|-------------------|--|
| <i>fc</i>         | The <code>vrna_fold_compound_t</code> filled with all relevant data for backtracking   |
| <i>i</i>          | 5' position of base pair closing the loop (will be set to 5' position of leftmost decomposed block upon successful backtracking)         |
| <i>j</i>          | 3' position of base pair closing the loop (will be set to 3' position of rightmost decomposed block upon successful backtracking)        |
| <i>k</i>          | Split position that delimits leftmost from rightmost block, [i,k] and [k+1, j], respectively. (Will be set upon successful backtracking) |
| <i>en</i>         | The energy contribution of the substructure enclosed by $(i, j)$   |
| <i>component1</i> | Type of leftmost block (1 = ML, 2 = C)   |
| <i>component2</i> | Type of rightmost block (1 = ML, 2 = C)  |

#### Returns

1, if backtracking succeeded, 0 otherwise.

## 16.20 Global Partition Function and Equilibrium Probabilities

Variations of the global partition function algorithm.

### 16.20.1 Detailed Description

Variations of the global partition function algorithm.

We provide implementations of the global partition function algorithm for

- Single sequences,
- Multiple sequence alignments (MSA), and
- RNA-RNA hybrids

Collaboration diagram for Global Partition Function and Equilibrium Probabilities:

## Modules

- [Computing Partition Functions of a Distance Based Partitioning](#)

*Compute the partition function and stochastically sample secondary structures for a partitioning of the secondary structure space according to the base pair distance to two fixed reference structures.*

- [Predicting various thermodynamic properties](#)

*Compute various thermodynamic properties using the partition function.*

- [Deprecated Interface for Global Partition Function Computation](#)

## Files

- file [part\\_func.h](#)

*Partition function implementations.*

## Data Structures

- struct [vrna\\_dimer\\_pf\\_s](#)  
*Data structure returned by [vrna\\_pf\\_dimer\(\)](#) More...*
- struct [vrna\\_multimer\\_pf\\_s](#)

## Functions

- [vrna\\_ep\\_t \\* vrna plist\\_from\\_probs \(vrna\\_fold\\_compound\\_t \\*vc, double cut\\_off\)](#)  
*Create a [vrna\\_ep\\_t](#) from base pair probability matrix.*

## Basic global partition function interface

- [FLT\\_OR\\_DBL vrna\\_pf \(vrna\\_fold\\_compound\\_t \\*vc, char \\*structure\)](#)  
*Compute the partition function  $Q$  for a given RNA sequence, or sequence alignment.*
- [vrna\\_dimer\\_pf\\_t vrna\\_pf\\_dimer \(vrna\\_fold\\_compound\\_t \\*vc, char \\*structure\)](#)  
*Calculate partition function and base pair probabilities of nucleic acid/nucleic acid dimers.*
- [FLT\\_OR\\_DBL \\* vrna\\_pf\\_substrands \(vrna\\_fold\\_compound\\_t \\*fc, size\\_t complex\\_size\)](#)
- [FLT\\_OR\\_DBL vrna\\_pf\\_add \(FLT\\_OR\\_DBL dG1, FLT\\_OR\\_DBL dG2, double kT\)](#)

## Simplified global partition function computation using sequence(s) or multiple sequence alignment(s)

- float [vrna\\_pf\\_fold \(const char \\*sequence, char \\*structure, vrna\\_ep\\_t \\*\\*pl\)](#)  
*Compute Partition function  $Q$  (and base pair probabilities) for an RNA sequence using a comparative method.*
- float [vrna\\_pf\\_circfold \(const char \\*sequence, char \\*structure, vrna\\_ep\\_t \\*\\*pl\)](#)  
*Compute Partition function  $Q$  (and base pair probabilities) for a circular RNA sequences using a comparative method.*
- float [vrna\\_pf\\_alifold \(const char \\*\\*sequences, char \\*structure, vrna\\_ep\\_t \\*\\*pl\)](#)  
*Compute Partition function  $Q$  (and base pair probabilities) for an RNA sequence alignment using a comparative method.*
- float [vrna\\_pf\\_circalifold \(const char \\*\\*sequences, char \\*structure, vrna\\_ep\\_t \\*\\*pl\)](#)  
*Compute Partition function  $Q$  (and base pair probabilities) for an alignment of circular RNA sequences using a comparative method.*
- [vrna\\_dimer\\_pf\\_t vrna\\_pf\\_co\\_fold \(const char \\*seq, char \\*structure, vrna\\_ep\\_t \\*\\*pl\)](#)  
*Calculate partition function and base pair probabilities of nucleic acid/nucleic acid dimers.*

### 16.20.2 Data Structure Documentation

#### 16.20.2.1 struct vrna\_dimer\_pf\_s

Data structure returned by [vrna\\_pf\\_dimer\(\)](#)

## Data Fields

- double **F0AB**  
*Null model without DuplexInit.*
- double **FAB**  
*all states with DuplexInit correction*
- double **FcAB**  
*true hybrid states only*

- double **FA**  
*monomer A*
- double **FB**  
*monomer B*

### 16.20.2.2 struct vrna\_multimer\_pf\_s

#### Data Fields

- double **F\_connected**  
*Fully connected ensemble (incl. DuplexInitiation and rotational symmetry correction.*
- double \* **F\_monomers**  
*monomers*
- size\_t **num\_monomers**  
*Number of monomers.*

### 16.20.3 Function Documentation

#### 16.20.3.1 vrna\_pf()

```
float vrna_pf (
    vrna_fold_compound_t * vc,
    char * structure )
#include <ViennaRNA/part_func.h>
```

Compute the partition function  $Q$  for a given RNA sequence, or sequence alignment.

If *structure* is not a NULL pointer on input, it contains on return a string consisting of the letters ". , | { } ( ) " denoting bases that are essentially unpaired, weakly paired, strongly paired without preference, weakly upstream (downstream) paired, or strongly up- (down-)stream paired bases, respectively. If the model's compute\_bpp is set to 0 base pairing probabilities will not be computed (saving CPU time), otherwise after calculations took place *pr* will contain the probability that bases  $i$  and  $j$  pair.

#### Note

This function is polymorphic. It accepts *vrna\_fold\_compound\_t* of type [VRNA\\_FC\\_TYPE\\_SINGLE](#), and [VRNA\\_FC\\_TYPE\\_COMPARATIVE](#).

This function may return [INF](#) / 100. in case of contradicting constraints or numerical over-/underflow. In the latter case, a corresponding warning will be issued to *stdout*.

#### See also

[vrna\\_fold\\_compound\\_t](#), [vrna\\_fold\\_compound\(\)](#), [vrna\\_pf\\_fold\(\)](#), [vrna\\_pf\\_circfold\(\)](#), [vrna\\_fold\\_compound\\_comparative\(\)](#), [vrna\\_pf\\_alifold\(\)](#), [vrna\\_pf\\_circalifold\(\)](#), [vrna\\_db\\_from\\_probs\(\)](#), [vrna\\_exp\\_params\(\)](#), [vrna\\_aln\\_pinfo\(\)](#)

#### Parameters

|         |                  |   |
|---------|------------------|---|
| in, out | <i>vc</i>        | The fold compound data structure  |
| in, out | <i>structure</i> | A pointer to the character array where position-wise pairing propensity will be stored.<br>(Maybe NULL) |

#### Returns

The ensemble free energy  $G = -RT \cdot \log(Q)$  in kcal/mol

**SWIG Wrapper Notes** This function is attached as method *pf()* to objects of type *fold\_compound*

### 16.20.3.2 vrna\_pf\_dimer()

```
vrna_dimer_pf_t vrna_pf_dimer (
    vrna_fold_compound_t * vc,
    char * structure )
#include <ViennaRNA/part_func.h>
```

Calculate partition function and base pair probabilities of nucleic acid/nucleic acid dimers.  
This is the cofold partition function folding.

#### Note

This function may return **INF / 100.** for the FA, FB, FAB, F0AB members of the output data structure in case of contradicting constraints or numerical over-/underflow. In the latter case, a corresponding warning will be issued to `stdout`.

#### See also

[vrna\\_fold\\_compound\(\)](#) for how to retrieve the necessary data structure

#### Parameters

|                        |  |
|------------------------|--|
| <code>vc</code>        | the fold compound data structure       |
| <code>structure</code> | Will hold the structure or constraints |

#### Returns

`vrna_dimer_pf_t` structure containing a set of energies needed for concentration computations.

**SWIG Wrapper Notes** This function is attached as method `pf_dimer()` to objects of type `fold_compound`

### 16.20.3.3 vrna\_pf\_fold()

```
float vrna_pf_fold (
    const char * sequence,
    char * structure,
    vrna_ep_t ** pl )
#include <ViennaRNA/part_func.h>
```

Compute Partition function  $Q$  (and base pair probabilities) for an RNA sequence using a comparative method. This simplified interface to [vrna\\_pf\(\)](#) computes the partition function and, if required, base pair probabilities for an RNA sequence using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing.

#### Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna\\_pf\(\)](#), and the data structure `vrna_fold_compound_t` instead.

#### See also

[vrna\\_pf\\_circfold\(\)](#), [vrna\\_pf\(\)](#), [vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\\_compound\\_t](#)

#### Parameters

|                        |  |
|------------------------|--|
| <code>sequence</code>  | RNA sequence   |
| <code>structure</code> | A pointer to the character array where position-wise pairing propensity will be stored. (Maybe NULL) |
| <code>pl</code>        | A pointer to a list of <code>vrna_ep_t</code> to store pairing probabilities (Maybe NULL)            |

**Returns**

The ensemble free energy  $G = -RT \cdot \log(Q)$  in kcal/mol

**16.20.3.4 vrna\_pf\_circfold()**

```
float vrna_pf_circfold (
    const char * sequence,
    char * structure,
    vrna_ep_t ** pl )
#include <ViennaRNA/part_func.h>
```

Compute Partition function  $Q$  (and base pair probabilities) for a circular RNA sequences using a comparative method.

This simplified interface to [vrna\\_pf\(\)](#) computes the partition function and, if required, base pair probabilities for a circular RNA sequence using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing.

**Note**

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna\\_pf\(\)](#), and the data structure [vrna\\_fold\\_compound\\_t](#) instead.

Folding of circular RNA sequences is handled as a post-processing step of the forward recursions. See [13] for further details.

**See also**

[vrna\\_pf\\_fold\(\)](#), [vrna\\_pf\(\)](#), [vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\\_compound\\_t](#)

**Parameters**

|                  |  |
|------------------|--|
| <i>sequence</i>  | A circular RNA sequence  |
| <i>structure</i> | A pointer to the character array where position-wise pairing propensity will be stored. (Maybe NULL) |
| <i>pl</i>        | A pointer to a list of <a href="#">vrna_ep_t</a> to store pairing probabilities (Maybe NULL)         |

**Returns**

The ensemble free energy  $G = -RT \cdot \log(Q)$  in kcal/mol

**16.20.3.5 vrna\_pf\_alifold()**

```
float vrna_pf_alifold (
    const char ** sequences,
    char * structure,
    vrna_ep_t ** pl )
#include <ViennaRNA/part_func.h>
```

Compute Partition function  $Q$  (and base pair probabilities) for an RNA sequence alignment using a comparative method.

This simplified interface to [vrna\\_pf\(\)](#) computes the partition function and, if required, base pair probabilities for an RNA sequence alignment using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing.

**Note**

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna\\_pf\(\)](#), and the data structure [vrna\\_fold\\_compound\\_t](#) instead.

**See also**

[vrna\\_pf\\_circalifold\(\)](#), [vrna\\_pf\(\)](#), [vrna\\_fold\\_compound\\_comparative\(\)](#), [vrna\\_fold\\_compound\\_t](#)

**Parameters**

|                  |  |
|------------------|--|
| <i>sequences</i> | RNA sequence alignment   |
| <i>structure</i> | A pointer to the character array where position-wise pairing propensity will be stored. (Maybe NULL) |
| <i>pl</i>        | A pointer to a list of <a href="#">vrna_ep_t</a> to store pairing probabilities (Maybe NULL)         |

**Returns**

The ensemble free energy  $G = -RT \cdot \log(Q)$  in kcal/mol

**16.20.3.6 vrna\_pf\_circalifold()**

```
float vrna_pf_circalifold (
    const char ** sequences,
    char * structure,
    vrna_ep_t ** pl )
#include <ViennaRNA/part_func.h>
```

Compute Partition function  $Q$  (and base pair probabilities) for an alignment of circular RNA sequences using a comparative method.

This simplified interface to [vrna\\_pf\(\)](#) computes the partition function and, if required, base pair probabilities for an RNA sequence alignment using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing.

**Note**

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna\\_pf\(\)](#), and the data structure [vrna\\_fold\\_compound\\_t](#) instead.

Folding of circular RNA sequences is handled as a post-processing step of the forward recursions. See [13] for further details.

**See also**

[vrna\\_pf\\_alifold\(\)](#), [vrna\\_pf\(\)](#), [vrna\\_fold\\_compound\\_comparative\(\)](#), [vrna\\_fold\\_compound\\_t](#)

**Parameters**

|                  |  |
|------------------|--|
| <i>sequences</i> | Sequence alignment of circular RNAs  |
| <i>structure</i> | A pointer to the character array where position-wise pairing propensity will be stored. (Maybe NULL) |
| <i>pl</i>        | A pointer to a list of <a href="#">vrna_ep_t</a> to store pairing probabilities (Maybe NULL)         |

**Returns**

The ensemble free energy  $G = -RT \cdot \log(Q)$  in kcal/mol

**16.20.3.7 vrna plist from\_probs()**

```
vrna_ep_t * vrna plist from_probs (
    vrna_fold_compound_t * vc,
    double cut_off )
#include <ViennaRNA/utils/structures.h>
```

Create a `vrna_ep_t` from base pair probability matrix.

The probability matrix provided via the `vrna_fold_compound_t` is parsed and all pair probabilities above the given threshold are used to create an entry in the plist

The end of the plist is marked by sequence positions i as well as j equal to 0. This condition should be used to stop looping over its entries

**Parameters**

|    |                      |                   |
|----|----------------------|-------------------|
| in | <code>vc</code>      | The fold compound |
| in | <code>cut_off</code> | The cutoff value  |

**Returns**

A pointer to the plist that is to be created

**16.20.3.8 vrna\_pf\_co\_fold()**

```
vrna_dimer_pf_t vrna_pf_co_fold (
    const char * seq,
    char * structure,
    vrna_ep_t ** pl )
#include <ViennaRNA/part_func.h>
```

Calculate partition function and base pair probabilities of nucleic acid/nucleic acid dimers.

This simplified interface to `vrna_pf_dimer()` computes the partition function and, if required, base pair probabilities for an RNA-RNA interaction using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing.

**Note**

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use `vrna_pf_dimer()`, and the data structure `vrna_fold_compound_t` instead.

**See also**

`vrna_pf_dimer()`

**Parameters**

|                        |  |
|------------------------|--|
| <code>seq</code>       | Two concatenated RNA sequences with a delimiting '&' in between                                      |
| <code>structure</code> | A pointer to the character array where position-wise pairing propensity will be stored. (Maybe NULL) |
| <code>pl</code>        | A pointer to a list of <code>vrna_ep_t</code> to store pairing probabilities (Maybe NULL)            |

**Returns**

`vrna_dimer_pf_t` structure containing a set of energies needed for concentration computations.

## 16.21 Local (sliding window) Partition Function and Equilibrium Probabilities

Scanning version using a sliding window approach to compute equilibrium probabilities.

### 16.21.1 Detailed Description

Scanning version using a sliding window approach to compute equilibrium probabilities.

Collaboration diagram for Local (sliding window) Partition Function and Equilibrium Probabilities:

### Modules

- [Deprecated Interface for Local \(Sliding Window\) Partition Function Computation](#)

### Files

- file [part\\_func\\_window.h](#)  
*Partition function and equilibrium probability implementation for the sliding window algorithm.*

### Macros

- `#define VRNA_EXT_LOOP 1U`  
*Exterior loop.*
- `#define VRNA_HP_LOOP 2U`  
*Hairpin loop.*
- `#define VRNA_INT_LOOP 4U`  
*Internal loop.*
- `#define VRNA_MB_LOOP 8U`  
*Multibranch loop.*
- `#define VRNA_ANY_LOOP (VRNA_EXT_LOOP | VRNA_HP_LOOP | VRNA_INT_LOOP | VRNA_MB_LOOP)`  
*Any loop.*
- `#define VRNA_PROBS_WINDOW_BPP 4096U`  
*Trigger base pairing probabilities.*
- `#define VRNA_PROBS_WINDOW_UP 8192U`  
*Trigger unpaired probabilities.*
- `#define VRNA_PROBS_WINDOW_STACKP 16384U`  
*Trigger base pair stack probabilities.*
- `#define VRNA_PROBS_WINDOW_UP_SPLIT 32768U`  
*Trigger detailed unpaired probabilities split up into different loop type contexts.*
- `#define VRNA_PROBS_WINDOW_PF 65536U`  
*Trigger partition function.*

### Typedefs

- `typedef void() vrna_probs_window_callback(FLT_OR_DBL *pr, int pr_size, int i, int max, unsigned int type, void *data)`  
*Sliding window probability computation callback.*

## Basic local partition function interface

- int [vrna\\_probs\\_window](#) (vrna\_fold\_compound\_t \*fc, int ulength, unsigned int options, vrna\_probs\_window\_callback \*cb, void \*data)

*Compute various equilibrium probabilities under a sliding window approach.*

## Simplified global partition function computation using sequence(s) or multiple sequence alignment(s)

- vrna\_ep\_t \* [vrna\\_pfl\\_fold](#) (const char \*sequence, int window\_size, int max\_bp\_span, float cutoff)  
*Compute base pair probabilities using a sliding-window approach.*
- int [vrna\\_pfl\\_fold\\_cb](#) (const char \*sequence, int window\_size, int max\_bp\_span, vrna\_probs\_window\_callback \*cb, void \*data)  
*Compute base pair probabilities using a sliding-window approach (callback version)*
- double \*\* [vrna\\_pfl\\_fold\\_up](#) (const char \*sequence, int ulength, int window\_size, int max\_bp\_span)  
*Compute probability of contiguous unpaired segments.*
- int [vrna\\_pfl\\_fold\\_up\\_cb](#) (const char \*sequence, int ulength, int window\_size, int max\_bp\_span, vrna\_probs\_window\_callback \*cb, void \*data)  
*Compute probability of contiguous unpaired segments.*

### 16.21.2 Macro Definition Documentation

#### 16.21.2.1 VRNA\_PROBS\_WINDOW\_BPP

```
#define VRNA_PROBS_WINDOW_BPP 4096U
#include <ViennaRNA/part_func_window.h>
```

Trigger base pairing probabilities.

Passing this flag to [vrna\\_probs\\_window\(\)](#) activates callback execution for base pairing probabilities. In turn, the corresponding callback receives this flag through the type argument whenever base pairing probabilities are provided.

Detailed information for the algorithm to compute unpaired probabilities can be taken from [3].

See also

[vrna\\_probs\\_window\(\)](#)

#### 16.21.2.2 VRNA\_PROBS\_WINDOW\_UP

```
#define VRNA_PROBS_WINDOW_UP 8192U
#include <ViennaRNA/part_func_window.h>
```

Trigger unpaired probabilities.

Passing this flag to [vrna\\_probs\\_window\(\)](#) activates callback execution for unpaired probabilities. In turn, the corresponding callback receives this flag through the type argument whenever unpaired probabilities are provided.

Detailed information for the algorithm to compute unpaired probabilities can be taken from [4].

See also

[vrna\\_probs\\_window\(\)](#)

### 16.21.2.3 VRNA\_PROBS\_WINDOW\_STACKP

```
#define VRNA_PROBS_WINDOW_STACKP 16384U
#include <ViennaRNA/part_func_window.h>
```

Trigger base pair stack probabilities.

Passing this flag to [vrna\\_probs\\_window\(\)](#) activates callback execution for stacking probabilities. In turn, the corresponding callback receives this flag through the `type` argument whenever stack probabilities are provided.

**Bug** Currently, this flag is a placeholder doing nothing as the corresponding implementation for stack probability computation is missing.

See also

[vrna\\_probs\\_window\(\)](#)

### 16.21.2.4 VRNA\_PROBS\_WINDOW\_UP\_SPLIT

```
#define VRNA_PROBS_WINDOW_UP_SPLIT 32768U
#include <ViennaRNA/part_func_window.h>
```

Trigger detailed unpaired probabilities split up into different loop type contexts.

Passing this flag to [vrna\\_probs\\_window\(\)](#) activates callback execution for unpaired probabilities. In contrast to [VRNA\\_PROBS\\_WINDOW\\_UP](#) this flag requests unpaired probabilities to be split up into different loop type contexts. In turn, the corresponding callback receives the [VRNA\\_PROBS\\_WINDOW\\_UP](#) flag OR-ed together with the corresponding loop type, i.e.:

- [VRNA\\_EXT\\_LOOP](#) - Exterior loop.
- [VRNA\\_HP\\_LOOP](#) - Hairpin loop.
- [VRNA\\_INT\\_LOOP](#) - Internal loop.
- [VRNA\\_MB\\_LOOP](#) - Multibranch loop.
- [VRNA\\_ANY\\_LOOP](#) - Any loop.

See also

[vrna\\_probs\\_window\(\)](#), [VRNA\\_PROBS\\_WINDOW\\_UP](#)

### 16.21.2.5 VRNA\_PROBS\_WINDOW\_PF

```
#define VRNA_PROBS_WINDOW_PF 65536U
#include <ViennaRNA/part_func_window.h>
```

Trigger partition function.

Passing this flag to [vrna\\_probs\\_window\(\)](#) activates callback execution for partition function. In turn, the corresponding callback receives this flag through its `type` argument whenever partition function data is provided.

Note

Instead of actually providing the partition function  $Z$ , the callback is always provided with the corresponding ensemble free energy  $\Delta G = -RT \ln Z$ .

See also

[vrna\\_probs\\_window\(\)](#)

## 16.21.3 Typedef Documentation

### 16.21.3.1 vrna\_probs\_window\_callback

```
typedef void() vrna_probs_window_callback(FLT_OR_DBL *pr, int pr_size, int i, int max, unsigned
int type, void *data)
#include <ViennaRNA/part_func_window.h>
Sliding window probability computation callback.
```

**Notes on Callback Functions** This function will be called for each probability data set in the sliding window probability computation implementation of [vrna\\_probs\\_window\(\)](#). The argument *type* specifies the type of probability that is passed to this function.

#### Types:

- [VRNA\\_PROBS\\_WINDOW\\_BPP](#) - Trigger base pairing probabilities.
- [VRNA\\_PROBS\\_WINDOW\\_UP](#) - Trigger unpaired probabilities.
- [VRNA\\_PROBS\\_WINDOW\\_PF](#) - Trigger partition function.

The above types usually come exclusively. However, for unpaired probabilities, the [VRNA\\_PROBS\\_WINDOW\\_UP](#) flag is OR-ed together with one of the loop type contexts

- [VRNA\\_EXT\\_LOOP](#) - Exterior loop.
- [VRNA\\_HP\\_LOOP](#) - Hairpin loop.
- [VRNA\\_INT\\_LOOP](#) - Internal loop.
- [VRNA\\_MB\\_LOOP](#) - Multibranch loop.
- [VRNA\\_ANY\\_LOOP](#) - Any loop.

to indicate the particular type of data available through the *pr* pointer.

#### See also

[vrna\\_probs\\_window\(\)](#), [vrna\\_pfl\\_fold\\_up\\_cb\(\)](#)

#### Parameters

|                |   |
|----------------|---|
| <i>pr</i>      | An array of probabilities                                 |
| <i>pr_size</i> | The length of the probability array                       |
| <i>i</i>       | The i-position (5') of the probabilities                  |
| <i>max</i>     | The (theoretical) maximum length of the probability array |
| <i>type</i>    | The type of data that is provided                         |
| <i>data</i>    | Auxiliary data  |

### 16.21.4 Function Documentation

#### 16.21.4.1 vrna\_probs\_window()

```
int vrna_probs_window (
    vrna_fold_compound_t * fc,
    int ulength,
    unsigned int options,
    vrna_probs_window_callback * cb,
    void * data )
```

```
#include <ViennaRNA/part_func_window.h>
```

Compute various equilibrium probabilities under a sliding window approach.

This function applies a sliding window scan for the sequence provided with the argument `fc` and reports back equilibrium probabilities through the callback function `cb`. The data reported to the callback depends on the `options` flag.

#### Note

The parameter `ulength` only affects computation and resulting data if unpaired probability computations are requested through the `options` flag.

#### Options:

- `VRNA_PROBS_WINDOW_BPP` - Trigger base pairing probabilities.
- `VRNA_PROBS_WINDOW_UP` - Trigger unpaired probabilities.
- `VRNA_PROBS_WINDOW_UP_SPLIT` - Trigger detailed unpaired probabilities split up into different loop type contexts.

Options may be OR-ed together

#### See also

`vrna_pfl_fold_cb()`, `vrna_pfl_fold_up_cb()`

#### Parameters

|                      |  |
|----------------------|--|
| <code>fc</code>      | The fold compound with sequence data, model settings and precomputed energy parameters |
| <code>ulength</code> | The maximal length of an unpaired segment (only for unpaired probability computations) |
| <code>cb</code>      | The callback function which collects the pair probability data for further processing  |
| <code>data</code>    | Some arbitrary data structure that is passed to the callback <code>cb</code>           |
| <code>options</code> | Option flags to control the behavior of this function                                  |

#### Returns

0 on failure, non-zero on success

#### 16.21.4.2 `vrna_pfl_fold()`

```
vrna_ep_t * vrna_pfl_fold (
    const char * sequence,
    int window_size,
    int max_bp_span,
    float cutoff )
#include <ViennaRNA/part_func_window.h>
```

Compute base pair probabilities using a sliding-window approach.

This is a simplified wrapper to `vrna_probs_window()` that given a nucleic acid sequence, a window size, a maximum base pair span, and a cutoff value computes the pair probabilities for any base pair in any window. The pair probabilities are returned as a list and the user has to take care to free() the memory occupied by the list.

#### Note

This function uses default model settings! For custom model settings, we refer to the function `vrna_probs_window()`.

In case of any computation errors, this function returns `NULL`.

## See also

[vrna\\_probs\\_window\(\)](#), [vrna\\_pfl\\_fold\\_cb\(\)](#), [vrna\\_pfl\\_fold\\_up\(\)](#)

## Parameters

|                    |  |
|--------------------|--|
| <i>sequence</i>    | The nucleic acid input sequence  |
| <i>window_size</i> | The size of the sliding window   |
| <i>max_bp_span</i> | The maximum distance along the backbone between two nucleotides that form a base pairs |
| <i>cutoff</i>      | A cutoff value that omits all pairs with lower probability                             |

## Returns

A list of base pair probabilities, terminated by an entry with [vrna\\_ep\\_t.i](#) and [vrna\\_ep\\_t.j](#) set to 0

**16.21.4.3 vrna\_pfl\_fold\_cb()**

```
int vrna_pfl_fold_cb (
    const char * sequence,
    int window_size,
    int max_bp_span,
    vrna_probs_window_callback * cb,
    void * data )
#include <ViennaRNA/part_func_window.h>
```

Compute base pair probabilities using a sliding-window approach (callback version)

This is a simplified wrapper to [vrna\\_probs\\_window\(\)](#) that given a nucleic acid sequence, a window size, a maximum base pair span, and a cutoff value computes the pair probabilities for any base pair in any window. It is similar to [vrna\\_pfl\\_fold\(\)](#) but uses a callback mechanism to return the pair probabilities.

Read the details for [vrna\\_probs\\_window\(\)](#) for details on the callback implementation!

## Note

This function uses default model settings! For custom model settings, we refer to the function [vrna\\_probs\\_window\(\)](#).

## See also

[vrna\\_probs\\_window\(\)](#), [vrna\\_pfl\\_fold\(\)](#), [vrna\\_pfl\\_fold\\_up\\_cb\(\)](#)

## Parameters

|                    |  |
|--------------------|--|
| <i>sequence</i>    | The nucleic acid input sequence  |
| <i>window_size</i> | The size of the sliding window   |
| <i>max_bp_span</i> | The maximum distance along the backbone between two nucleotides that form a base pairs |
| <i>cb</i>          | The callback function which collects the pair probability data for further processing  |
| <i>data</i>        | Some arbitrary data structure that is passed to the callback <i>cb</i>                 |

## Returns

0 on failure, non-zero on success

**16.21.4.4 vrna\_pfl\_fold\_up()**

```
double ** vrna_pfl_fold_up (
```

```

    const char * sequence,
    int ulength,
    int window_size,
    int max_bp_span )
#include <ViennaRNA/part_func_window.h>

```

Compute probability of contiguous unpaired segments.

This is a simplified wrapper to [vrna\\_probs\\_window\(\)](#) that given a nucleic acid sequence, a maximum length of unpaired segments (`ulength`), a window size, and a maximum base pair span computes the equilibrium probability of any segment not exceeding `ulength`. The probabilities to be unpaired are returned as a 1-based, 2-dimensional matrix with dimensions  $N \times M$ , where  $N$  is the length of the sequence and  $M$  is the maximum segment length. As an example, the probability of a segment of size 5 starting at position 100 is stored in the matrix entry  $X[100][5]$ . It is the users responsibility to free the memory occupied by this matrix.

#### Note

This function uses default model settings! For custom model settings, we refer to the function [vrna\\_probs\\_window\(\)](#).

#### Parameters

|                          |  |
|--------------------------|--|
| <code>sequence</code>    | The nucleic acid input sequence  |
| <code>ulength</code>     | The maximal length of an unpaired segment  |
| <code>window_size</code> | The size of the sliding window   |
| <code>max_bp_span</code> | The maximum distance along the backbone between two nucleotides that form a base pairs |

#### Returns

The probabilities to be unpaired for any segment not exceeding `ulength`

#### 16.21.4.5 vrna\_pfl\_fold\_up\_cb()

```

int vrna_pfl_fold_up_cb (
    const char * sequence,
    int ulength,
    int window_size,
    int max_bp_span,
    vrna_probs_window_callback * cb,
    void * data )
#include <ViennaRNA/part_func_window.h>

```

Compute probability of contiguous unpaired segments.

This is a simplified wrapper to [vrna\\_probs\\_window\(\)](#) that given a nucleic acid sequence, a maximum length of unpaired segments (`ulength`), a window size, and a maximum base pair span computes the equilibrium probability of any segment not exceeding `ulength`. It is similar to [vrna\\_pfl\\_fold\\_up\(\)](#) but uses a callback mechanism to return the unpaired probabilities.

Read the details for [vrna\\_probs\\_window\(\)](#) for details on the callback implementation!

#### Note

This function uses default model settings! For custom model settings, we refer to the function [vrna\\_probs\\_window\(\)](#).

#### Parameters

|                          |   |
|--------------------------|---|
| <code>sequence</code>    | The nucleic acid input sequence           |
| <code>ulength</code>     | The maximal length of an unpaired segment |
| <code>window_size</code> | The size of the sliding window            |

**Parameters**

|                          |  |
|--------------------------|--|
| <code>max_bp_span</code> | The maximum distance along the backbone between two nucleotides that form a base pairs |
| <code>cb</code>          | The callback function which collects the pair probability data for further processing  |
| <code>data</code>        | Some arbitrary data structure that is passed to the callback <code>cb</code>           |

**Returns**

0 on failure, non-zero on success

## 16.22 Suboptimals and Representative Structures

Sample and enumerate suboptimal secondary structures from RNA sequence data.

### 16.22.1 Detailed Description

Sample and enumerate suboptimal secondary structures from RNA sequence data.  
 Collaboration diagram for Suboptimals and Representative Structures:

## Modules

- [Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989](#)
- [Suboptimal Structures within an Energy Band around the MFE](#)
- [Random Structure Samples from the Ensemble](#)
  - Functions to draw random structure samples from the ensemble according to their equilibrium probability.*
- [Compute the Structure with Maximum Expected Accuracy \(MEA\)](#)
- [Compute the Centroid Structure](#)

## Files

- file [boltzmann\\_sampling.h](#)
  - Boltzmann Sampling of secondary structures from the ensemble.*
- file [centroid.h](#)
  - Centroid structure computation.*
- file [MEA.h](#)
  - Computes a MEA (maximum expected accuracy) structure.*
- file [mm.h](#)
  - Several Maximum Matching implementations.*
- file [subopt.h](#)
  - RNAsubopt and density of states declarations.*

## 16.23 Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989

### 16.23.1 Detailed Description

Collaboration diagram for Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989:

## Functions

- [`SOLUTION \* zukersubopt` \(`const char \*string`\)](#)
  - Compute Zuker type suboptimal structures.*
- [`SOLUTION \* zukersubopt\_par` \(`const char \*string, vrna\_param\_t \*parameters`\)](#)
  - Compute Zuker type suboptimal structures.*
- [`vrna\_subopt\_solution\_t \* vrna\_subopt\_zuker` \(`vrna\_fold\_compound\_t \*fc`\)](#)
  - Compute Zuker type suboptimal structures.*

## 16.23.2 Function Documentation

### 16.23.2.1 zukersubopt()

```
SOLUTION * zukersubopt (
    const char * string )
#include <ViennaRNA/subopt.h>
Compute Zuker type suboptimal structures.
Compute Suboptimal structures according to M. Zuker, i.e. for every possible base pair the minimum energy structure containing the resp. base pair. Returns a list of these structures and their energies.
```

**Deprecated** use vrna\_zukersubopt() instead

#### Parameters

|        |              |
|--------|--------------|
| string | RNA sequence |
|--------|--------------|

#### Returns

List of zuker suboptimal structures

### 16.23.2.2 zukersubopt\_par()

```
SOLUTION * zukersubopt_par (
    const char * string,
    vrna_param_t * parameters )
#include <ViennaRNA/subopt.h>
Compute Zuker type suboptimal structures.
```

**Deprecated** use vrna\_zukersubopt() instead

### 16.23.2.3 vrna\_subopt\_zuker()

```
vrna_subopt_solution_t * vrna_subopt_zuker (
    vrna_fold_compound_t * vc )
#include <ViennaRNA/subopt_zuker.h>
Compute Zuker type suboptimal structures.
Compute Suboptimal structures according to M. Zuker [27] , i.e. for every possible base pair the minimum energy structure containing the resp. base pair. Returns a list of these structures and their energies.
```

#### Note

This function internally uses the cofold implementation to compute the suboptimal structures. For that purpose, the function doubles the sequence and enlarges the DP matrices, which in fact will grow by a factor of 4 during the computation! At the end of the structure prediction, everything will be re-set to its original requirements, i.e. normal sequence, normal (empty) DP matrices.

**Bug** Due to resizing, any pre-existing constraints will be lost!

#### See also

[vrna\\_subopt\(\)](#), [zukersubopt\(\)](#), [zukersubopt\\_par\(\)](#)

**Parameters**

|    |               |
|----|---------------|
| vc | fold compound |
|----|---------------|

**Returns**

List of zuker suboptimal structures

**SWIG Wrapper Notes** This function is attached as method **subopt\_zuker()** to objects of type *fold\_compound*

## 16.24 Suboptimal Structures within an Energy Band around the MFE

### 16.24.1 Detailed Description

Collaboration diagram for Suboptimal Structures within an Energy Band around the MFE:

#### Typedefs

- `typedef void() vrna_subopt_callback(const char *structure, float energy, void *data)`  
*Callback for vrna\_subopt\_cb()*

#### Functions

- `vrna_subopt_solution_t * vrna_subopt (vrna_fold_compound_t *fc, int delta, int sorted, FILE *fp)`  
*Returns list of subopt structures or writes to fp.*
- `void vrna_subopt_cb (vrna_fold_compound_t *fc, int delta, vrna_subopt_callback *cb, void *data)`  
*Generate suboptimal structures within an energy band around the MFE.*
- `SOLUTION * subopt (char *seq, char *structure, int delta, FILE *fp)`  
*Returns list of subopt structures or writes to fp.*
- `SOLUTION * subopt_par (char *seq, char *structure, vrna_param_t *parameters, int delta, int is_left_constrained, int is_circular, FILE *fp)`  
*Returns list of subopt structures or writes to fp.*
- `SOLUTION * subopt_circ (char *seq, char *sequence, int delta, FILE *fp)`  
*Returns list of circular subopt structures or writes to fp.*

#### Variables

- `double print_energy`  
*printing threshold for use with logML*
- `int subopt_sorted`  
*Sort output by energy.*

### 16.24.2 Typedef Documentation

#### 16.24.2.1 vrna\_subopt\_callback

```
typedef void() vrna_subopt_callback(const char *structure, float energy, void *data)
#include <ViennaRNA/subopt.h>
Callback for vrna_subopt_cb()
```

**Notes on Callback Functions** This function will be called for each suboptimal secondary structure that is successfully backtraced.

#### See also

[vrna\\_subopt\\_cb\(\)](#)

### Parameters

|                        |  |
|------------------------|--|
| <code>structure</code> | The suboptimal secondary structure in dot-bracket notation                           |
| <code>energy</code>    | The free energy of the secondary structure in kcal/mol                               |
| <code>data</code>      | Some arbitrary, auxiliary data address as passed to <a href="#">vrna_subopt_cb()</a> |

## 16.24.3 Function Documentation

### 16.24.3.1 vrna\_subopt()

```
vrna_subopt_solution_t * vrna_subopt (
    vrna_fold_compound_t * vc,
    int delta,
    int sorted,
    FILE * fp )
#include <ViennaRNA/subopt.h>
```

Returns list of subopt structures or writes to fp.

This function produces **all** suboptimal secondary structures within '`delta`' \* 0.01 kcal/mol of the optimum, see [25]. The results are either directly written to a '`fp`' (if '`fp`' is not NULL), or (`fp==NULL`) returned in a `vrna_subopt_solution_t` \* list terminated by an entry were the '`structure`' member is NULL.

#### Note

This function requires all multibranch loop DP matrices for unique multibranch loop backtracing. Therefore, the supplied `vrna_fold_compound_t` `vc` (argument 1) must be initialized with `vrna_md_t.uniq_ML = 1`, for instance like this:

```
vrna_md_t md;
vrna_md_set_default(&md);
md.uniq_ML = 1;
vrna_fold_compound_t *vc=vrna_fold_compound("GGGGGAAAAAACCCCCC", &md, VRNA_OPTION_DEFAULT);
```

#### See also

[vrna\\_subopt\\_cb\(\)](#), [vrna\\_subopt\\_zuker\(\)](#)

### Parameters

|                     |   |
|---------------------|---|
| <code>fc</code>     |   |
| <code>delta</code>  |   |
| <code>sorted</code> | Sort results by energy in ascending order |
| <code>fp</code>     |   |

#### Returns

**SWIG Wrapper Notes** This function is attached as method `subopt()` to objects of type `fold_compound`

### 16.24.3.2 vrna\_subopt\_cb()

```
void vrna_subopt_cb (
    vrna_fold_compound_t * vc,
    int delta,
    vrna_subopt_callback * cb,
    void * data )
```

```
#include <ViennaRNA/subopt.h>
```

Generate suboptimal structures within an energy band around the MFE.

This is the most generic implementation of the suboptimal structure generator according to Wuchty et al. 1999 [25]. Identical to [vrna\\_subopt\(\)](#), it computes all secondary structures within an energy band `delta` around the MFE. However, this function does not print the resulting structures and their corresponding free energies to a file pointer, or returns them as a list. Instead, it calls a user-provided callback function which it passes the structure in dot-bracket format, the corresponding free energy in kcal/mol, and a user-provided data structure each time a structure was backtracked successfully. This function indicates the final output, i.e. the end of the backtracking procedure by passing NULL instead of an actual dot-bracket string to the callback.

#### Note

This function requires all multibranch loop DP matrices for unique multibranch loop backtracing. Therefore, the supplied `vrna_fold_compound_t` `vc` (argument 1) must be initialized with `vrna_md_t.uniq_ML = 1`, for instance like this:

```
vrna_md_t md;
vrna_md_set_default(&md);
md.uniq_ML = 1;
vrna_fold_compound_t *vc=vrna_fold_compound("GGGGGGAAAAAACCCCCC", &md, VRNA_OPTION_DEFAULT);
```

#### See also

[vrna\\_subopt\\_callback](#), [vrna\\_subopt\(\)](#), [vrna\\_subopt\\_zuker\(\)](#)

#### Parameters

|                    |   |
|--------------------|---|
| <code>fc</code>    | fold compound with the sequence data  |
| <code>delta</code> | Energy band around the MFE in 10cal/mol, i.e. deka-calories   |
| <code>cb</code>    | Pointer to a callback function that handles the backtracked structure and its free energy in kcal/mol |
| <code>data</code>  | Pointer to some data structure that is passed along to the callback                                   |

**SWIG Wrapper Notes** This function is attached as method `subopt_cb()` to objects of type `fold_compound`

#### 16.24.3.3 subopt()

```
SOLUTION * subopt (
    char * seq,
    char * structure,
    int delta,
    FILE * fp )
#include <ViennaRNA/subopt.h>
```

Returns list of subopt structures or writes to fp.

This function produces **all** suboptimal secondary structures within '`delta`' \* 0.01 kcal/mol of the optimum. The results are either directly written to a '`fp`' (if '`fp`' is not NULL), or (`fp==NULL`) returned in a `SOLUTION *` list terminated by an entry where the '`structure`' pointer is NULL.

#### Parameters

|                        |  |
|------------------------|--|
| <code>seq</code>       |  |
| <code>structure</code> |  |
| <code>delta</code>     |  |
| <code>fp</code>        |  |

Returns

#### 16.24.3.4 subopt\_circ()

```
SOLUTION * subopt_circ (
    char * seq,
    char * sequence,
    int delta,
    FILE * fp )
```

```
#include <ViennaRNA/subopt.h>
```

Returns list of circular subopt structures or writes to fp.

This function is similar to [subopt\(\)](#) but calculates secondary structures assuming the RNA sequence to be circular instead of linear

Parameters

|                 |  |
|-----------------|--|
| <i>seq</i>      |  |
| <i>sequence</i> |  |
| <i>delta</i>    |  |
| <i>fp</i>       |  |

Returns

## 16.25 Random Structure Samples from the Ensemble

Functions to draw random structure samples from the ensemble according to their equilibrium probability.

### 16.25.1 Detailed Description

Functions to draw random structure samples from the ensemble according to their equilibrium probability.  
Collaboration diagram for Random Structure Samples from the Ensemble:

### Modules

- [Stochastic Backtracking of Structures from Distance Based Partitioning](#)  
*Contains functions related to stochastic backtracking from a specified distance class.*
- [Deprecated Interface for Stochastic Backtracking](#)

### Macros

- `#define VRNA_PBACKTRACK_DEFAULT 0`  
*Boltzmann sampling flag indicating default backtracing mode.*
- `#define VRNA_PBACKTRACK_NON_REDUNDANT 1`  
*Boltzmann sampling flag indicating non-redundant backtracing mode.*

### Typedefs

- `typedef void() vrna_boltzmann_sampling_callback(const char *structure, void *data)`  
*Callback for Boltzmann sampling.*
- `typedef struct vrna_pbacktrack_memory_s * vrna_pbacktrack_mem_t`  
*Boltzmann sampling memory data structure.*

## Functions

- `char * vrna_pbacktrack5 (vrna_fold_compound_t *fc, unsigned int length)`  
*Sample a secondary structure of a subsequence from the Boltzmann ensemble according its probability.*
- `char ** vrna_pbacktrack5_num (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int length, unsigned int options)`  
*Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.*
- `unsigned int vrna_pbacktrack5_cb (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int length, vrna_boltzmann_sampling_callback *cb, void *data, unsigned int options)`  
*Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.*
- `char ** vrna_pbacktrack5_resume (vrna_fold_compound_t *vc, unsigned int num_samples, unsigned int length, vrna_pbacktrack_mem_t *nr_mem, unsigned int options)`  
*Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.*
- `unsigned int vrna_pbacktrack5_resume_cb (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int length, vrna_boltzmann_sampling_callback *cb, void *data, vrna_pbacktrack_mem_t *nr_mem, unsigned int options)`  
*Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.*
- `char * vrna_pbacktrack (vrna_fold_compound_t *fc)`  
*Sample a secondary structure from the Boltzmann ensemble according its probability.*
- `char ** vrna_pbacktrack_num (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int options)`  
*Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.*
- `unsigned int vrna_pbacktrack_cb (vrna_fold_compound_t *fc, unsigned int num_samples, vrna_boltzmann_sampling_callback *cb, void *data, unsigned int options)`  
*Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.*
- `char ** vrna_pbacktrack_resume (vrna_fold_compound_t *fc, unsigned int num_samples, vrna_pbacktrack_mem_t *nr_mem, unsigned int options)`  
*Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.*
- `unsigned int vrna_pbacktrack_resume_cb (vrna_fold_compound_t *fc, unsigned int num_samples, vrna_boltzmann_sampling_callback *cb, void *data, vrna_pbacktrack_mem_t *nr_mem, unsigned int options)`  
*Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.*
- `char * vrna_pbacktrack_sub (vrna_fold_compound_t *fc, unsigned int start, unsigned int end)`  
*Sample a secondary structure of a subsequence from the Boltzmann ensemble according its probability.*
- `char ** vrna_pbacktrack_sub_num (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int start, unsigned int end, unsigned int options)`  
*Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.*
- `unsigned int vrna_pbacktrack_sub_cb (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int start, unsigned int end, vrna_boltzmann_sampling_callback *cb, void *data, unsigned int options)`  
*Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.*
- `char ** vrna_pbacktrack_sub_resume (vrna_fold_compound_t *vc, unsigned int num_samples, unsigned int start, unsigned int end, vrna_pbacktrack_mem_t *nr_mem, unsigned int options)`  
*Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.*
- `unsigned int vrna_pbacktrack_sub_resume_cb (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int start, unsigned int end, vrna_boltzmann_sampling_callback *bs_cb, void *data, vrna_pbacktrack_mem_t *nr_mem, unsigned int options)`  
*Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.*
- `void vrna_pbacktrack_mem_free (vrna_pbacktrack_mem_t *s)`  
*Release memory occupied by a Boltzmann sampling memory data structure.*

## 16.25.2 Macro Definition Documentation

### 16.25.2.1 VRNA\_PBACKTRACK\_DEFAULT

```
#define VRNA_PBACKTRACK_DEFAULT 0
#include <ViennaRNA/boltzmann_sampling.h>
Boltzmann sampling flag indicating default backtracing mode.
```

See also

[vrna\\_pbacktrack5\\_num\(\)](#), [vrna\\_pbacktrack5\\_cb\(\)](#), [vrna\\_pbacktrack5\\_resume\(\)](#), [vrna\\_pbacktrack5\\_resume\\_cb\(\)](#),  
[vrna\\_pbacktrack\\_num\(\)](#), [vrna\\_pbacktrack\\_cb\(\)](#), [vrna\\_pbacktrack\\_resume\(\)](#), [vrna\\_pbacktrack\\_resume\\_cb\(\)](#)

### 16.25.2.2 VRNA\_PBACKTRACK\_NON\_REDUNDANT

```
#define VRNA_PBACKTRACK_NON_REDUNDANT 1
#include <ViennaRNA/boltzmann_sampling.h>
Boltzmann sampling flag indicating non-redundant backtracing mode.
This flag will turn the Boltzmann sampling into non-redundant backtracing mode along the lines of Michalik et al. 2017 [19]
```

See also

[vrna\\_pbacktrack5\\_num\(\)](#), [vrna\\_pbacktrack5\\_cb\(\)](#), [vrna\\_pbacktrack5\\_resume\(\)](#), [vrna\\_pbacktrack5\\_resume\\_cb\(\)](#),  
[vrna\\_pbacktrack\\_num\(\)](#), [vrna\\_pbacktrack\\_cb\(\)](#), [vrna\\_pbacktrack\\_resume\(\)](#), [vrna\\_pbacktrack\\_resume\\_cb\(\)](#)

## 16.25.3 Typedef Documentation

### 16.25.3.1 vrna\_boltzmann\_sampling\_callback

```
typedef void() vrna_boltzmann_sampling_callback(const char *structure, void *data)
#include <ViennaRNA/boltzmann_sampling.h>
Callback for Boltzmann sampling.
```

**Notes on Callback Functions** This function will be called for each secondary structure that has been successfully backtraced from the partition function DP matrices.

See also

[vrna\\_pbacktrack5\\_cb\(\)](#), [vrna\\_pbacktrack\\_cb\(\)](#), [vrna\\_pbacktrack5\\_resume\\_cb\(\)](#), [vrna\\_pbacktrack\\_resume\\_cb\(\)](#)

#### Parameters

|                        |  |
|------------------------|--|
| <code>structure</code> | The secondary structure in dot-bracket notation                            |
| <code>data</code>      | Some arbitrary, auxiliary data address as provided to the calling function |

### 16.25.3.2 vrna\_pbacktrack\_mem\_t

```
typedef struct vrna_pbacktrack_memory_s* vrna_pbacktrack_mem_t
#include <ViennaRNA/boltzmann_sampling.h>
```

Boltzmann sampling memory data structure.

This structure is required for properly resuming a previous sampling round in specialized Boltzmann sampling, such

as non-redundant backtracking.

Initialize with `NULL` and pass its address to the corresponding functions `vrna_pbacktrack5_resume()`, etc.

#### Note

Do not forget to release memory occupied by this data structure before losing its context! Use `vrna_pbacktrack_mem_free()`.

#### See also

`vrna_pbacktrack5_resume()`, `vrna_pbacktrack_resume()`, `vrna_pbacktrack5_resume_cb()`, `vrna_pbacktrack_resume_cb()`, `vrna_pbacktrack_mem_free()`

### 16.25.4 Function Documentation

#### 16.25.4.1 `vrna_pbacktrack5()`

```
char * vrna_pbacktrack5 (
    vrna_fold_compound_t * fc,
    unsigned int length )
#include <ViennaRNA/boltzmann_sampling.h>
```

Sample a secondary structure of a subsequence from the Boltzmann ensemble according its probability.  
Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a secondary structure.  
The parameter `length` specifies the length of the substructure starting from the 5' end.  
The structure  $s$  with free energy  $E(s)$  is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function  $Z = \sum_s \exp(-E(s)/kT)$ , Boltzmann constant  $k$  and thermodynamic temperature  $T$ .

#### Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar.  
This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML` = 1.

`vrna_pf()` has to be called first to fill the partition function matrices

#### Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

#### See also

`vrna_pbacktrack5_num()`, `vrna_pbacktrack5_cb()`, `vrna_pbacktrack()`

#### Parameters

|                     |  |
|---------------------|--|
| <code>fc</code>     | The fold compound data structure                                 |
| <code>length</code> | The length of the subsequence to consider (starting with 5' end) |

#### Returns

A sampled secondary structure in dot-bracket notation (or `NULL` on error)

**SWIG Wrapper Notes** This function is attached as overloaded method `pbacktrack5()` to objects of type `fold ← compound`. See also [Python Examples - Boltzmann Sampling](#)

### 16.25.4.2 `vrna_pbacktrack5_num()`

```
char ** vrna_pbacktrack5_num (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int length,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures. The parameter `length` specifies the length of the substructure starting from the 5' end.

Any structure  $s$  with free energy  $E(s)$  is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function  $Z = \sum_s \exp(-E(s)/kT)$ , Boltzmann constant  $k$  and thermodynamic temperature  $T$ .

Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [19].

#### Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

#### Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

#### Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

#### See also

`vrna_pbacktrack5()`, `vrna_pbacktrack5_cb()`, `vrna_pbacktrack_num()`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`

#### Parameters

|                          |  |
|--------------------------|--|
| <code>fc</code>          | The fold compound data structure                                 |
| <code>num_samples</code> | The size of the sample set, i.e. number of structures            |
| <code>length</code>      | The length of the subsequence to consider (starting with 5' end) |
| <code>options</code>     | A bitwise OR-flag indicating the backtracing mode.               |

**Returns**

A set of secondary structure samples in dot-bracket notation terminated by NULL (or NULL on error)

**SWIG Wrapper Notes** This function is attached as overloaded method `pbacktrack5()` to objects of type `fold_<compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

**16.25.4.3 vrna\_pbacktrack5\_cb()**

```
unsigned int vrna_pbacktrack5_cb (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int length,
    vrna_boltzmann_sampling_callback * cb,
    void * data,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures. The parameter `length` specifies the length of the substructure starting from the 5' end. Any structure  $s$  with free energy  $E(s)$  is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function  $Z = \sum_s \exp(-E(s)/kT)$ , Boltzmann constant  $k$  and thermodynamic temperature  $T$ .

Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [19].

In contrast to `vrna_pbacktrack5()` and `vrna_pbacktrack5_num()` this function yields the structure samples through a callback mechanism.

**Precondition**

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

**Note**

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

**Warning**

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

**See also**

`vrna_pbacktrack5()`, `vrna_pbacktrack5_num()`, `vrna_pbacktrack_cb()`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`

### Parameters

|                    |  |
|--------------------|--|
| <i>fc</i>          | The fold compound data structure                                 |
| <i>num_samples</i> | The size of the sample set, i.e. number of structures            |
| <i>length</i>      | The length of the subsequence to consider (starting with 5' end) |
| <i>cb</i>          | The callback that receives the sampled structure                 |
| <i>data</i>        | A data structure passed through to the callback <i>cb</i>        |
| <i>options</i>     | A bitwise OR-flag indicating the backtracing mode.               |

### Returns

The number of structures actually backtraced

**SWIG Wrapper Notes** This function is attached as overloaded method [pbacktrack5\(\)](#) to objects of type *fold* ← *compound* where the last argument *options* is optional with default value *options* = VRNA\_PBACKTRACK\_DEFAULT. See also [Python Examples - Boltzmann Sampling](#)

#### 16.25.4.4 vrna\_pbacktrack5\_resume()

```
char ** vrna_pbacktrack5_resume (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int length,
    vrna_pbacktrack_mem_t * nr_mem,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of *num\_samples* secondary structures. The parameter *length* specifies the length of the substructure starting from the 5' end.

Any structure *s* with free energy *E(s)* is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function  $Z = \sum_s \exp(-E(s)/kT)$ , Boltzmann constant *k* and thermodynamic temperature *T*.

Using the *options* flag one can switch between regular ([VRNA\\_PBACKTRACK\\_DEFAULT](#)) backtracing mode, and non-redundant sampling ([VRNA\\_PBACKTRACK\\_NON\\_REDUNDANT](#)) along the lines of Michalik et al. 2017 [19].

In contrast to [vrna\\_pbacktrack5\\_cb\(\)](#) this function allows for resuming a previous sampling round in specialized Boltzmann sampling, such as non-redundant backtracking. For that purpose, the user passes the address of a Boltzmann sampling data structure ([vrna\\_pbacktrack\\_mem\\_t](#)) which will be re-used in each round of sampling, i.e. each successive call to [vrna\\_pbacktrack5\\_resume\\_cb\(\)](#) or [vrna\\_pbacktrack5\\_resume\(\)](#).

A successive sample call to this function may look like:

```
vrna_pbacktrack_mem_t nonredundant_memory = NULL;
// sample the first 100 structures
vrna_pbacktrack5_resume(fc,
    100,
    fc->length,
    &nonredundant_memory,
    options);
// sample another 500 structures
vrna_pbacktrack5_resume(fc,
    500,
    fc->length,
    &nonredundant_memory,
    options);
// release memory occupied by the non-redundant memory data structure
vrna_pbacktrack_mem_free(nonredundant_memory);
```

**Precondition**

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

**Note**

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

**Warning**

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

**See also**

`vrna_pbacktrack5_resume_cb()`, `vrna_pbacktrack5_cb()`, `vrna_pbacktrack_resume()`, `vrna_pbacktrack_mem_t`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`, `vrna_pbacktrack_mem_free`

**Parameters**

|                          |  |
|--------------------------|--|
| <code>fc</code>          | The fold compound data structure                                 |
| <code>num_samples</code> | The size of the sample set, i.e. number of structures            |
| <code>length</code>      | The length of the subsequence to consider (starting with 5' end) |
| <code>nr_mem</code>      | The address of the Boltzmann sampling memory data structure      |
| <code>options</code>     | A bitwise OR-flag indicating the backtracing mode.               |

**Returns**

A set of secondary structure samples in dot-bracket notation terminated by NULL (or NULL on error)

**SWIG Wrapper Notes** This function is attached as overloaded method `pbacktrack5()` to objects of type `fold ← compound`. In addition to the list of structures, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

**16.25.4.5 vrna\_pbacktrack5\_resume\_cb()**

```
unsigned int vrna_pbacktrack5_resume_cb (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int length,
    vrna_boltzmann_sampling_callback * cb,
    void * data,
    vrna_pbacktrack_mem_t * nr_mem,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures. The parameter `length` specifies the length of the substructure starting from the 5' end.

Any structure  $s$  with free energy  $E(s)$  is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function  $Z = \sum_s \exp(-E(s)/kT)$ , Boltzmann constant  $k$  and thermodynamic temperature  $T$ . Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [19].

In contrast to `vrna_pbacktrack5_resume()` this function yields the structure samples through a callback mechanism.

A successive sample call to this function may look like:

```
vrna_pbacktrack_mem_t nonredundant_memory = NULL;
// sample the first 100 structures
vrna_pbacktrack5_resume_cb(fc,
                           100,
                           fc->length,
                           &callback_function,
                           (void *)&callback_data,
                           &nonredundant_memory,
                           options);

// sample another 500 structures
vrna_pbacktrack5_resume_cb(fc,
                           500,
                           fc->length,
                           &callback_function,
                           (void *)&callback_data,
                           &nonredundant_memory,
                           options);

// release memory occupied by the non-redundant memory data structure
vrna_pbacktrack_mem_free(nonredundant_memory);
```

### Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

### Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

### Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

### See also

`vrna_pbacktrack5_resume()`, `vrna_pbacktrack5_cb()`, `vrna_pbacktrack_resume_cb()`, `vrna_pbacktrack_mem_t`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`, `vrna_pbacktrack_mem_free`

### Parameters

|                          |  |
|--------------------------|--|
| <code>fc</code>          | The fold compound data structure                                 |
| <code>num_samples</code> | The size of the sample set, i.e. number of structures            |
| <code>length</code>      | The length of the subsequence to consider (starting with 5' end) |
| <code>cb</code>          | The callback that receives the sampled structure                 |
| <code>data</code>        | A data structure passed through to the callback <code>cb</code>  |
| <code>nr_mem</code>      | The address of the Boltzmann sampling memory data structure      |
| <code>options</code>     | A bitwise OR-flag indicating the backtracing mode.               |

**Returns**

The number of structures actually backtraced

**SWIG Wrapper Notes** This function is attached as overloaded method [pbacktrack5\(\)](#) to objects of type *fold ← compound*. In addition to the number of structures backtraced, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

**16.25.4.6 vrna\_pbacktrack()**

```
char * vrna_pbacktrack (
    vrna_fold_compound_t * fc )
#include <ViennaRNA/boltzmann_sampling.h>
```

Sample a secondary structure from the Boltzmann ensemble according its probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a secondary structure. The structure *s* with free energy  $E(s)$  is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function  $Z = \sum_s \exp(-E(s)/kT)$ , Boltzmann constant *k* and thermodynamic temperature *T*.

**Precondition**

Unique multiloop decomposition has to be active upon creation of `fc` with [vrna\\_fold\\_compound\(\)](#) or similar. This can be done easily by passing [vrna\\_fold\\_compound\(\)](#) a model details parameter with `vrna_md_t.uniq_ML = 1`.

[vrna\\_pf\(\)](#) has to be called first to fill the partition function matrices

**Note**

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

**See also**

[vrna\\_pbacktrack5\(\)](#), [vrna\\_pbacktrack\\_num\(\)](#), [vrna\\_pbacktrack\\_cb\(\)](#)

**Parameters**

|                 |                                  |
|-----------------|----------------------------------|
| <code>fc</code> | The fold compound data structure |
|-----------------|----------------------------------|

**Returns**

A sampled secondary structure in dot-bracket notation (or NULL on error)

**SWIG Wrapper Notes** This function is attached as overloaded method [pbacktrack\(\)](#) to objects of type *fold ← compound*. See also [Python Examples - Boltzmann Sampling](#)

**16.25.4.7 vrna\_pbacktrack\_num()**

```
char ** vrna_pbacktrack_num (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures.

Any structure  $s$  with free energy  $E(s)$  is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function  $Z = \sum_s \exp(-E(s)/kT)$ , Boltzmann constant  $k$  and thermodynamic temperature  $T$ .

Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [19].

#### Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

#### Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

#### Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

#### See also

`vrna_pbacktrack()`, `vrna_pbacktrack_cb()`, `vrna_pbacktrack5_num()`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`

#### Parameters

|                          |   |
|--------------------------|---|
| <code>fc</code>          | The fold compound data structure                      |
| <code>num_samples</code> | The size of the sample set, i.e. number of structures |
| <code>options</code>     | A bitwise OR-flag indicating the backtracing mode.    |

#### Returns

A set of secondary structure samples in dot-bracket notation terminated by NULL (or NULL on error)

**SWIG Wrapper Notes** This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

#### 16.25.4.8 `vrna_pbacktrack_cb()`

```
unsigned int vrna_pbacktrack_cb (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    vrna_boltzmann_sampling_callback * cb,
```

```

void * data,
unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>

```

Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.  
Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of num\_samples secondary structures.

Any structure  $s$  with free energy  $E(s)$  is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function  $Z = \sum_s \exp(-E(s)/kT)$ , Boltzmann constant  $k$  and thermodynamic temperature  $T$ . Using the options flag one can switch between regular ([VRNA\\_PBACKTRACK\\_DEFAULT](#)) backtracing mode, and non-redundant sampling ([VRNA\\_PBACKTRACK\\_NON\\_REDUNDANT](#)) along the lines of Michalik et al. 2017 [19].

In contrast to [vrna\\_pbacktrack\(\)](#) and [vrna\\_pbacktrack\\_num\(\)](#) this function yields the structure samples through a callback mechanism.

#### Precondition

Unique multiloop decomposition has to be active upon creation of fc with [vrna\\_fold\\_compound\(\)](#) or similar. This can be done easily by passing [vrna\\_fold\\_compound\(\)](#) a model details parameter with [vrna\\_md\\_t.uniq\\_ML](#) = 1.

[vrna\\_pf\(\)](#) has to be called first to fill the partition function matrices

#### Note

This function is polymorphic. It accepts [vrna\\_fold\\_compound\\_t](#) of type [VRNA\\_FC\\_TYPE\\_SINGLE](#), and [VRNA\\_FC\\_TYPE\\_COMPARATIVE](#).

#### Warning

In non-redundant sampling mode ([VRNA\\_PBACKTRACK\\_NON\\_REDUNDANT](#)), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

#### See also

[vrna\\_pbacktrack\(\)](#), [vrna\\_pbacktrack\\_num\(\)](#), [vrna\\_pbacktrack5\\_cb\(\)](#), [VRNA\\_PBACKTRACK\\_DEFAULT](#), [VRNA\\_PBACKTRACK\\_NON\\_REDUNDANT](#)

#### Parameters

|                    |   |
|--------------------|---|
| <i>fc</i>          | The fold compound data structure                      |
| <i>num_samples</i> | The size of the sample set, i.e. number of structures |
| <i>cb</i>          | The callback that receives the sampled structure      |
| <i>data</i>        | A data structure passed through to the callback cb    |
| <i>options</i>     | A bitwise OR-flag indicating the backtracing mode.    |

#### Returns

The number of structures actually backtraced

**SWIG Wrapper Notes** This function is attached as overloaded method [pbacktrack\(\)](#) to objects of type *fold ← compound* where the last argument *options* is optional with default value *options* = [VRNA\\_PBACKTRACK\\_DEFAULT](#). See also [Python Examples - Boltzmann Sampling](#)

#### 16.25.4.9 `vrna_pbacktrack_resume()`

```
char ** vrna_pbacktrack_resume (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    vrna_pbacktrack_mem_t * nr_mem,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.  
Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of num\_samples secondary structures.

Any structure  $s$  with free energy  $E(s)$  is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function  $Z = \sum_s \exp(-E(s)/kT)$ , Boltzmann constant  $k$  and thermodynamic temperature  $T$ . Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [19].

In contrast to `vrna_pbacktrack_cb()` this function allows for resuming a previous sampling round in specialized Boltzmann sampling, such as non-redundant backtracking. For that purpose, the user passes the address of a Boltzmann sampling data structure (`vrna_pbacktrack_mem_t`) which will be re-used in each round of sampling, i.e. each successive call to `vrna_pbacktrack_resume_cb()` or `vrna_pbacktrack_resume()`.

A successive sample call to this function may look like:

```
vrna_pbacktrack_mem_t nonredundant_memory = NULL;
// sample the first 100 structures
vrna_pbacktrack_resume(fc,
    100,
    &nonredundant_memory,
    options);
// sample another 500 structures
vrna_pbacktrack_resume(fc,
    500,
    &nonredundant_memory,
    options);
// release memory occupied by the non-redundant memory data structure
vrna_pbacktrack_mem_free(nonredundant_memory);
```

#### Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

#### Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

#### Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

#### See also

`vrna_pbacktrack_resume_cb()`, `vrna_pbacktrack_cb()`, `vrna_pbacktrack5_resume()`, `vrna_pbacktrack_mem_t`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`, `vrna_pbacktrack_mem_free`

### Parameters

|                          |   |
|--------------------------|---|
| <code>fc</code>          | The fold compound data structure                            |
| <code>num_samples</code> | The size of the sample set, i.e. number of structures       |
| <code>nr_mem</code>      | The address of the Boltzmann sampling memory data structure |
| <code>options</code>     | A bitwise OR-flag indicating the backtracing mode.          |

### Returns

A set of secondary structure samples in dot-bracket notation terminated by NULL (or NULL on error)

**SWIG Wrapper Notes** This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound`. In addition to the list of structures, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

### 16.25.4.10 `vrna_pbacktrack_resume_cb()`

```
unsigned int vrna_pbacktrack_resume_cb (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    vrna_boltzmann_sampling_callback * cb,
    void * data,
    vrna_pbacktrack_mem_t * nr_mem,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.  
Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures.

Any structure  $s$  with free energy  $E(s)$  is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function  $Z = \sum_s \exp(-E(s)/kT)$ , Boltzmann constant  $k$  and thermodynamic temperature  $T$ . Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [19].

In contrast to `vrna_pbacktrack5_resume()` this function yields the structure samples through a callback mechanism. A successive sample call to this function may look like:

```
vrna_pbacktrack_mem_t nonredundant_memory = NULL;
// sample the first 100 structures
vrna_pbacktrack5_resume_cb(fc,
    100,
    &callback_function,
    (void *)&callback_data,
    &nonredundant_memory,
    options);

// sample another 500 structures
vrna_pbacktrack5_resume_cb(fc,
    500,
    &callback_function,
    (void *)&callback_data,
    &nonredundant_memory,
    options);

// release memory occupied by the non-redundant memory data structure
vrna_pbacktrack_mem_free(nonredundant_memory);
```

### Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

**Note**

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

**Warning**

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

**See also**

`vrna_pbacktrack_resume()`, `vrna_pbacktrack_cb()`, `vrna_pbacktrack5_resume_cb()`, `vrna_pbacktrack_mem_t`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`, `vrna_pbacktrack_mem_free`

**Parameters**

|                          |   |
|--------------------------|---|
| <code>fc</code>          | The fold compound data structure                                |
| <code>num_samples</code> | The size of the sample set, i.e. number of structures           |
| <code>cb</code>          | The callback that receives the sampled structure                |
| <code>data</code>        | A data structure passed through to the callback <code>cb</code> |
| <code>nr_mem</code>      | The address of the Boltzmann sampling memory data structure     |
| <code>options</code>     | A bitwise OR-flag indicating the backtracing mode.              |

**Returns**

The number of structures actually backtraced

**SWIG Wrapper Notes** This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound`. In addition to the number of structures backtraced, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

**16.25.4.11 `vrna_pbacktrack_sub()`**

```
char * vrna_pbacktrack_sub (
    vrna_fold_compound_t * fc,
    unsigned int start,
    unsigned int end )
#include <ViennaRNA/boltzmann_sampling.h>
Sample a secondary structure of a subsequence from the Boltzmann ensemble according its probability.
Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a secondary structure.
The parameters start and end specify the interval  $[start : end]$  of the subsequence with  $1 \leq start < end \leq n$ 
for sequence length  $n$ , the structure  $s_{start,end}$  should be drawn from.
The resulting substructure  $s_{start,end}$  with free energy  $E(s_{start,end})$  is picked from the Boltzmann distributed sub
ensemble of all structures within the interval  $[start : end]$  according to its probability
```

$$p(s_{start,end}) = \frac{\exp(-E(s_{start,end})/kT)}{Z_{start,end}}$$

with partition function  $Z_{start,end} = \sum_{s_{start,end}} \exp(-E(s_{start,end})/kT)$ , Boltzmann constant  $k$  and thermodynamic temperature  $T$ .

### Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

### Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

### See also

`vrna_pbacktrack_sub_num()`, `vrna_pbacktrack_sub_cb()`, `vrna_pbacktrack()`

### Parameters

|                    |   |
|--------------------|---|
| <code>fc</code>    | The fold compound data structure  |
| <code>start</code> | The start of the subsequence to consider, i.e. 5'-end position(1-based) |
| <code>end</code>   | The end of the subsequence to consider, i.e. 3'-end position (1-based)  |

### Returns

A sampled secondary structure in dot-bracket notation (or NULL on error)

**SWIG Wrapper Notes** This function is attached as overloaded method `pbacktrack_sub()` to objects of type `fold_compound`. See also [Python Examples - Boltzmann Sampling](#)

### 16.25.4.12 `vrna_pbacktrack_sub_num()`

```
char ** vrna_pbacktrack_sub_num (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int start,
    unsigned int end,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures. The parameter `length` specifies the length of the substructure starting from the 5' end. Any structure  $s$  with free energy  $E(s)$  is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function  $Z = \sum_s \exp(-E(s)/kT)$ , Boltzmann constant  $k$  and thermodynamic temperature  $T$ .

Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [19].

### Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

**Note**

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

**Warning**

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

**See also**

`vrna_pbacktrack_sub()`, `vrna_pbacktrack_sub_cb()`, `vrna_pbacktrack_num()`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`

**Parameters**

|                          |   |
|--------------------------|---|
| <code>fc</code>          | The fold compound data structure  |
| <code>num_samples</code> | The size of the sample set, i.e. number of structures                   |
| <code>start</code>       | The start of the subsequence to consider, i.e. 5'-end position(1-based) |
| <code>end</code>         | The end of the subsequence to consider, i.e. 3'-end position (1-based)  |
| <code>options</code>     | A bitwise OR-flag indicating the backtracing mode.                      |

**Returns**

A set of secondary structure samples in dot-bracket notation terminated by NULL (or NULL on error)

**SWIG Wrapper Notes** This function is attached as overloaded method `pbacktrack_sub()` to objects of type `fold_compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

**16.25.4.13 vrna\_pbacktrack\_sub\_cb()**

```
unsigned int vrna_pbacktrack_sub_cb (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int start,
    unsigned int end,
    vrna_boltzmann_sampling_callback * cb,
    void * data,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures. The parameter `length` specifies the length of the substructure starting from the 5' end.

Any structure  $s$  with free energy  $E(s)$  is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function  $Z = \sum_s \exp(-E(s)/kT)$ , Boltzmann constant  $k$  and thermodynamic temperature  $T$ .

Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [19].

In contrast to `vrna_pbacktrack5()` and `vrna_pbacktrack5_num()` this function yields the structure samples through a callback mechanism.

### Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

### Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

### Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

### See also

`vrna_pbacktrack5()`, `vrna_pbacktrack5_num()`, `vrna_pbacktrack_cb()`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`

### Parameters

|                          |   |
|--------------------------|---|
| <code>fc</code>          | The fold compound data structure  |
| <code>num_samples</code> | The size of the sample set, i.e. number of structures                   |
| <code>start</code>       | The start of the subsequence to consider, i.e. 5'-end position(1-based) |
| <code>end</code>         | The end of the subsequence to consider, i.e. 3'-end position (1-based)  |
| <code>cb</code>          | The callback that receives the sampled structure                        |
| <code>data</code>        | A data structure passed through to the callback <code>cb</code>         |
| <code>options</code>     | A bitwise OR-flag indicating the backtracing mode.                      |

### Returns

The number of structures actually backtraced

**SWIG Wrapper Notes** This function is attached as overloaded method `pbacktrack()` to objects of type `fold ← compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

### 16.25.4.14 `vrna_pbacktrack_sub_resume()`

```
char ** vrna_pbacktrack_sub_resume (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int start,
    unsigned int end,
    vrna_pbacktrack_mem_t * nr_mem,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures. The parameter `length` specifies the length of the substructure starting from the 5' end.

Any structure  $s$  with free energy  $E(s)$  is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function  $Z = \sum_s \exp(-E(s)/kT)$ , Boltzmann constant  $k$  and thermodynamic temperature  $T$ . Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [19].

In contrast to `vrna_pbacktrack5_cb()` this function allows for resuming a previous sampling round in specialized Boltzmann sampling, such as non-redundant backtracking. For that purpose, the user passes the address of a Boltzmann sampling data structure (`vrna_pbacktrack_mem_t`) which will be re-used in each round of sampling, i.e. each successive call to `vrna_pbacktrack5_resume_cb()` or `vrna_pbacktrack5_resume()`.

A successive sample call to this function may look like:

```
vrna_pbacktrack_mem_t nonredundant_memory = NULL;
// sample the first 100 structures
vrna_pbacktrack5_resume(fc,
    100,
    fc->length,
    &nonredundant_memory,
    options);
// sample another 500 structures
vrna_pbacktrack5_resume(fc,
    500,
    fc->length,
    &nonredundant_memory,
    options);
// release memory occupied by the non-redundant memory data structure
vrna_pbacktrack_mem_free(nonredundant_memory);
```

#### Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

#### Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

#### Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

#### See also

`vrna_pbacktrack5_resume_cb()`, `vrna_pbacktrack5_cb()`, `vrna_pbacktrack_resume()`, `vrna_pbacktrack_mem_t`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`, `vrna_pbacktrack_mem_free`

#### Parameters

|                          |   |
|--------------------------|---|
| <code>fc</code>          | The fold compound data structure  |
| <code>num_samples</code> | The size of the sample set, i.e. number of structures                   |
| <code>start</code>       | The start of the subsequence to consider, i.e. 5'-end position(1-based) |
| <code>end</code>         | The end of the subsequence to consider, i.e. 3'-end position (1-based)  |
| <code>nr_mem</code>      | The address of the Boltzmann sampling memory data structure             |
| <code>options</code>     | A bitwise OR-flag indicating the backtracing mode.                      |

**Returns**

A set of secondary structure samples in dot-bracket notation terminated by NULL (or NULL on error)

**SWIG Wrapper Notes** This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound`. In addition to the list of structures, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

**16.25.4.15 vrna\_pbacktrack\_sub\_resume\_cb()**

```
unsigned int vrna_pbacktrack_sub_resume_cb (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int start,
    unsigned int end,
    vrna_boltzmann_sampling_callback * cb,
    void * data,
    vrna_pbacktrack_mem_t * nr_mem,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures. The parameter `length` specifies the length of the substructure starting from the 5' end.

Any structure  $s$  with free energy  $E(s)$  is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function  $Z = \sum_s \exp(-E(s)/kT)$ , Boltzmann constant  $k$  and thermodynamic temperature  $T$ .

Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [19].

In contrast to `vrna_pbacktrack5_resume()` this function yields the structure samples through a callback mechanism.

A successive sample call to this function may look like:

```
vrna_pbacktrack_mem_t nonredundant_memory = NULL;
// sample the first 100 structures
vrna_pbacktrack5_resume_cb(fc,
    100,
    fc->length,
    &callback_function,
    (void *)&callback_data,
    &nonredundant_memory,
    options);
// sample another 500 structures
vrna_pbacktrack5_resume_cb(fc,
    500,
    fc->length,
    &callback_function,
    (void *)&callback_data,
    &nonredundant_memory,
    options);
// release memory occupied by the non-redundant memory data structure
vrna_pbacktrack_mem_free(nonredundant_memory);
```

**Precondition**

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

**Note**

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

**Warning**

In non-redundant sampling mode ([VRNA\\_PBACKTRACK\\_NON\\_REDUNDANT](#)), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

**See also**

[vrna\\_pbacktrack5\\_resume\(\)](#), [vrna\\_pbacktrack5\\_cb\(\)](#), [vrna\\_pbacktrack\\_resume\\_cb\(\)](#), [vrna\\_pbacktrack\\_mem\\_t](#),  
[VRNA\\_PBACKTRACK\\_DEFAULT](#), [VRNA\\_PBACKTRACK\\_NON\\_REDUNDANT](#), [vrna\\_pbacktrack\\_mem\\_free](#)

**Parameters**

|                    |   |
|--------------------|---|
| <i>fc</i>          | The fold compound data structure  |
| <i>num_samples</i> | The size of the sample set, i.e. number of structures                   |
| <i>start</i>       | The start of the subsequence to consider, i.e. 5'-end position(1-based) |
| <i>end</i>         | The end of the subsequence to consider, i.e. 3'-end position (1-based)  |
| <i>cb</i>          | The callback that receives the sampled structure                        |
| <i>data</i>        | A data structure passed through to the callback <i>cb</i>               |
| <i>nr_mem</i>      | The address of the Boltzmann sampling memory data structure             |
| <i>options</i>     | A bitwise OR-flag indicating the backtracing mode.                      |

**Returns**

The number of structures actually backtraced

**SWIG Wrapper Notes** This function is attached as overloaded method [pbacktrack\(\)](#) to objects of type *fold\_* ← *compound*. In addition to the number of structures backtraced, this function also returns the *nr\_mem* data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

**16.25.4.16 vrna\_pbacktrack\_mem\_free()**

```
void vrna_pbacktrack_mem_free (
    vrna_pbacktrack_mem_t s )
#include <ViennaRNA/boltzmann_sampling.h>
Release memory occupied by a Boltzmann sampling memory data structure.
```

**See also**

[vrna\\_pbacktrack\\_mem\\_t](#), [vrna\\_pbacktrack5\\_resume\(\)](#), [vrna\\_pbacktrack5\\_resume\\_cb\(\)](#), [vrna\\_pbacktrack\\_resume\(\)](#),  
[vrna\\_pbacktrack\\_resume\\_cb\(\)](#)

**Parameters**

|          |  |
|----------|--|
| <i>s</i> | The non-redundancy memory data structure |
|----------|--|

**16.26 Compute the Structure with Maximum Expected Accuracy (MEA)****16.26.1 Detailed Description**

Collaboration diagram for Compute the Structure with Maximum Expected Accuracy (MEA):

## Functions

- `char * vrna_MEAs (vrna_fold_compound_t *fc, double gamma, float *mea)`  
`Compute a MEA (maximum expected accuracy) structure.`
- `char * vrna_MEAs_from_plist (vrna_ep_t *plist, const char *sequence, double gamma, vrna_md_t *md, float *mea)`  
`Compute a MEA (maximum expected accuracy) structure from a list of probabilities.`
- `float MEAs (plist *p, char *structure, double gamma)`  
`Computes a MEA (maximum expected accuracy) structure.`

### 16.26.2 Function Documentation

#### 16.26.2.1 vrna\_MEAs()

```
char * vrna_MEAs (
    vrna_fold_compound_t * fc,
    double gamma,
    float * mea )
#include <ViennaRNA/MEA.h>
Compute a MEA (maximum expected accuracy) structure.
The algorithm maximizes the expected accuracy
```

$$A(S) = \sum_{(i,j) \in S} 2\gamma p_{ij} + \sum_{i \notin S} p_i^u$$

Higher values of  $\gamma$  result in more base pairs of lower probability and thus higher sensitivity. Low values of  $\gamma$  result in structures containing only highly likely pairs (high specificity). The code of the MEA function also demonstrates the use of sparse dynamic programming scheme to reduce the time and memory complexity of folding.

#### Precondition

`vrna_pf()` must be executed on input parameter `fc`

#### Parameters

|                    |   |
|--------------------|---|
| <code>fc</code>    | The fold compound data structure with pre-filled base pair probability matrix |
| <code>gamma</code> | The weighting factor for base pairs vs. unpaired nucleotides                  |
| <code>mea</code>   | A pointer to a variable where the MEA value will be written to                |

#### Returns

An MEA structure (or NULL on any error)

**SWIG Wrapper Notes** This function is attached as overloaded method `MEA(gamma = 1.)` to objects of type `fold_compound`. Note, that it returns the MEA structure and MEA value as a tuple (`MEA_structure, MEA`)

#### 16.26.2.2 vrna\_MEAs\_from\_plist()

```
char * vrna_MEAs_from_plist (
    vrna_ep_t * plist,
    const char * sequence,
    double gamma,
    vrna_md_t * md,
    float * mea )
```

```
#include <ViennaRNA/MEA.h>
```

Compute a MEA (maximum expected accuracy) structure from a list of probabilities.

The algorithm maximizes the expected accuracy

$$A(S) = \sum_{(i,j) \in S} 2\gamma p_{ij} + \sum_{i \notin S} p_i^u$$

Higher values of  $\gamma$  result in more base pairs of lower probability and thus higher sensitivity. Low values of  $\gamma$  result in structures containing only highly likely pairs (high specificity). The code of the MEA function also demonstrates the use of sparse dynamic programming scheme to reduce the time and memory complexity of folding.

#### Note

The unpaired probabilities  $p_i^u = 1 - \sum_{j \neq i} p_{ij}$  are usually computed from the supplied pairing probabilities  $p_{ij}$  as stored in `plist` entries of type `VRNA_PLIST_TYPE_BASEPAIR`. To overwrite individual  $p_o^u$  values simply add entries with type `VRNA_PLIST_TYPE_UNPAIRED`

To include G-Quadruplex support, the corresponding field in `md` must be set.

#### Parameters

|                       |  |
|-----------------------|--|
| <code>plist</code>    | A list of base pair probabilities the MEA structure is computed from |
| <code>sequence</code> | The RNA sequence that corresponds to the list of probability values  |
| <code>gamma</code>    | The weighting factor for base pairs vs. unpaired nucleotides         |
| <code>md</code>       | A model details data structure (maybe NULL)                          |
| <code>mea</code>      | A pointer to a variable where the MEA value will be written to       |

#### Returns

An MEA structure (or NULL on any error)

**SWIG Wrapper Notes** This function is available as overloaded function `MEA_from_plist(gamma = 1., md = NULL)`. Note, that it returns the MEA structure and MEA value as a tuple (MEA\_structure, MEA)

### 16.26.2.3 MEA()

```
float MEA (
    plist * p,
    char * structure,
    double gamma )
#include <ViennaRNA/MEA.h>
```

Computes a MEA (maximum expected accuracy) structure.  
The algorithm maximizes the expected accuracy

$$A(S) = \sum_{(i,j) \in S} 2\gamma p_{ij} + \sum_{i \notin S} p_i^u$$

Higher values of  $\gamma$  result in more base pairs of lower probability and thus higher sensitivity. Low values of  $\gamma$  result in structures containing only highly likely pairs (high specificity). The code of the MEA function also demonstrates the use of sparse dynamic programming scheme to reduce the time and memory complexity of folding.

**Deprecated** Use `vrna_MEA()` or `vrna_MEA_from_plist()` instead!

## 16.27 Compute the Centroid Structure

### 16.27.1 Detailed Description

Collaboration diagram for Compute the Centroid Structure:

## Functions

- `char * vrna_centroid (vrna_fold_compound_t *vc, double *dist)`  
Get the centroid structure of the ensemble.
- `char * vrna_centroid_from plist (int length, double *dist, vrna_ep_t *pl)`  
Get the centroid structure of the ensemble.
- `char * vrna_centroid_from_probs (int length, double *dist, FLT_OR_DBL *probs)`  
Get the centroid structure of the ensemble.

### 16.27.2 Function Documentation

#### 16.27.2.1 vrna\_centroid()

```
char * vrna_centroid (
    vrna_fold_compound_t * vc,
    double * dist )
#include <ViennaRNA/centroid.h>
Get the centroid structure of the ensemble.
The centroid is the structure with the minimal average distance to all other structures
 $\langle d(S) \rangle = \sum_{(i,j) \in S} (1 - p_{ij}) + \sum_{(i,j) \notin S} p_{ij}$ 
Thus, the centroid is simply the structure containing all pairs with  $p_{ij} > 0.5$ . The distance of the centroid to the ensemble is written to the memory addressed by dist.
```

##### Parameters

|     |                   |   |
|-----|-------------------|---|
| in  | <code>vc</code>   | The fold compound data structure  |
| out | <code>dist</code> | A pointer to the distance variable where the centroid distance will be written to |

##### Returns

The centroid structure of the ensemble in dot-bracket notation (NULL on error)

#### 16.27.2.2 vrna\_centroid\_from plist()

```
char * vrna_centroid_from plist (
    int length,
    double * dist,
    vrna_ep_t * pl )
#include <ViennaRNA/centroid.h>
Get the centroid structure of the ensemble.
This function is a threadsafe replacement for centroid() with a vrna_ep_t input
The centroid is the structure with the minimal average distance to all other structures
 $\langle d(S) \rangle = \sum_{(i,j) \in S} (1 - p_{ij}) + \sum_{(i,j) \notin S} p_{ij}$ 
Thus, the centroid is simply the structure containing all pairs with  $p_{ij} > 0.5$ . The distance of the centroid to the ensemble is written to the memory addressed by dist.
```

##### Parameters

|     |                     |   |
|-----|---------------------|---|
| in  | <code>length</code> | The length of the sequence  |
| out | <code>dist</code>   | A pointer to the distance variable where the centroid distance will be written to |
| in  | <code>pl</code>     | A pair list containing base pair probability information about the ensemble       |

**Returns**

The centroid structure of the ensemble in dot-bracket notation (NULL on error)

**16.27.2.3 vrna\_centroid\_from\_probs()**

```
char * vrna_centroid_from_probs (
    int length,
    double * dist,
    FLT_OR_DBL * probs )
#include <ViennaRNA/centroid.h>
```

Get the centroid structure of the ensemble.

This function is a threadsafe replacement for [centroid\(\)](#) with a probability array input

The centroid is the structure with the minimal average distance to all other structures

$$\langle d(S) \rangle = \sum_{(i,j) \in S} (1 - p_{ij}) + \sum_{(i,j) \notin S} p_{ij}$$

Thus, the centroid is simply the structure containing all pairs with  $p_{ij} > 0.5$ . The distance of the centroid to the ensemble is written to the memory addressed by *dist*.

**Parameters**

|     |               |   |
|-----|---------------|---|
| in  | <i>length</i> | The length of the sequence  |
| out | <i>dist</i>   | A pointer to the distance variable where the centroid distance will be written to                               |
| in  | <i>probs</i>  | An upper triangular matrix containing base pair probabilities (access via <a href="#">vrna_idx_row_wise()</a> ) |

**Returns**

The centroid structure of the ensemble in dot-bracket notation (NULL on error)

**16.28 RNA-RNA Interaction****16.28.1 Detailed Description**

Collaboration diagram for RNA-RNA Interaction:

**Modules**

- [Partition Function for Two Hybridized Sequences](#)  
*Partition Function Cofolding.*
- [Partition Function for two Hybridized Sequences as a Stepwise Process](#)  
*RNA-RNA interaction as a stepwise process.*

**Files**

- file [concentrations.h](#)  
*Concentration computations for RNA-RNA interactions.*
- file [duplex.h](#)  
*Functions for simple RNA-RNA duplex interactions.*
- file [part\\_func\\_up.h](#)  
*Implementations for accessibility and RNA-RNA interaction as a stepwise process.*

**16.29 Classified Dynamic Programming Variants****16.29.1 Detailed Description**

Collaboration diagram for Classified Dynamic Programming Variants:

## Modules

- Distance Based Partitioning of the Secondary Structure Space
- Compute the Density of States

## 16.30 Distance Based Partitioning of the Secondary Structure Space

### 16.30.1 Detailed Description

Collaboration diagram for Distance Based Partitioning of the Secondary Structure Space:

## Modules

- Computing MFE representatives of a Distance Based Partitioning

*Compute the minimum free energy (MFE) and secondary structures for a partitioning of the secondary structure space according to the base pair distance to two fixed reference structures basepair distance to two fixed reference structures.*

- Computing Partition Functions of a Distance Based Partitioning

*Compute the partition function and stochastically sample secondary structures for a partitioning of the secondary structure space according to the base pair distance to two fixed reference structures.*

- Stochastic Backtracking of Structures from Distance Based Partitioning

*Contains functions related to stochastic backtracking from a specified distance class.*

## Files

- file `2Dfold.h`  
*MFE structures for base pair distance classes.*
- file `2Dpfold.h`  
*Partition function implementations for base pair distance classes.*

## 16.31 Computing MFE representatives of a Distance Based Partitioning

Compute the minimum free energy (MFE) and secondary structures for a partitioning of the secondary structure space according to the base pair distance to two fixed reference structures basepair distance to two fixed reference structures.

### 16.31.1 Detailed Description

Compute the minimum free energy (MFE) and secondary structures for a partitioning of the secondary structure space according to the base pair distance to two fixed reference structures basepair distance to two fixed reference structures.

#### See also

For further details, we refer to Lorenz et al. 2009 [15]

Collaboration diagram for Computing MFE representatives of a Distance Based Partitioning:

## Data Structures

- struct `vrna_sol_TwoD_t`  
*Solution element returned from `vrna_mfe_TwoD()` More...*
- struct `TwoDfold_vars`  
*Variables compound for 2Dfold MFE folding. More...*

## Typedefs

- **typedef struct vrna\_sol\_TwoD\_t vrna\_sol\_TwoD\_t**  
*Solution element returned from vrna\_mfe\_TwoD()*
- **typedef struct TwoDfold\_vars TwoDfold\_vars**  
*Variables compound for 2Dfold MFE folding.*

## Functions

- **vrna\_sol\_TwoD\_t \* vrna\_mfe\_TwoD (vrna\_fold\_compound\_t \*vc, int distance1, int distance2)**  
*Compute MFE's and representative for distance partitioning.*
- **char \* vrna\_backtrack5\_TwoD (vrna\_fold\_compound\_t \*vc, int k, int l, unsigned int j)**  
*Backtrack a minimum free energy structure from a 5' section of specified length.*
- **TwoDfold\_vars \* get\_TwoDfold\_variables (const char \*seq, const char \*structure1, const char \*structure2, int circ)**  
*Get a structure of type TwoDfold\_vars prefilled with current global settings.*
- **void destroy\_TwoDfold\_variables (TwoDfold\_vars \*our\_variables)**  
*Destroy a TwoDfold\_vars datastructure without memory loss.*
- **vrna\_sol\_TwoD\_t \* TwoDfoldList (TwoDfold\_vars \*vars, int distance1, int distance2)**  
*Compute MFE's and representative for distance partitioning.*
- **char \* TwoDfold\_backtrack\_f5 (unsigned int j, int k, int l, TwoDfold\_vars \*vars)**  
*Backtrack a minimum free energy structure from a 5' section of specified length.*

### 16.31.2 Data Structure Documentation

#### 16.31.2.1 struct vrna\_sol\_TwoD\_t

Solution element returned from [vrna\\_mfe\\_TwoD\(\)](#)

This element contains free energy and structure for the appropriate kappa (k), lambda (l) neighborhood. The data-structure contains two integer attributes 'k' and 'l' as well as an attribute 'en' of type float representing the free energy in kcal/mol and an attribute 's' of type char\* containing the secondary structure representative. A value of **INF** in k denotes the end of a list

See also

[vrna\\_mfe\\_TwoD\(\)](#)

#### Data Fields

- **int k**  
*Distance to first reference.*
- **int l**  
*Distance to second reference.*
- **float en**  
*Free energy in kcal/mol.*
- **char \* s**  
*MFE representative structure in dot-bracket notation.*

#### 16.31.2.2 struct TwoDfold\_vars

Variables compound for 2Dfold MFE folding.

**Deprecated** This data structure will be removed from the library soon! Use [vrna\\_fold\\_compound\\_t](#) and the corresponding functions [vrna\\_fold\\_compound\\_TwoD\(\)](#), [vrna\\_mfe\\_TwoD\(\)](#), and [vrna\\_fold\\_compound\\_free\(\)](#) instead!

Collaboration diagram for TwoDfold\_vars:

## Data Fields

- **vrna\_param\_t \* P**  
*Precomputed energy parameters and model details.*
- int **do\_backtrack**  
*Flag whether to do backtracing of the structure(s) or not.*
- char \* **ptype**  
*Precomputed array of pair types.*
- char \* **sequence**  
*The input sequence*
- short \* **S1**  
*The input sequences in numeric form.*
- unsigned int **maxD1**  
*Maximum allowed base pair distance to first reference.*
- unsigned int **maxD2**  
*Maximum allowed base pair distance to second reference.*
- unsigned int \* **mm1**  
*Maximum matching matrix, reference struct 1 disallowed.*
- unsigned int \* **mm2**  
*Maximum matching matrix, reference struct 2 disallowed.*
- int \* **my\_iindx**  
*Index for moving in quadratic distancy dimensions.*
- unsigned int \* **referenceBPs1**  
*Matrix containing number of basepairs of reference structure1 in interval [i,j].*
- unsigned int \* **referenceBPs2**  
*Matrix containing number of basepairs of reference structure2 in interval [i,j].*
- unsigned int \* **bpdist**  
*Matrix containing base pair distance of reference structure 1 and 2 on interval [i,j].*

### 16.31.3 Typedef Documentation

#### 16.31.3.1 vrna\_sol\_TwoD\_t

```
typedef struct vrna_sol_TwoD_t vrna_sol_TwoD_t
#include <ViennaRNA/2Dfold.h>
```

Solution element returned from [vrna\\_mfe\\_TwoD\(\)](#)

This element contains free energy and structure for the appropriate kappa (k), lambda (l) neighborhood. The data-structure contains two integer attributes 'k' and 'l' as well as an attribute 'en' of type float representing the free energy in kcal/mol and an attribute 's' of type char\* containing the secondary structure representative, A value of [INF](#) in k denotes the end of a list

See also

[vrna\\_mfe\\_TwoD\(\)](#)

#### 16.31.3.2 TwoDfold\_vars

```
typedef struct TwoDfold_vars TwoDfold_vars
#include <ViennaRNA/2Dfold.h>
```

Variables compound for 2Dfold MFE folding.

**Deprecated** This data structure will be removed from the library soon! Use [vrna\\_fold\\_compound\\_t](#) and the corresponding functions [vrna\\_fold\\_compound\\_TwoD\(\)](#), [vrna\\_mfe\\_TwoD\(\)](#), and [vrna\\_fold\\_compound\\_free\(\)](#) instead!

## 16.31.4 Function Documentation

### 16.31.4.1 vrna\_mfe\_TwoD()

```
vrna_sol_TwoD_t * vrna_mfe_TwoD (
    vrna_fold_compound_t * vc,
    int distance1,
    int distance2 )
#include <ViennaRNA/2Dfold.h>
```

Compute MFE's and representative for distance partitioning.

This function computes the minimum free energies and a representative secondary structure for each distance class according to the two references specified in the datastructure 'vars'. The maximum basepair distance to each of both references may be set by the arguments 'distance1' and 'distance2', respectively. If both distance arguments are set to '-1', no restriction is assumed and the calculation is performed for each distance class possible.

The returned list contains an entry for each distance class. If a maximum basepair distance to either of the references was passed, an entry with  $k=l=-1$  will be appended in the list, denoting the class where all structures exceeding the maximum will be thrown into. The end of the list is denoted by an attribute value of `INF` in the `k`-attribute of the list entry.

See also

[vrna\\_fold\\_compound\\_TwoD\(\)](#), [vrna\\_fold\\_compound\\_free\(\)](#), [vrna\\_pf\\_TwoD\(\)](#) [vrna\\_backtrack5\\_TwoD\(\)](#),  
[vrna\\_sol\\_TwoD\\_t](#), [vrna\\_fold\\_compound\\_t](#)

#### Parameters

|                        |   |
|------------------------|---|
| <code>vc</code>        | The datastructure containing all precomputed folding attributes |
| <code>distance1</code> | maximum distance to reference1 (-1 means no restriction)        |
| <code>distance2</code> | maximum distance to reference2 (-1 means no restriction)        |

#### Returns

A list of minimum free energies (and corresponding structures) for each distance class

### 16.31.4.2 vrna\_backtrack5\_TwoD()

```
char * vrna_backtrack5_TwoD (
    vrna_fold_compound_t * vc,
    int k,
    int l,
    unsigned int j )
#include <ViennaRNA/2Dfold.h>
```

Backtrack a minimum free energy structure from a 5' section of specified length.

This function allows one to backtrack a secondary structure beginning at the 5' end, a specified length and residing in a specific distance class. If the argument 'k' gets a value of -1, the structure that is backtracked is assumed to reside in the distance class where all structures exceeding the maximum basepair distance specified in [vrna\\_mfe\\_TwoD\(\)](#) belong to.

#### Note

The argument 'vars' must contain precalculated energy values in the energy matrices, i.e. a call to [vrna\\_mfe\\_TwoD\(\)](#) preceding this function is mandatory!

#### See also

[vrna\\_mfe\\_TwoD\(\)](#)

**Parameters**

|           |   |
|-----------|---|
| <i>vc</i> | The datastructure containing all precomputed folding attributes |
| <i>j</i>  | The length in nucleotides beginning from the 5' end             |
| <i>k</i>  | distance to reference1 (may be -1)                              |
| <i>l</i>  | distance to reference2  |

**16.31.4.3 get\_TwoDfold\_variables()**

```
TwoDfold_vars * get_TwoDfold_variables (
    const char * seq,
    const char * structure1,
    const char * structure2,
    int circ )
#include <ViennaRNA/2Dfold.h>
```

Get a structure of type `TwoDfold_vars` prefilled with current global settings.

This function returns a datastructure of type `TwoDfold_vars`. The data fields inside the `TwoDfold_vars` are prefilled by global settings and all memory allocations necessary to start a computation are already done for the convenience of the user

**Note**

Make sure that the reference structures are compatible with the sequence according to Watson-Crick- and Wobble-base pairing

**Deprecated** Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_TwoD()`, `vrna_mfe_TwoD()`, and `vrna_fold_compound_free()` instead!

**Parameters**

|                   |  |
|-------------------|--|
| <i>seq</i>        | The RNA sequence   |
| <i>structure1</i> | The first reference structure in dot-bracket notation                                      |
| <i>structure2</i> | The second reference structure in dot-bracket notation                                     |
| <i>circ</i>       | A switch to indicate the assumption to fold a circular instead of linear RNA (0=OFF, 1=ON) |

**Returns**

A datastructure prefilled with folding options and allocated memory

**16.31.4.4 destroy\_TwoDfold\_variables()**

```
void destroy_TwoDfold_variables (
    TwoDfold_vars * our_variables )
#include <ViennaRNA/2Dfold.h>
```

Destroy a `TwoDfold_vars` datastructure without memory loss.

This function free's all allocated memory that depends on the datastructure given.

**Deprecated** Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_TwoD()`, `vrna_mfe_TwoD()`, and `vrna_fold_compound_free()` instead!

**Parameters**

|                      |  |
|----------------------|--|
| <i>our_variables</i> | A pointer to the datastructure to be destroyed |
|----------------------|--|

#### 16.31.4.5 TwoDfoldList()

```
vrna_sol_TwoD_t * TwoDfoldList (
    TwoDfold_vars * vars,
    int distance1,
    int distance2 )
#include <ViennaRNA/2Dfold.h>
```

Compute MFE's and representative for distance partitioning.

This function computes the minimum free energies and a representative secondary structure for each distance class according to the two references specified in the datastructure 'vars'. The maximum basepair distance to each of both references may be set by the arguments 'distance1' and 'distance2', respectively. If both distance arguments are set to '-1', no restriction is assumed and the calculation is performed for each distance class possible.

The returned list contains an entry for each distance class. If a maximum basepair distance to either of the references was passed, an entry with  $k=l=-1$  will be appended in the list, denoting the class where all structures exceeding the maximum will be thrown into. The end of the list is denoted by an attribute value of [INF](#) in the  $k$ -attribute of the list entry.

**Deprecated** Use the new API that relies on [vrna\\_fold\\_compound\\_t](#) and the corresponding functions [vrna\\_fold\\_compound\\_TwoD\(\)](#), [vrna\\_mfe\\_TwoD\(\)](#), and [vrna\\_fold\\_compound\\_free\(\)](#) instead!

#### Parameters

|                  |  |
|------------------|--|
| <i>vars</i>      | the datastructure containing all predefined folding attributes |
| <i>distance1</i> | maximum distance to reference1 (-1 means no restriction)       |
| <i>distance2</i> | maximum distance to reference2 (-1 means no restriction)       |

#### 16.31.4.6 TwoDfold\_backtrack\_f5()

```
char * TwoDfold_backtrack_f5 (
    unsigned int j,
    int k,
    int l,
    TwoDfold_vars * vars )
#include <ViennaRNA/2Dfold.h>
```

Backtrack a minimum free energy structure from a 5' section of specified length.

This function allows one to backtrack a secondary structure beginning at the 5' end, a specified length and residing in a specific distance class. If the argument 'k' gets a value of -1, the structure that is backtracked is assumed to reside in the distance class where all structures exceeding the maximum basepair distance specified in TwoDfold() belong to.

#### Note

The argument 'vars' must contain precalculated energy values in the energy matrices, i.e. a call to TwoDfold() preceding this function is mandatory!

**Deprecated** Use the new API that relies on [vrna\\_fold\\_compound\\_t](#) and the corresponding functions [vrna\\_fold\\_compound\\_TwoD\(\)](#), [vrna\\_mfe\\_TwoD\(\)](#), [vrna\\_backtrack5\\_TwoD\(\)](#), and [vrna\\_fold\\_compound\\_free\(\)](#) instead!

#### Parameters

|             |  |
|-------------|--|
| <i>j</i>    | The length in nucleotides beginning from the 5' end            |
| <i>k</i>    | distance to reference1 (may be -1)                             |
| <i>l</i>    | distance to reference2   |
| <i>vars</i> | the datastructure containing all predefined folding attributes |

## 16.32 Computing Partition Functions of a Distance Based Partitioning

Compute the partition function and stochastically sample secondary structures for a partitioning of the secondary structure space according to the base pair distance to two fixed reference structures.

### 16.32.1 Detailed Description

Compute the partition function and stochastically sample secondary structures for a partitioning of the secondary structure space according to the base pair distance to two fixed reference structures.

Collaboration diagram for Computing Partition Functions of a Distance Based Partitioning:

### Data Structures

- struct [vrna\\_sol\\_TwoD\\_pf\\_t](#)  
*Solution element returned from [vrna\\_pf\\_TwoD\(\)](#) More...*

### Typedefs

- typedef struct [vrna\\_sol\\_TwoD\\_pf\\_t](#) [vrna\\_sol\\_TwoD\\_pf\\_t](#)  
*Solution element returned from [vrna\\_pf\\_TwoD\(\)](#)*

### Functions

- [vrna\\_sol\\_TwoD\\_pf\\_t \\* vrna\\_pf\\_TwoD \(\[vrna\\\_fold\\\_compound\\\_t\]\(#\) \\*vc, int maxDistance1, int maxDistance2\)](#)  
*Compute the partition function for all distance classes.*

### 16.32.2 Data Structure Documentation

#### 16.32.2.1 struct [vrna\\_sol\\_TwoD\\_pf\\_t](#)

Solution element returned from [vrna\\_pf\\_TwoD\(\)](#)

This element contains the partition function for the appropriate kappa (k), lambda (l) neighborhood The datastructure contains two integer attributes 'k' and 'l' as well as an attribute 'q' of type [FLT\\_OR\\_DBL](#)

A value of [INF](#) in k denotes the end of a list

See also

[vrna\\_pf\\_TwoD\(\)](#)

#### Data Fields

- int **k**  
*Distance to first reference.*
- int **l**  
*Distance to second reference.*
- [FLT\\_OR\\_DBL](#) **q**  
*partition function*

### 16.32.3 Typedef Documentation

#### 16.32.3.1 [vrna\\_sol\\_TwoD\\_pf\\_t](#)

```
typedef struct vrna\_sol\_TwoD\_pf\_t vrna\_sol\_TwoD\_pf\_t
#include <ViennaRNA/2Dpfold.h>
```

Solution element returned from [vrna\\_pf\\_TwoD\(\)](#)

This element contains the partition function for the appropriate kappa (k), lambda (l) neighborhood The datastructure contains two integer attributes 'k' and 'l' as well as an attribute 'q' of type [FLT\\_OR\\_DBL](#)

A value of [INF](#) in k denotes the end of a list

See also

[vrna\\_pf\\_TwoD\(\)](#)

#### 16.32.4 Function Documentation

##### 16.32.4.1 vrna\_pf\_TwoD()

```
vrna_sol_TwoD_pf_t * vrna_pf_TwoD (
    vrna_fold_compound_t * vc,
    int maxDistance1,
    int maxDistance2 )
#include <ViennaRNA/2Dpfold.h>
```

Compute the partition function for all distance classes.

This function computes the partition functions for all distance classes according the two reference structures specified in the datastructure 'vars'. Similar to [vrna\\_mfe\\_TwoD\(\)](#) the arguments maxDistance1 and maxDistance2 specify the maximum distance to both reference structures. A value of '-1' in either of them makes the appropriate distance restrictionless, i.e. all basepair distancies to the reference are taken into account during computation. In case there is a restriction, the returned solution contains an entry where the attribute k=l=-1 contains the partition function for all structures exceeding the restriction. A value of [INF](#) in the attribute 'k' of the returned list denotes the end of the list

See also

[vrna\\_fold\\_compound\\_TwoD\(\)](#), [vrna\\_fold\\_compound\\_free\(\)](#), [vrna\\_fold\\_compound](#) [vrna\\_sol\\_TwoD\\_pf\\_t](#)

##### Parameters

|                     |  |
|---------------------|--|
| <i>vc</i>           | The datastructure containing all necessary folding attributes and matrices |
| <i>maxDistance1</i> | The maximum basepair distance to reference1 (may be -1)                    |
| <i>maxDistance2</i> | The maximum basepair distance to reference2 (may be -1)                    |

##### Returns

A list of partition funtions for the corresponding distance classes

## 16.33 Stochastic Backtracking of Structures from Distance Based Partitioning

Contains functions related to stochastic backtracking from a specified distance class.

### 16.33.1 Detailed Description

Contains functions related to stochastic backtracking from a specified distance class.

Collaboration diagram for Stochastic Backtracking of Structures from Distance Based Partitioning:

### Functions

- char \* [vrna\\_pbacktrack\\_TwoD](#) ([vrna\\_fold\\_compound\\_t](#) \*vc, int d1, int d2)
 

*Sample secondary structure representatives from a set of distance classes according to their Boltzmann probability.*
- char \* [vrna\\_pbacktrack5\\_TwoD](#) ([vrna\\_fold\\_compound\\_t](#) \*vc, int d1, int d2, unsigned int length)
 

*Sample secondary structure representatives with a specified length from a set of distance classes according to their Boltzmann probability.*

## 16.33.2 Function Documentation

### 16.33.2.1 vrna\_pbacktrack\_TwoD()

```
char * vrna_pbacktrack_TwoD (
    vrna_fold_compound_t * vc,
    int d1,
    int d2 )
```

```
#include <ViennaRNA/2Dpfold.h>
```

Sample secondary structure representatives from a set of distance classes according to their Boltzmann probability. If the argument 'd1' is set to '-1', the structure will be backtracked in the distance class where all structures exceeding the maximum basepair distance to either of the references reside.

#### Precondition

The argument 'vars' must contain precalculated partition function matrices, i.e. a call to [vrna\\_pf\\_TwoD\(\)](#) preceding this function is mandatory!

#### See also

[vrna\\_pf\\_TwoD\(\)](#)

#### Parameters

|         |    |   |
|---------|----|---|
| in, out | vc | The <a href="#">vrna_fold_compound_t</a> datastructure containing all necessary folding attributes and matrices |
| in      | d1 | The distance to reference1 (may be -1)  |
| in      | d2 | The distance to reference2  |

#### Returns

A sampled secondary structure in dot-bracket notation

### 16.33.2.2 vrna\_pbacktrack5\_TwoD()

```
char * vrna_pbacktrack5_TwoD (
    vrna_fold_compound_t * vc,
    int d1,
    int d2,
    unsigned int length )
```

```
#include <ViennaRNA/2Dpfold.h>
```

Sample secondary structure representatives with a specified length from a set of distance classes according to their Boltzmann probability.

This function does essentially the same as [vrna\\_pbacktrack\\_TwoD\(\)](#) with the only difference that partial structures, i.e. structures beginning from the 5' end with a specified length of the sequence, are backtracked

#### Note

This function does not work (since it makes no sense) for circular RNA sequences!

#### Precondition

The argument 'vars' must contain precalculated partition function matrices, i.e. a call to [vrna\\_pf\\_TwoD\(\)](#) preceding this function is mandatory!

See also

[vrna\\_pbacktrack\\_TwoD\(\)](#), [vrna\\_pf\\_TwoD\(\)](#)

#### Parameters

|         |               |   |
|---------|---------------|---|
| in, out | <i>vc</i>     | The <a href="#">vrna_fold_compound_t</a> datastructure containing all necessary folding attributes and matrices |
| in      | <i>d1</i>     | The distance to reference1 (may be -1)  |
| in      | <i>d2</i>     | The distance to reference2  |
| in      | <i>length</i> | The length of the structure beginning from the 5' end   |

#### Returns

A sampled secondary structure in dot-bracket notation

## 16.34 Predicting various thermodynamic properties

Compute various thermodynamic properties using the partition function.

### 16.34.1 Detailed Description

Compute various thermodynamic properties using the partition function.

Many thermodynamic properties can be derived from the partition function

$$Q = \sum_{s \in \omega} e^{\frac{-E(s)}{kT}}.$$

In particular, for nucleic acids in equilibrium the probability  $p(F)$  of a particular structural feature  $F$  follows Boltzmann's law, i.e.

$$p(F) \propto \sum_{s|F \in s} e^{\frac{-E(s)}{kT}}.$$

The actual probabilities can then be obtained from the ratio of those structures containing  $F$  and *all* structures, i.e.

$$p(F) = \frac{1}{Q} \sum_{s|F \in s} e^{\frac{-E(s)}{kT}}.$$

Consequently, a particular secondary structure  $s$  has equilibrium probability

$$p(s) = \frac{1}{Q} e^{\frac{-E(s)}{kT}}$$

which can be easily computed once  $Q$  and  $E(s)$  are known.

On the other hand, efficient dynamic programming algorithms exist to compute the equilibrium probabilities

$$p_{ij} = \frac{1}{Q} \sum_{s|(i,j) \in s} e^{\frac{-E(s)}{kT}}$$

of base pairs  $(i, j)$  without the need for exhaustive enumeration of  $s$ .

This interface provides the functions for all thermodynamic property computations implemented in *RNALib*. Collaboration diagram for Predicting various thermodynamic properties:

## Files

- file [equilibrium\\_probs.h](#)  
*Equilibrium Probability implementations.*
- file [heat\\_capacity.h](#)  
*Compute heat capacity for an RNA.*

## Data Structures

- struct `vrna_heat_capacity_s`  
*A single result from heat capacity computations.* [More...](#)

## Typedefs

- `typedef void() vrna_heat_capacity_callback(float temp, float heat_capacity, void *data)`  
*The callback for heat capacity predictions.*
- `typedef struct vrna_heat_capacity_s vrna_heat_capacity_t`  
*A single result from heat capacity computations.*

## Base pair probabilities and derived computations

- `int vrna_pairing_probs (vrna_fold_compound_t *vc, char *structure)`
- `double vrna_mean_bp_distance_pr (int length, FLT_OR_DBL *pr)`  
*Get the mean base pair distance in the thermodynamic ensemble from a probability matrix.*
- `double vrna_mean_bp_distance (vrna_fold_compound_t *vc)`  
*Get the mean base pair distance in the thermodynamic ensemble.*
- `double vrna_ensemble_defect_pt (vrna_fold_compound_t *fc, const short *pt)`  
*Compute the Ensemble Defect for a given target structure provided as a `vrna_ptable`.*
- `double vrna_ensemble_defect (vrna_fold_compound_t *fc, const char *structure)`  
*Compute the Ensemble Defect for a given target structure.*
- `double * vrna_positional_entropy (vrna_fold_compound_t *fc)`  
*Compute a vector of positional entropies.*
- `vrna_ep_t * vrna_stack_prob (vrna_fold_compound_t *vc, double cutoff)`  
*Compute stacking probabilities.*

## Multimer probabilities computations

- `void vrna_pf_dimer_probs (double FAB, double FA, double FB, vrna_ep_t *prAB, const vrna_ep_t *prA, const vrna_ep_t *prB, int Alength, const vrna_exp_param_t *exp_params)`  
*Compute Boltzmann probabilities of dimerization without homodimers.*

## Structure probability computations

- `double vrna_pr_structure (vrna_fold_compound_t *fc, const char *structure)`  
*Compute the equilibrium probability of a particular secondary structure.*
- `double vrna_pr_energy (vrna_fold_compound_t *vc, double e)`

## Basic heat capacity function interface

- `vrna_heat_capacity_t * vrna_heat_capacity (vrna_fold_compound_t *fc, float T_min, float T_max, float T_increment, unsigned int mpoints)`  
*Compute the specific heat for an RNA.*
- `int vrna_heat_capacity_cb (vrna_fold_compound_t *fc, float T_min, float T_max, float T_increment, unsigned int mpoints, vrna_heat_capacity_callback *cb, void *data)`  
*Compute the specific heat for an RNA (callback variant)*

## Simplified heat capacity computation

- `vrna_heat_capacity_t * vrna_heat_capacity_simple (const char *sequence, float T_min, float T_max, float T_increment, unsigned int mpoints)`  
*Compute the specific heat for an RNA (simplified variant)*

## 16.34.2 Data Structure Documentation

### 16.34.2.1 struct vrna\_heat\_capacity\_s

A single result from heat capacity computations.

See also

[vrna\\_heat\\_capacity\(\)](#)

#### Data Fields

- float **temperature**  
*The temperature in °C.*
- float **heat\_capacity**  
*The specific heat at this temperature in Kcal/(Mol \* K)*

## 16.34.3 Typedef Documentation

### 16.34.3.1 vrna\_heat\_capacity\_callback

```
typedef void() vrna_heat_capacity_callback(float temp, float heat_capacity, void *data)
#include <ViennaRNA/heat_capacity.h>
```

The callback for heat capacity predictions.

**Notes on Callback Functions** This function will be called for each evaluated temperature in the heat capacity prediction.

See also

[vrna\\_heat\\_capacity\\_cb\(\)](#)

#### Parameters

|                      |   |
|----------------------|---|
| <i>temp</i>          | The current temperature this results corresponds to in °C                         |
| <i>heat_capacity</i> | The heat capacity in Kcal/(Mol * K)   |
| <i>data</i>          | Some arbitrary data pointer passed through by the function executing the callback |

### 16.34.3.2 vrna\_heat\_capacity\_t

```
typedef struct vrna_heat_capacity_s vrna_heat_capacity_t
#include <ViennaRNA/heat_capacity.h>
```

A single result from heat capacity computations.

This is a convenience typedef for [vrna\\_heat\\_capacity\\_s](#), i.e. results as obtained from [vrna\\_heat\\_capacity\(\)](#)

## 16.34.4 Function Documentation

### 16.34.4.1 vrna\_mean\_bp\_distance\_pr()

```
double vrna_mean_bp_distance_pr (
    int length,
    FLT_OR_DBL * pr )
#include <ViennaRNA/equilibrium_probs.h>
```

Get the mean base pair distance in the thermodynamic ensemble from a probability matrix.

$$\langle d \rangle = \sum_{a,b} p_a p_b d(S_a, S_b)$$

this can be computed from the pair probs  $p_{ij}$  as

$$\langle d \rangle = \sum_{ij} p_{ij}(1 - p_{ij})$$

#### Parameters

|               |   |
|---------------|---|
| <i>length</i> | The length of the sequence                        |
| <i>pr</i>     | The matrix containing the base pair probabilities |

#### Returns

The mean pair distance of the structure ensemble

#### 16.34.4.2 vrna\_mean\_bp\_distance()

```
double vrna_mean_bp_distance (
    vrna_fold_compound_t * vc )
#include <ViennaRNA/equilibrium_probs.h>
Get the mean base pair distance in the thermodynamic ensemble.
```

$$\langle d \rangle = \sum_{a,b} p_a p_b d(S_a, S_b)$$

this can be computed from the pair probs  $p_{ij}$  as

$$\langle d \rangle = \sum_{ij} p_{ij}(1 - p_{ij})$$

#### Parameters

|           |                                  |
|-----------|----------------------------------|
| <i>vc</i> | The fold compound data structure |
|-----------|----------------------------------|

#### Returns

The mean pair distance of the structure ensemble

**SWIG Wrapper Notes** This function is attached as method [mean\\_bp\\_distance\(\)](#) to objects of type *fold\_compound*

#### 16.34.4.3 vrna\_ensemble\_defect\_pt()

```
double vrna_ensemble_defect_pt (
    vrna_fold_compound_t * fc,
    const short * pt )
#include <ViennaRNA/equilibrium_probs.h>
```

Compute the Ensemble Defect for a given target structure provided as a **vrna\_ptable**.

Given a target structure  $s$ , compute the average dissimilarity of a randomly drawn structure from the ensemble, i.e.:

$$ED(s) = 1 - \frac{1}{n} \sum_{ij,(i,j) \in s} p_{ij} - \frac{1}{n} \sum_i (1 - s_i) q_i$$

with sequence length  $n$ , the probability  $p_{ij}$  of a base pair  $(i, j)$ , the probability  $q_i = 1 - \sum_j p_{ij}$  of nucleotide  $i$  being unpaired, and the indicator variable  $s_i = 1$  if  $\exists(i, j) \in s$ , and  $s_i = 0$  otherwise.

#### Precondition

The `vrna_fold_compound_t` input parameter `fc` must contain a valid base pair probability matrix. This means that partition function and base pair probabilities must have been computed using `fc` before execution of this function!

#### See also

`vrna_pf()`, `vrna_pairing_probs()`, `vrna_ensemble_defect()`

#### Parameters

|                 |   |
|-----------------|---|
| <code>fc</code> | A fold_compound with pre-computed base pair probabilities |
| <code>pt</code> | A pair table representing a target structure              |

#### Returns

The ensemble defect with respect to the target structure, or -1. upon failure, e.g. pre-conditions are not met

**SWIG Wrapper Notes** This function is attached as overloaded method `ensemble_defect()` to objects of type `fold_compound`.

#### 16.34.4.4 `vrna_ensemble_defect()`

```
double vrna_ensemble_defect (
    vrna_fold_compound_t * fc,
    const char * structure )
#include <ViennaRNA/equilibrium_probs.h>
```

Compute the Ensemble Defect for a given target structure.

This is a wrapper around `vrna_ensemble_defect_pt()`. Given a target structure  $s$ , compute the average dissimilarity of a randomly drawn structure from the ensemble, i.e.:

$$ED(s) = 1 - \frac{1}{n} \sum_{ij, (i,j) \in s} p_{ij} - \frac{1}{n} \sum_i (1 - s_i) q_i$$

with sequence length  $n$ , the probability  $p_{ij}$  of a base pair  $(i, j)$ , the probability  $q_i = 1 - \sum_j p_{ij}$  of nucleotide  $i$  being unpaired, and the indicator variable  $s_i = 1$  if  $\exists(i, j) \in s$ , and  $s_i = 0$  otherwise.

#### Precondition

The `vrna_fold_compound_t` input parameter `fc` must contain a valid base pair probability matrix. This means that partition function and base pair probabilities must have been computed using `fc` before execution of this function!

#### See also

`vrna_pf()`, `vrna_pairing_probs()`, `vrna_ensemble_defect_pt()`

#### Parameters

|                        |   |
|------------------------|---|
| <code>fc</code>        | A fold_compound with pre-computed base pair probabilities |
| <code>structure</code> | A target structure in dot-bracket notation                |

**Returns**

The ensemble defect with respect to the target structure, or -1. upon failure, e.g. pre-conditions are not met

**SWIG Wrapper Notes** This function is attached as method `ensemble_defect()` to objects of type `fold_compound`. Note that the SWIG wrapper takes a structure in dot-bracket notation and converts it into a pair table using `vrna_ptable_from_string()`. The resulting pair table is then internally passed to `vrna_ensemble_defect_pt()`. To control which kind of matching brackets will be used during conversion, the optional argument `options` can be used. See also the description of `vrna_ptable_from_string()` for available options. (default: `VRNA_BRACKETS_RND`).

**16.34.4.5 vrna\_positional\_entropy()**

```
double * vrna_positional_entropy (
    vrna_fold_compound_t * fc )
#include <ViennaRNA/equilibrium_probs.h>
```

Compute a vector of positional entropies.

This function computes the positional entropies from base pair probabilities as

$$S(i) = - \sum_j p_{ij} \log(p_{ij}) - q_i \log(q_i)$$

with unpaired probabilities  $q_i = 1 - \sum_j p_{ij}$ .

Low entropy regions have little structural flexibility and the reliability of the predicted structure is high. High entropy implies many structural alternatives. While these alternatives may be functionally important, they make structure prediction more difficult and thus less reliable.

**Precondition**

This function requires pre-computed base pair probabilities! Thus, `vrna_pf()` must be called beforehand.

**Parameters**

|                 |   |
|-----------------|---|
| <code>fc</code> | A fold_compound with pre-computed base pair probabilities |
|-----------------|---|

**Returns**

A 1-based vector of positional entropies  $S(i)$ . (position 0 contains the sequence length)

**SWIG Wrapper Notes** This function is attached as method `positional_entropy()` to objects of type `fold_compound`

**16.34.4.6 vrna\_stack\_prob()**

```
vrna_ep_t * vrna_stack_prob (
    vrna_fold_compound_t * vc,
    double cutoff )
#include <ViennaRNA/equilibrium_probs.h>
```

Compute stacking probabilities.

For each possible base pair  $(i, j)$ , compute the probability of a stack  $(i, j), (i + 1, j - 1)$ .

**Parameters**

|                     |  |
|---------------------|--|
| <code>vc</code>     | The fold compound data structure with precomputed base pair probabilities  |
| <code>cutoff</code> | A cutoff value that limits the output to stacks with $p > \text{cutoff}$ . |

**Returns**

A list of stacks with enclosing base pair  $(i, j)$  and probability  $p$

**16.34.4.7 vrna\_pf\_dimer\_probs()**

```
void vrna_pf_dimer_probs (
    double FAB,
    double FA,
    double FB,
    vrna_ep_t * prAB,
    const vrna_ep_t * prA,
    const vrna_ep_t * prB,
    int Alength,
    const vrna_exp_param_t * exp_params )
#include <ViennaRNA/equilibrium_probs.h>
```

Compute Boltzmann probabilities of dimerization without homodimers.

Given the pair probabilities and free energies (in the null model) for a dimer AB and the two constituent monomers A and B, compute the conditional pair probabilities given that a dimer AB actually forms. Null model pair probabilities are given as a list as produced by [vrna plist from probs\(\)](#), the dimer probabilities 'prAB' are modified in place.

**Parameters**

|                   |                                   |
|-------------------|-----------------------------------|
| <i>FAB</i>        | free energy of dimer AB           |
| <i>FA</i>         | free energy of monomer A          |
| <i>FB</i>         | free energy of monomer B          |
| <i>prAB</i>       | pair probabilities for dimer      |
| <i>prA</i>        | pair probabilities monomer        |
| <i>prB</i>        | pair probabilities monomer        |
| <i>Alength</i>    | Length of molecule A              |
| <i>exp_params</i> | The precomputed Boltzmann factors |

**16.34.4.8 vrna\_pr\_structure()**

```
double vrna_pr_structure (
    vrna_fold_compound_t * fc,
    const char * structure )
#include <ViennaRNA/equilibrium_probs.h>
```

Compute the equilibrium probability of a particular secondary structure.

The probability  $p(s)$  of a particular secondary structure  $s$  can be computed as

$$p(s) = \frac{\exp(-\beta E(s))}{Z}$$

from the structures free energy  $E(s)$  and the partition function

$$Z = \sum_s \exp(-\beta E(s)), \quad \text{with } \beta = \frac{1}{RT}$$

where  $R$  is the gas constant and  $T$  the thermodynamic temperature.

**Precondition**

The fold compound `fc` must have went through a call to [vrna\\_pf\(\)](#) to fill the dynamic programming matrices with the corresponding partition function.

**Parameters**

|                  |  |
|------------------|--|
| <i>fc</i>        | The fold compound data structure with precomputed partition function           |
| <i>structure</i> | The secondary structure to compute the probability for in dot-bracket notation |

**Returns**

The probability of the input structure (range [0 : 1])

**SWIG Wrapper Notes** This function is attached as method **pr\_structure()** to objects of type *fold\_compound*

**16.34.4.9 vrna\_pr\_energy()**

```
double vrna_pr_energy (
    vrna_fold_compound_t * fc,
    double e )
#include <ViennaRNA/equilibrium_probs.h>
```

**SWIG Wrapper Notes** This function is attached as method **pr\_energy()** to objects of type *fold\_compound*

**16.34.4.10 vrna\_heat\_capacity()**

```
vrna_heat_capacity_t * vrna_heat_capacity (
    vrna_fold_compound_t * fc,
    float T_min,
    float T_max,
    float T_increment,
    unsigned int mpoints )
#include <ViennaRNA/heat_capacity.h>
```

Compute the specific heat for an RNA.

This function computes an RNAs specific heat in a given temperature range from the partition function by numeric differentiation. The result is returned as a list of pairs of temperature in °C and specific heat in Kcal/(Mol\*K).

Users can specify the temperature range for the computation from *T\_min* to *T\_max*, as well as the increment step size *T\_increment*. The latter also determines how many times the partition function is computed. Finally, the parameter *mpoints* determines how smooth the curve should be. The algorithm itself fits a parabola to  $2 \cdot mpoints + 1$  data points to calculate 2nd derivatives. Increasing this parameter produces a smoother curve.

**See also**

[vrna\\_heat\\_capacity\\_cb\(\)](#), [vrna\\_heat\\_capacity\\_t](#), [vrna\\_heat\\_capacity\\_s](#)

**Parameters**

|                    |   |
|--------------------|---|
| <i>fc</i>          | The <a href="#">vrna_fold_compound_t</a> with the RNA sequence to analyze   |
| <i>T_min</i>       | Lowest temperature in °C  |
| <i>T_max</i>       | Highest temperature in °C   |
| <i>T_increment</i> | Stepsize for temperature incrementation in °C (a reasonable choice might be 1 °C)                                 |
| <i>mpoints</i>     | The number of interpolation points to calculate 2nd derivative (a reasonable choice might be 2, min: 1, max: 100) |

**Returns**

A list of pairs of temperatures and corresponding heat capacity or *NULL* upon any failure. The last entry of the list is indicated by a **temperature** field set to a value smaller than *T\_min*

**SWIG Wrapper Notes** This function is attached as overloaded method **heat\_capacity()** to objects of type *fold\_compound*. If the optional function arguments *T\_min*, *T\_max*, *T\_increment*, and *mpoints* are omitted, they default to 0.0, 100.0, 1.0 and 2, respectively.

#### 16.34.4.11 vrna\_heat\_capacity\_cb()

```
int vrna_heat_capacity_cb (
    vrna_fold_compound_t * fc,
    float T_min,
    float T_max,
    float T_increment,
    unsigned int mpoints,
    vrna_heat_capacity_callback * cb,
    void * data )
```

#include <ViennaRNA/heat\_capacity.h>

Compute the specific heat for an RNA (callback variant)

Similar to [vrna\\_heat\\_capacity\(\)](#), this function computes an RNAs specific heat in a given temperature range from the partition function by numeric differentiation. Instead of returning a list of temperature/specific heat pairs, however, this function returns the individual results through a callback mechanism. The provided function will be called for each result and passed the corresponding temperature and specific heat values along with the arbitrary data as provided through the *data* pointer argument.

Users can specify the temperature range for the computation from *T\_min* to *T\_max*, as well as the increment step size *T\_increment*. The latter also determines how many times the partition function is computed. Finally, the parameter *mpoints* determines how smooth the curve should be. The algorithm itself fits a parabola to  $2 \cdot mpoints + 1$  data points to calculate 2nd derivatives. Increasing this parameter produces a smoother curve.

##### See also

[vrna\\_heat\\_capacity\(\)](#), [vrna\\_heat\\_capacity\\_callback](#)

##### Parameters

|                    |   |
|--------------------|---|
| <i>fc</i>          | The <a href="#">vrna_fold_compound_t</a> with the RNA sequence to analyze   |
| <i>T_min</i>       | Lowest temperature in °C  |
| <i>T_max</i>       | Highest temperature in °C   |
| <i>T_increment</i> | Stepsize for temperature incrementation in °C (a reasonable choice might be 1 °C)                                 |
| <i>mpoints</i>     | The number of interpolation points to calculate 2nd derivative (a reasonable choice might be 2, min: 1, max: 100) |
| <i>cb</i>          | The user-defined callback function that receives the individual results   |
| <i>data</i>        | An arbitrary data structure that will be passed to the callback in conjunction with the results                   |

##### Returns

Returns 0 upon failure, and non-zero otherwise

**SWIG Wrapper Notes** This function is attached as method **heat\_capacity\_cb()** to objects of type *fold\_compound*

#### 16.34.4.12 vrna\_heat\_capacity\_simple()

```
vrna_heat_capacity_t * vrna_heat_capacity_simple (
    const char * sequence,
    float T_min,
    float T_max,
    float T_increment,
    unsigned int mpoints )
```

```
#include <ViennaRNA/heat_capacity.h>
```

Compute the specific heat for an RNA (simplified variant)

Similar to [vrna\\_heat\\_capacity\(\)](#), this function computes an RNAs specific heat in a given temperature range from the partition function by numeric differentiation. This simplified version, however, only requires the RNA sequence as input instead of a `vrna_fold_compound_t` data structure. The result is returned as a list of pairs of temperature in °C and specific heat in Kcal/(Mol\*K).

Users can specify the temperature range for the computation from `T_min` to `T_max`, as well as the increment step size `T_increment`. The latter also determines how many times the partition function is computed. Finally, the parameter `mpoints` determines how smooth the curve should be. The algorithm itself fits a parabola to  $2 \cdot mpoints + 1$  data points to calculate 2nd derivatives. Increasing this parameter produces a smoother curve.

#### See also

[vrna\\_heat\\_capacity\(\)](#), [vrna\\_heat\\_capacity\\_cb\(\)](#), [vrna\\_heat\\_capacity\\_t](#), [vrna\\_heat\\_capacity\\_s](#)

#### Parameters

|                          |   |
|--------------------------|---|
| <code>sequence</code>    | The RNA sequence input (must be uppercase)  |
| <code>T_min</code>       | Lowest temperature in °C  |
| <code>T_max</code>       | Highest temperature in °C   |
| <code>T_increment</code> | Stepsize for temperature incrementation in °C (a reasonable choice might be 1 °C)                                 |
| <code>mpoints</code>     | The number of interpolation points to calculate 2nd derivative (a reasonable choice might be 2, min: 1, max: 100) |

#### Returns

A list of pairs of temperatures and corresponding heat capacity or `NULL` upon any failure. The last entry of the list is indicated by a `temperature` field set to a value smaller than `T_min`.

**SWIG Wrapper Notes** This function is available as overloaded function `heat_capacity()`. If the optional function arguments `T_min`, `T_max`, `T_increment`, and `mpoints` are omitted, they default to 0.0, 100.0, 1.0 and 2, respectively.

## 16.35 Compute the Density of States

### 16.35.1 Detailed Description

Collaboration diagram for Compute the Density of States:

#### Variables

- int `density_of_states` [MAXDOS+1]

*The Density of States.*

### 16.35.2 Variable Documentation

#### 16.35.2.1 density\_of\_states

```
int density_of_states[MAXDOS+1]  [extern]
#include <ViennaRNA/subopt.h>
```

The Density of States.

This array contains the density of states for an RNA sequences after a call to `subopt_par()`, `subopt()` or `subopt_circ()`.

### Precondition

Call one of the functions [subopt\\_par\(\)](#), [subopt\(\)](#) or [subopt\\_circ\(\)](#) prior accessing the contents of this array

### See also

[subopt\\_par\(\)](#), [subopt\(\)](#), [subopt\\_circ\(\)](#)

## 16.36 Inverse Folding (Design)

RNA sequence design.

### 16.36.1 Detailed Description

RNA sequence design.

### Files

- file [inverse.h](#)

*Inverse folding routines.*

### Functions

- float [inverse\\_fold](#) (char \*start, const char \*target)  
*Find sequences with predefined structure.*
- float [inverse\\_pf\\_fold](#) (char \*start, const char \*target)  
*Find sequence that maximizes probability of a predefined structure.*

### Variables

- char \* [symbolset](#)  
*This global variable points to the allowed bases, initially "AUGC". It can be used to design sequences from reduced alphabets.*
- float [final\\_cost](#)
- int [give\\_up](#)
- int [inv\\_verbose](#)

### 16.36.2 Function Documentation

#### 16.36.2.1 [inverse\\_fold\(\)](#)

```
float inverse_fold (
    char * start,
    const char * target )
#include <ViennaRNA/inverse.h>
```

Find sequences with predefined structure.

This function searches for a sequence with minimum free energy structure provided in the parameter 'target', starting with sequence 'start'. It returns 0 if the search was successful, otherwise a structure distance in terms of the energy difference between the search result and the actual target 'target' is returned. The found sequence is returned in 'start'. If [give\\_up](#) is set to 1, the function will return as soon as it is clear that the search will be unsuccessful, this speeds up the algorithm if you are only interested in exact solutions.

#### Parameters

|               |  |
|---------------|--|
| <i>start</i>  | The start sequence                                     |
| <i>target</i> | The target secondary structure in dot-bracket notation |

**Returns**

The distance to the target in case a search was unsuccessful, 0 otherwise

**16.36.2.2 inverse\_pf\_fold()**

```
float inverse_pf_fold (
    char * start,
    const char * target )
#include <ViennaRNA/inverse.h>
```

Find sequence that maximizes probability of a predefined structure.

This function searches for a sequence with maximum probability to fold into the provided structure 'target' using the partition function algorithm. It returns  $-kT \cdot \log(p)$  where  $p$  is the frequency of 'target' in the ensemble of possible structures. This is usually much slower than [inverse\\_fold\(\)](#).

**Parameters**

|               |  |
|---------------|--|
| <i>start</i>  | The start sequence                                     |
| <i>target</i> | The target secondary structure in dot-bracket notation |

**Returns**

The distance to the target in case a search was unsuccessful, 0 otherwise

**16.36.3 Variable Documentation****16.36.3.1 final\_cost**

```
float final_cost [extern]
#include <ViennaRNA/inverse.h>
when to stop inverse\_pf\_fold\(\)
```

**16.36.3.2 give\_up**

```
int give_up [extern]
#include <ViennaRNA/inverse.h>
default 0: try to minimize structure distance even if no exact solution can be found
```

**16.36.3.3 inv\_verbose**

```
int inv_verbose [extern]
#include <ViennaRNA/inverse.h>
print out substructure on which inverse\_fold\(\) fails
```

**16.37 Neighborhood Relation and Move Sets for Secondary Structures**

Different functions to generate structural neighbors of a secondary structure according to a particular Move Set.

**16.37.1 Detailed Description**

Different functions to generate structural neighbors of a secondary structure according to a particular Move Set. This module contains methods to compute the neighbors of an RNA secondary structure. Neighbors of a given structure are all structures that differ in exactly one base pair. That means one can insert or delete base pairs in the given structure. These insertions and deletions of base pairs are usually called moves. A third move which is considered in these methods is a shift move. A shifted base pair has one stable position and one position that

changes. These moves are encoded as follows:

- insertion:  $(i, j)$  where  $i, j > 0$
- deletion:  $(i, j)$  where  $i, j < 0$   
 shift:  $(i, j)$  where either  $i > 0, j < 0$  or  $i < 0, j > 0$   
 The negative position of a shift indicates the position that has changed.

Example:

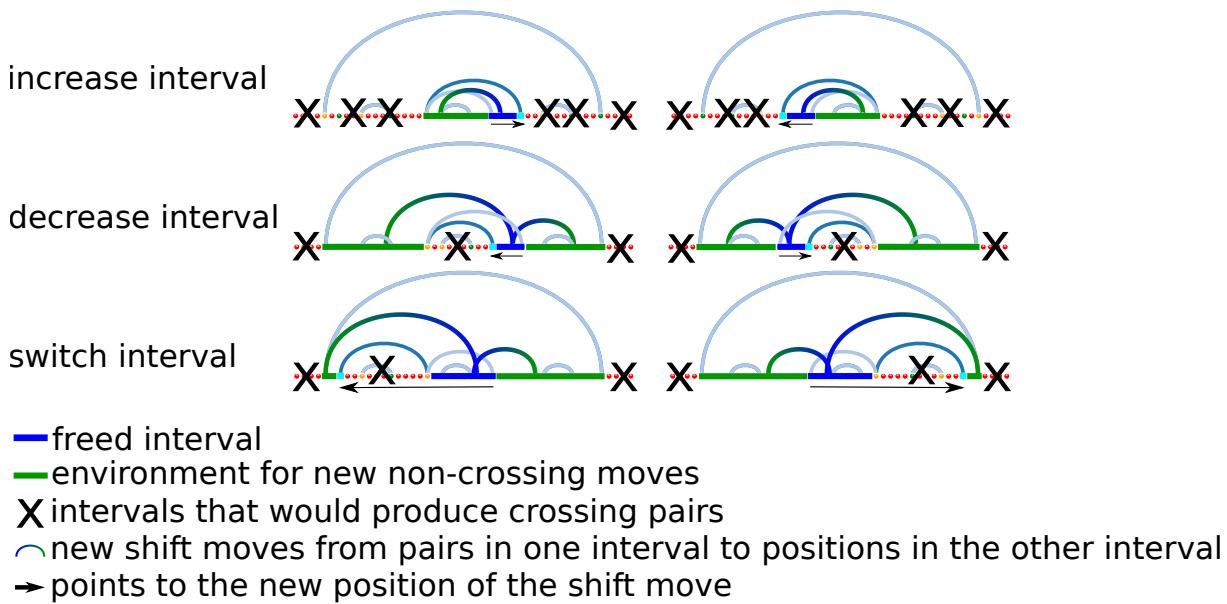
```
We have given a sequence and a structure.
Sequence AAGGAAACC
Structure ..(....)
Indices 123456789
The given base pair is (3,9) and the neighbors are the insertion (4, 8), the deletion (-3,-9), the
shift (3,-8)
and the shift (-4, 9).
This leads to the neighbored structures:
....(....)
.....
...(...).
....(...)
```

A simple method to construct all insertions is to iterate over the positions of a sequence twice. The first iteration has the index  $i$  in  $[1, \text{sequence length}]$ , the second iteration has the index  $j$  in  $[i+1, \text{sequence length}]$ . All pairs  $(i,j)$  with compatible letters and which are non-crossing with present base pairs are valid neighbored insertion moves. Valid deletion moves are all present base pairs with negative sign. Valid shift moves are constructed by taking all paired positions as fix position of a shift move and iterating over all positions of the sequence. If the letters of a position are compatible and if it the move is non-crossing with existing base pairs, we have a valid shift move. The method of generating shift moves can be accelerated by skipping neighbored base pairs.

If we need to construct all neighbors several times for subsequent moves, we can speed up the task by using the move set of the previous structure. The previous move set has to be filtered, such that all moves that would cross the next selected move are non-crossing. Next, the selected move has to be removed. Then one has to only to generate all moves that were not possible before. One move is the inverted selected move (if it was an insertion, simply make the indices negative). The generation of all other new moves is different and depends on the selected move. It is easy for an insertion move, because we have only to include all non-crossing shift moves, that are possible with the new base pair. For that we can either iterate over the sequence or we can select all crossing shift moves in the filter procedure and convert them into shifts.

The generation of new moves given a deletion is a little bit more complex, because we can create more moves. At first we can insert the deleted pair as insertion move. Then we generate all insertions that would have crossed the deleted base pair. Finally we construct all crossing shift moves.

If the given move is a shift, we can save much time by specifying the intervals for the generation of new moves. The interval which was enclosed by the positive position of the shift move and the previous paired position is the freed interval after applying the move. This freed interval includes all positions and base pairs that we need to construct new insertions and shifts. All these new moves have one position in the freed interval and the other position in the environment of the freed interval. The environment are all position which are outside the freed interval, but within the same enclosing loop of the shift move. The environment for valid base pairs can be divided into one or more intervals, depending on the shift move. The following examples describe a few scenarios to specify the intervals of the environment.



Given the intervals of the environment and the freed interval, the new shift moves can be constructed quickly. One has to take all positions of pairs from the environment in order to create valid pairs with positions in the freed interval. The same procedure can be applied for the other direction. This is taking all paired positions within the freed interval in order to look for pairs with valid positions in the intervals of the environment. Collaboration diagram for Neighborhood Relation and Move Sets for Secondary Structures:

## Files

- file `move.h`  
*Methods to operate with structural neighbors of RNA secondary structures.*
- file `neighbor.h`  
*Methods to compute the neighbors of an RNA secondary structure.*

## Data Structures

- struct `vrna_move_s`  
*An atomic representation of the transition / move from one structure to its neighbor.* [More...](#)

## Macros

- #define `VRNA_MOVESET_INSERTION` 4  
*Option flag indicating insertion move.*
- #define `VRNA_MOVESET_DELETION` 8  
*Option flag indicating deletion move.*
- #define `VRNA_MOVESET_SHIFT` 16  
*Option flag indicating shift move.*
- #define `VRNA_MOVESET_NO_LP` 32  
*Option flag indicating moves without lonely base pairs.*
- #define `VRNA_MOVESET_DEFAULT` (`VRNA_MOVESET_INSERTION` | `VRNA_MOVESET_DELETION`)  
*Option flag indicating default move set, i.e. insertions/deletion of a base pair.*
- #define `VRNA_NEIGHBOR_CHANGE` 1  
*State indicator for a neighbor that has been changed.*
- #define `VRNA_NEIGHBOR_INVALID` 2  
*State indicator for a neighbor that has been invalidated.*
- #define `VRNA_NEIGHBOR_NEW` 3  
*State indicator for a neighbor that has become newly available.*

## Typedefs

- `typedef struct vrna_move_s vrna_move_t`  
*A single move that transforms a secondary structure into one of its neighbors.*
- `typedef void() vrna_callback_move_update(vrna_fold_compound_t *fc, vrna_move_t neighbor, unsigned int state, void *data)`  
*Prototype of the neighborhood update callback.*

## Functions

- `vrna_move_t vrna_move_init (int pos_5, int pos_3)`  
*Create an atomic move.*
- `void vrna_move_list_free (vrna_move_t *moves)`
- `void vrna_move_apply (short *pt, const vrna_move_t *m)`  
*Apply a particular move / transition to a secondary structure, i.e. transform a structure.*
- `int vrna_move_is_removal (const vrna_move_t *m)`  
*Test whether a move is a base pair removal.*
- `int vrna_move_is_insertion (const vrna_move_t *m)`  
*Test whether a move is a base pair insertion.*
- `int vrna_move_is_shift (const vrna_move_t *m)`  
*Test whether a move is a base pair shift.*
- `int vrna_move_compare (const vrna_move_t *a, const vrna_move_t *b, const short *pt)`  
*Compare two moves.*
- `void vrna_loopidx_update (int *loopidx, const short *pt, int length, const vrna_move_t *m)`  
*Alters the loopIndices array that was constructed with `vrna_loopidx_from_ptable()`.*
- `vrna_move_t * vrna_neighbors (vrna_fold_compound_t *vc, const short *pt, unsigned int options)`  
*Generate neighbors of a secondary structure.*
- `vrna_move_t * vrna_neighbors_successive (const vrna_fold_compound_t *vc, const vrna_move_t *curr←_move, const short *prev_pt, const vrna_move_t *prev_neighbors, int size_prev_neighbors, int *size←_neighbors, unsigned int options)`  
*Generate neighbors of a secondary structure (the fast way)*
- `int vrna_move_neighbor_diff_cb (vrna_fold_compound_t *fc, short *ptable, vrna_move_t move, vrna_callback_move_update *cb, void *data, unsigned int options)`  
*Apply a move to a secondary structure and indicate which neighbors have changed consequentially.*
- `vrna_move_t * vrna_move_neighbor_diff (vrna_fold_compound_t *fc, short *ptable, vrna_move_t move, vrna_move_t **invalid_moves, unsigned int options)`  
*Apply a move to a secondary structure and indicate which neighbors have changed consequentially.*

### 16.37.2 Data Structure Documentation

#### 16.37.2.1 struct vrna\_move\_s

An atomic representation of the transition / move from one structure to its neighbor.  
An atomic transition / move may be one of the following:

- a **base pair insertion**,
- a **base pair removal**, or
- a **base pair shift** where an existing base pair changes one of its pairing partner.

These moves are encoded by two integer values that represent the affected 5' and 3' nucleotide positions. Furthermore, we use the following convention on the signedness of these encodings:

- both values are positive for *insertion moves*
- both values are negative for *base pair removals*
- both values have different signedness for *shift moves*, where the positive value indicates the nucleotide that stays constant, and the others absolute value is the new pairing partner

**Note**

A value of 0 in either field is used as list-end indicator and doesn't represent any valid move.

Collaboration diagram for vrna\_move\_s:

**Data Fields**

- int **pos\_5**  
*The (absolute value of the) 5' position of a base pair, or any position of a shifted pair.*
- int **pos\_3**  
*The (absolute value of the) 3' position of a base pair, or any position of a shifted pair.*
- **vrna\_move\_t \* next**  
*The next base pair (if an elementary move changes more than one base pair), or NULL. Has to be terminated with move 0,0.*

**16.37.3 Macro Definition Documentation****16.37.3.1 VRNA\_MOVESET\_INSERTION**

```
#define VRNA_MOVESET_INSERTION 4
#include <ViennaRNA/landscape/move.h>
Option flag indicating insertion move.
```

See also

[vrna\\_neighbors\(\)](#), [vrna\\_neighbors\\_successive\(\)](#), [vrna\\_path\(\)](#)

**16.37.3.2 VRNA\_MOVESET\_DELETION**

```
#define VRNA_MOVESET_DELETION 8
#include <ViennaRNA/landscape/move.h>
Option flag indicating deletion move.
```

See also

[vrna\\_neighbors\(\)](#), [vrna\\_neighbors\\_successive\(\)](#), [vrna\\_path\(\)](#)

**16.37.3.3 VRNA\_MOVESET\_SHIFT**

```
#define VRNA_MOVESET_SHIFT 16
#include <ViennaRNA/landscape/move.h>
Option flag indicating shift move.
```

See also

[vrna\\_neighbors\(\)](#), [vrna\\_neighbors\\_successive\(\)](#), [vrna\\_path\(\)](#)

**16.37.3.4 VRNA\_MOVESET\_NO\_LP**

```
#define VRNA_MOVESET_NO_LP 32
#include <ViennaRNA/landscape/move.h>
Option flag indicating moves without lonely base pairs.
```

See also

[vrna\\_neighbors\(\)](#), [vrna\\_neighbors\\_successive\(\)](#), [vrna\\_path\(\)](#)

### 16.37.3.5 VRNA\_MOVESET\_DEFAULT

```
#define VRNA_MOVESET_DEFAULT (VRNA_MOVESET_INSERTION | VRNA_MOVESET_DELETION)
#include <ViennaRNA/landscape/move.h>
```

Option flag indicating default move set, i.e. insertions/deletion of a base pair.

See also

[vrna\\_neighbors\(\)](#), [vrna\\_neighbors\\_successive\(\)](#), [vrna\\_path\(\)](#)

### 16.37.3.6 VRNA\_NEIGHBOR\_CHANGE

```
#define VRNA_NEIGHBOR_CHANGE 1
#include <ViennaRNA/landscape/neighbor.h>
```

State indicator for a neighbor that has been changed.

See also

[vrna\\_move\\_neighbor\\_diff\\_cb\(\)](#)

### 16.37.3.7 VRNA\_NEIGHBOR\_INVALID

```
#define VRNA_NEIGHBOR_INVALID 2
#include <ViennaRNA/landscape/neighbor.h>
```

State indicator for a neighbor that has been invalidated.

See also

[vrna\\_move\\_neighbor\\_diff\\_cb\(\)](#)

### 16.37.3.8 VRNA\_NEIGHBOR\_NEW

```
#define VRNA_NEIGHBOR_NEW 3
#include <ViennaRNA/landscape/neighbor.h>
```

State indicator for a neighbor that has become newly available.

See also

[vrna\\_move\\_neighbor\\_diff\\_cb\(\)](#)

## 16.37.4 Typedef Documentation

### 16.37.4.1 vrna\_callback\_move\_update

```
typedef void() vrna_callback_move_update(vrna_fold_compound_t *fc, vrna_move_t neighbor, unsigned
int state, void *data)
#include <ViennaRNA/landscape/neighbor.h>
```

Prototype of the neighborhood update callback.

See also

[vrna\\_move\\_neighbor\\_diff\\_cb\(\)](#), [VRNA\\_NEIGHBOR\\_CHANGE](#), [VRNA\\_NEIGHBOR\\_INVALID](#), [VRNA\\_NEIGHBOR\\_NEW](#)

#### Parameters

|                 |   |
|-----------------|---|
| <i>fc</i>       | The fold compound the calling function is working on                                  |
| <i>neighbor</i> | The move that generates the (changed or new) neighbor                                 |
| <i>state</i>    | State of the neighbor (move) as supplied by argument <i>neighbor</i>                  |
| <i>data</i>     | Some arbitrary data pointer as passed to <a href="#">vrna_move_neighbor_diff_cb()</a> |

## 16.37.5 Function Documentation

### 16.37.5.1 vrna\_move\_init()

```
vrna_move_t vrna_move_init (
    int pos_5,
    int pos_3 )
#include <ViennaRNA/landscape/move.h>
Create an atomic move.
```

See also

[vrna\\_move\\_s](#)

#### Parameters

|                          |  |
|--------------------------|--|
| <code>pos←<br/>_5</code> | The 5' position of the move (positive for insertions, negative for removal, any value for shift moves) |
| <code>pos←<br/>_3</code> | The 3' position of the move (positive for insertions, negative for removal, any value for shift moves) |

#### Returns

An atomic move as specified by `pos_5` and `pos_3`

### 16.37.5.2 vrna\_move\_list\_free()

```
void vrna_move_list_free (
    vrna_move_t * moves )
#include <ViennaRNA/landscape/move.h>
delete all moves in a zero terminated list.
```

### 16.37.5.3 vrna\_move\_apply()

```
void vrna_move_apply (
    short * pt,
    const vrna_move_t * m )
#include <ViennaRNA/landscape/move.h>
Apply a particular move / transition to a secondary structure, i.e. transform a structure.
```

#### Parameters

|                      |                 |  |
|----------------------|-----------------|--|
| <code>in, out</code> | <code>pt</code> | The pair table representation of the secondary structure |
| <code>in</code>      | <code>m</code>  | The move to apply  |

### 16.37.5.4 vrna\_move\_is\_removal()

```
int vrna_move_is_removal (
    const vrna_move_t * m )
#include <ViennaRNA/landscape/move.h>
Test whether a move is a base pair removal.
```

**Parameters**

|          |                          |
|----------|--------------------------|
| <i>m</i> | The move to test against |
|----------|--------------------------|

**Returns**

Non-zero if the move is a base pair removal, 0 otherwise

**16.37.5.5 vrna\_move\_is\_insertion()**

```
int vrna_move_is_insertion (
    const vrna_move_t * m )
#include <ViennaRNA/landscape/move.h>
Test whether a move is a base pair insertion.
```

**Parameters**

|          |                          |
|----------|--------------------------|
| <i>m</i> | The move to test against |
|----------|--------------------------|

**Returns**

Non-zero if the move is a base pair insertion, 0 otherwise

**16.37.5.6 vrna\_move\_is\_shift()**

```
int vrna_move_is_shift (
    const vrna_move_t * m )
#include <ViennaRNA/landscape/move.h>
Test whether a move is a base pair shift.
```

**Parameters**

|          |                          |
|----------|--------------------------|
| <i>m</i> | The move to test against |
|----------|--------------------------|

**Returns**

Non-zero if the move is a base pair shift, 0 otherwise

**16.37.5.7 vrna\_move\_compare()**

```
int vrna_move_compare (
    const vrna_move_t * a,
    const vrna_move_t * b,
    const short * pt )
#include <ViennaRNA/landscape/move.h>
Compare two moves.
The function compares two moves a and b and returns whether move a is lexicographically smaller (-1), larger (1) or equal to move b.
If any of the moves a or b is a shift move, this comparison only makes sense in a structure context. Thus, the third argument with the current structure must be provided.
```

**Note**

This function returns 0 (equality) upon any error, e.g. missing input

**Warning**

Currently, shift moves are not supported!

**Parameters**

|           |   |
|-----------|---|
| <i>a</i>  | The first move of the comparison  |
| <i>b</i>  | The second move of the comparison   |
| <i>pt</i> | The pair table of the current structure that is compatible with both moves (maybe NULL if moves are guaranteed to be no shifts) |

**Returns**

-1 if *a* < *b*, 1 if *a* > *b*, 0 otherwise

**16.37.5.8 vrna\_loopidx\_update()**

```
void vrna_loopidx_update (
    int * loopidx,
    const short * pt,
    int length,
    const vrna_move_t * m )
#include <ViennaRNA/landscape/neighbor.h>
```

Alters the loopIndices array that was constructed with [vrna\\_loopidx\\_from\\_ptable\(\)](#).

The loopIndex of the current move will be inserted. The correctness of the input will not be checked because the speed should be optimized.

**Parameters**

|               |                |  |
|---------------|----------------|--|
| <i>in,out</i> | <i>loopidx</i> | The loop index data structure that needs an update |
| <i>in</i>     | <i>pt</i>      | A pair table on which the move will be executed    |
|               | <i>length</i>  | The length of the structure                        |
| <i>in</i>     | <i>m</i>       | The move that is applied to the current structure  |

**16.37.5.9 vrna\_neighbors()**

```
vrna_move_t * vrna_neighbors (
    vrna_fold_compound_t * vc,
    const short * pt,
    unsigned int options )
#include <ViennaRNA/landscape/neighbor.h>
```

Generate neighbors of a secondary structure.

This function allows one to generate all structural neighbors (according to a particular move set) of an RNA secondary structure. The neighborhood is then returned as a list of transitions / moves required to transform the current structure into the actual neighbor.

**See also**

[vrna\\_neighbors\\_successive\(\)](#), [vrna\\_move\\_apply\(\)](#), [VRNA\\_MOVESET\\_INSERTION](#), [VRNA\\_MOVESET\\_DELETION](#), [VRNA\\_MOVESET\\_SHIFT](#), [VRNA\\_MOVESET\\_DEFAULT](#)

**Parameters**

|    |                |   |
|----|----------------|---|
| in | <i>vc</i>      | A vrna_fold_compound_t containing the energy parameters and model details |
| in | <i>pt</i>      | The pair table representation of the structure                            |
|    | <i>options</i> | Options to modify the behavior of this function, e.g. available move set  |

**Returns**

Neighbors as a list of moves / transitions (the last element in the list has both of its fields set to 0)

**SWIG Wrapper Notes** This function is attached as an overloaded method *neighbors()* to objects of type *fold ← compound*. The optional parameter *options* defaults to [VRNA\\_MOVESET\\_DEFAULT](#) if it is omitted.

**16.37.5.10 vrna\_neighbors\_successive()**

```
vrna_move_t * vrna_neighbors_successive (
    const vrna_fold_compound_t * vc,
    const vrna_move_t * curr_move,
    const short * prev_pt,
    const vrna_move_t * prev_neighbors,
    int size_prev_neighbors,
    int * size_neighbors,
    unsigned int options )
```

```
#include <ViennaRNA/landscape/neighbor.h>
```

Generate neighbors of a secondary structure (the fast way)

This function implements a fast way to generate all neighbors of a secondary structure that results from successive applications of individual moves. The speed-up results from updating an already known list of valid neighbors before the individual move towards the current structure took place. In essence, this function removes neighbors that are not accessible anymore and inserts neighbors emerging after a move took place.

**See also**

[vrna\\_neighbors\(\)](#), [vrna\\_move\\_apply\(\)](#), [VRNA\\_MOVESET\\_INSERTION](#), [VRNA\\_MOVESET\\_DELETION](#), [VRNA\\_MOVESET\\_SHIFT](#), [VRNA\\_MOVESET\\_DEFAULT](#)

**Parameters**

|     |                            |   |
|-----|----------------------------|---|
| in  | <i>vc</i>                  | A vrna_fold_compound_t containing the energy parameters and model details           |
| in  | <i>curr_move</i>           | The move that was/will be applied to <i>prev_pt</i>                                 |
| in  | <i>prev_pt</i>             | A pair table representation of the structure before <i>curr_move</i> is/was applied |
| in  | <i>prev_neighbors</i>      | The list of neighbors of <i>prev_pt</i>   |
|     | <i>size_prev_neighbors</i> | The size of <i>prev_neighbors</i> , i.e. the lists length                           |
| out | <i>size_neighbors</i>      | A pointer to store the size / length of the new neighbor list                       |
|     | <i>options</i>             | Options to modify the behavior of this function, e.g. available move set            |

**Returns**

Neighbors as a list of moves / transitions (the last element in the list has both of its fields set to 0)

**16.37.5.11 vrna\_move\_neighbor\_diff\_cb()**

```
int vrna_move_neighbor_diff_cb (
    vrna_fold_compound_t * fc,
```

```

    short * ptable,
    vrna_move_t move,
    vrna_callback_move_update * cb,
    void * data,
    unsigned int options )
#include <ViennaRNA/landscape/neighbor.h>

```

Apply a move to a secondary structure and indicate which neighbors have changed consequentially.

This function applies a move to a secondary structure and explores the local neighborhood of the affected loop. Any changes to previously compatible neighbors that have been affected by this loop will be reported through a callback function. In particular, any of the three cases might appear:

- A previously available neighbor move has changed, usually the free energy change of the move ([VRNA\\_NEIGHBOR\\_CHANGE](#))
- A previously available neighbor move became invalid ([VRNA\\_NEIGHBOR\\_INVALID](#))
- A new neighbor move becomes available ([VRNA\\_NEIGHBOR\\_NEW](#))

#### See also

[vrna\\_move\\_neighbor\\_diff\(\)](#), [VRNA\\_NEIGHBOR\\_CHANGE](#), [VRNA\\_NEIGHBOR\\_INVALID](#), [VRNA\\_NEIGHBOR\\_NEW](#), [vrna\\_callback\\_move\\_update](#)

#### Parameters

|                |  |
|----------------|--|
| <i>fc</i>      | A fold compound for the RNA sequence(s) that this function operates on                   |
| <i>ptable</i>  | The current structure as pair table  |
| <i>move</i>    | The move to apply  |
| <i>cb</i>      | The address of the callback function that is passed the neighborhood changes             |
| <i>data</i>    | An arbitrary data pointer that will be passed through to the callback function <i>cb</i> |
| <i>options</i> | Options to modify the behavior of this function, .e.g available move set                 |

#### Returns

Non-zero on success, 0 otherwise

#### 16.37.5.12 [vrna\\_move\\_neighbor\\_diff\(\)](#)

```

vrna_move_t * vrna_move_neighbor_diff (
    vrna_fold_compound_t * fc,
    short * ptable,
    vrna_move_t move,
    vrna_move_t ** invalid_moves,
    unsigned int options )
#include <ViennaRNA/landscape/neighbor.h>

```

Apply a move to a secondary structure and indicate which neighbors have changed consequentially.

Similar to [vrna\\_move\\_neighbor\\_diff\\_cb\(\)](#), this function applies a move to a secondary structure and reports back the neighbors of the current structure become affected by this move. Instead of executing a callback for each of the affected neighbors, this function compiles two lists of neighbor moves, one that is returned and consists of all moves that are novel or may have changed in energy, and a second, *invalid\_moves*, that consists of all the neighbor moves that become invalid, respectively.

#### Parameters

|               |  |
|---------------|--|
| <i>fc</i>     | A fold compound for the RNA sequence(s) that this function operates on |
| <i>ptable</i> | The current structure as pair table                                    |

**Parameters**

|                      |  |
|----------------------|--|
| <i>move</i>          | The move to apply  |
| <i>invalid_moves</i> | The address of a move list where the function stores those moves that become invalid |
| <i>options</i>       | Options to modify the behavior of this function, e.g available move set              |

**Returns**

A list of moves that might have changed in energy or are novel compared to the structure before application of the move

## 16.38 (Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima

API for various RNA folding path algorithms.

### 16.38.1 Detailed Description

API for various RNA folding path algorithms.

This part of our API allows for generating RNA secondary structure (re-)folding paths between two secondary structures or simply starting from a single structure. This is most important if an estimate of the refolding energy barrier between two structures is required, or a structure's corresponding local minimum needs to be determined, e.g. through a gradient-descent walk.

This part of the interface is further split into the following sections:

- [Direct Refolding Paths between two Secondary Structures](#), and
- [Folding Paths that start at a single Secondary Structure](#)

Collaboration diagram for (Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima:

### Modules

- [Direct Refolding Paths between two Secondary Structures](#)  
*Heuristics to explore direct, optimal (re-)folding paths between two secondary structures.*
- [Folding Paths that start at a single Secondary Structure](#)  
*Implementation of gradient- and random walks starting from a single secondary structure.*
- [Deprecated Interface for \(Re-\)folding Paths, Saddle Points, and Energy Barriers](#)

### Files

- file [findpath.h](#)  
*A breadth-first search heuristic for optimal direct folding paths.*
- file [paths.h](#)  
*API for computing (optimal) (re-)folding paths between secondary structures.*
- file [walk.h](#)  
*Methods to generate particular paths such as gradient or random walks through the energy landscape of an RNA sequence.*

### Data Structures

- struct [vrna\\_path\\_s](#)  
*An element of a refolding path list. [More...](#)*

## Macros

- `#define VRNA_PATH_TYPE_DOT_BRACKET 1U`  
*Flag to indicate producing a (re-)folding path as list of dot-bracket structures.*
- `#define VRNA_PATH_TYPE_MOVES 2U`  
*Flag to indicate producing a (re-)folding path as list of transition moves.*

## Typedefs

- `typedef struct vrna_path_s vrna_path_t`  
*Typename for the refolding path data structure `vrna_path_s`.*
- `typedef struct vrna_path_options_s * vrna_path_options_t`  
*Options data structure for (re-)folding path implementations.*

## Functions

- `void vrna_path_free (vrna_path_t *path)`  
*Release (free) memory occupied by a (re-)folding path.*
- `void vrna_path_options_free (vrna_path_options_t options)`  
*Release (free) memory occupied by an options data structure for (re-)folding path implementations.*

### 16.38.2 Data Structure Documentation

#### 16.38.2.1 struct vrna\_path\_s

An element of a refolding path list.

Usually, one has to deal with an array of `vrna_path_s`, e.g. returned from one of the refolding-path algorithms. Since in most cases the length of the list is not known in advance, such lists have an *end-of-list* marker, which is either:

- a value of `NULL` for `vrna_path_s::s` if `vrna_path_s::type = VRNA_PATH_TYPE_DOT_BRACKET`, or
- a `vrna_path_s::move` with zero in both fields `vrna_move_t::pos_5` and `vrna_move_t::pos_3` if `vrna_path_s::type = VRNA_PATH_TYPE_MOVES`.

In the following we show an example for how to cover both cases of iteration:

```
vrna_path_t *ptr = path; // path was returned from one of the refolding path functions, e.g.
    vrna_path_direct()
if (ptr) {
    if (ptr->type == VRNA_PATH_TYPE_DOT_BRACKET) {
        for (; ptr->s; ptr++)
            printf("%s [%6.2f]\n", ptr->s, ptr->en);
    } else if (ptr->type == VRNA_PATH_TYPE_MOVES) {
        for (; ptr->move.pos_5 != 0; ptr++)
            printf("move %d:%d, dG = %6.2f\n", ptr->move.pos_5, ptr->move.pos_3, ptr->en);
    }
}
```

See also

[vrna\\_path\\_free\(\)](#)

Collaboration diagram for `vrna_path_s`:

#### Data Fields

- `unsigned int type`  
*The type of the path element.*
- `double en`  
*Free energy of current structure.*
- `char * s`  
*Secondary structure in dot-bracket notation.*
- `vrna_move_t move`  
*Move that transforms the previous structure into it's next neighbor along the path.*

### 16.38.2.1.1 Field Documentation

#### 16.38.2.1.1.1 type `unsigned int vrna_path_s::type`

The type of the path element.

A value of `VRNA_PATH_TYPE_DOT_BRACKET` indicates that `vrna_path_s::s` consists of the secondary structure in dot-bracket notation, and `vrna_path_s::en` the corresponding free energy.

On the other hand, if the value is `VRNA_PATH_TYPE_MOVES`, `vrna_path_s::s` is `NULL` and `vrna_path_s::move` is set to the transition move that transforms a previous structure into its neighbor along the path. In this case, the attribute `vrna_path_s::en` states the change in free energy with respect to the structure before application of `vrna_path_s::move`.

### 16.38.3 Macro Definition Documentation

#### 16.38.3.1 VRNA\_PATH\_TYPE\_DOT\_BRACKET

```
#define VRNA_PATH_TYPE_DOT_BRACKET 1U
#include <ViennaRNA/landscape/paths.h>
Flag to indicate producing a (re-)folding path as list of dot-bracket structures.
```

See also

[vrna\\_path\\_t](#), [vrna\\_path\\_options\\_findpath\(\)](#), [vrna\\_path\\_direct\(\)](#), [vrna\\_path\\_direct\\_ub\(\)](#)

#### 16.38.3.2 VRNA\_PATH\_TYPE\_MOVES

```
#define VRNA_PATH_TYPE_MOVES 2U
#include <ViennaRNA/landscape/paths.h>
Flag to indicate producing a (re-)folding path as list of transition moves.
```

See also

[vrna\\_path\\_t](#), [vrna\\_path\\_options\\_findpath\(\)](#), [vrna\\_path\\_direct\(\)](#), [vrna\\_path\\_direct\\_ub\(\)](#)

### 16.38.4 Function Documentation

#### 16.38.4.1 vrna\_path\_free()

```
void vrna_path_free (
    vrna_path_t * path )
#include <ViennaRNA/landscape/paths.h>
Release (free) memory occupied by a (re-)folding path.
```

See also

[vrna\\_path\\_direct\(\)](#), [vrna\\_path\\_direct\\_ub\(\)](#), [vrna\\_path\\_findpath\(\)](#), [vrna\\_path\\_findpath\\_ub\(\)](#)

Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <code>path</code> | The refolding path to be free'd |
|-------------------|---------------------------------|

#### 16.38.4.2 vrna\_path\_options\_free()

```
void vrna_path_options_free (
    vrna_path_options_t options )
#include <ViennaRNA/landscape/paths.h>
```

Release (free) memory occupied by an options data structure for (re-)folding path implementations.

See also

[vrna\\_path\\_options\\_findpath\(\)](#), [vrna\\_path\\_direct\(\)](#), [vrna\\_path\\_direct\\_ub\(\)](#)

Parameters

|                |   |
|----------------|---|
| <i>options</i> | The options data structure to be free'd |
|----------------|---|

## 16.39 Direct Refolding Paths between two Secondary Structures

Heuristics to explore direct, optimal (re-)folding paths between two secondary structures.

### 16.39.1 Detailed Description

Heuristics to explore direct, optimal (re-)folding paths between two secondary structures.

Collaboration diagram for Direct Refolding Paths between two Secondary Structures:

### Functions

- int [vrna\\_path\\_findpath\\_saddle](#) (*vrna\_fold\_compound\_t* \*fc, const char \*s1, const char \*s2, int width)
   
*Find energy of a saddle point between 2 structures (search only direct path)*
- int [vrna\\_path\\_findpath\\_saddle\\_ub](#) (*vrna\_fold\_compound\_t* \*fc, const char \*s1, const char \*s2, int width, int maxE)
   
*Find energy of a saddle point between 2 structures (search only direct path)*
- *vrna\_path\_t* \* [vrna\\_path\\_findpath](#) (*vrna\_fold\_compound\_t* \*fc, const char \*s1, const char \*s2, int width)
   
*Find refolding path between 2 structures (search only direct path)*
- *vrna\_path\_t* \* [vrna\\_path\\_findpath\\_ub](#) (*vrna\_fold\_compound\_t* \*fc, const char \*s1, const char \*s2, int width, int maxE)
   
*Find refolding path between 2 structures (search only direct path)*
- *vrna\_path\_options\_t* [vrna\\_path\\_options\\_findpath](#) (int width, unsigned int type)
   
*Create options data structure for findpath direct (re-)folding path heuristic.*
- *vrna\_path\_t* \* [vrna\\_path\\_direct](#) (*vrna\_fold\_compound\_t* \*fc, const char \*s1, const char \*s2, *vrna\_path\_options\_t* options)
   
*Determine an optimal direct (re-)folding path between two secondary structures.*
- *vrna\_path\_t* \* [vrna\\_path\\_direct\\_ub](#) (*vrna\_fold\_compound\_t* \*fc, const char \*s1, const char \*s2, int maxE, *vrna\_path\_options\_t* options)
   
*Determine an optimal direct (re-)folding path between two secondary structures.*

### 16.39.2 Function Documentation

#### 16.39.2.1 vrna\_path\_findpath\_saddle()

```
int vrna_path_findpath_saddle (
    vrna_fold_compound_t * fc,
    const char * s1,
```

```

    const char * s2,
    int width )
#include <ViennaRNA/landscape/findpath.h>
Find energy of a saddle point between 2 structures (search only direct path)
This function uses an implementation of the findpath algorithm [7] for near-optimal direct refolding path prediction.
Model details, and energy parameters are used as provided via the parameter 'fc'. The vrna_fold_compound_t does
not require memory for any DP matrices, but requires all most basic init values as one would get from a call like this:
fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_DEFAULT);

```

#### See also

[vrna\\_path\\_findpath\\_saddle\\_ub\(\)](#), [vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\\_compound\\_t](#), [vrna\\_path\\_findpath\(\)](#)

#### Parameters

|              |  |
|--------------|--|
| <i>fc</i>    | The <b>vrna_fold_compound_t</b> with precomputed sequence encoding and model details |
| <i>s1</i>    | The start structure in dot-bracket notation  |
| <i>s2</i>    | The target structure in dot-bracket notation   |
| <i>width</i> | A number specifying how many strutures are being kept at each step during the search |

#### Returns

The saddle energy in 10cal/mol

**SWIG Wrapper Notes** This function is attached as an overloaded method *path\_findpath\_saddle()* to objects of type *fold\_compound*. The optional parameter *width* defaults to 1 if it is omitted.

#### 16.39.2.2 vrna\_path\_findpath\_saddle\_ub()

```

int vrna_path_findpath_saddle_ub (
    vrna_fold_compound_t * fc,
    const char * s1,
    const char * s2,
    int width,
    int maxE )
#include <ViennaRNA/landscape/findpath.h>
Find energy of a saddle point between 2 structures (search only direct path)
This function uses an implementation of the findpath algorithm [7] for near-optimal direct refolding path prediction.
Model details, and energy parameters are used as provided via the parameter 'fc'. The vrna_fold_compound_t does
not require memory for any DP matrices, but requires all most basic init values as one would get from a call like this:
fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_DEFAULT);

```

#### Warning

The argument *maxE* ( $E_{max}$ ) enables one to specify an upper bound, or maximum free energy for the saddle point between the two input structures. If no path with  $E_{saddle} < E_{max}$  is found, the function simply returns *maxE*

#### See also

[vrna\\_path\\_findpath\\_saddle\(\)](#), [vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\\_compound\\_t](#), [vrna\\_path\\_findpath\(\)](#)

#### Parameters

|              |  |
|--------------|--|
| <i>fc</i>    | The <b>vrna_fold_compound_t</b> with precomputed sequence encoding and model details |
| <i>s1</i>    | The start structure in dot-bracket notation  |
| <i>s2</i>    | The target structure in dot-bracket notation   |
| <i>width</i> | A number specifying how many strutures are being kept at each step during the search |

*maxE* An upper bound for the saddle point energy in 10cal/mol

Generated by Doxygen

**Returns**

The saddle energy in 10cal/mol

**SWIG Wrapper Notes** This function is attached as an overloaded method `path_findpath_saddle()` to objects of type `fold_compound`. The optional parameter `width` defaults to 1 if it is omitted, while the optional parameter `maxE` defaults to `INF`. In case the function did not find a path with  $E_{saddle} < E_{max}$  the function returns a `NULL` object, i.e. `undef` for Perl and `None` for Python.

**16.39.2.3 vrna\_path\_findpath()**

```
vrna_path_t * vrna_path_findpath (
    vrna_fold_compound_t * fc,
    const char * s1,
    const char * s2,
    int width )
#include <ViennaRNA/landscape/findpath.h>
```

Find refolding path between 2 structures (search only direct path)

This function uses an implementation of the `findpath` algorithm [7] for near-optimal direct refolding path prediction. Model details, and energy parameters are used as provided via the parameter 'fc'. The `vrna_fold_compound_t` does not require memory for any DP matrices, but requires all most basic init values as one would get from a call like this:  
`fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_DEFAULT);`

**See also**

[vrna\\_path\\_findpath\\_ub\(\)](#), [vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\\_compound\\_t](#), [vrna\\_path\\_findpath\\_saddle\(\)](#)

**Parameters**

|                    |  |
|--------------------|--|
| <code>fc</code>    | The <code>vrna_fold_compound_t</code> with precomputed sequence encoding and model details |
| <code>s1</code>    | The start structure in dot-bracket notation  |
| <code>s2</code>    | The target structure in dot-bracket notation   |
| <code>width</code> | A number specifying how many strutures are being kept at each step during the search       |

**Returns**

The saddle energy in 10cal/mol

**SWIG Wrapper Notes** This function is attached as an overloaded method `path_findpath()` to objects of type `fold_compound`. The optional parameter `width` defaults to 1 if it is omitted.

**16.39.2.4 vrna\_path\_findpath\_ub()**

```
vrna_path_t * vrna_path_findpath_ub (
    vrna_fold_compound_t * fc,
    const char * s1,
    const char * s2,
    int width,
    int maxE )
#include <ViennaRNA/landscape/findpath.h>
```

Find refolding path between 2 structures (search only direct path)

This function uses an implementation of the `findpath` algorithm [7] for near-optimal direct refolding path prediction. Model details, and energy parameters are used as provided via the parameter 'fc'. The `vrna_fold_compound_t` does not require memory for any DP matrices, but requires all most basic init values as one would get from a call like this:  
`fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_DEFAULT);`

**Warning**

The argument `maxE` enables one to specify an upper bound, or maximum free energy for the saddle point between the two input structures. If no path with  $E_{saddle} < E_{max}$  is found, the function simply returns `NULL`.

**See also**

[vrna\\_path\\_findpath\(\)](#), [vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\\_compound\\_t](#), [vrna\\_path\\_findpath\\_saddle\(\)](#)

**Parameters**

|                    |   |
|--------------------|---|
| <code>fc</code>    | The <a href="#">vrna_fold_compound_t</a> with precomputed sequence encoding and model details |
| <code>s1</code>    | The start structure in dot-bracket notation   |
| <code>s2</code>    | The target structure in dot-bracket notation  |
| <code>width</code> | A number specifying how many structures are being kept at each step during the search         |
| <code>maxE</code>  | An upper bound for the saddle point energy in 10cal/mol                                       |

**Returns**

The saddle energy in 10cal/mol

**SWIG Wrapper Notes** This function is attached as an overloaded method `path_findpath()` to objects of type `fold_compound`. The optional parameter `width` defaults to 1 if it is omitted, while the optional parameter `maxE` defaults to [INF](#). In case the function did not find a path with  $E_{saddle} < E_{max}$  the function returns an empty list.

**16.39.2.5 vrna\_path\_options\_findpath()**

```
vrna_path_options_t vrna_path_options_findpath (
    int width,
    unsigned int type )
#include <ViennaRNA/landscape/paths.h>
```

Create options data structure for findpath direct (re-)folding path heuristic.

This function returns an options data structure that switches the [vrna\\_path\\_direct\(\)](#) and [vrna\\_path\\_direct\\_ub\(\)](#) API functions to use the `findpath` [7] heuristic. The parameter `width` specifies the width of the breadth-first search while the second parameter `type` allows one to set the type of the returned (re-)folding path.

Currently, the following return types are available:

- A list of dot-bracket structures and corresponding free energy (flag: [VRNA\\_PATH\\_TYPE\\_DOT\\_BRACKET](#))
- A list of transition moves and corresponding free energy changes (flag: [VRNA\\_PATH\\_TYPE\\_MOVES](#))

**See also**

[VRNA\\_PATH\\_TYPE\\_DOT\\_BRACKET](#), [VRNA\\_PATH\\_TYPE\\_MOVES](#), [vrna\\_path\\_options\\_free\(\)](#), [vrna\\_path\\_direct\(\)](#), [vrna\\_path\\_direct\\_ub\(\)](#)

**Parameters**

|                    |   |
|--------------------|---|
| <code>width</code> | Width of the breath-first search strategy                                 |
| <code>type</code>  | Setting that specifies how the return (re-)folding path should be encoded |

**Returns**

An options data structure with settings for the findpath direct path heuristic

**SWIG Wrapper Notes** This function is available as overloaded function `path_options_findpath()`. The optional parameter `width` defaults to 10 if omitted, while the optional parameter `type` defaults to `VRNA_PATH_TYPE_DOT_BRACKET`.

**16.39.2.6 vrna\_path\_direct()**

```
vrna_path_t * vrna_path_direct (
    vrna_fold_compound_t * fc,
    const char * s1,
    const char * s2,
    vrna_path_options_t options )
#include <ViennaRNA/landscape/paths.h>
```

Determine an optimal direct (re-)folding path between two secondary structures.

This is the generic wrapper function to retrieve (an optimal) (re-)folding path between two secondary structures `s1` and `s2`. The actual algorithm that is used to generate the (re-)folding path is determined by the settings specified in the `options` data structure. This data structure also determines the return type, which might be either:

- a list of dot-bracket structures with corresponding free energy, or
- a list of transition moves with corresponding free energy change

If the `options` parameter is passed a `NULL` pointer, this function defaults to the *findpath heuristic* [7] with a breadth-first search width of 10, and the returned path consists of dot-bracket structures with corresponding free energies.

**See also**

[vrna\\_path\\_direct\\_ub\(\)](#), [vrna\\_path\\_options\\_findpath\(\)](#), [vrna\\_path\\_options\\_free\(\)](#), [vrna\\_path\\_free\(\)](#)

**Parameters**

|                      |   |
|----------------------|---|
| <code>fc</code>      | The <code>vrna_fold_compound_t</code> with precomputed sequence encoding and model details                        |
| <code>s1</code>      | The start structure in dot-bracket notation   |
| <code>s2</code>      | The target structure in dot-bracket notation  |
| <code>options</code> | An options data structure that specifies the path heuristic and corresponding settings (maybe <code>NULL</code> ) |

**Returns**

An optimal (re-)folding path between the two input structures

**SWIG Wrapper Notes** This function is attached as an overloaded method `path_direct()` to objects of type `fold_compound`. The optional parameter `options` defaults to `NULL` if it is omitted.

**16.39.2.7 vrna\_path\_direct\_ub()**

```
vrna_path_t * vrna_path_direct_ub (
    vrna_fold_compound_t * fc,
    const char * s1,
    const char * s2,
    int maxE,
    vrna_path_options_t options )
#include <ViennaRNA/landscape/paths.h>
```

Determine an optimal direct (re-)folding path between two secondary structures.

This function is similar to [vrna\\_path\\_direct\(\)](#), but allows to specify an *upper-bound* for the saddle point energy. The underlying algorithms will stop determining an (optimal) (re-)folding path, if none can be found that has a saddle point below the specified upper-bound threshold `maxE`.

#### Warning

The argument `maxE` enables one to specify an upper bound, or maximum free energy for the saddle point between the two input structures. If no path with  $E_{saddle} < E_{max}$  is found, the function simply returns `NULL`

#### See also

[vrna\\_path\\_direct\\_ub\(\)](#), [vrna\\_path\\_options\\_findpath\(\)](#), [vrna\\_path\\_options\\_free\(\)](#), [vrna\\_path\\_free\(\)](#)

#### Parameters

|                      |   |
|----------------------|---|
| <code>fc</code>      | The <a href="#">vrna_fold_compound_t</a> with precomputed sequence encoding and model details                     |
| <code>s1</code>      | The start structure in dot-bracket notation   |
| <code>s2</code>      | The target structure in dot-bracket notation  |
| <code>maxE</code>    | Upper bound for the saddle point along the (re-)folding path  |
| <code>options</code> | An options data structure that specifies the path heuristic and corresponding settings (maybe <code>NULL</code> ) |

#### Returns

An optimal (re-)folding path between the two input structures

**SWIG Wrapper Notes** This function is attached as an overloaded method `path_direct()` to objects of type `fold ← compound`. The optional parameter `maxE` defaults to `#INT_MAX - 1` if it is omitted, while the optional parameter `options` defaults to `NULL`. In case the function did not find a path with  $E_{saddle} < E_{max}$  it returns an empty list.

## 16.40 Folding Paths that start at a single Secondary Structure

Implementation of gradient- and random walks starting from a single secondary structure.

### 16.40.1 Detailed Description

Implementation of gradient- and random walks starting from a single secondary structure.  
Collaboration diagram for Folding Paths that start at a single Secondary Structure:

#### Macros

- `#define VRNA_PATH_STEEPEST_DESCENT 128`  
*Option flag to request a steepest descent / gradient path.*
- `#define VRNA_PATH_RANDOM 256`  
*Option flag to request a random walk path.*
- `#define VRNA_PATH_NO_TRANSITION_OUTPUT 512`  
*Option flag to omit returning the transition path.*
- `#define VRNA_PATH_DEFAULT (VRNA_PATH_STEEPEST_DESCENT | VRNA_MOVESET_DEFAULT)`  
*Option flag to request defaults (steepest descent / default move set)*

#### Functions

- [vrna\\_move\\_t \\* vrna\\_path \(vrna\\_fold\\_compound\\_t \\*vc, short \\*pt, unsigned int steps, unsigned int options\)](#)  
*Compute a path, store the final structure, and return a list of transition moves from the start to the final structure.*
- [vrna\\_move\\_t \\* vrna\\_path\\_gradient \(vrna\\_fold\\_compound\\_t \\*vc, short \\*pt, unsigned int options\)](#)

*Compute a steepest descent / gradient path, store the final structure, and return a list of transition moves from the start to the final structure.*

- [vrna\\_move\\_t \\* vrna\\_path\\_random](#) (`vrna_fold_compound_t *vc, short *pt, unsigned int steps, unsigned int options)`

*Generate a random walk / path of a given length, store the final structure, and return a list of transition moves from the start to the final structure.*

## 16.40.2 Macro Definition Documentation

### 16.40.2.1 VRNA\_PATH\_STEEPEST\_DESCENT

```
#define VRNA_PATH_STEEPEST_DESCENT 128
#include <ViennaRNA/landscape/walk.h>
Option flag to request a steepest descent / gradient path.
```

See also

[vrna\\_path\(\)](#)

### 16.40.2.2 VRNA\_PATH\_RANDOM

```
#define VRNA_PATH_RANDOM 256
#include <ViennaRNA/landscape/walk.h>
Option flag to request a random walk path.
```

See also

[vrna\\_path\(\)](#)

### 16.40.2.3 VRNA\_PATH\_NO\_TRANSITION\_OUTPUT

```
#define VRNA_PATH_NO_TRANSITION_OUTPUT 512
#include <ViennaRNA/landscape/walk.h>
Option flag to omit returning the transition path.
```

See also

[vrna\\_path\(\)](#), [vrna\\_path\\_gradient\(\)](#), [vrna\\_path\\_random\(\)](#)

### 16.40.2.4 VRNA\_PATH\_DEFAULT

```
#define VRNA_PATH_DEFAULT (VRNA_PATH_STEEPEST_DESCENT | VRNA_MOVESET_DEFAULT)
#include <ViennaRNA/landscape/walk.h>
Option flag to request defaults (steepest descent / default move set)
```

See also

[vrna\\_path\(\)](#), [VRNA\\_PATH\\_STEEPEST\\_DESCENT](#), [VRNA\\_MOVESET\\_DEFAULT](#)

## 16.40.3 Function Documentation

### 16.40.3.1 vrna\_path()

```
vrna_move_t * vrna_path (
    vrna_fold_compound_t * vc,
    short * pt,
    unsigned int steps,
    unsigned int options )
#include <ViennaRNA/landscape/walk.h>
```

Compute a path, store the final structure, and return a list of transition moves from the start to the final structure. This function computes, given a start structure in pair table format, a transition path, updates the pair table to the final structure of the path. Finally, if not requested otherwise by using the `VRNA_PATH_NO_TRANSITION_OUTPUT` flag in the `options` field, this function returns a list of individual transitions that lead from the start to the final structure if requested.

The currently available transition paths are

- Steepest Descent / Gradient walk (flag: `VRNA_PATH_STEEPEST_DESCENT`)
- Random walk (flag: `VRNA_PATH_RANDOM`)

The type of transitions must be set through the `options` parameter

#### Note

Since the result is written to the input structure you may want to use `vrna_ptable_copy()` before calling this function to keep the initial structure

#### See also

`vrna_path_gradient()`, `vrna_path_random()`, `vrna_ptable()`, `vrna_ptable_copy()`, `vrna_fold_compound()`, `VRNA_PATH_STEEPEST_DESCENT`, `VRNA_PATH_RANDOM`, `VRNA_MOVESET_DEFAULT`, `VRNA_MOVESET_SHIFT`, `VRNA_PATH_NO_TRANSITION_OUTPUT`

#### Parameters

|         |                      |   |
|---------|----------------------|---|
| in      | <code>vc</code>      | A <code>vrna_fold_compound_t</code> containing the energy parameters and model details                                |
| in, out | <code>pt</code>      | The pair table containing the start structure. Used to update to the final structure after execution of this function |
| in      | <code>options</code> | Options to modify the behavior of this function   |

#### Returns

A list of transition moves (default), or NULL (if `options & VRNA_PATH_NO_TRANSITION_OUTPUT`)

**SWIG Wrapper Notes** This function is attached as an overloaded method `path()` to objects of type `fold_compound`. The optional parameter `options` defaults to `VRNA_PATH_DEFAULT` if it is omitted.

### 16.40.3.2 vrna\_path\_gradient()

```
vrna_move_t * vrna_path_gradient (
    vrna_fold_compound_t * vc,
    short * pt,
    unsigned int options )
#include <ViennaRNA/landscape/walk.h>
```

Compute a steepest descent / gradient path, store the final structure, and return a list of transition moves from the start to the final structure.

This function computes, given a start structure in pair table format, a steepest descent path, updates the pair table to the final structure of the path. Finally, if not requested otherwise by using the `VRNA_PATH_NO_TRANSITION_OUTPUT` flag in the `options` field, this function returns a list of individual transitions that lead from the start to the final structure if requested.

**Note**

Since the result is written to the input structure you may want to use `vrna_phtable_copy()` before calling this function to keep the initial structure

**See also**

`vrna_path_random()`, `vrna_path()`, `vrna_phtable()`, `vrna_phtable_copy()`, `vrna_fold_compound()` `VRNA_MOVESET_DEFAULT`, `VRNA_MOVESET_SHIFT`, `VRNA_PATH_NO_TRANSITION_OUTPUT`

**Parameters**

|         |                      |   |
|---------|----------------------|---|
| in      | <code>vc</code>      | A <code>vrna_fold_compound_t</code> containing the energy parameters and model details                                |
| in, out | <code>pt</code>      | The pair table containing the start structure. Used to update to the final structure after execution of this function |
| in      | <code>options</code> | Options to modify the behavior of this function   |

**Returns**

A list of transition moves (default), or NULL (if options & `VRNA_PATH_NO_TRANSITION_OUTPUT`)

**SWIG Wrapper Notes** This function is attached as an overloaded method `path_gradient()` to objects of type `fold_compound`. The optional parameter `options` defaults to `VRNA_PATH_DEFAULT` if it is omitted.

**16.40.3.3 vrna\_path\_random()**

```
vrna_move_t * vrna_path_random (
    vrna_fold_compound_t * vc,
    short * pt,
    unsigned int steps,
    unsigned int options )
#include <ViennaRNA/landscape/walk.h>
```

Generate a random walk / path of a given length, store the final structure, and return a list of transition moves from the start to the final structure.

This function generates, given a start structure in pair table format, a random walk / path, updates the pair table to the final structure of the path. Finally, if not requested otherwise by using the `VRNA_PATH_NO_TRANSITION_OUTPUT` flag in the `options` field, this function returns a list of individual transitions that lead from the start to the final structure if requested.

**Note**

Since the result is written to the input structure you may want to use `vrna_phtable_copy()` before calling this function to keep the initial structure

**See also**

`vrna_path_gradient()`, `vrna_path()`, `vrna_phtable()`, `vrna_phtable_copy()`, `vrna_fold_compound()` `VRNA_MOVESET_DEFAULT`, `VRNA_MOVESET_SHIFT`, `VRNA_PATH_NO_TRANSITION_OUTPUT`

**Parameters**

|         |                      |   |
|---------|----------------------|---|
| in      | <code>vc</code>      | A <code>vrna_fold_compound_t</code> containing the energy parameters and model details                                |
| in, out | <code>pt</code>      | The pair table containing the start structure. Used to update to the final structure after execution of this function |
| in      | <code>steps</code>   | The length of the path, i.e. the total number of transitions / moves  |
| in      | <code>options</code> | Options to modify the behavior of this function   |

**Returns**

A list of transition moves (default), or NULL (if options & `VRNA_PATH_NO_TRANSITION_OUTPUT`)

**SWIG Wrapper Notes** This function is attached as an overloaded method `path_gradient()` to objects of type `fold_compound`. The optional parameter `options` defaults to `VRNA_PATH_DEFAULT` if it is omitted.

## 16.41 Experimental Structure Probing Data

Include Experimental Structure Probing Data to Guide Structure Predictions.

### 16.41.1 Detailed Description

Include Experimental Structure Probing Data to Guide Structure Predictions.

Collaboration diagram for Experimental Structure Probing Data:

## Modules

- **SHAPE Reactivity Data**

*Incorporate SHAPE reactivity structure probing data into the folding recursions by means of soft constraints.*

- **Generate Soft Constraints from Data**

*Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments.*

## 16.42 SHAPE Reactivity Data

Incorporate SHAPE reactivity structure probing data into the folding recursions by means of soft constraints.

### 16.42.1 Detailed Description

Incorporate SHAPE reactivity structure probing data into the folding recursions by means of soft constraints.

Details for our implementation to incorporate SHAPE reactivity data to guide secondary structure prediction can be found in [17] Collaboration diagram for SHAPE Reactivity Data:

## Files

- file `SHAPE.h`

*This module provides function to incorporate SHAPE reactivity data into the folding recursions by means of soft constraints.*

## Functions

- int `vrna_sc_add_SHAPE_deigan` (`vrna_fold_compound_t` \*vc, const double \*reactivities, double m, double b, unsigned int options)
 

*Add SHAPE reactivity data as soft constraints (Deigan et al. method)*
- int `vrna_sc_add_SHAPE_deigan_ali` (`vrna_fold_compound_t` \*vc, const char \*\*shape\_files, const int \*shape\_file\_association, double m, double b, unsigned int options)
 

*Add SHAPE reactivity data from files as soft constraints for consensus structure prediction (Deigan et al. method)*
- int `vrna_sc_add_SHAPE_zarringhalam` (`vrna_fold_compound_t` \*vc, const double \*reactivities, double b, double default\_value, const char \*shape\_conversion, unsigned int options)
 

*Add SHAPE reactivity data as soft constraints (Zarringhalam et al. method)*
- int `vrna_sc_SHAPE_to_pr` (const char \*shape\_conversion, double \*values, int length, double default\_value)
 

*Convert SHAPE reactivity values to probabilities for being unpaired.*

## 16.42.2 Function Documentation

### 16.42.2.1 vrna\_sc\_add\_SHAPE\_deigan()

```
int vrna_sc_add_SHAPE_deigan (
    vrna_fold_compound_t * vc,
    const double * reactivities,
    double m,
    double b,
    unsigned int options )
#include <ViennaRNA/constraints/SHAPE.h>
```

Add SHAPE reactivity data as soft constraints (Deigan et al. method)

This approach of SHAPE directed RNA folding uses the simple linear ansatz

$$\Delta G_{\text{SHAPE}}(i) = m \ln(\text{SHAPE reactivity}(i) + 1) + b$$

to convert SHAPE reactivity values to pseudo energies whenever a nucleotide  $i$  contributes to a stacked pair. A positive slope  $m$  penalizes high reactivities in paired regions, while a negative intercept  $b$  results in a confirmatory ‘bonus’ free energy for correctly predicted base pairs. Since the energy evaluation of a base pair stack involves two pairs, the pseudo energies are added for all four contributing nucleotides. Consequently, the energy term is applied twice for pairs inside a helix and only once for pairs adjacent to other structures. For all other loop types the energy model remains unchanged even when the experimental data highly disagrees with a certain motif.

#### See also

For further details, we refer to [6].

[vrna\\_sc\\_remove\(\)](#), [vrna\\_sc\\_add\\_SHAPE\\_zarringhalam\(\)](#), [vrna\\_sc\\_minimize\\_perturbation\(\)](#)

#### Parameters

|                     |   |
|---------------------|---|
| <i>vc</i>           | The <a href="#">vrna_fold_compound_t</a> the soft constraints are associated with |
| <i>reactivities</i> | A vector of normalized SHAPE reactivities   |
| <i>m</i>            | The slope of the conversion function  |
| <i>b</i>            | The intercept of the conversion function  |
| <i>options</i>      | The options flag indicating how/where to store the soft constraints               |

#### Returns

1 on successful extraction of the method, 0 on errors

**SWIG Wrapper Notes** This function is attached as method `sc_add_SHAPE_deigan()` to objects of type `fold_<-compound`

### 16.42.2.2 vrna\_sc\_add\_SHAPE\_deigan\_ali()

```
int vrna_sc_add_SHAPE_deigan_ali (
    vrna_fold_compound_t * vc,
    const char ** shape_files,
    const int * shape_file_association,
    double m,
    double b,
    unsigned int options )
#include <ViennaRNA/constraints/SHAPE.h>
```

Add SHAPE reactivity data from files as soft constraints for consensus structure prediction (Deigan et al. method)

## Parameters

|                               |  |
|-------------------------------|--|
| <i>vc</i>                     | The <code>vrna_fold_compound_t</code> the soft constraints are associated with |
| <i>shape_files</i>            | A set of filenames that contain normalized SHAPE reactivity data               |
| <i>shape_file_association</i> | An array of integers that associate the files with sequences in the alignment  |
| <i>m</i>                      | The slope of the conversion function   |
| <i>b</i>                      | The intercept of the conversion function                                       |
| <i>options</i>                | The options flag indicating how/where to store the soft constraints            |

## Returns

1 on successful extraction of the method, 0 on errors

**SWIG Wrapper Notes** This function is attached as method `sc_add_SHAPE_deigan_ali()` to objects of type `fold_compound`

**16.42.2.3 `vrna_sc_add_SHAPE_zarringhalam()`**

```
int vrna_sc_add_SHAPE_zarringhalam (
    vrna_fold_compound_t * vc,
    const double * reactivities,
    double b,
    double default_value,
    const char * shape_conversion,
    unsigned int options )
#include <ViennaRNA/constraints/SHAPE.h>
```

Add SHAPE reactivity data as soft constraints (Zarringhalam et al. method)

This method first converts the observed SHAPE reactivity of nucleotide  $i$  into a probability  $q_i$  that position  $i$  is unpaired by means of a non-linear map. Then pseudo-energies of the form

$$\Delta G_{\text{SHAPE}}(x, i) = \beta |x_i - q_i|$$

are computed, where  $x_i = 0$  if position  $i$  is unpaired and  $x_i = 1$  if  $i$  is paired in a given secondary structure. The parameter  $\beta$  serves as scaling factor. The magnitude of discrepancy between prediction and experimental observation is represented by  $|x_i - q_i|$ .

## See also

For further details, we refer to [26]

[vrna\\_sc\\_remove\(\)](#), [vrna\\_sc\\_add\\_SHAPE\\_deigan\(\)](#), [vrna\\_sc\\_minimize\\_perturbation\(\)](#)

## Parameters

|                         |  |
|-------------------------|--|
| <i>vc</i>               | The <code>vrna_fold_compound_t</code> the soft constraints are associated with |
| <i>reactivities</i>     | A vector of normalized SHAPE reactivities                                      |
| <i>b</i>                | The scaling factor $\beta$ of the conversion function                          |
| <i>default_value</i>    | The default value for a nucleotide where reactivity data is missing for        |
| <i>shape_conversion</i> | A flag that specifies how to convert reactivities to probabilities             |
| <i>options</i>          | The options flag indicating how/where to store the soft constraints            |

**Returns**

1 on successful extraction of the method, 0 on errors

**SWIG Wrapper Notes** This function is attached as method `sc_add_SHAPE_zarringhalam()` to objects of type `fold_compound`

**16.42.2.4 vrna\_sc\_SHAPE\_to\_pr()**

```
int vrna_sc_SHAPE_to_pr (
    const char * shape_conversion,
    double * values,
    int length,
    double default_value )
#include <ViennaRNA/constraints/SHAPE.h>
```

Convert SHAPE reactivity values to probabilities for being unpaired.

This function parses the informations from a given file and stores the result in the preallocated string sequence and the `FLT_OR_DBL` array values.

**See also**

[vrna\\_file\\_SHAPE\\_read\(\)](#)

**Parameters**

|                               |   |
|-------------------------------|---|
| <code>shape_conversion</code> | String definining the method used for the conversion process    |
| <code>values</code>           | Pointer to an array of SHAPE reactivities                       |
| <code>length</code>           | Length of the array of SHAPE reactivities                       |
| <code>default_value</code>    | Result used for position with invalid/missing reactivity values |

**16.43 Generate Soft Constraints from Data**

Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of neccessary adjustments.

**16.43.1 Detailed Description**

Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of neccessary adjustments.

Collaboration diagram for Generate Soft Constraints from Data:

**Files**

- file [perturbation\\_fold.h](#)

*Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of neccessary adjustments.*

**Macros**

- `#define VRNA_OBJECTIVE_FUNCTION_QUADRATIC 0`  
*Use the sum of squared aberrations as objective function.*
- `#define VRNA_OBJECTIVE_FUNCTION_ABSOLUTE 1`  
*Use the sum of absolute aberrations as objective function.*
- `#define VRNA_MINIMIZER_DEFAULT 0`

*Use a custom implementation of the gradient descent algorithm to minimize the objective function.*

- `#define VRNA_MINIMIZER_CONJUGATE_FR 1`  
*Use the GNU Scientific Library implementation of the Fletcher-Reeves conjugate gradient algorithm to minimize the objective function.*
- `#define VRNA_MINIMIZER_CONJUGATE_PR 2`  
*Use the GNU Scientific Library implementation of the Polak-Ribiere conjugate gradient algorithm to minimize the objective function.*
- `#define VRNA_MINIMIZER_VECTOR_BFGS 3`  
*Use the GNU Scientific Library implementation of the vector Broyden-Fletcher-Goldfarb-Shanno algorithm to minimize the objective function.*
- `#define VRNA_MINIMIZER_VECTOR_BFGS2 4`  
*Use the GNU Scientific Library implementation of the vector Broyden-Fletcher-Goldfarb-Shanno algorithm to minimize the objective function.*
- `#define VRNA_MINIMIZER_STEEPEST_DESCENT 5`  
*Use the GNU Scientific Library implementation of the steepest descent algorithm to minimize the objective function.*

## Typedefs

- `typedef void(* progress_callback) (int iteration, double score, double *epsilon)`  
*Callback for following the progress of the minimization process.*

## Functions

- `void vrna_sc_minimize_perturbation (vrna_fold_compound_t *vc, const double *q_prob_unpaired, int objective_function, double sigma_squared, double tau_squared, int algorithm, int sample_size, double *epsilon, double initialStepSize, double minStepSize, double minImprovement, double minimizerTolerance, progress_callback callback)`  
*Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments.*

### 16.43.2 Macro Definition Documentation

#### 16.43.2.1 VRNA\_OBJECTIVE\_FUNCTION\_QUADRATIC

```
#define VRNA_OBJECTIVE_FUNCTION_QUADRATIC 0
#include <ViennaRNA/perturbation_fold.h>
Use the sum of squared aberrations as objective function.
```

$$F(\vec{\epsilon}) = \sum_{i=1}^n \frac{\epsilon_i^2}{\tau^2} + \sum_{i=1}^n \frac{(p_i(\vec{\epsilon}) - q_i)^2}{\sigma^2} \rightarrow \min$$

#### 16.43.2.2 VRNA\_OBJECTIVE\_FUNCTION\_ABSOLUTE

```
#define VRNA_OBJECTIVE_FUNCTION_ABSOLUTE 1
#include <ViennaRNA/perturbation_fold.h>
Use the sum of absolute aberrations as objective function.
```

$$F(\vec{\epsilon}) = \sum_{i=1}^n \frac{|\epsilon_i|}{\tau^2} + \sum_{i=1}^n \frac{|p_i(\vec{\epsilon}) - q_i|}{\sigma^2} \rightarrow \min$$

#### 16.43.2.3 VRNA\_MINIMIZER\_CONJUGATE\_FR

```
#define VRNA_MINIMIZER_CONJUGATE_FR 1
#include <ViennaRNA/perturbation_fold.h>
Use the GNU Scientific Library implementation of the Fletcher-Reeves conjugate gradient algorithm to minimize the objective function.
Please note that this algorithm can only be used when the GNU Scientific Library is available on your system
```

#### 16.43.2.4 VRNA\_MINIMIZER\_CONJUGATE\_PR

```
#define VRNA_MINIMIZER_CONJUGATE_PR 2
#include <ViennaRNA/perturbation_fold.h>
```

Use the GNU Scientific Library implementation of the Polak-Ribiere conjugate gradient algorithm to minimize the objective function.

Please note that this algorithm can only be used when the GNU Scientific Library is available on your system

#### 16.43.2.5 VRNA\_MINIMIZER\_VECTOR\_BFGS

```
#define VRNA_MINIMIZER_VECTOR_BFGS 3
#include <ViennaRNA/perturbation_fold.h>
```

Use the GNU Scientific Library implementation of the vector Broyden-Fletcher-Goldfarb-Shanno algorithm to minimize the objective function.

Please note that this algorithm can only be used when the GNU Scientific Library is available on your system

#### 16.43.2.6 VRNA\_MINIMIZER\_VECTOR\_BFGS2

```
#define VRNA_MINIMIZER_VECTOR_BFGS2 4
#include <ViennaRNA/perturbation_fold.h>
```

Use the GNU Scientific Library implementation of the vector Broyden-Fletcher-Goldfarb-Shanno algorithm to minimize the objective function.

Please note that this algorithm can only be used when the GNU Scientific Library is available on your system

#### 16.43.2.7 VRNA\_MINIMIZER\_STEEPEST\_DESCENT

```
#define VRNA_MINIMIZER_STEEPEST_DESCENT 5
#include <ViennaRNA/perturbation_fold.h>
```

Use the GNU Scientific Library implementation of the steepest descent algorithm to minimize the objective function.

Please note that this algorithm can only be used when the GNU Scientific Library is available on your system

### 16.43.3 Typedef Documentation

#### 16.43.3.1 progress\_callback

```
typedef void(* progress_callback) (int iteration, double score, double *epsilon)
#include <ViennaRNA/perturbation_fold.h>
```

Callback for following the progress of the minimization process.

#### Parameters

|                  |   |
|------------------|---|
| <i>iteration</i> | The number of the current iteration                 |
| <i>score</i>     | The score of the objective function                 |
| <i>epsilon</i>   | The perturbation vector yielding the reported score |

### 16.43.4 Function Documentation

#### 16.43.4.1 vrna\_sc\_minimize\_perturbation()

```
void vrna_sc_minimize_perturbation (
    vrna_fold_compound_t * vc,
    const double * q_prob_unpaired,
    int objective_function,
    double sigma_squared,
```

```

    double tau_squared,
    int algorithm,
    int sample_size,
    double * epsilon,
    double initialStepSize,
    double minStepSize,
    double minImprovement,
    double minimizerTolerance,
    progress_callback callback )
#include <ViennaRNA/perturbation_fold.h>

```

Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments.

Use an iterative minimization algorithm to find a vector of perturbation energies whose incorporation as soft constraints shifts the predicted pairing probabilities closer to the experimentally observed probabilities. The algorithm aims to minimize an objective function that penalizes discrepancies between predicted and observed pairing probabilities and energy model adjustments, i.e. an appropriate vector of perturbation energies satisfies

$$F(\vec{\epsilon}) = \sum_{\mu} \frac{\epsilon_{\mu}^2}{\tau^2} + \sum_{i=1}^n \frac{(p_i(\vec{\epsilon}) - q_i)^2}{\sigma^2} \rightarrow \min .$$

An initialized fold compound and an array containing the observed probability for each nucleotide to be unbound are required as input data. The parameters `objective_function`, `sigma_squared` and `tau_squared` are responsible for adjusting the aim of the objective function. Dependend on which type of objective function is selected, either squared or absolute aberrations are contributing to the objective function. The ratio of the parameters `sigma_squared` and `tau_squared` can be used to adjust the algorithm to find a solution either close to the thermodynamic prediction (`sigma_squared >> tau_squared`) or close to the experimental data (`tau_squared >> sigma_squared`). The minimization can be performed by makeing use of a custom gradient descent implementation or using one of the minimizing algorithms provided by the GNU Scientific Library. All algorithms require the evaluation of the gradient of the objective function, which includes the evaluation of conditional pairing probabilites. Since an exact evaluation is expensive, the probabilities can also be estimated from sampling by setting an appropriate sample size. The found vector of perturbation energies will be stored in the array `epsilon`. The progress of the minimization process can be tracked by implementing and passing a callback function.

#### See also

For further details we refere to [23].

#### Parameters

|                                 |   |
|---------------------------------|---|
| <code>vc</code>                 | Pointer to a fold compound  |
| <code>q_prob_unpaired</code>    | Pointer to an array containing the probability to be unpaired for each nucleotide   |
| <code>objective_function</code> | The type of objective function to be used (VRNA_OBJECTIVE_FUNCTION_QUADRATIC / VRNA_OBJECTIVE_FUNCTION_LINEAR)  |
| <code>sigma_squared</code>      | A factor used for weighting the objective function. More weight on this factor will lead to a solution close to the null vector.                                    |
| <code>tau_squared</code>        | A factor used for weighting the objective function. More weight on this factor will lead to a solution close to the data provided in <code>q_prob_unpaired</code> . |
| <code>algorithm</code>          | The minimization algorithm (VRNA_MINIMIZER_*)   |
| <code>sample_size</code>        | The number of sampled sequences used for estimating the pairing probabilities. A value $\leq 0$ will lead to an exact evaluation.                                   |
| <code>epsilon</code>            | A pointer to an array used for storing the calculated vector of perturbation energies   |
| <code>callback</code>           | A pointer to a callback function used for reporting the current minimization progress   |

## 16.44 Ligands Binding to RNA Structures

Simple Extensions to Model Ligand Binding to RNA Structures.

### 16.44.1 Detailed Description

Simple Extensions to Model Ligand Binding to RNA Structures.

Collaboration diagram for Ligands Binding to RNA Structures:

#### Modules

- [Ligands Binding to Unstructured Domains](#)

*Add ligand binding to loop regions using the [Unstructured Domains](#) feature.*

- [Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints](#)

*Ligand binding to specific hairpin/interior loop like motifs using the [Soft Constraints](#) feature.*

#### Files

- file [ligand.h](#)

*Functions for incorporation of ligands binding to hairpin and interior loop motifs using the soft constraints framework.*

## 16.45 Ligands Binding to Unstructured Domains

Add ligand binding to loop regions using the [Unstructured Domains](#) feature.

Add ligand binding to loop regions using the [Unstructured Domains](#) feature.

Sometime, certain ligands, like single strand binding (SSB) proteins, compete with intramolecular base pairing of the RNA. In situations, where the dissociation constant of the ligand is known and the ligand binds to a consecutive stretch of single-stranded nucleotides we can use the [Unstructured Domains](#) functionality to extend the RNA folding grammar. This module provides a convenience default implementation that covers most of the application scenarios. The function [vrna\\_ud\\_add\\_motif\(\)](#) attaches a ligands sequence motif and corresponding binding free energy to the list of known ligand motifs within a [vrna\\_fold\\_compound\\_t.domains\\_up](#) attribute. The first call to this function initializes the [Unstructured Domains](#) feature with our default implementation. Subsequent calls of secondary structure prediction algorithms with the modified [vrna\\_fold\\_compound\\_t](#) then directly include the competition of the ligand with regules base pairing. Since we utilize the unstructured domain extension, The ligand binding model can be removed again using the [vrna\\_ud\\_remove\(\)](#) function. Collaboration diagram for Ligands Binding to Unstructured Domains:

## 16.46 Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints

Ligand binding to specific hairpin/interior loop like motifs using the [Soft Constraints](#) feature.

### 16.46.1 Detailed Description

Ligand binding to specific hairpin/interior loop like motifs using the [Soft Constraints](#) feature.

Collaboration diagram for Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints:

#### Data Structures

- struct [vrna\\_sc\\_motif\\_s](#)

#### Typedefs

- typedef struct [vrna\\_sc\\_motif\\_s](#) [vrna\\_sc\\_motif\\_t](#)

*Type definition for soft constraint motif.*

## Functions

- int `vrna_sc_add_hi_motif` (`vrna_fold_compound_t *fc, const char *seq, const char *structure, FLT_OR_DBL energy, unsigned int options)`

Add soft constraints for hairpin or interior loop binding motif.

### 16.46.2 Data Structure Documentation

#### 16.46.2.1 struct vrna\_sc\_motif\_s

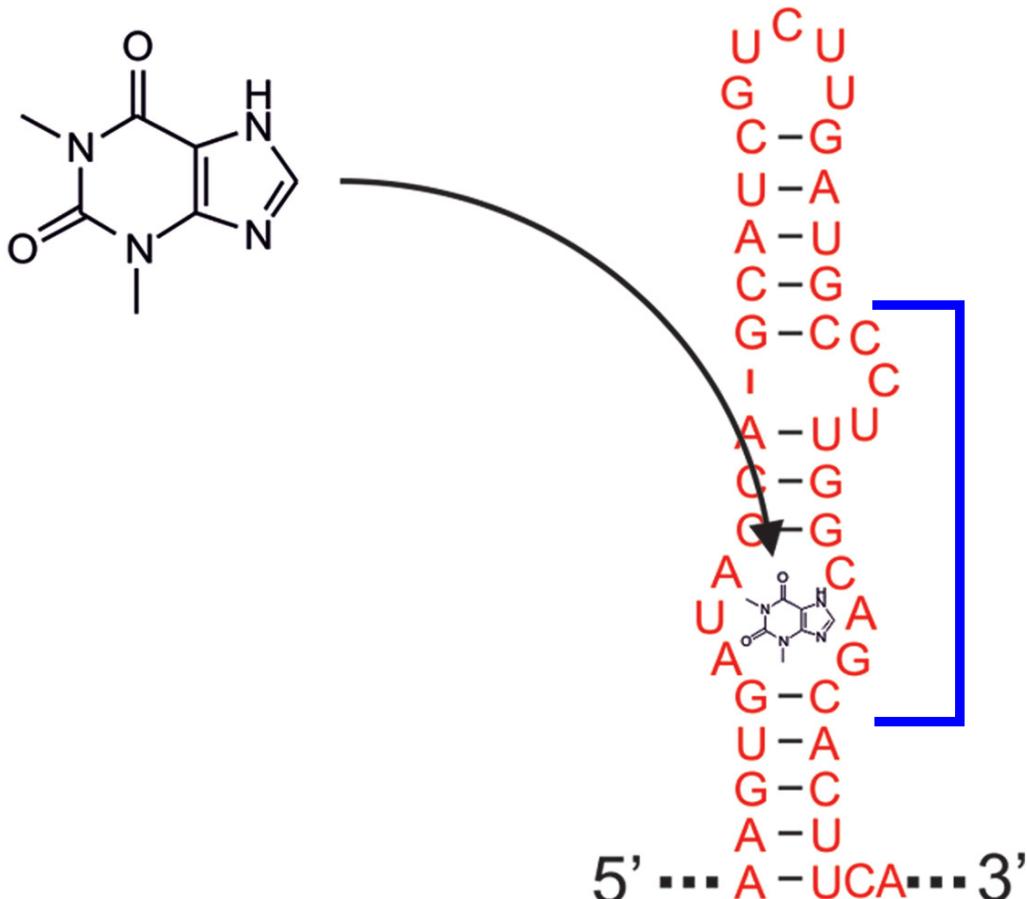
### 16.46.3 Function Documentation

#### 16.46.3.1 vrna\_sc\_add\_hi\_motif()

```
int vrna_sc_add_hi_motif (
    vrna_fold_compound_t * fc,
    const char * seq,
    const char * structure,
    FLT_OR_DBL energy,
    unsigned int options )
#include <ViennaRNA/constraints/ligand.h>
```

Add soft constraints for hairpin or interior loop binding motif.

Here is an example that adds a theophylline binding motif. Free energy contribution is derived from  $k_d = 0.1\mu M$ , taken from Jenison et al. 1994. At 1M concentration the corresponding binding free energy amounts to  $-9.93 \text{ kcal/mol}$ .



```
vrna_sc_add_hi_motif(fc,
                      "GAUACCAG&CCCUUGGCAGC",
```

```
" . . . ( ( ( & ) . . . ) ) . . . )",
-9.93, VRNA_OPTION_DEFAULT);
```

**Parameters**

|                  |   |
|------------------|---|
| <i>fc</i>        | The <code>vrna_fold_compound_t</code> the motif is applied to |
| <i>seq</i>       | The sequence motif (may be interspaced by '&' character)      |
| <i>structure</i> | The structure motif (may be interspaced by '&' character)     |
| <i>energy</i>    | The free energy of the motif (e.g. binding free energy)       |
| <i>options</i>   | Options   |

**Returns**

non-zero value if application of the motif using soft constraints was successful

**SWIG Wrapper Notes** This function is attached as method `sc_add_hi_motif()` to objects of type `fold_compound`

## 16.47 Structure Modules and Pseudoknots

### 16.47.1 Detailed Description

Collaboration diagram for Structure Modules and Pseudoknots:

#### Modules

- [Pseudoknots](#)  
*Implementations to predict pseudoknotted structures.*
- [G-Quadruplexes](#)  
*Various functions related to G-quadruplex computations.*

#### Files

- file [gquad.h](#)  
*G-quadruplexes.*

## 16.48 Pseudoknots

Implementations to predict pseudoknotted structures.

### 16.48.1 Detailed Description

Implementations to predict pseudoknotted structures.

Collaboration diagram for Pseudoknots:

#### Files

- file [pk\\_plex.h](#)  
*Heuristics for two-step pseudoknot forming interaction predictions.*

#### Data Structures

- struct [vrna\\_pk\\_plex\\_result\\_s](#)  
*A result of the RNA PKplex interaction prediction. [More...](#)*

## Typedefs

- `typedef int() vrna_callback_pk_plex_score(const short *pt, int start_5, int end_5, int start_3, int end_3, void *data)`  
*Pseudoknot loop scoring function prototype.*
- `typedef struct vrna_pk_plex_option_s * vrna_pk_plex_opt_t`  
*RNA PKplex options object.*
- `typedef struct vrna_pk_plex_result_s vrna_pk_plex_t`  
*Convenience typedef for results of the RNA PKplex prediction.*

## Functions

- `vrna_pk_plex_t * vrna_pk_plex (vrna_fold_compound_t *fc, const int **accessibility, vrna_pk_plex_opt_t options)`  
*Predict Pseudoknot interactions in terms of a two-step folding process.*
- `int ** vrna_pk_plex_accessibility (const char *sequence, unsigned int unpaired, double cutoff)`  
*Obtain a list of opening energies suitable for PKplex computations.*
- `vrna_pk_plex_opt_t vrna_pk_plex_opt_defaults (void)`  
*Default options for PKplex algorithm.*
- `vrna_pk_plex_opt_t vrna_pk_plex_opt (unsigned int delta, unsigned int max_interaction_length, int pk← penalty)`  
*Simple options for PKplex algorithm.*
- `vrna_pk_plex_opt_t vrna_pk_plex_opt_fun (unsigned int delta, unsigned int max_interaction_length, vrna_callback_pk_plex_score *scoring_function, void *scoring_data)`  
*Simple options for PKplex algorithm.*

### 16.48.2 Data Structure Documentation

#### 16.48.2.1 struct vrna\_pk\_plex\_result\_s

A result of the RNA PKplex interaction prediction.

See also

[vrna\\_pk\\_plex\\_t](#)

#### Data Fields

- `char * structure`  
*Secondary Structure in dot-bracket notation.*
- `double energy`  
*Net free energy in kcal/mol.*
- `double dGpk`  
*Free energy of PK loop in kcal/mol.*
- `double dGint`  
*Free energy of PK forming duplex interaction.*
- `double dG1`  
*Opening energy for the 5' interaction site used in the heuristic.*
- `double dG2`  
*Opening energy for the 3' interaction site used in the heuristic.*
- `unsigned int start_5`  
*Start coordinate of the 5' interaction site.*
- `unsigned int end_5`  
*End coordinate of the 5' interaction site.*
- `unsigned int start_3`  
*Start coordinate of the 3' interaction site.*
- `unsigned int end_3`  
*End coordinate of the 3' interaction site.*

### 16.48.3 Typedef Documentation

#### 16.48.3.1 vrna\_callback\_pk\_plex\_score

```
typedef int() vrna_callback_pk_plex_score(const short *pt, int start_5, int end_5, int start_3, int end_3, void *data)
#include <ViennaRNA/pk_plex.h>
```

Pseudoknot loop scoring function prototype.

This function is used to evaluate a formed pseudoknot (PK) interaction in [vrna\\_pk\\_plex\(\)](#). It is supposed to take a PK-free secondary structure as input and coordinates of an additional interaction site. From this data, the energy (penalty) to score the PK loop is derived and returned in decakal/mol. Upon passing zero in any of the interaction site coordinates (`start_5`, `end_5`, `start_3`, `end_3`) or a *NULL* pointer in `pt`, the function must return a PK loop score. This minimum PK loop score is used in the first phase of the heuristic implemented in [vrna\\_pk\\_plex\(\)](#) to assess whether a particular interaction is further taken into account in a later, more thorough evaluation step. The simplest scoring function would simply return a constant score for any PK loop, no matter what type of loop is formed and how large the loop is. This is the default if [vrna\\_pk\\_plex\\_opt\\_defaults\(\)](#) or [vrna\\_pk\\_plex\\_opt\(\)](#) is used to generate options for [vrna\\_pk\\_plex\(\)](#).

See also

[vrna\\_pk\\_plex\\_opt\\_fun\(\)](#), [vrna\\_pk\\_plex\(\)](#)

#### Parameters

|                      |   |
|----------------------|---|
| <code>pt</code>      | The secondary structure (without pseudoknot) in pair table notation |
| <code>start_5</code> | The start coordinate of the 5' site of the pseudoknot interaction   |
| <code>end_5</code>   | The end coordinate of the 5' site of the pseudoknot interaction     |
| <code>start_3</code> | The start coordinate of the 3' site of the pseudoknot interaction   |
| <code>end_3</code>   | The end coordinate of the 3' site of the pseudoknot interaction     |
| <code>data</code>    | An arbitrary data structure passed from the calling function        |

#### Returns

The energy (penalty) of the resulting pseudoknot

#### 16.48.3.2 vrna\_pk\_plex\_opt\_t

```
typedef struct vrna_pk_plex_option_s* vrna_pk_plex_opt_t
#include <ViennaRNA/pk_plex.h>
```

RNA PKplex options object.

See also

[vrna\\_pk\\_plex\\_opt\\_defaults\(\)](#), [vrna\\_pk\\_plex\\_opt\(\)](#), [vrna\\_pk\\_plex\\_opt\\_fun\(\)](#), [vrna\\_pk\\_plex\(\)](#), [vrna\\_callback\\_pk\\_plex\\_score](#)

#### 16.48.3.3 vrna\_pk\_plex\_t

```
typedef struct vrna_pk_plex_result_s vrna_pk_plex_t
#include <ViennaRNA/pk_plex.h>
```

Convenience typedef for results of the RNA PKplex prediction.

See also

[#vrna\\_pk\\_plex\\_results\\_s, vrna\\_pk\\_plex\(\)](#)

## 16.48.4 Function Documentation

### 16.48.4.1 vrna\_pk\_plex()

```
vrna_pk_plex_t * vrna_pk_plex (
    vrna_fold_compound_t * fc,
    const int ** accessibility,
    vrna_pk_plex_opt_t options )
#include <ViennaRNA/pk_plex.h>
```

Predict Pseudoknot interactions in terms of a two-step folding process.

Computes simple pseudoknot interactions according to the PKplex algorithm. This simple heuristic first compiles a list of potential interaction sites that may form a pseudoknot. The resulting candidate interactions are then fixed and an PK-free MFE structure for the remainder of the sequence is computed.

The `accessibility` argument is a list of opening energies for potential interaction sites. It is used in the first step of the algorithm to identify potential interactions. Upon passing `NULL`, the opening energies are determined automatically based on the current model settings.

Depending on the `options`, the function can return the MFE (incl. PK loops) or suboptimal structures within an energy band around the MFE. The PK loop is internally scored by a scoring function that in the simplest cases assigns a constant value for each PK loop. More complicated scoring functions can be passed as well, see [vrna\\_callback\\_pk\\_plex\\_score](#) and [vrna\\_pk\\_plex\\_opt\\_fun\(\)](#).

The function returns `NULL` on any error. Otherwise, a list of structures and interaction coordinates with corresponding energy contributions is returned. If no PK-interaction that satisfies the options is found, the list only consists of the PK-free MFE structure.

#### Parameters

|                            |   |
|----------------------------|---|
| <code>fc</code>            | fold compound with the input sequence and model settings  |
| <code>accessibility</code> | An array of opening energies for the implemented heuristic (maybe <code>NULL</code> )                 |
| <code>options</code>       | An <a href="#">vrna_pk_plex_opt_t</a> options data structure that determines the algorithm parameters |

#### Returns

A list of potentially pseudoknotted structures (Last element in the list indicated by `NULL` value in [vrna\\_pk\\_plex\\_result\\_s.structure](#))

### 16.48.4.2 vrna\_pk\_plex\_accessibility()

```
int ** vrna_pk_plex_accessibility (
    const char * sequence,
    unsigned int unpaired,
    double cutoff )
#include <ViennaRNA/pk_plex.h>
```

Obtain a list of opening energies suitable for PKplex computations.

#### See also

[vrna\\_pk\\_plex\(\)](#)

#### Parameters

|                       |  |
|-----------------------|--|
| <code>sequence</code> | The RNA sequence   |
| <code>unpaired</code> | The maximum number of unpaired nucleotides, i.e. length of interaction |
| <code>cutoff</code>   | A cutoff value for unpaired probabilities                              |

**Returns**

Opening energies as required for [vrna\\_pk\\_plex\(\)](#)

**16.48.4.3 vrna\_pk\_plex\_opt\_defaults()**

```
vrna_pk_plex_opt_t vrna_pk_plex_opt_defaults (
    void )
#include <ViennaRNA/pk_plex.h>
```

Default options for PKplex algorithm.

**See also**

[vrna\\_pk\\_plex\(\)](#), [vrna\\_pk\\_plex\\_opt\(\)](#), [vrna\\_pk\\_plex\\_opt\\_fun\(\)](#)

**Returns**

An options data structure suitable for PKplex computations

**16.48.4.4 vrna\_pk\_plex\_opt()**

```
vrna_pk_plex_opt_t vrna_pk_plex_opt (
    unsigned int delta,
    unsigned int max_interaction_length,
    int pk_penalty )
#include <ViennaRNA/pk_plex.h>
```

Simple options for PKplex algorithm.

**See also**

[vrna\\_pk\\_plex\(\)](#), [vrna\\_pk\\_plex\\_opt\\_defaults\(\)](#), [vrna\\_pk\\_plex\\_opt\\_fun\(\)](#)

**Parameters**

|                               |   |
|-------------------------------|---|
| <i>delta</i>                  | Size of energy band around MFE for suboptimal results in dekalcal/mol |
| <i>max_interaction_length</i> | Maximum length of interaction   |
| <i>pk_penalty</i>             | Energy constant to score the PK forming loop                          |

**Returns**

An options data structure suitable for PKplex computations

**16.48.4.5 vrna\_pk\_plex\_opt\_fun()**

```
vrna_pk_plex_opt_t vrna_pk_plex_opt_fun (
    unsigned int delta,
    unsigned int max_interaction_length,
    vrna_callback_pk_plex_score * scoring_function,
    void * scoring_data )
#include <ViennaRNA/pk_plex.h>
```

Simple options for PKplex algorithm.

**See also**

[vrna\\_pk\\_plex\(\)](#), [vrna\\_pk\\_plex\\_opt\\_defaults\(\)](#), [vrna\\_pk\\_plex\\_opt\(\)](#), [vrna\\_callback\\_pk\\_plex\\_score](#)

**Parameters**

|                               |  |
|-------------------------------|--|
| <i>delta</i>                  | Size of energy band around MFE for suboptimal results in dekalcal/mol          |
| <i>max_interaction_length</i> | Maximum length of interaction  |
| <i>scoring_function</i>       | Energy evaluating function to score the PK forming loop                        |
| <i>scoring_data</i>           | An arbitrary data structure passed to the scoring function (maybe <i>NUL</i> ) |

**Returns**

An options data structure suitable for PKplex computations

## 16.49 G-Quadruplexes

Various functions related to G-quadruplex computations.

### 16.49.1 Detailed Description

Various functions related to G-quadruplex computations.

Collaboration diagram for G-Quadruplexes:

## Functions

- int \* [get\\_gquad\\_matrix](#) (short \*S, [vrna\\_param\\_t](#) \*P)  
*Get a triangular matrix prefilled with minimum free energy contributions of G-quadruplexes.*
- int [parse\\_gquad](#) (const char \*struc, int \*L, int I[3])
- PRIVATE int [backtrack\\_GQuad\\_IntLoop](#) (int c, int i, int j, int type, short \*S, int \*ggg, int \*index, int \*p, int \*q, [vrna\\_param\\_t](#) \*P)
- PRIVATE int [backtrack\\_GQuad\\_IntLoop\\_L](#) (int c, int i, int j, int type, short \*S, int \*\*ggg, int maxdist, int \*p, int \*q, [vrna\\_param\\_t](#) \*P)

### 16.49.2 Function Documentation

#### 16.49.2.1 [get\\_gquad\\_matrix\(\)](#)

```
int * get_gquad_matrix (
    short * S,
    vrna\_param\_t * P )
#include <ViennaRNA/gquad.h>
```

Get a triangular matrix prefilled with minimum free energy contributions of G-quadruplexes.

At each position ij in the matrix, the minimum free energy of any G-quadruplex delimited by i and j is stored. If no G-quadruplex formation is possible, the matrix element is set to INF. Access the elements in the matrix via matrix[indx[j]+i]. To get the integer array indx see [get\\_jindx\(\)](#).

See also

[get\\_jindx\(\)](#), [encode\\_sequence\(\)](#)

**Parameters**

|          |   |
|----------|---|
| <i>S</i> | The encoded sequence  |
| <i>P</i> | A pointer to the data structure containing the precomputed energy contributions |

**Returns**

A pointer to the G-quadruplex contribution matrix

**16.49.2.2 parse\_gquad()**

```
int parse_gquad (
    const char * struc,
    int * L,
    int l[3] )
#include <ViennaRNA/gquad.h>
given a dot-bracket structure (possibly) containing gquads encoded by '+' signs, find first gquad, return end position or 0 if none found Upon return L and l[] contain the number of stacked layers, as well as the lengths of the linker regions. To parse a string with many gquads, call parse_gquad repeatedly e.g. end1 = parse_gquad(struc, &L, l); ... ; end2 = parse_gquad(struc+end1, &L, l); end2+=end1; ... ; end3 = parse_gquad(struc+end2, &L, l); end3+=end2; ... ;
```

**16.49.2.3 backtrack\_GQuad\_IntLoop()**

```
PRIVATE int backtrack_GQuad_IntLoop (
    int c,
    int i,
    int j,
    int type,
    short * S,
    int * ggg,
    int * index,
    int * p,
    int * q,
    vrna_param_t * P )
#include <ViennaRNA/gquad.h>
backtrack an interior loop like enclosed g-quadruplex with closing pair (i,j)
```

**Parameters**

|              |  |
|--------------|--|
| <i>c</i>     | The total contribution the loop should resemble              |
| <i>i</i>     | position i of enclosing pair                                 |
| <i>j</i>     | position j of enclosing pair                                 |
| <i>type</i>  | base pair type of enclosing pair (must be reverse type)      |
| <i>S</i>     | integer encoded sequence                                     |
| <i>ggg</i>   | triangular matrix containing g-quadruplex contributions      |
| <i>index</i> | the index for accessing the triangular matrix                |
| <i>p</i>     | here the 5' position of the gquad is stored                  |
| <i>q</i>     | here the 3' position of the gquad is stored                  |
| <i>P</i>     | the datastructure containing the precalculated contributions |

**Returns**

1 on success, 0 if no gquad found

**16.49.2.4 backtrack\_GQuad\_IntLoop\_L()**

```
PRIVATE int backtrack_GQuad_IntLoop_L (
    int c,
```

```

int i,
int j,
int type,
short * S,
int ** ggg,
int maxdist,
int * p,
int * q,
vrna_param_t * P )
#include <ViennaRNA/gquad.h>
backtrack an interior loop like enclosed g-quadruplex with closing pair (i,j) with underlying Lfold matrix

```

**Parameters**

|             |  |
|-------------|--|
| <i>c</i>    | The total contribution the loop should resemble              |
| <i>i</i>    | position i of enclosing pair                                 |
| <i>j</i>    | position j of enclosing pair                                 |
| <i>type</i> | base pair type of enclosing pair (must be reverse type)      |
| <i>S</i>    | integer encoded sequence                                     |
| <i>ggg</i>  | triangular matrix containing g-quadruplex contributions      |
| <i>p</i>    | here the 5' position of the gquad is stored                  |
| <i>q</i>    | here the 3' position of the gquad is stored                  |
| <i>P</i>    | the datastructure containing the precalculated contributions |

**Returns**

1 on success, 0 if no gquad found

## 16.50 Utilities

### 16.50.1 Detailed Description

Collaboration diagram for Utilities:

### Modules

- Utilities to deal with Nucleotide Alphabets

*Functions to cope with various aspects related to the nucleotide sequence alphabet.*

- (Nucleic Acid Sequence) String Utilitites

*Functions to parse, convert, manipulate, create, and compare (nucleic acid sequence) strings.*

- Secondary Structure Utilities

*Functions to create, parse, convert, manipulate, and compare secondary structure representations.*

- Multiple Sequence Alignment Utilities

*Functions to extract features from and to manipulate multiple sequence alignments.*

- Files and I/O

*Functions to parse, write, and convert various file formats and to deal with file system related issues.*

- Plotting

*Functions for Creating Secondary Structure Plots, Dot-Plots, and More.*

- Search Algorithms

*Implementations of various search algorithms to detect strings of objects within other strings of objects.*

- Combinatorics Algorithms

*Implementations to solve various combinatorial aspects for strings of objects.*

- (Abstract) Data Structures

*All datastructures and typedefs shared among the ViennaRNA Package can be found here.*

- **Messages**

*Functions to print various kind of messages.*

- **Unit Conversion**

*Functions to convert between various physical units.*

## Files

- file **alphabet.h**

*Functions to process, convert, and generally handle different nucleotide and/or base pair alphabets.*

- file **combinatorics.h**

*Various implementations that deal with combinatorial aspects of objects.*

- file **commands.h**

*Parse and apply different commands that alter the behavior of secondary structure prediction and evaluation.*

- file **sequence.h**

*Functions and data structures related to sequence representations ..*

- file **file\_formats\_msa.h**

*Functions dealing with file formats for Multiple Sequence Alignments (MSA)*

- file **utils.h**

*Several utilities for file handling.*

- file **utils.h**

*Various utilities to assist in plotting secondary structures and consensus structures.*

- file **alignments.h**

*Various utility- and helper-functions for sequence alignments and comparative structure prediction.*

- file **basic.h**

*General utility- and helper-functions used throughout the ViennaRNA Package.*

- file **strings.h**

*General utility- and helper-functions for RNA sequence and structure strings used throughout the ViennaRNA Package.*

- file **units.h**

*Physical Units and Functions to convert them into each other.*

- file **BoyerMoore.h**

*Variants of the Boyer-Moore string search algorithm.*

- file **char\_stream.h**

*Implementation of a dynamic, buffered character stream.*

- file **stream\_output.h**

*An implementation of a buffered, ordered stream output data structure.*

## Macros

- #define **VRNA\_INPUT\_ERROR** 1U

*Output flag of `get_input_line()`: "An ERROR has occured, maybe EOF".*

- #define **VRNA\_INPUT\_QUIT** 2U

*Output flag of `get_input_line()`: "the user requested quitting the program".*

- #define **VRNA\_INPUT\_MISC** 4U

*Output flag of `get_input_line()`: "something was read".*

- #define **VRNA\_INPUT\_FASTA\_HEADER** 8U

*Input/Output flag of `get_input_line()`:*

*if used as input option this tells `get_input_line()` that the data to be read should comply with the FASTA format.*

- #define **VRNA\_INPUT\_CONSTRAINT** 32U

*Input flag for `get_input_line()`:*

*Tell `get_input_line()` that we assume to read a structure constraint.*

- `#define VRNA_INPUT_NO_TRUNCATION 256U`  
*Input switch for `get_input_line()`: "do not truncate the line by eliminating white spaces at end of line".*
- `#define VRNA_INPUT_NO_REST 512U`  
*Input switch for `vrna_file_fasta_read_record()`: "do fill rest array".*
- `#define VRNA_INPUT_NO_SPAN 1024U`  
*Input switch for `vrna_file_fasta_read_record()`: "never allow data to span more than one line".*
- `#define VRNA_INPUT_NOSKIP_BLANK_LINES 2048U`  
*Input switch for `vrna_file_fasta_read_record()`: "do not skip empty lines".*
- `#define VRNA_INPUT_BLANK_LINE 4096U`  
*Output flag for `vrna_file_fasta_read_record()`: "read an empty line".*
- `#define VRNA_INPUT_NOSKIP_COMMENTS 128U`  
*Input switch for `get_input_line()`: "do not skip comment lines".*
- `#define VRNA_INPUT_COMMENT 8192U`  
*Output flag for `vrna_file_fasta_read_record()`: "read a comment".*
- `#define MIN2(A, B) ((A) < (B) ? (A) : (B))`  
*Get the minimum of two comparable values.*
- `#define MAX2(A, B) ((A) > (B) ? (A) : (B))`  
*Get the maximum of two comparable values.*
- `#define MIN3(A, B, C) (MIN2((MIN2((A), (B))), (C)))`  
*Get the minimum of three comparable values.*
- `#define MAX3(A, B, C) (MAX2((MAX2((A), (B))), (C)))`  
*Get the maximum of three comparable values.*

## Functions

- `void * vrna_alloc (unsigned size)`  
*Allocate space safely.*
- `void * vrna_realloc (void *p, unsigned size)`  
*Reallocate space safely.*
- `void vrna_init_rand (void)`  
*Initialize seed for random number generator.*
- `void vrna_init_rand_seed (unsigned int seed)`  
*Initialize the random number generator with a pre-defined seed.*
- `double vrna_urn (void)`  
*get a random number from [0..1]*
- `int vrna_int_urn (int from, int to)`  
*Generates a pseudo random integer in a specified range.*
- `char * vrna_time_stamp (void)`  
*Get a timestamp.*
- `unsigned int get_input_line (char **string, unsigned int options)`
- `int * vrna_idx_row_wise (unsigned int length)`  
*Get an index mapper array (*iindx*) for accessing the energy matrices, e.g. in partition function related functions.*
- `int * vrna_idx_col_wise (unsigned int length)`  
*Get an index mapper array (*indx*) for accessing the energy matrices, e.g. in MFE related functions.*

## Variables

- `unsigned short xsubi [3]`  
*Current 48 bit random number.*

### 16.50.2 Macro Definition Documentation

### 16.50.2.1 VRNA\_INPUT\_FASTA\_HEADER

```
#define VRNA_INPUT_FASTA_HEADER 8U
#include <ViennaRNA/utils/basic.h>
Input/Output flag of get\_input\_line\(\):
if used as input option this tells get\_input\_line\(\) that the data to be read should comply with the FASTA format.
the function will return this flag if a fasta header was read
```

## 16.50.3 Function Documentation

### 16.50.3.1 vrna\_alloc()

```
void * vrna_alloc (
    unsigned size )
#include <ViennaRNA/utils/basic.h>
Allocate space safely.
```

#### Parameters

|             |   |
|-------------|---|
| <i>size</i> | The size of the memory to be allocated in bytes |
|-------------|---|

#### Returns

A pointer to the allocated memory

### 16.50.3.2 vrna\_realloc()

```
void * vrna_realloc (
    void * p,
    unsigned size )
#include <ViennaRNA/utils/basic.h>
Reallocate space safely.
```

#### Parameters

|             |  |
|-------------|--|
| <i>p</i>    | A pointer to the memory region to be reallocated |
| <i>size</i> | The size of the memory to be allocated in bytes  |

#### Returns

A pointer to the newly allocated memory

### 16.50.3.3 vrna\_init\_rand()

```
void vrna_init_rand (
    void )
#include <ViennaRNA/utils/basic.h>
Initialize seed for random number generator.
```

#### See also

[vrna\\_init\\_rand\\_seed\(\)](#), [vrna\\_urn\(\)](#)

#### 16.50.3.4 vrna\_init\_rand\_seed()

```
void vrna_init_rand_seed (
    unsigned int seed )
#include <ViennaRNA/utils/basic.h>
Initialize the random number generator with a pre-defined seed.
```

See also

[vrna\\_init\\_rand\(\)](#), [vrna\\_urn\(\)](#)

Parameters

|             |  |
|-------------|--|
| <i>seed</i> | The seed for the random number generator |
|-------------|--|

**SWIG Wrapper Notes** This function is available as an overloaded function [init\\_rand\(\)](#) where the argument *seed* is optional.

#### 16.50.3.5 vrna\_urn()

```
double vrna_urn (
    void )
#include <ViennaRNA/utils/basic.h>
get a random number from [0..1]
```

See also

[vrna\\_int\\_urn\(\)](#), [vrna\\_init\\_rand\(\)](#), [vrna\\_init\\_rand\\_seed\(\)](#)

Note

Usually implemented by calling *erand48()*.

Returns

A random number in range [0..1]

#### 16.50.3.6 vrna\_int\_urn()

```
int vrna_int_urn (
    int from,
    int to )
#include <ViennaRNA/utils/basic.h>
Generates a pseudo random integer in a specified range.
```

See also

[vrna\\_urn\(\)](#), [vrna\\_init\\_rand\(\)](#)

Parameters

|             |                           |
|-------------|---------------------------|
| <i>from</i> | The first number in range |
| <i>to</i>   | The last number in range  |

**Returns**

A pseudo random number in range [from, to]

**16.50.3.7 vrna\_time\_stamp()**

```
char * vrna_time_stamp (
    void )
#include <ViennaRNA/utils/basic.h>
Get a timestamp.
Returns a string containing the current date in the format
```

Fri Mar 19 21:10:57 1993

**Returns**

A string containing the timestamp

**16.50.3.8 get\_input\_line()**

```
unsigned int get_input_line (
    char ** string,
    unsigned int options )
#include <ViennaRNA/utils/basic.h>
```

Retrieve a line from 'stdin' safely while skipping comment characters and other features. This function returns the type of input it has read if recognized. An option argument allows one to switch between different reading modes. Currently available options are:

[VRNA\\_INPUT\\_COMMENT](#), [VRNA\\_INPUT\\_NOSKIP\\_COMMENTS](#), [VRNA\\_INPUT\\_NO\\_TRUNCATION](#)  
pass a collection of options as one value like this:

`get_input_line(string, option_1 | option_2 | option_n)`

If the function recognizes the type of input, it will report it in the return value. It also reports if a user defined 'quit' command (-sign on 'stdin') was given. Possible return values are:

[VRNA\\_INPUT\\_FASTA\\_HEADER](#), [VRNA\\_INPUT\\_ERROR](#), [VRNA\\_INPUT\\_MISC](#), [VRNA\\_INPUT\\_QUIT](#)

**Parameters**

|                |  |
|----------------|--|
| <i>string</i>  | A pointer to the character array that contains the line read |
| <i>options</i> | A collection of options for switching the functions behavior |

**Returns**

A flag with information about what has been read

**16.50.3.9 vrna\_idx\_row\_wise()**

```
int * vrna_idx_row_wise (
    unsigned int length )
#include <ViennaRNA/utils/basic.h>
```

Get an index mapper array (iindx) for accessing the energy matrices, e.g. in partition function related functions. Access of a position "(i,j)" is then accomplished by using

`(i,j) ~ iindx[i]-j`

This function is necessary as most of the two-dimensional energy matrices are actually one-dimensional arrays throughout the ViennaRNA Package

Consult the implemented code to find out about the mapping formula ;)

See also

[vrna\\_idx\\_col\\_wise\(\)](#)

Parameters

|               |                                |
|---------------|--------------------------------|
| <i>length</i> | The length of the RNA sequence |
|---------------|--------------------------------|

Returns

The mapper array

#### 16.50.3.10 vrna\_idx\_col\_wise()

```
int * vrna_idx_col_wise (
    unsigned int length )
#include <ViennaRNA/utils/basic.h>
```

Get an index mapper array (indx) for accessing the energy matrices, e.g. in MFE related functions.  
Access of a position "(i,j)" is then accomplished by using

```
(i, j) ~ indx[j]+i
```

This function is necessary as most of the two-dimensional energy matrices are actually one-dimensional arrays throughout the ViennaRNAPackage

Consult the implemented code to find out about the mapping formula ;)

See also

[vrna\\_idx\\_row\\_wise\(\)](#)

Parameters

|               |                                |
|---------------|--------------------------------|
| <i>length</i> | The length of the RNA sequence |
|---------------|--------------------------------|

Returns

The mapper array

### 16.50.4 Variable Documentation

#### 16.50.4.1 xsubi

```
unsigned short xsubi[3] [extern]
#include <ViennaRNA/utils/basic.h>
```

Current 48 bit random number.

This variable is used by [vrna\\_urn\(\)](#). These should be set to some random number seeds before the first call to [vrna\\_urn\(\)](#).

See also

[vrna\\_urn\(\)](#)

## 16.51 Exterior Loops

Functions to evaluate the free energy contributions for exterior loops.

### 16.51.1 Detailed Description

Functions to evaluate the free energy contributions for exterior loops.

Collaboration diagram for Exterior Loops:

#### Files

- file [external.h](#)

*Energy evaluation of exterior loops for MFE and partition function calculations.*

#### Boltzmann weight (partition function) interface

- `typedef struct vrna_mx_pf_aux_el_s * vrna_mx_pf_aux_el_t`  
*Auxiliary helper arrays for fast exterior loop computations.*
- `FLT_OR_DBL vrna_exp_E_ext_stem` (unsigned int type, int n5d, int n3d, [vrna\\_exp\\_param\\_t](#) \*p)  
*Evaluate a stem branching off the exterior loop (Boltzmann factor version)*
- `vrna_mx_pf_aux_el_t vrna_exp_E_ext_fast_init` ([vrna\\_fold\\_compound\\_t](#) \*fc)
- `void vrna_exp_E_ext_fast_rotate` ([vrna\\_mx\\_pf\\_aux\\_el\\_t](#) aux\_mx)
- `void vrna_exp_E_ext_fast_free` ([vrna\\_mx\\_pf\\_aux\\_el\\_t](#) aux\_mx)
- `FLT_OR_DBL vrna_exp_E_ext_fast` ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j, [vrna\\_mx\\_pf\\_aux\\_el\\_t](#) aux\_mx)
- `void vrna_exp_E_ext_fast_update` ([vrna\\_fold\\_compound\\_t](#) \*fc, int j, [vrna\\_mx\\_pf\\_aux\\_el\\_t](#) aux\_mx)

#### Basic free energy interface

- `int vrna_E_ext_stem` (unsigned int type, int n5d, int n3d, [vrna\\_param\\_t](#) \*p)  
*Evaluate a stem branching off the exterior loop.*
- `int vrna_eval_ext_stem` ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j)  
*Evaluate the free energy of a base pair in the exterior loop.*
- `int vrna_E_ext_loop_5` ([vrna\\_fold\\_compound\\_t](#) \*fc)
- `int vrna_E_ext_loop_3` ([vrna\\_fold\\_compound\\_t](#) \*fc, int i)

### 16.51.2 Typedef Documentation

#### 16.51.2.1 [vrna\\_mx\\_pf\\_aux\\_el\\_t](#)

```
typedef struct vrna_mx_pf_aux_el_s* vrna_mx_pf_aux_el_t
#include <ViennaRNA/loops/external.h>
Auxiliary helper arrays for fast exterior loop computations.
```

#### See also

[vrna\\_exp\\_E\\_ext\\_fast\\_init\(\)](#), [vrna\\_exp\\_E\\_ext\\_fast\\_rotate\(\)](#), [vrna\\_exp\\_E\\_ext\\_fast\\_free\(\)](#), [vrna\\_exp\\_E\\_ext\\_fast\(\)](#)

### 16.51.3 Function Documentation

#### 16.51.3.1 [vrna\\_E\\_ext\\_stem\(\)](#)

```
int vrna_E_ext_stem (
    unsigned int type,
    int n5d,
    int n3d,
    vrna_param_t * p )
#include <ViennaRNA/loops/external.h>
```

Evaluate a stem branching off the exterior loop.

Given a base pair  $(i, j)$  encoded by *type*, compute the energy contribution including dangling-end/terminal-mismatch contributions. Instead of returning the energy contribution per-se, this function returns the corresponding Boltzmann factor. If either of the adjacent nucleotides  $(i - 1)$  and  $(j + 1)$  must not contribute stacking energy, the corresponding encoding must be  $-1$ .

See also

`vrna_E_exp_stem()`

#### Parameters

|             |   |
|-------------|---|
| <i>type</i> | The base pair encoding  |
| <i>n5d</i>  | The encoded nucleotide directly adjacent at the 5' side of the base pair (may be $-1$ ) |
| <i>n3d</i>  | The encoded nucleotide directly adjacent at the 3' side of the base pair (may be $-1$ ) |
| <i>p</i>    | The pre-computed energy parameters  |

#### Returns

The energy contribution of the introduced exterior-loop stem

#### 16.51.3.2 `vrna_eval_ext_stem()`

```
int vrna_eval_ext_stem (
    vrna_fold_compound_t * fc,
    int i,
    int j )
#include <ViennaRNA/loops/external.h>
```

Evaluate the free energy of a base pair in the exterior loop.

Evaluate the free energy of a base pair connecting two nucleotides in the exterior loop and take hard constraints into account.

Typically, this is simply dangling end contributions of the adjacent nucleotides, potentially a terminal A-U mismatch penalty, and maybe some generic soft constraint contribution for that decomposition.

#### Note

For *dangles* == 1 || 3 this function also evaluates the three additional pairs  $(i + 1, j)$ ,  $(i, j - 1)$ , and  $(i + 1, j - 1)$  and returns the minimum for all four possibilities in total.

#### Parameters

|           |   |
|-----------|---|
| <i>fc</i> | Fold compound to work on (defines the model and parameters) |
| <i>i</i>  | 5' position of the base pair                                |
| <i>j</i>  | 3' position of the base pair                                |

#### Returns

Free energy contribution that arises when this pair is formed in the exterior loop

#### 16.51.3.3 `vrna_exp_E_ext_stem()`

```
FLT_OR_DBL vrna_exp_E_ext_stem (
    unsigned int type,
    int n5d,
```

```

int n3d,
    vrna_exp_param_t * p )
#include <ViennaRNA/loops/external.h>
Evaluate a stem branching off the exterior loop (Boltzmann factor version)
Given a base pair  $(i, j)$  encoded by type, compute the energy contribution including dangling-end/terminal-mismatch contributions. Instead of returning the energy contribution per-se, this function returns the corresponding Boltzmann factor. If either of the adjacent nucleotides  $(i - 1)$  and  $(j + 1)$  must not contribute stacking energy, the corresponding encoding must be  $-1$ .

```

#### See also

[vrna\\_E\\_ext\\_stem\(\)](#)

#### Parameters

|             |   |
|-------------|---|
| <i>type</i> | The base pair encoding  |
| <i>n5d</i>  | The encoded nucleotide directly adjacent at the 5' side of the base pair (may be $-1$ ) |
| <i>n3d</i>  | The encoded nucleotide directly adjacent at the 3' side of the base pair (may be $-1$ ) |
| <i>p</i>    | The pre-computed energy parameters (Boltzmann factor version)                           |

#### Returns

The Boltzmann weighted energy contribution of the introduced exterior-loop stem

## 16.52 Hairpin Loops

Functions to evaluate the free energy contributions for hairpin loops.

### 16.52.1 Detailed Description

Functions to evaluate the free energy contributions for hairpin loops.

Collaboration diagram for Hairpin Loops:

#### Files

- file [hairpin.h](#)  
*Energy evaluation of hairpin loops for MFE and partition function calculations.*

#### Basic free energy interface

- int [vrna\\_E\\_hp\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j)  
*Evaluate the free energy of a hairpin loop and consider hard constraints if they apply.*
- int [vrna\\_E\\_ext\\_hp\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j)  
*Evaluate the free energy of an exterior hairpin loop and consider possible hard constraints.*
- int [vrna\\_eval\\_ext\\_hp\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j)  
*Evaluate free energy of an exterior hairpin loop.*
- int [vrna\\_eval\\_hp\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j)  
*Evaluate free energy of a hairpin loop.*
- PRIVATE int [E\\_Hairpin](#) (int size, int type, int si1, int sj1, const char \*string, [vrna\\_param\\_t](#) \*P)  
*Compute the Energy of a hairpin-loop.*

## Boltzmann weight (partition function) interface

- PRIVATE `FLT_OR_DBL exp_E_Hairpin (int u, int type, short si1, short sj1, const char *string, vrna_exp_param_t *P)`  
 $e^{-\Delta G/kT}$  of a hairpin loop.
- `FLT_OR_DBL vrna_exp_E_hp_loop (vrna_fold_compound_t *fc, int i, int j)`  
High-Level function for hairpin loop energy evaluation (partition function variant)

### 16.52.2 Function Documentation

#### 16.52.2.1 vrna\_E\_hp\_loop()

```
int vrna_E_hp_loop (
    vrna_fold_compound_t * fc,
    int i,
    int j )
#include <ViennaRNA/loops/hairpin.h>
```

Evaluate the free energy of a hairpin loop and consider hard constraints if they apply.

This function evaluates the free energy of a hairpin loop

In case the base pair is not allowed due to a constraint conflict, this function returns `INF`.

#### Note

This function is polymorphic! The provided `vrna_fold_compound_t` may be of type `VRNA_FC_TYPE_SINGLE` or `VRNA_FC_TYPE_COMPARATIVE`

#### Parameters

|                 |   |
|-----------------|---|
| <code>fc</code> | The <code>vrna_fold_compound_t</code> that stores all relevant model settings         |
| <code>i</code>  | The 5' nucleotide of the base pair (3' to evaluate the pair as exterior hairpin loop) |
| <code>j</code>  | The 3' nucleotide of the base pair (5' to evaluate the pair as exterior hairpin loop) |

#### Returns

The free energy of the hairpin loop in 10cal/mol

#### 16.52.2.2 vrna\_E\_ext\_hp\_loop()

```
int vrna_E_ext_hp_loop (
    vrna_fold_compound_t * fc,
    int i,
    int j )
#include <ViennaRNA/loops/hairpin.h>
```

Evaluate the free energy of an exterior hairpin loop and consider possible hard constraints.

#### Note

This function is polymorphic! The provided `vrna_fold_compound_t` may be of type `VRNA_FC_TYPE_SINGLE` or `VRNA_FC_TYPE_COMPARATIVE`

#### 16.52.2.3 vrna\_eval\_hp\_loop()

```
int vrna_eval_hp_loop (
    vrna_fold_compound_t * fc,
```

```

    int i,
    int j )
#include <ViennaRNA/loops/hairpin.h>
Evaluate free energy of a hairpin loop.
```

**Note**

This function is polymorphic! The provided `vrna_fold_compound_t` may be of type `VRNA_FC_TYPE_SINGLE` or `VRNA_FC_TYPE_COMPARATIVE`

**Parameters**

|                 |  |
|-----------------|--|
| <code>fc</code> | The <code>vrna_fold_compound_t</code> for the particular energy evaluation |
| <code>i</code>  | 5'-position of the base pair   |
| <code>j</code>  | 3'-position of the base pair   |

**Returns**

Free energy of the hairpin loop closed by  $(i, j)$  in deka-kal/mol

**SWIG Wrapper Notes** This function is attached as method `eval_hp_loop()` to objects of type `fold_compound`

**16.52.2.4 E\_Hairpin()**

```

PRIVATE int E_Hairpin (
    int size,
    int type,
    int sil,
    int sjl,
    const char * string,
    vrna_param_t * P )
#include <ViennaRNA/loops/hairpin.h>
```

Compute the Energy of a hairpin-loop.

To evaluate the free energy of a hairpin-loop, several parameters have to be known. A general hairpin-loop has this structure:

```

      a3 a4
      a2     a5
      a1     a6
      X - Y
      |   |
  5'   3'
```

where X-Y marks the closing pair [e.g. a (G,C) pair]. The length of this loop is 6 as there are six unpaired nucleotides (a1-a6) enclosed by (X,Y). The 5' mismatching nucleotide is a1 while the 3' mismatch is a6. The nucleotide sequence of this loop is "a1.a2.a3.a4.a5.a6"

**Note**

The parameter sequence should contain the sequence of the loop in capital letters of the nucleic acid alphabet if the loop size is below 7. This is useful for unusually stable tri-, tetra- and hexa-loops which are treated differently (based on experimental data) if they are tabulated.

**See also**

`scale_parameters()`  
`vrna_param_t`

**Warning**

Not (really) thread safe! A threadsafe implementation will replace this function in a future release!  
 Energy evaluation may change due to updates in global variable "tetra\_loop"

**Parameters**

|               |   |
|---------------|---|
| <i>size</i>   | The size of the loop (number of unpaired nucleotides)                                   |
| <i>type</i>   | The pair type of the base pair closing the hairpin                                      |
| <i>si1</i>    | The 5'-mismatching nucleotide   |
| <i>sj1</i>    | The 3'-mismatching nucleotide   |
| <i>string</i> | The sequence of the loop (May be NULL, otherwise must be at least <i>size</i> + 2 long) |
| <i>P</i>      | The datastructure containing scaled energy parameters                                   |

**Returns**

The Free energy of the Hairpin-loop in dcal/mol

**16.52.2.5 exp\_E\_Hairpin()**

```
PRIVATE FLT_OR_DBL exp_E_Hairpin (
    int u,
    int type,
    short si1,
    short sj1,
    const char * string,
    vrna_exp_param_t * P )
#include <ViennaRNA/loops/hairpin.h>
Compute Boltzmann weight  $e^{-\Delta G/kT}$  of a hairpin loop.
multiply by scale[u+2]
```

**See also**

[get\\_scaled\\_pf\\_parameters\(\)](#)  
[vrna\\_exp\\_param\\_t](#)  
[E\\_Hairpin\(\)](#)

**Warning**

Not (really) thread safe! A threadsafe implementation will replace this function in a future release!  
 Energy evaluation may change due to updates in global variable "tetra\_loop"

**Parameters**

|               |   |
|---------------|---|
| <i>u</i>      | The size of the loop (number of unpaired nucleotides)                                   |
| <i>type</i>   | The pair type of the base pair closing the hairpin                                      |
| <i>si1</i>    | The 5'-mismatching nucleotide   |
| <i>sj1</i>    | The 3'-mismatching nucleotide   |
| <i>string</i> | The sequence of the loop (May be NULL, otherwise must be at least <i>size</i> + 2 long) |
| <i>P</i>      | The datastructure containing scaled Boltzmann weights of the energy parameters          |

**Returns**

The Boltzmann weight of the Hairpin-loop

**16.52.2.6 vrna\_exp\_E\_hp\_loop()**

```
FLT_OR_DBL vrna_exp_E_hp_loop (
    vrna_fold_compound_t * fc,
    int i,
    int j )
#include <ViennaRNA/loops/hairpin.h>
High-Level function for hairpin loop energy evaluation (partition function variant)
```

**See also**

[vrna\\_E\\_hp\\_loop\(\)](#) for it's free energy counterpart

**Note**

This function is polymorphic! The provided [vrna\\_fold\\_compound\\_t](#) may be of type [VRNA\\_FC\\_TYPE\\_SINGLE](#) or [VRNA\\_FC\\_TYPE\\_COMPARATIVE](#)

## 16.53 Internal Loops

Functions to evaluate the free energy contributions for internal loops.

### 16.53.1 Detailed Description

Functions to evaluate the free energy contributions for internal loops.

Collaboration diagram for Internal Loops:

#### Files

- file [internal.h](#)

*Energy evaluation of interior loops for MFE and partition function calculations.*

#### Basic free energy interface

- int [vrna\\_E\\_int\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j)
  - int [vrna\\_eval\\_int\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j, int k, int l)
- Evaluate the free energy contribution of an interior loop with delimiting base pairs (i, j) and (k, l).*
- int [vrna\\_E\\_ext\\_int\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j, int \*ip, int \*iq)
  - int [vrna\\_E\\_stack](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j)

#### Boltzmann weight (partition function) interface

- [FLT\\_OR\\_DBL](#) [vrna\\_exp\\_E\\_int\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j)
- [FLT\\_OR\\_DBL](#) [vrna\\_exp\\_E\\_interior\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j, int k, int l)

### 16.53.2 Function Documentation

### 16.53.2.1 vrna\_eval\_int\_loop()

```
int vrna_eval_int_loop (
    vrna_fold_compound_t * fc,
    int i,
    int j,
    int k,
    int l )
#include <ViennaRNA/loops/internal.h>
```

Evaluate the free energy contribution of an interior loop with delimiting base pairs  $(i, j)$  and  $(k, l)$ .

#### Note

This function is polymorphic, i.e. it accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE` as well as `VRNA_FC_TYPE_COMPARATIVE`

**SWIG Wrapper Notes** This function is attached as method `eval_int_loop()` to objects of type `fold_compound`

## 16.54 Multibranch Loops

Functions to evaluate the free energy contributions for multibranch loops.

### 16.54.1 Detailed Description

Functions to evaluate the free energy contributions for multibranch loops.

Collaboration diagram for Multibranch Loops:

#### Files

- file `multibranch.h`

*Energy evaluation of multibranch loops for MFE and partition function calculations.*

### Boltzmann weight (partition function) interface

- `typedef struct vrna_mx_pf_aux_ml_s * vrna_mx_pf_aux_ml_t`  
*Auxiliary helper arrays for fast exterior loop computations.*
  - `FLT_OR_DBL vrna_exp_E_mb_loop_fast (vrna_fold_compound_t *fc, int i, int j, vrna_mx_pf_aux_ml_t aux_mx)`
  - `vrna_mx_pf_aux_ml_t vrna_exp_E_ml_fast_init (vrna_fold_compound_t *fc)`
  - `void vrna_exp_E_ml_fast_rotate (vrna_mx_pf_aux_ml_t aux_mx)`
  - `void vrna_exp_E_ml_fast_free (vrna_mx_pf_aux_ml_t aux_mx)`
  - `const FLT_OR_DBL * vrna_exp_E_ml_fast_qqm (vrna_mx_pf_aux_ml_t aux_mx)`
  - `const FLT_OR_DBL * vrna_exp_E_ml_fast_qqm1 (vrna_mx_pf_aux_ml_t aux_mx)`
  - `FLT_OR_DBL vrna_exp_E_ml_fast (vrna_fold_compound_t *fc, int i, int j, vrna_mx_pf_aux_ml_t aux_mx)`

### Basic free energy interface

- `int vrna_E_mb_loop_stack (vrna_fold_compound_t *fc, int i, int j)`  
*Evaluate energy of a multi branch helices stacking onto closing pair  $(i, j)$* 
  - `int vrna_E_mb_loop_fast (vrna_fold_compound_t *fc, int i, int j, int *dml1, int *dml2)`
  - `int E_ml_rightmost_stem (int i, int j, vrna_fold_compound_t *fc)`
  - `int vrna_E_ml_stems_fast (vrna_fold_compound_t *fc, int i, int j, int *fmi, int *dml)`

### 16.54.2 Typedef Documentation

### 16.54.2.1 vrna\_mx\_pf\_aux\_ml\_t

```
typedef struct vrna_mx_pf_aux_ml_s* vrna_mx_pf_aux_ml_t
#include <ViennaRNA/loops/multibranch.h>
Auxiliary helper arrays for fast exterior loop computations.
```

See also

[vrna\\_exp\\_E\\_ml\\_fast\\_init\(\)](#), [vrna\\_exp\\_E\\_ml\\_fast\\_rotate\(\)](#), [vrna\\_exp\\_E\\_ml\\_fast\\_free\(\)](#), [vrna\\_exp\\_E\\_ml\\_fast\(\)](#)

## 16.54.3 Function Documentation

### 16.54.3.1 vrna\_E\_mb\_loop\_stack()

```
int vrna_E_mb_loop_stack (
    vrna_fold_compound_t * fc,
    int i,
    int j )
#include <ViennaRNA/loops/multibranch.h>
Evaluate energy of a multi branch helices stacking onto closing pair (i,j)
Computes total free energy for coaxial stacking of (i,j) with (i+1,k) or (k+1,j-1)
```

## 16.55 Partition Function for Two Hybridized Sequences

Partition Function Cofolding.

### 16.55.1 Detailed Description

Partition Function Cofolding.

To simplify the implementation the partition function computation is done internally in a null model that does not include the duplex initiation energy, i.e. the entropic penalty for producing a dimer from two monomers). The resulting free energies and pair probabilities are initially relative to that null model. In a second step the free energies can be corrected to include the dimerization penalty, and the pair probabilities can be divided into the conditional pair probabilities given that a re dimer is formed or not formed. See [2] for further details.

As for folding one RNA molecule, this computes the partition function of all possible structures and the base pair probabilities. Uses the same global [pf\\_scale](#) variable to avoid overflows.

After computing the partition functions of all possible dimers one can compute the probabilities of base pairs, the concentrations out of start concentrations and sofar and soaway.

Dimer formation is inherently concentration dependent. Given the free energies of the monomers A and B and dimers AB, AA, and BB one can compute the equilibrium concentrations, given input concentrations of A and B, see e.g. Dimitrov & Zuker (2004) Collaboration diagram for Partition Function for Two Hybridized Sequences:

## Files

- file [concentrations.h](#)  
*Concentration computations for RNA-RNA interactions.*
- file [part\\_func\\_up.h](#)  
*Implementations for accessibility and RNA-RNA interaction as a stepwise process.*

## Typedefs

- typedef struct [vrna\\_dimer\\_pf\\_s](#) [vrna\\_dimer\\_pf\\_t](#)  
*Typename for the data structure that stores the dimer partition functions, [vrna\\_dimer\\_pf\\_s](#), as returned by [vrna\\_pf\\_dimer\(\)](#)*
- typedef struct [vrna\\_dimer\\_pf\\_s](#) [cofoldF](#)  
*Backward compatibility typedef for [vrna\\_dimer\\_pf\\_s](#).*

## Variables

- int **mirnatog**  
*Toggles no intrabp in 2nd mol.*
- double **F\_monomer** [2]  
*Free energies of the two monomers.*
- typedef struct [vrna\\_dimer\\_conc\\_s](#) **vrna\_dimer\_conc\_t**  
*Typename for the data structure that stores the dimer concentrations, [vrna\\_dimer\\_conc\\_s](#), as required by [vrna\\_pf\\_dimer\\_concentration\(\)](#)*
- typedef struct [vrna\\_dimer\\_conc\\_s](#) **ConcEnt**  
*Backward compatibility typedef for [vrna\\_dimer\\_conc\\_s](#).*
- **vrna\_dimer\_conc\_t \* vrna\_pf\_dimer\_concentrations** (double FcAB, double FcAA, double FcBB, double FEA, double FEB, const double \*startconc, const [vrna\\_exp\\_param\\_t](#) \*exp\_params)  
*Given two start monomer concentrations a and b, compute the concentrations in thermodynamic equilibrium of all dimers and the monomers.*

## Simplified global partition function computation using sequence(s) or multiple sequence alignment(s)

- **vrna\_dimer\_pf\_t vrna\_pf\_co\_fold** (const char \*seq, char \*structure, [vrna\\_ep\\_t](#) \*\*pl)  
*Calculate partition function and base pair probabilities of nucleic acid/nucleic acid dimers.*

### 16.55.2 Function Documentation

#### 16.55.2.1 [vrna\\_pf\\_co\\_fold\(\)](#)

```
vrna_dimer_pf_t vrna_pf_co_fold (
    const char * seq,
    char * structure,
    vrna_ep_t ** pl )
```

```
#include <ViennaRNA/part_func.h>
```

Calculate partition function and base pair probabilities of nucleic acid/nucleic acid dimers.

This simplified interface to [vrna\\_pf\\_dimer\(\)](#) computes the partition function and, if required, base pair probabilities for an RNA-RNA interaction using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing.

#### Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna\\_pf\\_dimer\(\)](#), and the data structure [vrna\\_fold\\_compound\\_t](#) instead.

#### See also

[vrna\\_pf\\_dimer\(\)](#)

#### Parameters

|                  |  |
|------------------|--|
| <b>seq</b>       | Two concatenated RNA sequences with a delimiting '&' in between                                      |
| <b>structure</b> | A pointer to the character array where position-wise pairing propensity will be stored. (Maybe NULL) |
| <b>pl</b>        | A pointer to a list of <a href="#">vrna_ep_t</a> to store pairing probabilities (Maybe NULL)         |

**Returns**

`vrna_dimer_pf_t` structure containing a set of energies needed for concentration computations.

**16.55.2.2 vrna\_pf\_dimer\_concentrations()**

```
vrna_dimer_conc_t * vrna_pf_dimer_concentrations (
    double FcAB,
    double FcAA,
    double FcBB,
    double FEA,
    double FEB,
    const double * startconc,
    const vrna_exp_param_t * exp_params )
#include <ViennaRNA/concentrations.h>
```

Given two start monomer concentrations  $a$  and  $b$ , compute the concentrations in thermodynamic equilibrium of all dimers and the monomers.

This function takes an array 'startconc' of input concentrations with alternating entries for the initial concentrations of molecules A and B (terminated by two zeroes), then computes the resulting equilibrium concentrations from the free energies for the dimers. Dimer free energies should be the dimer-only free energies, i.e. the FcAB entries from the `vrna_dimer_pf_t` struct.

**Parameters**

|                         |  |
|-------------------------|--|
| <code>FcAB</code>       | Free energy of AB dimer (FcAB entry)                                   |
| <code>FcAA</code>       | Free energy of AA dimer (FcAB entry)                                   |
| <code>FcBB</code>       | Free energy of BB dimer (FcAB entry)                                   |
| <code>FEA</code>        | Free energy of monomer A   |
| <code>FEB</code>        | Free energy of monomer B   |
| <code>startconc</code>  | List of start concentrations [a0],[b0],[a1],[b1],...,[an],[bn],[0],[0] |
| <code>exp_params</code> | The precomputed Boltzmann factors                                      |

**Returns**

`vrna_dimer_conc_t` array containing the equilibrium energies and start concentrations

**16.56 Partition Function for two Hybridized Sequences as a Stepwise Process**

RNA-RNA interaction as a stepwise process.

**16.56.1 Detailed Description**

RNA-RNA interaction as a stepwise process.

In this approach to cofolding the interaction between two RNA molecules is seen as a stepwise process. In a first step, the target molecule has to adopt a structure in which a binding site is accessible. In a second step, the ligand molecule will hybridize with a region accessible to an interaction. Consequently the algorithm is designed as a two step process: The first step is the calculation of the probability that a region within the target is unpaired, or equivalently, the calculation of the free energy needed to expose a region. In the second step we compute the free energy of an interaction for every possible binding site. Collaboration diagram for Partition Function for two Hybridized Sequences as a Stepwise Process:

**Functions**

- `pu_contrib * pf_unstru` (`char *sequence, int max_w`)

*Calculate the partition function over all unpaired regions of a maximal length.*

- `interact * pf_interact (const char *s1, const char *s2, pu_contrib *p_c, pu_contrib *p_c2, int max_w, char *cstruc, int incr3, int incr5)`

*Calculates the probability of a local interaction between two sequences.*

- `void free_interact (interact *pin)`

*Frees the output of function `pf_interact()`.*

- `void free_pu_contrib_struct (pu_contrib *pu)`

*Frees the output of function `pf_unstru()`.*

## 16.56.2 Function Documentation

### 16.56.2.1 pf\_unstru()

```
pu_contrib * pf_unstru (
    char * sequence,
    int max_w )
#include <ViennaRNA/part_func_up.h>
```

Calculate the partition function over all unpaired regions of a maximal length.

You have to call function `pf_fold()` providing the same sequence before calling `pf_unstru()`. If you want to calculate unpaired regions for a constrained structure, set variable 'structure' in function '`pf_fold()`' to the constrain string. It returns a `pu_contrib` struct containing four arrays of dimension  $[i = 1 \text{ to } \text{length}(\text{sequence})][j = 0 \text{ to } u-1]$  containing all possible contributions to the probabilities of unpaired regions of maximum length  $u$ . Each array in `pu_contrib` contains one of the contributions to the total probability of being unpaired: The probability of being unpaired within an exterior loop is in array `pu_contrib->E`, the probability of being unpaired within a hairpin loop is in array `pu_contrib->H`, the probability of being unpaired within an interior loop is in array `pu_contrib->I` and probability of being unpaired within a multi-loop is in array `pu_contrib->M`. The total probability of being unpaired is the sum of the four arrays of `pu_contrib`.

This function frees everything allocated automatically. To free the output structure call `free_pu_contrib()`.

#### Parameters

|                       |  |
|-----------------------|--|
| <code>sequence</code> |  |
| <code>max_w</code>    |  |

#### Returns

### 16.56.2.2 pf\_interact()

```
interact * pf_interact (
    const char * s1,
    const char * s2,
    pu_contrib * p_c,
    pu_contrib * p_c2,
    int max_w,
    char * cstruc,
    int incr3,
    int incr5 )
#include <ViennaRNA/part_func_up.h>
```

Calculates the probability of a local interaction between two sequences.

The function considers the probability that the region of interaction is unpaired within 's1' and 's2'. The longer sequence has to be given as 's1'. The shorter sequence has to be given as 's2'. Function `pf_unstru()` has to be called for 's1' and 's2', where the probabilities of being unpaired have to be given in 'p\_c' and 'p\_c2', respectively.

If you do not want to include the probabilities of being unpaired for 's2' set 'p\_c2' to NULL. If variable 'cstruc' is not NULL, constrained folding is done: The available constrains for intermolecular interaction are: '' (no constrain), 'x' (the base has no intermolecular interaction) and 'l' (the corresponding base has to be paired intermolecularly).

The parameter 'w' determines the maximal length of the interaction. The parameters 'incr5' and 'incr3' allows inclusion of unpaired residues left ('incr5') and right ('incr3') of the region of interaction in 's1'. If the 'incr' options are used, function [pf\\_unstru\(\)](#) has to be called with  $w=w+incr5+incr3$  for the longer sequence 's1'.

It returns a structure of type [interact](#) which contains the probability of the best local interaction including residue i in Pi and the minimum free energy in Gi, where i is the position in sequence 's1'. The member Gikjl of structure [interact](#) is the best interaction between region [k,i]  $k < i$  in longer sequence 's1' and region [j,l]  $j < l$  in 's2'. Gikjl\_wo is Gikjl without the probability of being unpaired.

Use [free\\_interact\(\)](#) to free the returned structure, all other stuff is freed inside [pf\\_interact\(\)](#).

#### Parameters

|        |  |
|--------|--|
| s1     |  |
| s2     |  |
| p_c    |  |
| p_c2   |  |
| max↔_w |  |
| cstruc |  |
| incr3  |  |
| incr5  |  |

#### Returns

## 16.57 Reading/Writing Energy Parameter Sets from/to File

Read and Write energy parameter sets from and to files or strings.

### 16.57.1 Detailed Description

Read and Write energy parameter sets from and to files or strings.

Collaboration diagram for Reading/Writing Energy Parameter Sets from/to File:

## Modules

- [Converting Energy Parameter Files](#)

*Convert energy parameter files into the latest format.*

## Macros

- `#define VRNA_PARAMETER_FORMAT_DEFAULT 0`

*Default Energy Parameter File format.*

## Functions

- `int vrna_params_load (const char fname[], unsigned int options)`  
*Load energy parameters from a file.*
- `int vrna_params_save (const char fname[], unsigned int options)`  
*Save energy parameters to a file.*
- `int vrna_params_load_from_string (const char *string, const char *name, unsigned int options)`  
*Load energy parameters from string.*

- int [vrna\\_params\\_load\\_defaults](#) (void)  
*Load default RNA energy parameter set.*
- int [vrna\\_params\\_load\\_RNA\\_Turner2004](#) (void)  
*Load Turner 2004 RNA energy parameter set.*
- int [vrna\\_params\\_load\\_RNA\\_Turner1999](#) (void)  
*Load Turner 1999 RNA energy parameter set.*
- int [vrna\\_params\\_load\\_RNA\\_Andronescu2007](#) (void)  
*Load Andronescu 2007 RNA energy parameter set.*
- int [vrna\\_params\\_load\\_RNA\\_Langdon2018](#) (void)  
*Load Langdon 2018 RNA energy parameter set.*
- int [vrna\\_params\\_load\\_RNA\\_misc\\_special\\_hairpins](#) (void)  
*Load Misc Special Hairpin RNA energy parameter set.*
- int [vrna\\_params\\_load\\_DNA\\_Mathews2004](#) (void)  
*Load Mathews 2004 DNA energy parameter set.*
- int [vrna\\_params\\_load\\_DNA\\_Mathews1999](#) (void)  
*Load Mathews 1999 DNA energy parameter set.*
- const char \* [last\\_parameter\\_file](#) (void)  
*Get the file name of the parameter file that was most recently loaded.*
- void [read\\_parameter\\_file](#) (const char fname[ ])  
*Read energy parameters from a file.*
- void [write\\_parameter\\_file](#) (const char fname[ ])  
*Write energy parameters to a file.*

## 16.57.2 Macro Definition Documentation

### 16.57.2.1 VRNA\_PARAMETER\_FORMAT\_DEFAULT

```
#define VRNA_PARAMETER_FORMAT_DEFAULT 0
#include <ViennaRNA/params/io.h>
Default Energy Parameter File format.
```

See also

[vrna\\_params\\_load\(\)](#), [vrna\\_params\\_load\\_from\\_string\(\)](#), [vrna\\_params\\_save\(\)](#)

## 16.57.3 Function Documentation

### 16.57.3.1 vrna\_params\_load()

```
int vrna_params_load (
    const char fname[],
    unsigned int options )
#include <ViennaRNA/params/io.h>
Load energy parameters from a file.
```

See also

[vrna\\_params\\_load\\_from\\_string\(\)](#), [vrna\\_params\\_save\(\)](#), [vrna\\_params\\_load\\_defaults\(\)](#), [vrna\\_params\\_load\\_RNA\\_Turner2004\(\)](#), [vrna\\_params\\_load\\_RNA\\_Turner1999\(\)](#), [vrna\\_params\\_load\\_RNA\\_Andronescu2007\(\)](#), [vrna\\_params\\_load\\_RNA\\_Langdon2018\(\)](#), [vrna\\_params\\_load\\_RNA\\_misc\\_special\\_hairpins\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathews2004\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathev](#)

### Parameters

|                |   |
|----------------|---|
| <i>fname</i>   | The path to the file containing the energy parameters                         |
| <i>options</i> | File format bit-mask (usually <a href="#">VRNA_PARAMETER_FORMAT_DEFAULT</a> ) |

### Returns

Non-zero on success, 0 on failure

**SWIG Wrapper Notes** This function is available as overloaded function [params\\_load\(fname="", options=VRNA\\_PARAMETER\\_FORMAT\\_DEFAULT\)](#). Here, the empty filename string indicates to load default RNA parameters, i.e. this is equivalent to calling [vrna\\_params\\_load\\_defaults\(\)](#).

### 16.57.3.2 vrna\_params\_save()

```
int vrna_params_save (
    const char *fname[],
    unsigned int options )
#include <ViennaRNA/params/io.h>
Save energy parameters to a file.
```

#### See also

[vrna\\_params\\_load\(\)](#)

### Parameters

|                |   |
|----------------|---|
| <i>fname</i>   | A filename (path) for the file where the current energy parameters will be written to |
| <i>options</i> | File format bit-mask (usually <a href="#">VRNA_PARAMETER_FORMAT_DEFAULT</a> )         |

### Returns

Non-zero on success, 0 on failure

**SWIG Wrapper Notes** This function is available as overloaded function [params\\_save\(fname, options=VRNA\\_PARAMETER\\_FORMAT\\_DEFAULT\)](#).

### 16.57.3.3 vrna\_params\_load\_from\_string()

```
int vrna_params_load_from_string (
    const char * string,
    const char * name,
    unsigned int options )
#include <ViennaRNA/params/io.h>
```

Load energy paramters from string.

The string must follow the default energy parameter file convention! The optional `name` argument allows one to specify a name for the parameter set which is stored internally.

#### See also

[vrna\\_params\\_load\(\)](#), [vrna\\_params\\_save\(\)](#), [vrna\\_params\\_load\\_defaults\(\)](#), [vrna\\_params\\_load\\_RNA\\_Turner2004\(\)](#), [vrna\\_params\\_load\\_RNA\\_Turner1999\(\)](#), [vrna\\_params\\_load\\_RNA\\_Andronescu2007\(\)](#), [vrna\\_params\\_load\\_RNA\\_Langdon2018\(\)](#), [vrna\\_params\\_load\\_RNA\\_misc\\_special\\_hairpins\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathews2004\(\)](#), [vrna\\_params\\_load\\_DNA\\_Matthews2004\(\)](#)

**Parameters**

|                |   |
|----------------|---|
| <i>string</i>  | A 0-terminated string containing energy parameters                            |
| <i>name</i>    | A name for the parameter set in <i>string</i> (Maybe NULL)                    |
| <i>options</i> | File format bit-mask (usually <a href="#">VRNA_PARAMETER_FORMAT_DEFAULT</a> ) |

**Returns**

Non-zero on success, 0 on failure

**SWIG Wrapper Notes** This function is available as overloaded function [params\\_load\\_from\\_string\(string, name="", options=VRNA\\_PARAMETER\\_FORMAT\\_DEFAULT\)](#).

**16.57.3.4 vrna\_params\_load\_defaults()**

```
int vrna_params_load_defaults (
    void )
#include <ViennaRNA/params/io.h>
```

Load default RNA energy parameter set.

This is a convenience function to load the Turner 2004 RNA free energy parameters. It's the same as calling [vrna\\_params\\_load\\_RNA\\_Turner2004\(\)](#)

**See also**

[vrna\\_params\\_load\(\)](#), [vrna\\_params\\_load\\_from\\_string\(\)](#), [vrna\\_params\\_save\(\)](#), [vrna\\_params\\_load\\_RNA\\_Turner2004\(\)](#),  
[vrna\\_params\\_load\\_RNA\\_Turner1999\(\)](#), [vrna\\_params\\_load\\_RNA\\_Andronescu2007\(\)](#), [vrna\\_params\\_load\\_RNA\\_Langdon2018\(\)](#)  
[vrna\\_params\\_load\\_RNA\\_misc\\_special\\_hairpins\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathews2004\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathe](#)

**Returns**

Non-zero on success, 0 on failure

**SWIG Wrapper Notes** This function is available as overloaded function [params\\_load\(\)](#).

**16.57.3.5 vrna\_params\_load\_RNA\_Turner2004()**

```
int vrna_params_load_RNA_Turner2004 (
    void )
#include <ViennaRNA/params/io.h>
```

Load Turner 2004 RNA energy parameter set.

**See also**

[vrna\\_params\\_load\(\)](#), [vrna\\_params\\_load\\_from\\_string\(\)](#), [vrna\\_params\\_save\(\)](#), [vrna\\_params\\_load\\_defaults\(\)](#),  
[vrna\\_params\\_load\\_RNA\\_Turner1999\(\)](#), [vrna\\_params\\_load\\_RNA\\_Andronescu2007\(\)](#), [vrna\\_params\\_load\\_RNA\\_Langdon2018\(\)](#)  
[vrna\\_params\\_load\\_RNA\\_misc\\_special\\_hairpins\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathews2004\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathe](#)

**Returns**

Non-zero on success, 0 on failure

**SWIG Wrapper Notes** This function is available as function [params\\_load\\_RNA\\_Turner2004\(\)](#).

### 16.57.3.6 vrna\_params\_load\_RNA\_Turner1999()

```
int vrna_params_load_RNA_Turner1999 (
    void )
#include <ViennaRNA/params/io.h>
Load Turner 1999 RNA energy parameter set.
```

#### See also

[vrna\\_params\\_load\(\)](#), [vrna\\_params\\_load\\_from\\_string\(\)](#), [vrna\\_params\\_save\(\)](#), [vrna\\_params\\_load\\_RNA\\_Turner2004\(\)](#),  
[vrna\\_params\\_load\\_defaults\(\)](#), [vrna\\_params\\_load\\_RNA\\_Andronescu2007\(\)](#), [vrna\\_params\\_load\\_RNA\\_Langdon2018\(\)](#),  
[vrna\\_params\\_load\\_RNA\\_misc\\_special\\_hairpins\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathews2004\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathe](#)

#### Returns

Non-zero on success, 0 on failure

**SWIG Wrapper Notes** This function is available as function **params\_load\_RNA\_Turner1999()**.

### 16.57.3.7 vrna\_params\_load\_RNA\_Andronescu2007()

```
int vrna_params_load_RNA_Andronescu2007 (
    void )
#include <ViennaRNA/params/io.h>
Load Andronescu 2007 RNA energy parameter set.
```

#### See also

[vrna\\_params\\_load\(\)](#), [vrna\\_params\\_load\\_from\\_string\(\)](#), [vrna\\_params\\_save\(\)](#), [vrna\\_params\\_load\\_RNA\\_Turner2004\(\)](#),  
[vrna\\_params\\_load\\_RNA\\_Turner1999\(\)](#), [vrna\\_params\\_load\\_defaults\(\)](#), [vrna\\_params\\_load\\_RNA\\_Langdon2018\(\)](#),  
[vrna\\_params\\_load\\_RNA\\_misc\\_special\\_hairpins\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathews2004\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathe](#)

#### Returns

Non-zero on success, 0 on failure

**SWIG Wrapper Notes** This function is available as function **params\_load\_RNA\_Andronescu2007()**.

### 16.57.3.8 vrna\_params\_load\_RNA\_Langdon2018()

```
int vrna_params_load_RNA_Langdon2018 (
    void )
#include <ViennaRNA/params/io.h>
Load Langdon 2018 RNA energy parameter set.
```

#### See also

[vrna\\_params\\_load\(\)](#), [vrna\\_params\\_load\\_from\\_string\(\)](#), [vrna\\_params\\_save\(\)](#), [vrna\\_params\\_load\\_RNA\\_Turner2004\(\)](#),  
[vrna\\_params\\_load\\_RNA\\_Turner1999\(\)](#), [vrna\\_params\\_load\\_RNA\\_Andronescu2007\(\)](#), [vrna\\_params\\_load\\_defaults\(\)](#),  
[vrna\\_params\\_load\\_RNA\\_misc\\_special\\_hairpins\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathews2004\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathe](#)

#### Returns

Non-zero on success, 0 on failure

**SWIG Wrapper Notes** This function is available as function **params\_load\_RNA\_Langdon2018()**.

### 16.57.3.9 vrna\_params\_load\_RNA\_misc\_special\_hairpins()

```
int vrna_params_load_RNA_misc_special_hairpins (
    void )
#include <ViennaRNA/params/io.h>
Load Misc Special Hairpin RNA energy parameter set.
```

#### See also

[vrna\\_params\\_load\(\)](#), [vrna\\_params\\_load\\_from\\_string\(\)](#), [vrna\\_params\\_save\(\)](#), [vrna\\_params\\_load\\_RNA\\_Turner2004\(\)](#),  
[vrna\\_params\\_load\\_RNA\\_Turner1999\(\)](#), [vrna\\_params\\_load\\_RNA\\_Andronescu2007\(\)](#), [vrna\\_params\\_load\\_RNA\\_Langdon2018\(\)](#)  
[vrna\\_params\\_load\\_defaults\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathews2004\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathews1999\(\)](#)

#### Returns

Non-zero on success, 0 on failure

**SWIG Wrapper Notes** This function is available as function **params\_load\_RNA\_misc\_special\_hairpins()**.

### 16.57.3.10 vrna\_params\_load\_DNA\_Mathews2004()

```
int vrna_params_load_DNA_Mathews2004 (
    void )
#include <ViennaRNA/params/io.h>
Load Mathews 2004 DNA energy parameter set.
```

#### See also

[vrna\\_params\\_load\(\)](#), [vrna\\_params\\_load\\_from\\_string\(\)](#), [vrna\\_params\\_save\(\)](#), [vrna\\_params\\_load\\_RNA\\_Turner2004\(\)](#),  
[vrna\\_params\\_load\\_RNA\\_Turner1999\(\)](#), [vrna\\_params\\_load\\_RNA\\_Andronescu2007\(\)](#), [vrna\\_params\\_load\\_RNA\\_Langdon2018\(\)](#)  
[vrna\\_params\\_load\\_RNA\\_misc\\_special\\_hairpins\(\)](#), [vrna\\_params\\_load\\_defaults\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathews1999\(\)](#)

#### Returns

Non-zero on success, 0 on failure

**SWIG Wrapper Notes** This function is available as function **params\_load\_DNA\_Mathews2004()**.

### 16.57.3.11 vrna\_params\_load\_DNA\_Mathews1999()

```
int vrna_params_load_DNA_Mathews1999 (
    void )
#include <ViennaRNA/params/io.h>
Load Mathews 1999 DNA energy parameter set.
```

#### See also

[vrna\\_params\\_load\(\)](#), [vrna\\_params\\_load\\_from\\_string\(\)](#), [vrna\\_params\\_save\(\)](#), [vrna\\_params\\_load\\_RNA\\_Turner2004\(\)](#),  
[vrna\\_params\\_load\\_RNA\\_Turner1999\(\)](#), [vrna\\_params\\_load\\_RNA\\_Andronescu2007\(\)](#), [vrna\\_params\\_load\\_RNA\\_Langdon2018\(\)](#)  
[vrna\\_params\\_load\\_RNA\\_misc\\_special\\_hairpins\(\)](#), [vrna\\_params\\_load\\_DNA\\_Mathews2004\(\)](#), [vrna\\_params\\_load\\_defaults\(\)](#)

#### Returns

Non-zero on success, 0 on failure

**SWIG Wrapper Notes** This function is available as function **params\_load\_DNA\_Mathews1999()**.

### 16.57.3.12 last\_parameter\_file()

```
const char * last_parameter_file (
    void )
#include <ViennaRNA/params/io.h>
Get the file name of the parameter file that was most recently loaded.
```

#### Returns

The file name of the last parameter file, or NULL if parameters are still at defaults

### 16.57.3.13 read\_parameter\_file()

```
void read_parameter_file (
    const char fname[] )
#include <ViennaRNA/params/io.h>
Read energy parameters from a file.
```

**Deprecated** Use [vrna\\_params\\_load\(\)](#) instead!

#### Parameters

|              |   |
|--------------|---|
| <i>fname</i> | The path to the file containing the energy parameters |
|--------------|---|

### 16.57.3.14 write\_parameter\_file()

```
void write_parameter_file (
    const char fname[] )
#include <ViennaRNA/params/io.h>
Write energy parameters to a file.
```

**Deprecated** Use [vrna\\_params\\_save\(\)](#) instead!

#### Parameters

|              |   |
|--------------|---|
| <i>fname</i> | A filename (path) for the file where the current energy parameters will be written to |
|--------------|---|

## 16.58 Converting Energy Parameter Files

Convert energy parameter files into the latest format.

### 16.58.1 Detailed Description

Convert energy parameter files into the latest format.

To preserve some backward compatibility the RNAlib also provides functions to convert energy parameter files from the format used in version 1.4-1.8 into the new format used since version 2.0 Collaboration diagram for Converting Energy Parameter Files:

### Files

- file [1.8.4\\_epars.h](#)

*Free energy parameters for parameter file conversion.*

- file [1.8.4\\_intloops.h](#)

*Free energy parameters for interior loop contributions needed by the parameter file conversion functions.*

## Macros

- `#define VRNA_CONVERT_OUTPUT_ALL 1U`
- `#define VRNA_CONVERT_OUTPUT_HP 2U`
- `#define VRNA_CONVERT_OUTPUT_STACK 4U`
- `#define VRNA_CONVERT_OUTPUT_MM_HP 8U`
- `#define VRNA_CONVERT_OUTPUT_MM_INT 16U`
- `#define VRNA_CONVERT_OUTPUT_MM_INT_1N 32U`
- `#define VRNA_CONVERT_OUTPUT_MM_INT_23 64U`
- `#define VRNA_CONVERT_OUTPUT_MM_MULTI 128U`
- `#define VRNA_CONVERT_OUTPUT_MM_EXT 256U`
- `#define VRNA_CONVERT_OUTPUT_DANGLE5 512U`
- `#define VRNA_CONVERT_OUTPUT_DANGLE3 1024U`
- `#define VRNA_CONVERT_OUTPUT_INT_11 2048U`
- `#define VRNA_CONVERT_OUTPUT_INT_21 4096U`
- `#define VRNA_CONVERT_OUTPUT_INT_22 8192U`
- `#define VRNA_CONVERT_OUTPUT_BULGE 16384U`
- `#define VRNA_CONVERT_OUTPUT_INT 32768U`
- `#define VRNA_CONVERT_OUTPUT_ML 65536U`
- `#define VRNA_CONVERT_OUTPUT_MISC 131072U`
- `#define VRNA_CONVERT_OUTPUT_SPECIAL_HP 262144U`
- `#define VRNA_CONVERT_OUTPUT_VANILLA 524288U`
- `#define VRNA_CONVERT_OUTPUT_NINIO 1048576U`
- `#define VRNA_CONVERT_OUTPUT_DUMP 2097152U`

## Functions

- void `convert_parameter_file (const char *iname, const char *oname, unsigned int options)`

### 16.58.2 Macro Definition Documentation

#### 16.58.2.1 VRNA\_CONVERT\_OUTPUT\_ALL

```
#define VRNA_CONVERT_OUTPUT_ALL 1U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of a complete parameter set
```

#### 16.58.2.2 VRNA\_CONVERT\_OUTPUT\_HP

```
#define VRNA_CONVERT_OUTPUT_HP 2U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of hairpin contributions
```

#### 16.58.2.3 VRNA\_CONVERT\_OUTPUT\_STACK

```
#define VRNA_CONVERT_OUTPUT_STACK 4U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of base pair stack contributions
```

#### 16.58.2.4 VRNA\_CONVERT\_OUTPUT\_MM\_HP

```
#define VRNA_CONVERT_OUTPUT_MM_HP 8U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of hairpin mismatch contribution
```

#### 16.58.2.5 VRNA\_CONVERT\_OUTPUT\_MM\_INT

```
#define VRNA_CONVERT_OUTPUT_MM_INT 16U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of interior loop mismatch contribution
```

#### 16.58.2.6 VRNA\_CONVERT\_OUTPUT\_MM\_INT\_1N

```
#define VRNA_CONVERT_OUTPUT_MM_INT_1N 32U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of 1:n interior loop mismatch contribution
```

#### 16.58.2.7 VRNA\_CONVERT\_OUTPUT\_MM\_INT\_23

```
#define VRNA_CONVERT_OUTPUT_MM_INT_23 64U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of 2:3 interior loop mismatch contribution
```

#### 16.58.2.8 VRNA\_CONVERT\_OUTPUT\_MM\_MULTI

```
#define VRNA_CONVERT_OUTPUT_MM_MULTI 128U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of multi loop mismatch contribution
```

#### 16.58.2.9 VRNA\_CONVERT\_OUTPUT\_MM\_EXT

```
#define VRNA_CONVERT_OUTPUT_MM_EXT 256U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of exterior loop mismatch contribution
```

#### 16.58.2.10 VRNA\_CONVERT\_OUTPUT\_DANGLE5

```
#define VRNA_CONVERT_OUTPUT_DANGLE5 512U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of 5' dangle contribution
```

#### 16.58.2.11 VRNA\_CONVERT\_OUTPUT\_DANGLE3

```
#define VRNA_CONVERT_OUTPUT_DANGLE3 1024U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of 3' dangle contribution
```

#### 16.58.2.12 VRNA\_CONVERT\_OUTPUT\_INT\_11

```
#define VRNA_CONVERT_OUTPUT_INT_11 2048U
#include <ViennaRNA/params/convert.h>
```

Flag to indicate printing of 1:1 interior loop contribution

#### 16.58.2.13 VRNA\_CONVERT\_OUTPUT\_INT\_21

```
#define VRNA_CONVERT_OUTPUT_INT_21 4096U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of 2:1 interior loop contribution
```

#### 16.58.2.14 VRNA\_CONVERT\_OUTPUT\_INT\_22

```
#define VRNA_CONVERT_OUTPUT_INT_22 8192U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of 2:2 interior loop contribution
```

#### 16.58.2.15 VRNA\_CONVERT\_OUTPUT\_BULGE

```
#define VRNA_CONVERT_OUTPUT_BULGE 16384U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of bulge loop contribution
```

#### 16.58.2.16 VRNA\_CONVERT\_OUTPUT\_INT

```
#define VRNA_CONVERT_OUTPUT_INT 32768U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of interior loop contribution
```

#### 16.58.2.17 VRNA\_CONVERT\_OUTPUT\_ML

```
#define VRNA_CONVERT_OUTPUT_ML 65536U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of multi loop contribution
```

#### 16.58.2.18 VRNA\_CONVERT\_OUTPUT\_MISC

```
#define VRNA_CONVERT_OUTPUT_MISC 131072U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of misc contributions (such as terminalAU)
```

#### 16.58.2.19 VRNA\_CONVERT\_OUTPUT\_SPECIAL\_HP

```
#define VRNA_CONVERT_OUTPUT_SPECIAL_HP 262144U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of special hairpin contributions (tri-, tetra-, hexa-loops)
```

#### 16.58.2.20 VRNA\_CONVERT\_OUTPUT\_VANILLA

```
#define VRNA_CONVERT_OUTPUT_VANILLA 524288U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of given parameters only
```

**Note**

This option overrides all other output options, except `VRNA_CONVERT_OUTPUT_DUMP` !

**16.58.2.21 VRNA\_CONVERT\_OUTPUT\_NINIO**

```
#define VRNA_CONVERT_OUTPUT_NINIO 1048576U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of interior loop asymmetry contribution
```

**16.58.2.22 VRNA\_CONVERT\_OUTPUT\_DUMP**

```
#define VRNA_CONVERT_OUTPUT_DUMP 2097152U
#include <ViennaRNA/params/convert.h>
Flag to indicate dumping the energy contributions from the library instead of an input file
```

**16.58.3 Function Documentation****16.58.3.1 convert\_parameter\_file()**

```
void convert_parameter_file (
    const char * iname,
    const char * oname,
    unsigned int options)
#include <ViennaRNA/params/convert.h>
Convert/dump a Vienna 1.8.4 formatted energy parameter file
The options argument allows one to control the different output modes.
Currently available options are:
VRNA_CONVERT_OUTPUT_ALL, VRNA_CONVERT_OUTPUT_HP, VRNA_CONVERT_OUTPUT_STACK
VRNA_CONVERT_OUTPUT_MM_HP, VRNA_CONVERT_OUTPUT_MM_INT, VRNA_CONVERT_OUTPUT_MM_INT_1N
VRNA_CONVERT_OUTPUT_MM_INT_23, VRNA_CONVERT_OUTPUT_MM_MULTI, VRNA_CONVERT_OUTPUT_MM_EXT
VRNA_CONVERT_OUTPUT_DANGLE5, VRNA_CONVERT_OUTPUT_DANGLE3, VRNA_CONVERT_OUTPUT_INT_11
VRNA_CONVERT_OUTPUT_INT_21, VRNA_CONVERT_OUTPUT_INT_22, VRNA_CONVERT_OUTPUT_BULGE
VRNA_CONVERT_OUTPUT_INT, VRNA_CONVERT_OUTPUT_DL, VRNA_CONVERT_OUTPUT_MISC
VRNA_CONVERT_OUTPUT_SPECIAL_HP, VRNA_CONVERT_OUTPUT_VANILLA, VRNA_CONVERT_OUTPUT_NINIO
VRNA_CONVERT_OUTPUT_DUMP
```

The defined options are fine for bitwise compare- and assignment-operations, e. g.: pass a collection of options as a single value like this:

```
convert_parameter_file(ifile, ofile, option_1 | option_2 | option_n)
```

**Parameters**

|                |  |
|----------------|--|
| <i>iname</i>   | The input file name (If NULL input is read from stdin)     |
| <i>oname</i>   | The output file name (If NULL output is written to stdout) |
| <i>options</i> | The options (as described above)                           |

**16.59 Utilities to deal with Nucleotide Alphabets**

Functions to cope with various aspects related to the nucleotide sequence alphabet.

### 16.59.1 Detailed Description

Functions to cope with various aspects related to the nucleotide sequence alphabet.  
 Collaboration diagram for Utilities to deal with Nucleotide Alphabets:

#### Files

- file `alphabet.h`  
*Functions to process, convert, and generally handle different nucleotide and/or base pair alphabets.*
- file `sequence.h`  
*Functions and data structures related to sequence representations ..*

#### Data Structures

- struct `vrna_sequence_s`  
*Data structure representing a nucleotide sequence. More...*
- struct `vrna_alignment_s`

#### Typedefs

- typedef struct `vrna_sequence_s` `vrna_seq_t`  
*Typename for nucleotide sequence representation data structure `vrna_sequence_s`.*

#### Enumerations

- enum `vrna_seq_type_e` { `VRNA_SEQ_UNKNOWN` , `VRNA_SEQ_RNA` , `VRNA_SEQ_DNA` }  
*A enumerator used in `vrna_sequence_s` to distinguish different nucleotide sequences.*

#### Functions

- `char * vrna_ptypes (const short *S, vrna_md_t *md)`  
*Get an array of the numerical encoding for each possible base pair (i,j)*
- `short * vrna_seq_encode (const char *sequence, vrna_md_t *md)`  
*Get a numerical representation of the nucleotide sequence.*
- `short * vrna_seq_encode_simple (const char *sequence, vrna_md_t *md)`  
*Get a numerical representation of the nucleotide sequence (simple version)*
- `int vrna_nucleotide_encode (char c, vrna_md_t *md)`  
*Encode a nucleotide character to numerical value.*
- `char vrna_nucleotide_decode (int enc, vrna_md_t *md)`  
*Decode a numerical representation of a nucleotide back into nucleotide alphabet.*

### 16.59.2 Data Structure Documentation

#### 16.59.2.1 struct `vrna_sequence_s`

Data structure representing a nucleotide sequence.

##### Data Fields

- `vrna_seq_type_e type`  
*The type of sequence.*
- `char * string`  
*The string representation of the sequence.*
- `short * encoding`  
*The integer representation of the sequence.*
- `unsigned int length`  
*The length of the sequence.*

### 16.59.2.2 struct vrna\_alignment\_s

Collaboration diagram for vrna\_alignment\_s:

## 16.59.3 Enumeration Type Documentation

### 16.59.3.1 vrna\_seq\_type\_e

```
enum vrna_seq_type_e
```

```
#include <ViennaRNA/sequence.h>
```

A enumerator used in [vrna\\_sequence\\_s](#) to distinguish different nucleotide sequences.

Enumerator

|                  |  |
|------------------|--|
| VRNA_SEQ_UNKNOWN | Nucleotide sequence represents an Unkown type. |
| VRNA_SEQ_RNA     | Nucleotide sequence represents an RNA type.    |
| VRNA_SEQ_DNA     | Nucleotide sequence represents a DNA type.     |

## 16.59.4 Function Documentation

### 16.59.4.1 vrna\_ptypes()

```
char * vrna_ptypes (
    const short * s,
    vrna_md_t * md )
```

```
#include <ViennaRNA/alphabet.h>
```

Get an array of the numerical encoding for each possible base pair (i,j)

Note

This array is always indexed in column-wise order, in contrast to previously different indexing between mfe and pf variants!

See also

[vrna\\_idx\\_col\\_wise\(\)](#), [vrna\\_fold\\_compound\\_t](#)

### 16.59.4.2 vrna\_seq\_encode()

```
short * vrna_seq_encode (
    const char * sequence,
    vrna_md_t * md )
```

```
#include <ViennaRNA/alphabet.h>
```

Get a numerical representation of the nucleotide sequence.

Parameters

|                 |  |
|-----------------|--|
| <i>sequence</i> | The input sequence in upper-case letters   |
| <i>md</i>       | A pointer to a <a href="#">vrna_md_t</a> data structure that specifies the conversion type |

**Returns**

A list of integer encodings for each sequence letter (1-based). Position 0 denotes the length of the list

**SWIG Wrapper Notes** In the target scripting language, this function is wrapped as overloaded function `seq_encode()` where the last parameter, the `model_details` data structure, is optional. If it is omitted, default model settings are applied, i.e. default nucleotide letter conversion. The wrapped function returns a list/tuple of integer representations of the input sequence.

**16.59.4.3 vrna\_nucleotide\_encode()**

```
int vrna_nucleotide_encode (
    char c,
    vrna_md_t * md )
#include <ViennaRNA/alphabet.h>
```

Encode a nucleotide character to numerical value.

This function encodes a nucleotide character to its numerical representation as required by many functions in RNAlib.

**See also**

[vrna\\_nucleotide\\_decode\(\)](#), [vrna\\_seq\\_encode\(\)](#)

**Parameters**

|                 |   |
|-----------------|---|
| <code>c</code>  | The nucleotide character to encode                    |
| <code>md</code> | The model details that determine the kind of encoding |

**Returns**

The encoded nucleotide

**16.59.4.4 vrna\_nucleotide\_decode()**

```
char vrna_nucleotide_decode (
    int enc,
    vrna_md_t * md )
#include <ViennaRNA/alphabet.h>
```

Decode a numerical representation of a nucleotide back into nucleotide alphabet.

This function decodes a numerical representation of a nucleotide character back into nucleotide alphabet

**See also**

[vrna\\_nucleotide\\_encode\(\)](#), [vrna\\_seq\\_encode\(\)](#)

**Parameters**

|                  |   |
|------------------|---|
| <code>enc</code> | The encoded nucleotide                                |
| <code>md</code>  | The model details that determine the kind of decoding |

**Returns**

The decoded nucleotide character

## 16.60 (Nucleic Acid Sequence) String Utilitites

Functions to parse, convert, manipulate, create, and compare (nucleic acid sequence) strings.

### 16.60.1 Detailed Description

Functions to parse, convert, manipulate, create, and compare (nucleic acid sequence) strings.  
 Collaboration diagram for (Nucleic Acid Sequence) String Utilitites:

#### Files

- file `strings.h`

*General utility- and helper-functions for RNA sequence and structure strings used throughout the ViennaRNA Package.*

#### Macros

- #define `XSTR(s)` `STR(s)`  
*Stringify a macro after expansion.*
- #define `STR(s)` `#s`  
*Stringify a macro argument.*
- #define `FILENAME_MAX_LENGTH` 80  
*Maximum length of filenames that are generated by our programs.*
- #define `FILENAME_ID_LENGTH` 42  
*Maximum length of id taken from fasta header for filename generation.*
- #define `VRNA_TRIM.LEADING` 1U  
*Trim only characters leading the string.*
- #define `VRNA_TRIM.TRAILING` 2U  
*Trim only characters trailing the string.*
- #define `VRNA_TRIM.IN_BETWEEN` 4U  
*Trim only characters within the string.*
- #define `VRNA_TRIM.SUBST_BY_FIRST` 8U  
*Replace remaining characters after trimming with the first delimiter in list.*
- #define `VRNA_TRIM.DEFAULT` ( `VRNA_TRIM.LEADING` | `VRNA_TRIM.TRAILING` )  
*Default settings for trimming, i.e. trim leading and trailing.*
- #define `VRNA_TRIM.ALL` ( `VRNA_TRIM.DEFAULT` | `VRNA_TRIM.IN_BETWEEN` )  
*Trim characters anywhere in the string.*

#### Functions

- char \* `vrna_strdup_printf` (const char \*format,...)  
*Safely create a formatted string.*
- char \* `vrna_strdup_vprintf` (const char \*format, va\_list argp)  
*Safely create a formatted string.*
- int `vrna_strcat_printf` (char \*\*dest, const char \*format,...)  
*Safely append a formatted string to another string.*
- int `vrna_strcat_vprintf` (char \*\*dest, const char \*format, va\_list args)  
*Safely append a formatted string to another string.*
- unsigned int `vrna_strtrim` (char \*string, const char \*delimiters, unsigned int keep, unsigned int options)  
*Trim a string by removing (multiple) occurrences of a particular character.*

- `char ** vrna_strsplit (const char *string, const char *delimiter)`  
*Split a string into tokens using a delimiting character.*
- `char * vrna_random_string (int l, const char symbols[])`  
*Create a random string using characters from a specified symbol set.*
- `int vrna_hamming_distance (const char *s1, const char *s2)`  
*Calculate hamming distance between two sequences.*
- `int vrna_hamming_distance_bound (const char *s1, const char *s2, int n)`  
*Calculate hamming distance between two sequences up to a specified length.*
- `void vrna_seq_toRNA (char *sequence)`  
*Convert an input sequence (possibly containing DNA alphabet characters) to RNA alphabet.*
- `void vrna_seq_toupper (char *sequence)`  
*Convert an input sequence to uppercase.*
- `void vrna_seq_reverse (char *sequence)`  
*Reverse a string in-place.*
- `char * vrna_DNA_complement (const char *sequence)`  
*Retrieve a DNA sequence which resembles the complement of the input sequence.*
- `char * vrna_seq_ungapped (const char *sequence)`  
*Remove gap characters from a nucleotide sequence.*
- `char * vrna_cut_point_insert (const char *string, int cp)`  
*Add a separating '&' character into a string according to cut-point position.*
- `char * vrna_cut_point_remove (const char *string, int *cp)`  
*Remove a separating '&' character from a string.*

## 16.60.2 Macro Definition Documentation

### 16.60.2.1 FILENAME\_MAX\_LENGTH

```
#define FILENAME_MAX_LENGTH 80
#include <ViennaRNA/utils/strings.h>
```

Maximum length of filenames that are generated by our programs.

This definition should be used throughout the complete ViennaRNA package wherever a static array holding filenames of output files is declared.

### 16.60.2.2 FILENAME\_ID\_LENGTH

```
#define FILENAME_ID_LENGTH 42
#include <ViennaRNA/utils/strings.h>
```

Maximum length of id taken from fasta header for filename generation.

this has to be smaller than FILENAME\_MAX\_LENGTH since in most cases, some suffix will be appended to the ID

### 16.60.2.3 VRNA\_TRIM.LEADING

```
#define VRNA_TRIM_LEADING 1U
#include <ViennaRNA/utils/strings.h>
```

Trim only characters leading the string.

See also

[vrna\\_strtrim\(\)](#)

#### 16.60.2.4 VRNA\_TRIM\_TRAILING

```
#define VRNA_TRIM_TRAILING 2U
#include <ViennaRNA/utils/strings.h>
Trim only characters trailing the string.
```

See also

[vrna\\_strtrim\(\)](#)

#### 16.60.2.5 VRNA\_TRIM\_IN\_BETWEEN

```
#define VRNA_TRIM_IN_BETWEEN 4U
#include <ViennaRNA/utils/strings.h>
Trim only characters within the string.
```

See also

[vrna\\_strtrim\(\)](#)

#### 16.60.2.6 VRNA\_TRIM\_SUBST\_BY\_FIRST

```
#define VRNA_TRIM_SUBST_BY_FIRST 8U
#include <ViennaRNA/utils/strings.h>
Replace remaining characters after trimming with the first delimiter in list.
```

See also

[vrna\\_strtrim\(\)](#)

#### 16.60.2.7 VRNA\_TRIM\_DEFAULT

```
#define VRNA_TRIM_DEFAULT ( VRNA_TRIM.LEADING | VRNA_TRIM.TRAILING )
#include <ViennaRNA/utils/strings.h>
Default settings for trimming, i.e. trim leading and trailing.
```

See also

[vrna\\_strtrim\(\)](#)

#### 16.60.2.8 VRNA\_TRIM\_ALL

```
#define VRNA_TRIM_ALL ( VRNA_TRIM_DEFAULT | VRNA_TRIM_IN_BETWEEN )
#include <ViennaRNA/utils/strings.h>
Trim characters anywhere in the string.
```

See also

[vrna\\_strtrim\(\)](#)

### 16.60.3 Function Documentation

### 16.60.3.1 vrna\_strdup\_printf()

```
char * vrna_strdup_printf (
    const char * format,
    ...
)
#include <ViennaRNA/utils/strings.h>
```

Safely create a formatted string.

This function is a safe implementation for creating a formatted character array, similar to *sprintf*. Internally, it uses the *asprintf* function if available to dynamically allocate a large enough character array to store the supplied content. If *asprintf* is not available, mimic its behavior using *vsnprintf*.

#### Note

The returned pointer of this function should always be passed to *free()* to release the allocated memory

#### See also

[vrna\\_strdup\\_vprintf\(\)](#), [vrna\\_strdup\\_printf\(\)](#)

#### Parameters

|               |  |
|---------------|--|
| <i>format</i> | The format string (See also <i>asprintf</i> )        |
| ...           | The list of variables used to fill the format string |

#### Returns

The formatted, null-terminated string, or NULL if something has gone wrong

### 16.60.3.2 vrna\_strdup\_vprintf()

```
char * vrna_strdup_vprintf (
    const char * format,
    va_list argp )
#include <ViennaRNA/utils/strings.h>
```

Safely create a formatted string.

This function is the *va\_list* version of [vrna\\_strdup\\_printf\(\)](#)

#### Note

The returned pointer of this function should always be passed to *free()* to release the allocated memory

#### See also

[vrna\\_strdup\\_printf\(\)](#), [vrna\\_strdup\\_vprintf\(\)](#), [vrna\\_strdup\\_vprintf\(\)](#)

#### Parameters

|               |   |
|---------------|---|
| <i>format</i> | The format string (See also <i>asprintf</i> )   |
| <i>argp</i>   | The list of arguments to fill the format string |

**Returns**

The formatted, null-terminated string, or NULL if something has gone wrong

**16.60.3.3 vrna\_strcat\_printf()**

```
int vrna_strcat_printf (
    char ** dest,
    const char * format,
    ...
)
```

```
#include <ViennaRNA/utils/strings.h>
```

Safely append a formatted string to another string.

This function is a safe implementation for appending a formatted character array, similar to a combination of *strcat* and *sprintf*. The function automatically allocates enough memory to store both, the previous content stored at *dest* and the appended format string. If the *dest* pointer is NULL, the function allocate memory only for the format string. The function returns the number of characters in the resulting string or -1 in case of an error.

**See also**

[vrna\\_strcat\\_vprintf\(\)](#), [vrna\\_strdup\\_printf\(\)](#), [vrna\\_strdup\\_vprintf\(\)](#)

**Parameters**

|               |   |
|---------------|---|
| <i>dest</i>   | The address of a char *pointer where the formatted string is to be appended |
| <i>format</i> | The format string (See also <i>sprintf</i> )                                |
| ...           | The list of variables used to fill the format string                        |

**Returns**

The number of characters in the final string, or -1 on error

**16.60.3.4 vrna\_strcat\_vprintf()**

```
int vrna_strcat_vprintf (
    char ** dest,
    const char * format,
    va_list args
)
```

```
#include <ViennaRNA/utils/strings.h>
```

Safely append a formatted string to another string.

This function is the *va\_list* version of [vrna\\_strcat\\_printf\(\)](#)

**See also**

[vrna\\_strcat\\_printf\(\)](#), [vrna\\_strdup\\_printf\(\)](#), [vrna\\_strdup\\_vprintf\(\)](#)

**Parameters**

|               |   |
|---------------|---|
| <i>dest</i>   | The address of a char *pointer where the formatted string is to be appended |
| <i>format</i> | The format string (See also <i>sprintf</i> )                                |
| <i>args</i>   | The list of argument to fill the format string                              |

**Returns**

The number of characters in the final string, or -1 on error

**16.60.3.5 vrna\_strtrim()**

```
unsigned int vrna_strtrim (
    char * string,
    const char * delimiters,
    unsigned int keep,
    unsigned int options )
#include <ViennaRNA/utils/strings.h>
```

Trim a string by removing (multiple) occurrences of a particular character.

This function removes (multiple) consecutive occurrences of a set of characters (`delimiters`) within an input string. It may be used to remove leading and/or trailing whitespaces or to restrict the maximum number of consecutive occurrences of the delimiting characters `delimiters`. Setting `keep=0` removes all occurrences, while other values reduce multiple consecutive occurrences to at most `keep` delimiters. This might be useful if one would like to reduce multiple whitespaces to a single one, or to remove empty fields within a comma-separated value string. The parameter `delimiters` may be a pointer to a 0-terminated char string containing a set of any ASCII character. If `NULL` is passed as delimiter set or an empty char string, all whitespace characters are trimmed. The `options` parameter is a bit vector that specifies which part of the string should undergo trimming. The implementation distinguishes the leading ([VRNA\\_TRIM.LEADING](#)), trailing ([VRNA\\_TRIM.TRAILING](#)), and in-between ([VRNA\\_TRIM.IN\\_BETWEEN](#)) part with respect to the delimiter set. Combinations of these parts can be specified by using logical-or operator.

The following example code removes all leading and trailing whitespace characters from the input string:

```
char      string[20] = " \t blablabla ";
unsigned int r        = vrna_strtrim(&(string[0]),
                                     NULL,
                                     0,
                                     VRNA_TRIM_DEFAULT);
```

**Note**

The delimiter always consists of a single character from the set of characters provided. In case of alternative delimiters and non-null `keep` parameter, the first `keep` delimiters are preserved within the string. Use [VRNA\\_TRIM.SUBST\\_BY\\_FIRST](#) to substitute all remaining delimiting characters with the first from the `delimiters` list.

**See also**

[VRNA\\_TRIM.LEADING](#), [VRNA\\_TRIM.TRAILING](#), [VRNA\\_TRIM.IN\\_BETWEEN](#), [VRNA\\_TRIM.SUBST\\_BY\\_FIRST](#),  
[VRNA\\_TRIM.DEFAULT](#), [VRNA\\_TRIM.ALL](#)

**Parameters**

|                         |   |
|-------------------------|---|
| <code>string</code>     | The '\0'-terminated input string to trim  |
| <code>delimiters</code> | The delimiter characters as 0-terminated char array (or <code>NULL</code> )         |
| <code>keep</code>       | The maximum number of consecutive occurrences of the delimiter in the output string |
| <code>options</code>    | The option bit vector specifying the mode of operation                              |

**Returns**

The number of delimiters removed from the string

**SWIG Wrapper Notes** Since many scripting languages treat strings as immutable objects, this function does not modify the input string directly. Instead, it returns the modified string as second return value, together with the number of removed delimiters.

The scripting language interface provides an overloaded version of this function, with default parameters `delimiters=NULL`, `keep=0`, and `options=VRNA_TRIM.DEFAULT`.

### 16.60.3.6 vrna\_strsplit()

```
char ** vrna_strsplit (
    const char * string,
    const char * delimiter )
#include <ViennaRNA/utils/strings.h>
```

Split a string into tokens using a delimiting character.

This function splits a string into an array of strings using a single character that delimits the elements within the string. The default delimiter is the ampersand '&' and will be used when `NULL` is passed as a second argument. The returned list is `NULL` terminated, i.e. the last element is `NULL`. If the delimiter is not found, the returned list contains exactly one element: the input string.

For instance, the following code:

```
char **tok = vrna_strsplit ("GGGG&CCCC&AAAAAA", NULL);
for (char **ptr = tok; *ptr; ptr++) {
    printf ("%s\n", *ptr);
    free (*ptr);
}
free (tok);
```

produces this output:

```
* GGGG
* CCCC
* AAAAAA
*
```

and properly free's the memory occupied by the returned element array.

#### Note

This function internally uses `strtok_r()` and is therefore considered to be thread-safe. Also note, that it is the users responsibility to free the memory of the array and that of the individual element strings!

In case the input string consists of consecutive delimiters, starts or ends with one or multiple delimiters, empty strings are produced in the output list, indicating the empty fields of data resulting from the split. Use [vrna\\_trim\(\)](#) prior to a call to this function to remove any leading, trailing, or in-between empty fields.

#### See also

[vrna\\_trim\(\)](#)

#### Parameters

|                        |   |
|------------------------|---|
| <code>string</code>    | The input string that should be split into elements                   |
| <code>delimiter</code> | The delimiting character. If <code>NULL</code> , the delimiter is "&" |

#### Returns

A `NULL` terminated list of the elements in the string

### 16.60.3.7 vrna\_random\_string()

```
char * vrna_random_string (
    int l,
    const char symbols[] )
#include <ViennaRNA/utils/strings.h>
```

Create a random string using characters from a specified symbol set.

**Parameters**

|                |                            |
|----------------|----------------------------|
| <i>l</i>       | The length of the sequence |
| <i>symbols</i> | The symbol set             |

**Returns**

A random string of length '*l*' containing characters from the symbolset

**16.60.3.8 vrna\_hamming\_distance()**

```
int vrna_hamming_distance (
    const char * s1,
    const char * s2 )
#include <ViennaRNA/utils/strings.h>
```

Calculate hamming distance between two sequences.

**Parameters**

|           |                     |
|-----------|---------------------|
| <i>s1</i> | The first sequence  |
| <i>s2</i> | The second sequence |

**Returns**

The hamming distance between *s1* and *s2*

**16.60.3.9 vrna\_hamming\_distance\_bound()**

```
int vrna_hamming_distance_bound (
    const char * s1,
    const char * s2,
    int n )
#include <ViennaRNA/utils/strings.h>
```

Calculate hamming distance between two sequences up to a specified length.

This function is similar to [vrna\\_hamming\\_distance\(\)](#) but instead of comparing both sequences up to their actual length only the first '*n*' characters are taken into account

**Parameters**

|           |   |
|-----------|---|
| <i>s1</i> | The first sequence  |
| <i>s2</i> | The second sequence   |
| <i>n</i>  | The length of the subsequences to consider (starting from the 5' end) |

**Returns**

The hamming distance between *s1* and *s2*

**16.60.3.10 vrna\_seq\_toRNA()**

```
void vrna_seq_toRNA (
    char * sequence )
#include <ViennaRNA/utils/strings.h>
```

Convert an input sequence (possibly containing DNA alphabet characters) to RNA alphabet.  
This function substitutes *T* and *t* with *U* and *u*, respectively

#### Parameters

|                 |                              |
|-----------------|------------------------------|
| <i>sequence</i> | The sequence to be converted |
|-----------------|------------------------------|

### 16.60.3.11 vrna\_seq\_toupper()

```
void vrna_seq_toupper (
    char * sequence )
#include <ViennaRNA/utils/strings.h>
Convert an input sequence to uppercase.
```

#### Parameters

|                 |                              |
|-----------------|------------------------------|
| <i>sequence</i> | The sequence to be converted |
|-----------------|------------------------------|

### 16.60.3.12 vrna\_seq\_reverse()

```
void vrna_seq_reverse (
    char * sequence )
#include <ViennaRNA/utils/strings.h>
Reverse a string in-place.  
This function reverses a character string in the form of an array of characters in-place, i.e. it changes the input parameter.
```

#### Postcondition

After execution, the input *sequence* consists of the reverse string prior to the execution.

#### See also

[vrna\\_DNA\\_complement\(\)](#)

#### Parameters

|                 |                       |
|-----------------|-----------------------|
| <i>sequence</i> | The string to reverse |
|-----------------|-----------------------|

### 16.60.3.13 vrna\_DNA\_complement()

```
char * vrna_DNA_complement (
    const char * sequence )
#include <ViennaRNA/utils/strings.h>
Retrieve a DNA sequence which resembles the complement of the input sequence.  
This function returns a new DNA string which is the complement of the input, i.e. the nucleotide letters A,C,G, and T are substituted by their complements T,G,C, and A, respectively.  
Any characters not belonging to the alphabet of the 4 canonical bases of DNA are not altered.
```

#### Note

This function also handles lower-case input sequences and treats U of the RNA alphabet equally to T

## See also

[vrna\\_seq\\_reverse\(\)](#)

## Parameters

|                 |                        |
|-----------------|------------------------|
| <i>sequence</i> | the input DNA sequence |
|-----------------|------------------------|

## Returns

The complement of the input DNA sequence

**16.60.3.14 vrna\_seq\_ungapped()**

```
char * vrna_seq_ungapped (
    const char * sequence )
#include <ViennaRNA/utils/strings.h>
Remove gap characters from a nucleotide sequence.
```

## Parameters

|                 |   |
|-----------------|---|
| <i>sequence</i> | The original, null-terminated nucleotide sequence |
|-----------------|---|

## Returns

A copy of the input sequence with all gap characters removed

**16.60.3.15 vrna\_cut\_point\_insert()**

```
char * vrna_cut_point_insert (
    const char * string,
    int cp )
#include <ViennaRNA/utils/strings.h>
Add a separating '&' character into a string according to cut-point position.
If the cut-point position is less or equal to zero, this function just returns a copy of the provided string. Otherwise,
the cut-point character is set at the corresponding position
```

## Parameters

|               |                        |
|---------------|------------------------|
| <i>string</i> | The original string    |
| <i>cp</i>     | The cut-point position |

## Returns

A copy of the provided string including the cut-point character

**16.60.3.16 vrna\_cut\_point\_remove()**

```
char * vrna_cut_point_remove (
    const char * string,
    int * cp )
#include <ViennaRNA/utils/strings.h>
```

Remove a separating '&' character from a string.

This function removes the cut-point indicating '&' character from a string and memorizes its position in a provided integer variable. If not '&' is found in the input, the integer variable is set to -1. The function returns a copy of the input string with the '&' being sliced out.

#### Parameters

|               |                        |
|---------------|------------------------|
| <i>string</i> | The original string    |
| <i>cp</i>     | The cut-point position |

#### Returns

A copy of the input string with the '&' being sliced out

## 16.61 Secondary Structure Utilities

Functions to create, parse, convert, manipulate, and compare secondary structure representations.

### 16.61.1 Detailed Description

Functions to create, parse, convert, manipulate, and compare secondary structure representations.  
Collaboration diagram for Secondary Structure Utilities:

#### Modules

- [Dot-Bracket Notation of Secondary Structures](#)

*The Dot-Bracket notation as introduced already in the early times of the ViennaRNA Package denotes base pairs by matching pairs of parenthesis () and unpaired nucleotides by dots ..*

- [Washington University Secondary Structure \(WUSS\) notation](#)

*The WUSS notation, as frequently used for consensus secondary structures in Stockholm 1.0 format.*

- [Pair Table Representation of Secondary Structures](#)
- [Pair List Representation of Secondary Structures](#)
- [Abstract Shapes Representation of Secondary Structures](#)

*Abstract Shapes, introduced by Giegerich et al. in (2004) [10], collapse the secondary structure while retaining the nestedness of helices and hairpin loops.*

- [Helix List Representation of Secondary Structures](#)
- [Tree Representation of Secondary Structures](#)

*Secondary structures can be readily represented as trees, where internal nodes represent base pairs, and leaves represent unpaired nucleotides. The dot-bracket structure string already is a tree represented by a string of parenthesis (base pairs) and dots for the leaf nodes (unpaired nucleotides).*

- [Distance measures between Secondary Structures](#)
- [Deprecated Interface for Secondary Structure Utilities](#)

#### Files

- file [structures.h](#)

*Various utility- and helper-functions for secondary structure parsing, converting, etc.*

#### Functions

- `int * vrna_loopidx_from_ptable (const short *pt)`  
*Get a loop index representation of a structure.*
- `unsigned int * vrna_refBPCnt_matrix (const short *reference_pt, unsigned int turn)`  
*Make a reference base pair count matrix.*
- `unsigned int * vrna_refBPdist_matrix (const short *pt1, const short *pt2, unsigned int turn)`

- `char * vrna_db_from_probs (const FLT_OR_DBL *pr, unsigned int length)`  
*Create a dot-bracket like structure string from base pair probability matrix.*
- `char vrna_bpp_symbol (const float *x)`  
*Get a pseudo dot bracket notation for a given probability information.*
- `char * vrna_db_from_bp_stack (vrna_bp_stack_t *bp, unsigned int length)`  
*Create a dot-bracket/parenthesis structure from backtracking stack.*

## 16.61.2 Function Documentation

### 16.61.2.1 vrna\_refBPcnt\_matrix()

```
unsigned int * vrna_refBPcnt_matrix (
    const short * reference_pt,
    unsigned int turn )
#include <ViennaRNA/utils/structures.h>
Make a reference base pair count matrix.
Get an upper triangular matrix containing the number of basepairs of a reference structure for each interval [i,j] with i<j. Access it via iindx!!!
```

### 16.61.2.2 vrna\_refBPDist\_matrix()

```
unsigned int * vrna_refBPDist_matrix (
    const short * pt1,
    const short * pt2,
    unsigned int turn )
#include <ViennaRNA/utils/structures.h>
Make a reference base pair distance matrix.
Get an upper triangular matrix containing the base pair distance of two reference structures for each interval [i,j] with i<j. Access it via iindx!!!
```

### 16.61.2.3 vrna\_db\_from\_probs()

```
char * vrna_db_from_probs (
    const FLT_OR_DBL * pr,
    unsigned int length )
#include <ViennaRNA/utils/structures.h>
Create a dot-bracket like structure string from base pair probability matrix.
```

**SWIG Wrapper Notes** This function is available as parameter-less method `db_from_probs()` bound to objects of type `fold_compound`. Parameters `pr` and `length` are implicitly taken from the `fold_compound` object the method is bound to. Upon missing base pair probabilities, this method returns an empty string.

### 16.61.2.4 vrna\_db\_from\_bp\_stack()

```
char * vrna_db_from_bp_stack (
    vrna_bp_stack_t * bp,
    unsigned int length )
#include <ViennaRNA/utils/structures.h>
Create a dot-bracket/parenthesis structure from backtracking stack.
This function is capable to create dot-bracket structures from suboptimal structure prediction sensu M. Zuker
```

## Parameters

|               |  |
|---------------|--|
| <i>bp</i>     | Base pair stack containing the traced base pairs |
| <i>length</i> | The length of the structure                      |

## Returns

The secondary structure in dot-bracket notation as provided in the input

## 16.62 Dot-Bracket Notation of Secondary Structures

The Dot-Bracket notation as introduced already in the early times of the ViennaRNA Package denotes base pairs by matching pairs of parenthesis () and unpaired nucleotides by dots ..

### 16.62.1 Detailed Description

The Dot-Bracket notation as introduced already in the early times of the ViennaRNA Package denotes base pairs by matching pairs of parenthesis () and unpaired nucleotides by dots ..

As a simple example, consider a helix of size 4 enclosing a hairpin of size 4. In dot-bracket notation, this is annotated as

```
(( (( . . . ) ) ))
```

#### Extended Dot-Bracket Notation

A more generalized version of the original Dot-Bracket notation may use additional pairs of brackets, such as <>, {}, and [], and matching pairs of uppercase/lowercase letters. This allows for annotating pseudo-knots, since different pairs of brackets are not required to be nested.

The following annotations of a simple structure with two crossing helices of size 4 are equivalent:

```
<<<<[ [ [ . . . >>> ] ] ]]
```

```
(( (( AAAA . . . ) ) )) aaaa
```

AAAA{ {{ { . . . aaaa } } } } Collaboration diagram for Dot-Bracket Notation of Secondary Structures:

## Macros

- #define VRNA\_BRACKETS\_ALPHA 4U  
*Bitflag to indicate secondary structure notations using uppercase/lowercase letters from the latin alphabet.*
- #define VRNA\_BRACKETS\_RND 8U  
*Bitflag to indicate secondary structure notations using round brackets (parenthesis), ()*
- #define VRNA\_BRACKETS\_CLY 16U  
*Bitflag to indicate secondary structure notations using curly brackets, {}*
- #define VRNA\_BRACKETS\_ANG 32U  
*Bitflag to indicate secondary structure notations using angular brackets, <>*
- #define VRNA\_BRACKETS\_SQR 64U  
*Bitflag to indicate secondary structure notations using square brackets, []*
- #define VRNA\_BRACKETS\_DEFAULT  
*Default bitmask to indicate secondary structure notation using any pair of brackets.*
- #define VRNA\_BRACKETS\_ANY  
*Bitmask to indicate secondary structure notation using any pair of brackets or uppercase/lowercase alphabet letters.*

## Functions

- char \* vrna\_db\_pack (const char \*struc)  
*Pack secondary secondary structure, 5:1 compression using base 3 encoding.*
- char \* vrna\_db\_unpack (const char \*packed)  
*Unpack secondary structure previously packed with vrna\_db\_pack()*
- void vrna\_db\_flatten (char \*structure, unsigned int options)

- Substitute pairs of brackets in a string with parenthesis.*
- void [vrna\\_db\\_flatten\\_to](#) (char \*string, const char target[3], unsigned int options)  
*Substitute pairs of brackets in a string with another type of pair characters.*
  - char \* [vrna\\_db\\_from\\_ptable](#) (const short \*pt)  
*Convert a pair table into dot-parenthesis notation.*
  - char \* [vrna\\_db\\_from plist](#) ([vrna\\_ep\\_t](#) \*pairs, unsigned int n)  
*Convert a list of base pairs into dot-bracket notation.*
  - char \* [vrna\\_db\\_to\\_element\\_string](#) (const char \*structure)  
*Convert a secondary structure in dot-bracket notation to a nucleotide annotation of loop contexts.*
  - char \* [vrna\\_db\\_pk\\_remove](#) (const char \*structure, unsigned int options)  
*Remove pseudo-knots from an input structure.*

## 16.62.2 Macro Definition Documentation

### 16.62.2.1 VRNA\_BRACKETS\_ALPHA

```
#define VRNA_BRACKETS_ALPHA 4U
#include <ViennaRNA/utils/structures.h>
```

Bitflag to indicate secondary structure notations using uppercase/lowercase letters from the latin alphabet.

See also

[vrna\\_ptable\\_from\\_string\(\)](#)

### 16.62.2.2 VRNA\_BRACKETS\_RND

```
#define VRNA_BRACKETS_RND 8U
#include <ViennaRNA/utils/structures.h>
```

Bitflag to indicate secondary structure notations using round brackets (parenthesis), ()

See also

[vrna\\_ptable\\_from\\_string\(\)](#), [vrna\\_db\\_flatten\(\)](#), [vrna\\_db\\_flatten\\_to\(\)](#)

### 16.62.2.3 VRNA\_BRACKETS\_CLY

```
#define VRNA_BRACKETS_CLY 16U
#include <ViennaRNA/utils/structures.h>
```

Bitflag to indicate secondary structure notations using curly brackets, {}

See also

[vrna\\_ptable\\_from\\_string\(\)](#), [vrna\\_db\\_flatten\(\)](#), [vrna\\_db\\_flatten\\_to\(\)](#)

### 16.62.2.4 VRNA\_BRACKETS\_ANG

```
#define VRNA_BRACKETS_ANG 32U
#include <ViennaRNA/utils/structures.h>
```

Bitflag to indicate secondary structure notations using angular brackets, <>

See also

[vrna\\_ptable\\_from\\_string\(\)](#), [vrna\\_db\\_flatten\(\)](#), [vrna\\_db\\_flatten\\_to\(\)](#)

### 16.62.2.5 VRNA\_BRACKETS\_SQR

```
#define VRNA_BRACKETS_SQR 64U
#include <ViennaRNA/utils/structures.h>
Bitflag to indicate secondary structure notations using square brackets, [ ]
```

See also

[vrna\\_ptable\\_from\\_string\(\)](#), [vrna\\_db\\_flatten\(\)](#), [vrna\\_db\\_flatten\\_to\(\)](#)

### 16.62.2.6 VRNA\_BRACKETS\_DEFAULT

```
#define VRNA_BRACKETS_DEFAULT
#include <ViennaRNA/utils/structures.h>
Value:
(VRNA_BRACKETS_RND | \
VRNA_BRACKETS_CLY | \
VRNA_BRACKETS_ANG | \
VRNA_BRACKETS_SQR)
```

Default bitmask to indicate secondary structure notation using any pair of brackets.

This set of matching brackets/parenthesis is always nested, i.e. pseudo-knot free, in WUSS format. However, in general different kinds of brackets are mostly used for annotating pseudo-knots. Thus special care has to be taken to remove pseudo-knots if this bitmask is used in functions that return secondary structures without pseudo-knots!

See also

[vrna\\_ptable\\_from\\_string\(\)](#), [vrna\\_db\\_flatten\(\)](#), [vrna\\_db\\_flatten\\_to\(\)](#), [vrna\\_db\\_pk\\_remove\(\)](#) [vrna\\_pt\\_pk\\_remove\(\)](#)

### 16.62.2.7 VRNA\_BRACKETS\_ANY

```
#define VRNA_BRACKETS_ANY
#include <ViennaRNA/utils/structures.h>
Value:
(VRNA_BRACKETS_RND | \
VRNA_BRACKETS_CLY | \
VRNA_BRACKETS_ANG | \
VRNA_BRACKETS_SQR | \
VRNA_BRACKETS_ALPHA)
```

Bitmask to indicate secondary structure notation using any pair of brackets or uppercase/lowercase alphabet letters.

See also

[vrna\\_ptable\\_from\\_string\(\)](#), [vrna\\_db\\_pk\\_remove\(\)](#), [vrna\\_db\\_flatten\(\)](#), [vrna\\_db\\_flatten\\_to\(\)](#)

## 16.62.3 Function Documentation

### 16.62.3.1 vrna\_db\_pack()

```
char * vrna_db_pack (
    const char * struc )
#include <ViennaRNA/utils/structures.h>
```

Pack secondary secondary structure, 5:1 compression using base 3 encoding.

Returns a binary string encoding of the secondary structure using a 5:1 compression scheme. The string is NULL terminated and can therefore be used with standard string functions such as strcmp(). Useful for programs that need to keep many structures in memory.

See also

[vrna\\_db\\_unpack\(\)](#)

**Parameters**

|              |   |
|--------------|---|
| <i>struc</i> | The secondary structure in dot-bracket notation |
|--------------|---|

**Returns**

The binary encoded structure

**16.62.3.2 vrna\_db\_unpack()**

```
char * vrna_db_unpack (
    const char * packed )
#include <ViennaRNA/utils/structures.h>
Unpack secondary structure previously packed with vrna\_db\_pack\(\)
Translate a compressed binary string produced by vrna\_db\_pack\(\) back into the familiar dot-bracket notation.
```

**See also**

[vrna\\_db\\_pack\(\)](#)

**Parameters**

|               |   |
|---------------|---|
| <i>packed</i> | The binary encoded packed secondary structure |
|---------------|---|

**Returns**

The unpacked secondary structure in dot-bracket notation

**16.62.3.3 vrna\_db\_flatten()**

```
void vrna_db_flatten (
    char * structure,
    unsigned int options )
#include <ViennaRNA/utils/structures.h>
Substitute pairs of brackets in a string with parenthesis.
This function can be used to replace brackets of unusual types, such as angular brackets <>, to dot-bracket format.
The options parameter is used to specify which types of brackets will be replaced by round parenthesis () .
```

**See also**

[vrna\\_db\\_flatten\\_to\(\)](#), [VRNA\\_BRACKETS\\_RND](#), [VRNA\\_BRACKETS\\_ANG](#), [VRNA\\_BRACKETS\\_CLY](#),  
[VRNA\\_BRACKETS\\_SQR](#), [VRNA\\_BRACKETS\\_DEFAULT](#)

**Parameters**

|                  |  |
|------------------|--|
| <i>structure</i> | The structure string where brackets are flattened in-place           |
| <i>options</i>   | A bitmask to specify which types of brackets should be flattened out |

**SWIG Wrapper Notes** This function flattens an input structure string in-place! The second parameter is optional and defaults to [VRNA\\_BRACKETS\\_DEFAULT](#).

An overloaded version of this function exists, where an additional second parameter can be passed to specify the target brackets, i.e. the type of matching pair characters all brackets will be flattened to. Therefore, in the scripting language interface this function is a replace-

ment for `vrna_db_flatten_to()`.

#### 16.62.3.4 `vrna_db_flatten_to()`

```
void vrna_db_flatten_to (
    char * string,
    const char target[3],
    unsigned int options )
#include <ViennaRNA/utils/structures.h>
```

Substitute pairs of brackets in a string with another type of pair characters.

This function can be used to replace brackets in a structure annotation string, such as square brackets [] , to another type of pair characters, e.g. angular brackets <> .

The `target` array must contain a character for the 'pair open' annotation at position 0, and one for 'pair close' at position 1. `options` parameter is used to specify which types of brackets will be replaced by the new pairs.

**See also**

[vrna\\_db\\_flatten\(\)](#), [VRNA\\_BRACKETS\\_RND](#), [VRNA\\_BRACKETS\\_ANG](#), [VRNA\\_BRACKETS\\_CLY](#), [VRNA\\_BRACKETS\\_SQR](#), [VRNA\\_BRACKETS\\_DEFAULT](#)

**Parameters**

|                      |  |
|----------------------|--|
| <code>string</code>  | The structure string where brackets are flattened in-place           |
| <code>target</code>  | The new pair characters the string will be flattened to              |
| <code>options</code> | A bitmask to specify which types of brackets should be flattened out |

**SWIG Wrapper Notes** This function is available as an overloaded version of `vrna_db_flatten()`

#### 16.62.3.5 `vrna_db_from_ptable()`

```
char * vrna_db_from_ptable (
    const short * pt )
#include <ViennaRNA/utils/structures.h>
```

Convert a pair table into dot-parenthesis notation.

This function also converts pair table formatted structures that contain pseudoknots. Non-nested base pairs result in additional pairs of parenthesis and brackets within the resulting dot-parenthesis string. The following pairs are available: (), [], {}, <>, as well as pairs of matching upper-/lower-case characters from the alphabet A-Z.

**Note**

In cases where the level of non-nested base pairs exceeds the maximum number of 30 different base pair indicators (4 parenthesis/brackets, 26 matching characters), a warning is printed and the remaining base pairs are left out from the conversion.

**Parameters**

|                 |                             |
|-----------------|-----------------------------|
| <code>pt</code> | The pair table to be copied |
|-----------------|-----------------------------|

**Returns**

A char pointer to the dot-bracket string

**16.62.3.6 vrna\_db\_from\_plist()**

```
char * vrna_db_from_plist (
    vrna_ep_t * pairs,
    unsigned int n )
#include <ViennaRNA/utils/structures.h>
```

Convert a list of base pairs into dot-bracket notation.

**See also**

[vrna\\_plist\(\)](#)

**Parameters**

|              |   |
|--------------|---|
| <i>pairs</i> | A <a href="#">vrna_ep_t</a> containing the pairs to be included in the dot-bracket string |
| <i>n</i>     | The length of the structure (number of nucleotides)                                       |

**Returns**

The dot-bracket string containing the provided base pairs

**16.62.3.7 vrna\_db\_to\_element\_string()**

```
char * vrna_db_to_element_string (
    const char * structure )
#include <ViennaRNA/utils/structures.h>
```

Convert a secondary structure in dot-bracket notation to a nucleotide annotation of loop contexts.

**Parameters**

|                  |   |
|------------------|---|
| <i>structure</i> | The secondary structure in dot-bracket notation |
|------------------|---|

**Returns**

A string annotating each nucleotide according to it's structural context

**16.62.3.8 vrna\_db\_pk\_remove()**

```
char * vrna_db_pk_remove (
    const char * structure,
    unsigned int options )
#include <ViennaRNA/utils/structures.h>
```

Remove pseudo-knots from an input structure.

This function removes pseudo-knots from an input structure by determining the minimum number of base pairs that need to be removed to make the structure pseudo-knot free.

To accomplish that, we use a dynamic programming algorithm similar to the Nussinov maximum matching approach.

The input structure must be in a dot-bracket string like form where crossing base pairs are denoted by the use of additional types of matching brackets, e.g. <>, {}, [], {}. Furthermore, crossing pairs may be annotated by matching uppercase/lowercase letters from the alphabet A–Z. For the latter, the uppercase letter must be the 5' and the lowercase letter the 3' nucleotide of the base pair. The actual type of brackets to be recognized by this function must be specified through the *options* parameter.

**Note**

Brackets in the input structure string that are not covered by the `options` bitmask will be silently ignored!

**See also**

[vrna\\_pt\\_pk\\_remove\(\)](#), [vrna\\_db\\_flatten\(\)](#), [VRNA\\_BRACKETS\\_RND](#), [VRNA\\_BRACKETS\\_ANG](#), [VRNA\\_BRACKETS\\_CLY](#),  
[VRNA\\_BRACKETS\\_SQR](#), [VRNA\\_BRACKETS\\_ALPHA](#), [VRNA\\_BRACKETS\\_DEFAULT](#), [VRNA\\_BRACKETS\\_ANY](#)

**Parameters**

|                        |   |
|------------------------|---|
| <code>structure</code> | Input structure in dot-bracket format that may include pseudo-knots |
| <code>options</code>   | A bitmask to specify which types of brackets should be processed    |

**Returns**

The input structure devoid of pseudo-knots in dot-bracket notation

**SWIG Wrapper Notes** This function is available as an overloaded function `db_pk_remove()` where the optional second parameter `options` defaults to [VRNA\\_BRACKETS\\_ANY](#).

## 16.63 Washington University Secondary Structure (WUSS) notation

The WUSS notation, as frequently used for consensus secondary structures in [Stockholm 1.0 format](#).

### 16.63.1 Detailed Description

The WUSS notation, as frequently used for consensus secondary structures in [Stockholm 1.0 format](#).

This notation allows for a fine-grained annotation of base pairs and unpaired nucleotides, including pseudo-knots. Below, you'll find a list of secondary structure elements and their corresponding WUSS annotation (See also the infernal user guide at <http://eddylab.org/infernal/Userguide.pdf>)

- **Base pairs**

Nested base pairs are annotated by matching pairs of the symbols `<>`, `( )`, `{ }`, and `[ ]`. Each of the matching pairs of parenthesis have their special meaning, however, when used as input in our programs, e.g. structure constraint, these details are usually ignored. Furthermore, base pairs that constitute as pseudo-knot are denoted by letters from the latin alphabet and are, if not denoted otherwise, ignored entirely in our programs.

- **Hairpin loops**

Unpaired nucleotides that constitute the hairpin loop are indicated by underscores, `_`.

Example: `<<<<_____>>>>`

- **Bulges and interior loops**

Residues that constitute a bulge or interior loop are denoted by dashes, `-`.

Example: `((--<<_____>>-))`

- **Multibranch loops**

Unpaired nucleotides in multibranch loops are indicated by commas `,`.

Example: `((,,<<_____>>,<<_____>>))`

- **External residues**

Single stranded nucleotides in the exterior loop, i.e. not enclosed by any other pair are denoted by colons, `:`.

Example: `<<<_____>>>:::`

- **Insertions**

In cases where an alignment represents the consensus with a known structure, insertions relative to the known structure are denoted by periods, `..`. Regions where local structural alignment was invoked, leaving regions of both target and query sequence unaligned, are indicated by tildes, `~`.

**Note**

These symbols only appear in alignments of a known (query) structure annotation to a target sequence of unknown structure.

- **Pseudo-knots**

The WUSS notation allows for annotation of pseudo-knots using pairs of upper-case/lower-case letters.

**Note**

Our programs and library functions usually ignore pseudo-knots entirely treating them as unpaired nucleotides, if not stated otherwise.

Example: <<<\_AAA\_\_\_\_>>>aaa

Collaboration diagram for Washington University Secondary Structure (WUSS) notation:

## Functions

- `char * vrna_db_from_WUSS (const char *wuss)`

*Convert a WUSS annotation string to dot-bracket format.*

### 16.63.2 Function Documentation

#### 16.63.2.1 `vrna_db_from_WUSS()`

```
char * vrna_db_from_WUSS (
    const char * wuss )
#include <ViennaRNA/utils/structures.h>
Convert a WUSS annotation string to dot-bracket format.
```

**Note**

This function flattens all brackets, and treats pseudo-knots annotated by matching pairs of upper/lowercase letters as unpaired nucleotides

**Parameters**

|                   |                                   |
|-------------------|-----------------------------------|
| <code>wuss</code> | The input string in WUSS notation |
|-------------------|-----------------------------------|

**Returns**

A dot-bracket notation of the input secondary structure

## 16.64 Pair Table Representation of Secondary Structures

### 16.64.1 Detailed Description

Collaboration diagram for Pair Table Representation of Secondary Structures:

## Functions

- `short * vrna_ptable (const char *structure)`  
*Create a pair table from a dot-bracket notation of a secondary structure.*
- `short * vrna_ptable_from_string (const char *structure, unsigned int options)`  
*Create a pair table for a secondary structure string.*
- `short * vrna_pt_pk_get (const char *structure)`

- Create a pair table of a secondary structure (pseudo-knot version)*
- `short * vrna_ptable_copy (const short *pt)`  
*Get an exact copy of a pair table.*
  - `short * vrna_pt_ali_get (const char *structure)`  
*Create a pair table of a secondary structure (snoop align version)*
  - `short * vrna_pt_snoop_get (const char *structure)`  
*Create a pair table of a secondary structure (snoop version)*
  - `short * vrna_pt_pk_remove (const short *ptable, unsigned int options)`  
*Remove pseudo-knots from a pair table.*

## 16.64.2 Function Documentation

### 16.64.2.1 vrna\_ptable()

```
short * vrna_ptable (
    const char * structure )
#include <ViennaRNA/utils/structures.h>
```

Create a pair table from a dot-bracket notation of a secondary structure.

Returns a newly allocated table, such that table[i]=j if (i,j) pair or 0 if i is unpaired, table[0] contains the length of the structure.

See also

[vrna\\_ptable\\_from\\_string\(\)](#), [vrna\\_db\\_from\\_ptable\(\)](#)

#### Parameters

|                        |   |
|------------------------|---|
| <code>structure</code> | The secondary structure in dot-bracket notation |
|------------------------|---|

#### Returns

A pointer to the created pair\_table

**SWIG Wrapper Notes** This function is wrapped as overloaded function `ptable()` that takes an optional argument `options` to specify which type of matching brackets should be considered during conversion. The default set is round brackets, i.e. [VRNA\\_BRACKETS\\_RND](#).

### 16.64.2.2 vrna\_ptable\_from\_string()

```
short * vrna_ptable_from_string (
    const char * structure,
    unsigned int options )
#include <ViennaRNA/utils/structures.h>
```

Create a pair table for a secondary structure string.

This function takes an input string of a secondary structure annotation in [Dot-Bracket Notation](#) (a.k.a. [Dot-Parenthesis Notation](#)) or dot-bracket-ext-notation, and converts it into a pair table representation.

#### Note

This function also extracts crossing base pairs, i.e. pseudo-knots if more than a single matching bracket type is allowed through the bitmask `options`.

See also

[vrna\\_ptable\(\)](#), [vrna\\_db\\_from\\_ptable\(\)](#), [vrna\\_db\\_flatten\\_to\(\)](#), [vrna\\_pt\\_pk\\_remove\(\)](#) [VRNA\\_BRACKETS\\_RND](#),  
[VRNA\\_BRACKETS\\_ANG](#), [VRNA\\_BRACKETS\\_CLY](#), [VRNA\\_BRACKETS\\_SQR](#), [VRNA\\_BRACKETS\\_ALPHA](#),  
[VRNA\\_BRACKETS\\_DEFAULT](#), [VRNA\\_BRACKETS\\_ANY](#)

**Parameters**

|                  |  |
|------------------|--|
| <i>structure</i> | Secondary structure in dot-bracket-ext-notation                                    |
| <i>options</i>   | A bitmask to specify which brackets are recognized during conversion to pair table |

**Returns**

A pointer to a new pair table of the provided secondary structure

**SWIG Wrapper Notes** This function is wrapped as overloaded function `ptable()` that takes an optional argument `options` to specify which type of matching brackets should be considered during conversion. The default set is round brackets, i.e. `VRNA_BRACKETS_RND`.

**16.64.2.3 vrna\_pt\_pk\_get()**

```
short * vrna_pt_pk_get (
    const char * structure )
#include <ViennaRNA/utils/structures.h>
Create a pair table of a secondary structure (pseudo-knot version)
Returns a newly allocated table, such that table[i]=j if (i,j) pair or 0 if i is unpaired, table[0] contains the length of the
structure.
In contrast to vrna_ptable() this function also recognizes the base pairs denoted by '[' and ']' brackets. Thus, this
function behaves like
vrna_ptable_from_string(structure, #VRNA_BRACKETS_RND | VRNA_BRACKETS_SQR)
```

**See also**

[vrna\\_ptable\\_from\\_string\(\)](#)

**Parameters**

|                  |  |
|------------------|--|
| <i>structure</i> | The secondary structure in (extended) dot-bracket notation |
|------------------|--|

**Returns**

A pointer to the created pair\_table

**16.64.2.4 vrna\_ptable\_copy()**

```
short * vrna_ptable_copy (
    const short * pt )
#include <ViennaRNA/utils/structures.h>
Get an exact copy of a pair table.
```

**Parameters**

|           |                             |
|-----------|-----------------------------|
| <i>pt</i> | The pair table to be copied |
|-----------|-----------------------------|

**Returns**

A pointer to the copy of 'pt'

**16.64.2.5 vrna\_pt\_snoop\_get()**

```
short * vrna_pt_snoop_get (
    const char * structure )
#include <ViennaRNA/utils/structures.h>
Create a pair table of a secondary structure (snoop version)
returns a newly allocated table, such that: table[i]=j if (i,j) pair or 0 if i is unpaired, table[0] contains the length of the
structure. The special pseudoknotted H/ACA-mRNA structure is taken into account.
```

**16.64.2.6 vrna\_pt\_pk\_remove()**

```
short * vrna_pt_pk_remove (
    const short * ptable,
    unsigned int options )
#include <ViennaRNA/utils/structures.h>
```

Remove pseudo-knots from a pair table.

This function removes pseudo-knots from an input structure by determining the minimum number of base pairs that need to be removed to make the structure pseudo-knot free.

To accomplish that, we use a dynamic programming algorithm similar to the Nussinov maximum matching approach.

**See also**

[vrna\\_db\\_pk\\_remove\(\)](#)

**Parameters**

|                |   |
|----------------|---|
| <i>ptable</i>  | Input structure that may include pseudo-knots |
| <i>options</i> |   |

**Returns**

The input structure devoid of pseudo-knots

## 16.65 Pair List Representation of Secondary Structures

### 16.65.1 Detailed Description

Collaboration diagram for Pair List Representation of Secondary Structures:

#### Data Structures

- struct [vrna\\_elem\\_prob\\_s](#)

*Data structure representing a single entry of an element probability list (e.g. list of pair probabilities)* [More...](#)

#### Macros

- #define **VRNA\_PLIST\_TYPE\_BASEPAIR** 0  
*A Base Pair element.*
- #define **VRNA\_PLIST\_TYPE\_GQUAD** 1  
*A G-Quadruplex element.*
- #define **VRNA\_PLIST\_TYPE\_H\_MOTIF** 2

- #define **VRNA\_PLIST\_TYPE\_I\_MOTIF** 3  
*An Hairpin loop motif element.*
- #define **VRNA\_PLIST\_TYPE\_UD\_MOTIF** 4  
*An Internal loop motif element.*
- #define **VRNA\_PLIST\_TYPE\_STACK** 5  
*An Unstructured Domain motif element.*
- #define **VRNA\_PLIST\_TYPE\_UNPAIRED** 6  
*An unpaired base.*
- #define **VRNA\_PLIST\_TYPE\_TRIPLE** 7  
*One pair of a base triplet.*

## Typedefs

- **typedef struct vrna\_elem\_prob\_s vrna\_ep\_t**  
*Convenience typedef for data structure vrna\_elem\_prob\_s.*

## Functions

- **vrna\_ep\_t \* vrna\_plist** (const char \*struc, float pr)  
*Create a vrna\_ep\_t from a dot-bracket string.*

### 16.65.2 Data Structure Documentation

#### 16.65.2.1 struct vrna\_elem\_prob\_s

Data structure representing a single entry of an element probability list (e.g. list of pair probabilities)

See also

[vrna\\_plist\(\)](#), [vrna\\_plist\\_from\\_probs\(\)](#), [vrna\\_db\\_from\\_plist\(\)](#), [VRNA\\_PLIST\\_TYPE\\_BASEPAIR](#), [VRNA\\_PLIST\\_TYPE\\_GQUAD](#), [VRNA\\_PLIST\\_TYPE\\_H\\_MOTIF](#), [VRNA\\_PLIST\\_TYPE\\_I\\_MOTIF](#), [VRNA\\_PLIST\\_TYPE\\_UD\\_MOTIF](#), [VRNA\\_PLIST\\_TYPE\\_STACK](#)

#### Data Fields

- int **i**  
*Start position (usually 5' nucleotide that starts the element, e.g. base pair)*
- int **j**  
*End position (usually 3' nucleotide that ends the element, e.g. base pair)*
- float **p**  
*Probability of the element.*
- int **type**  
*Type of the element.*

### 16.65.3 Function Documentation

#### 16.65.3.1 vrna\_plist()

```
vrna_ep_t * vrna_plist (
    const char * struc,
    float pr )
#include <ViennaRNA/utils/structures.h>
Create a vrna_ep_t from a dot-bracket string.
```

The dot-bracket string is parsed and for each base pair an entry in the plist is created. The probability of each pair in the list is set by a function parameter.

The end of the plist is marked by sequence positions i as well as j equal to 0. This condition should be used to stop looping over its entries

#### Parameters

|              |  |
|--------------|--|
| <i>struc</i> | The secondary structure in dot-bracket notation      |
| <i>pr</i>    | The probability for each base pair used in the plist |

#### Returns

The plist array

## 16.66 Abstract Shapes Representation of Secondary Structures

Abstract Shapes, introduced by Giegerich et al. in (2004) [10], collapse the secondary structure while retaining the nestedness of helices and hairpin loops.

### 16.66.1 Detailed Description

Abstract Shapes, introduced by Giegerich et al. in (2004) [10], collapse the secondary structure while retaining the nestedness of helices and hairpin loops.

The abstract shapes representation abstracts the structure from individual base pairs and their corresponding location in the sequence, while retaining the inherent nestedness of helices and hairpin loops.

Below is a description of what is included in the abstract shapes abstraction for each respective level together with an example structure:

```
CGUCUAAAACUCAUCACCGUGUGGAGCUGCGACCCUCCCCUAGAUUCGAAGACGAG
((((((...(((.....)))))...(((.....))...))))))))..
```

| Shape Level | Description  | Result                   |
|-------------|--|--------------------------|
| 1           | Most accurate - all loops and all unpaired   | [__[_[]]_[_[]_↔]<br>] ]_ |
| 2           | Nesting pattern for all loop types and unpaired regions in external loop and multiloop | [[_[]][_[]_]]            |
| 3           | Nesting pattern for all loop types but no unpaired regions                             | [[[[]][[[[]]]]           |
| 4           | Helix nesting pattern in external loop and multiloop                                   | [[[[[[]]]]]              |
| 5           | Most abstract - helix nesting pattern and no unpaired regions                          | [[[[]]]]                 |

#### Note

Our implementations also provide the special Shape Level 0, which does not collapse any structural features but simply convert base pairs and unpaired nucleotides into their corresponding set of symbols for abstract shapes.

Collaboration diagram for Abstract Shapes Representation of Secondary Structures:

### Functions

- `char * vrna_abstract_shapes (const char *structure, unsigned int level)`  
*Convert a secondary structure in dot-bracket notation to its abstract shapes representation.*
- `char * vrna_abstract_shapes_pt (const short *pt, unsigned int level)`  
*Convert a secondary structure to its abstract shapes representation.*

## 16.66.2 Function Documentation

### 16.66.2.1 vrna\_abstract\_shapes()

```
char * vrna_abstract_shapes (
    const char * structure,
    unsigned int level )
#include <ViennaRNA/utils/structures.h>
```

Convert a secondary structure in dot-bracket notation to its abstract shapes representation.

This function converts a secondary structure into its abstract shapes representation as presented by Giegerich et al. 2004 [10].

See also

[vrna\\_abstract\\_shapes\\_pt\(\)](#)

#### Parameters

|                  |  |
|------------------|--|
| <i>structure</i> | A secondary structure in dot-bracket notation          |
| <i>level</i>     | The abstraction level (integer in the range of 0 to 5) |

#### Returns

The secondary structure in abstract shapes notation

**SWIG Wrapper Notes** This function is available as an overloaded function `abstract_shapes()` where the optional second parameter `level` defaults to 5.

### 16.66.2.2 vrna\_abstract\_shapes\_pt()

```
char * vrna_abstract_shapes_pt (
    const short * pt,
    unsigned int level )
#include <ViennaRNA/utils/structures.h>
```

Convert a secondary structure to its abstract shapes representation.

This function converts a secondary structure into its abstract shapes representation as presented by Giegerich et al. 2004 [10]. This function is equivalent to `vrna_db_to_shapes()`, but requires a pair table input instead of a dot-bracket structure.

#### Note

The length of the structure must be present at `pt[0]`!

See also

[vrna\\_abstract\\_shapes\(\)](#)

#### Parameters

|              |  |
|--------------|--|
| <i>pt</i>    | A secondary structure in pair table format             |
| <i>level</i> | The abstraction level (integer in the range of 0 to 5) |

#### Returns

The secondary structure in abstract shapes notation

**SWIG Wrapper Notes** This function is available as an overloaded function `abstract_shapes()` where the optional second parameter `level` defaults to 5.

## 16.67 Helix List Representation of Secondary Structures

### 16.67.1 Detailed Description

Collaboration diagram for Helix List Representation of Secondary Structures:

#### Data Structures

- struct `vrna_hx_s`  
*Data structure representing an entry of a helix list. [More...](#)*

#### Typedefs

- typedef struct `vrna_hx_s` `vrna_hx_t`  
*Convenience typedef for data structure `vrna_hx_s`.*

#### Functions

- `vrna_hx_t * vrna_hx_from_ptable (short *pt)`  
*Convert a pair table representation of a secondary structure into a helix list.*
- `vrna_hx_t * vrna_hx_merge (const vrna_hx_t *list, int maxdist)`  
*Create a merged helix list from another helix list.*

### 16.67.2 Data Structure Documentation

#### 16.67.2.1 struct `vrna_hx_s`

Data structure representing an entry of a helix list.

#### 16.67.3 Function Documentation

##### 16.67.3.1 `vrna_hx_from_ptable()`

```
vrna_hx_t * vrna_hx_from_ptable (
    short * pt )
#include <ViennaRNA/utils/structures.h>
Convert a pair table representation of a secondary structure into a helix list.
```

#### Parameters

|                 |  |
|-----------------|--|
| <code>pt</code> | The secondary structure in pair table representation |
|-----------------|--|

**Returns**

The secondary structure represented as a helix list

## 16.68 Tree Representation of Secondary Structures

Secondary structures can be readily represented as trees, where internal nodes represent base pairs, and leaves represent unpaired nucleotides. The dot-bracket structure string already is a tree represented by a string of parenthesis (base pairs) and dots for the leaf nodes (unpaired nucleotides).

### 16.68.1 Detailed Description

Secondary structures can be readily represented as trees, where internal nodes represent base pairs, and leaves represent unpaired nucleotides. The dot-bracket structure string already is a tree represented by a string of parenthesis (base pairs) and dots for the leaf nodes (unpaired nucleotides).

Alternatively, one may find representations with two types of node labels, **P** for paired and **U** for unpaired; a dot is then replaced by **(U)**, and each closed bracket is assigned an additional identifier **P**. We call this the expanded notation. In [8] a condensed representation of the secondary structure is proposed, the so-called homeomorphically irreducible tree (HIT) representation. Here a stack is represented as a single pair of matching brackets labeled **P** and weighted by the number of base pairs. Correspondingly, a contiguous strain of unpaired bases is shown as one pair of matching brackets labeled **U** and weighted by its length. Generally any string consisting of matching brackets and identifiers is equivalent to a plane tree with as many different types of nodes as there are identifiers.

Bruce Shapiro proposed a coarse grained representation [21], which, does not retain the full information of the secondary structure. He represents the different structure elements by single matching brackets and labels them as

- **H** (hairpin loop),
- **I** (interior loop),
- **B** (bulge),
- **M** (multi-loop), and
- **S** (stack).

We extend his alphabet by an extra letter for external elements **E**. Again these identifiers may be followed by a weight corresponding to the number of unpaired bases or base pairs in the structure element. All tree representations (except for the dot-bracket form) can be encapsulated into a virtual root (labeled **R**).

The following example illustrates the different linear tree representations used by the package:

Consider the secondary structure represented by the dot-bracket string (full tree) **.((...((...))...((...))))**. which is the most convenient condensed notation used by our programs and library functions.

Then, the following tree representations are equivalent:

- Expanded tree:  

$$((U)((U)(U)((((U)(U)(U)P)P)P)(U)(U)((((U)(U)P)P)P)P)(U)R)$$
- HIT representation (Fontana et al. 1993 [8]):  

$$((U1)((U2)((U3)P3)(U2)((U2)P2)P2)(U1)R)$$
- Coarse Grained **Tree** Representation (Shapiro 1988 [21]):
  - Short (with root node **R**, without stem nodes **S**):  

$$((H)((H)M)R)$$
  - Full (with root node **R**):  

$$(((((H)S)((H)S)M)S)R)$$
  - Extended (with root node **R**, with external nodes **E**):  

$$((((((H)S)((H)S)M)S)E)R)$$
  - Weighted (with root node **R**, with external nodes **E**):  

$$((((((H)S3)((H)S2)M4)S2)E2)R)$$

The Expanded tree is rather clumsy and mostly included for the sake of completeness. The different versions of Coarse Grained [Tree](#) Representations are variations of Shapiro's linear tree notation.

For the output of aligned structures from string editing, different representations are needed, where we put the label on both sides. The above examples for tree representations would then look like:

```
* a) (UU) (P (P (P (P (UU) (UU) (P (P (P (UU) (UU) P) P) P) (UU) (UU) (P (P (UU) (U...
* b) (UU) (P2 (P2 (U2U2) (P2 (U3U3) P3) (U2U2) (P2 (U2U2) P2) P2) (UU) P2) (UU)
* c) (B (M (HH) (HH) M) B)
*   (S (B (S (M (S (HH) S) (S (HH) S) M) S) B) S)
*   (E (S (B (S (M (S (HH) S) (S (HH) S) M) S) B) S) E)
* d) (R (E2 (S2 (B1 (S2 (M4 (S3 (H3) S3) ((H2) S2) M4) S2) B1) S2) E2) R)
*
```

Aligned structures additionally contain the gap character `_`. Collaboration diagram for Tree Representation of Secondary Structures:

## Macros

- `#define VRNA_STRUCTURE_TREE_HIT 1U`  
*Homeomorphically Irreducible [Tree](#) (HIT) representation of a secondary structure.*
- `#define VRNA_STRUCTURE_TREE_SHAPIRO_SHORT 2U`  
*(short) Coarse Grained representation of a secondary structure*
- `#define VRNA_STRUCTURE_TREE_SHAPIRO 3U`  
*(full) Coarse Grained representation of a secondary structure*
- `#define VRNA_STRUCTURE_TREE_SHAPIRO_EXT 4U`  
*(extended) Coarse Grained representation of a secondary structure*
- `#define VRNA_STRUCTURE_TREE_SHAPIRO_WEIGHT 5U`  
*(weighted) Coarse Grained representation of a secondary structure*
- `#define VRNA_STRUCTURE_TREE_EXPANDED 6U`  
*Expanded [Tree](#) representation of a secondary structure.*

## Functions

- `char * vrna_db_to_tree_string (const char *structure, unsigned int type)`  
*Convert a Dot-Bracket structure string into tree string representation.*
- `char * vrna_tree_string_unweight (const char *structure)`  
*Remove weights from a linear string tree representation of a secondary structure.*
- `char * vrna_tree_string_to_db (const char *tree)`  
*Convert a linear tree string representation of a secondary structure back to Dot-Bracket notation.*

### 16.68.2 Macro Definition Documentation

#### 16.68.2.1 VRNA\_STRUCTURE\_TREE\_HIT

```
#define VRNA_STRUCTURE_TREE_HIT 1U
#include <ViennaRNA/utils/structures.h>
Homeomorphically Irreducible Tree (HIT) representation of a secondary structure.
```

See also

[vrna\\_db\\_to\\_tree\\_string\(\)](#)

### 16.68.2.2 VRNA\_STRUCTURE\_TREE\_SHAPIRO\_SHORT

```
#define VRNA_STRUCTURE_TREE_SHAPIRO_SHORT 2U
#include <ViennaRNA/utils/structures.h>
(short) Coarse Grained representation of a secondary structure
```

See also

[vrna\\_db\\_to\\_tree\\_string\(\)](#)

### 16.68.2.3 VRNA\_STRUCTURE\_TREE\_SHAPIRO

```
#define VRNA_STRUCTURE_TREE_SHAPIRO 3U
#include <ViennaRNA/utils/structures.h>
(full) Coarse Grained representation of a secondary structure
```

See also

[vrna\\_db\\_to\\_tree\\_string\(\)](#)

### 16.68.2.4 VRNA\_STRUCTURE\_TREE\_SHAPIRO\_EXT

```
#define VRNA_STRUCTURE_TREE_SHAPIRO_EXT 4U
#include <ViennaRNA/utils/structures.h>
(extended) Coarse Grained representation of a secondary structure
```

See also

[vrna\\_db\\_to\\_tree\\_string\(\)](#)

### 16.68.2.5 VRNA\_STRUCTURE\_TREE\_SHAPIRO\_WEIGHT

```
#define VRNA_STRUCTURE_TREE_SHAPIRO_WEIGHT 5U
#include <ViennaRNA/utils/structures.h>
(weighted) Coarse Grained representation of a secondary structure
```

See also

[vrna\\_db\\_to\\_tree\\_string\(\)](#)

### 16.68.2.6 VRNA\_STRUCTURE\_TREE\_EXPANDED

```
#define VRNA_STRUCTURE_TREE_EXPANDED 6U
#include <ViennaRNA/utils/structures.h>
Expanded Tree representation of a secondary structure.
```

See also

[vrna\\_db\\_to\\_tree\\_string\(\)](#)

## 16.68.3 Function Documentation

### 16.68.3.1 vrna\_db\_to\_tree\_string()

```
char * vrna_db_to_tree_string (
    const char * structure,
    unsigned int type )
#include <ViennaRNA/utils/structures.h>
```

Convert a Dot-Bracket structure string into tree string representation.

This function allows one to convert a secondary structure in dot-bracket notation into one of the various tree representations for secondary structures. The resulting tree is then represented as a string of parenthesis and node symbols, similar to the Newick format.

Currently we support conversion into the following formats, denoted by the value of parameter `type`:

- `VRNA_STRUCTURE_TREE_HIT` - Homeomorphically Irreducible [Tree](#) (HIT) representation of a secondary structure. (See also Fontana et al. 1993 [8])
- `VRNA_STRUCTURE_TREE_SHAPIRO_SHORT` - (short) Coarse Grained representation of a secondary structure (same as Shapiro 1988 [21], but with root node `R` and without `S` nodes for the stems)
- `VRNA_STRUCTURE_TREE_SHAPIRO` - (full) Coarse Grained representation of a secondary structure (See also Shapiro 1988 [21])
- `VRNA_STRUCTURE_TREE_SHAPIRO_EXT` - (extended) Coarse Grained representation of a secondary structure (same as Shapiro 1988 [21], but external nodes denoted as `E`)
- `VRNA_STRUCTURE_TREE_SHAPIRO_WEIGHT` - (weighted) Coarse Grained representation of a secondary structure (same as `VRNA_STRUCTURE_TREE_SHAPIRO_EXT` but with additional weights for number of unpaired nucleotides in loop, and number of pairs in stems)
- `VRNA_STRUCTURE_TREE_EXPANDED` - Expanded [Tree](#) representation of a secondary structure.

#### See also

[Tree Representations of Secondary Structures](#)

#### Parameters

|                        |  |
|------------------------|--|
| <code>structure</code> | The null-terminated dot-bracket structure string             |
| <code>type</code>      | A switch to determine the type of tree string representation |

#### Returns

A tree representation of the input `structure`

### 16.68.3.2 vrna\_tree\_string\_unweight()

```
char * vrna_tree_string_unweight (
    const char * structure )
#include <ViennaRNA/utils/structures.h>
```

Remove weights from a linear string tree representation of a secondary structure.

This function strips the weights of a linear string tree representation such as HIT, or Coarse Grained [Tree](#) sensu Shapiro [21]

#### See also

[vrna\\_db\\_to\\_tree\\_string\(\)](#)

#### Parameters

|                        |   |
|------------------------|---|
| <code>structure</code> | A linear string tree representation of a secondary structure with weights |
|------------------------|---|

**Returns**

A linear string tree representation of a secondary structure without weights

**16.68.3.3 vrna\_tree\_string\_to\_db()**

```
char * vrna_tree_string_to_db (
    const char * tree )
#include <ViennaRNA/utils/structures.h>
```

Convert a linear tree string representation of a secondary structure back to Dot-Bracket notation.

**Warning**

This function only accepts *Expanded* and *HIT* tree representations!

**See also**

[vrna\\_db\\_to\\_tree\\_string\(\)](#), [VRNA\\_STRUCTURE\\_TREE\\_EXPANDED](#), [VRNA\\_STRUCTURE\\_TREE\\_HIT](#), [Tree Representations of Secondary Structures](#)

**Parameters**

|             |  |
|-------------|--|
| <i>tree</i> | A linear tree string representation of a secondary structure |
|-------------|--|

**Returns**

A dot-bracket notation of the secondary structure provided in *tree*

**16.69 Distance measures between Secondary Structures****16.69.1 Detailed Description**

Collaboration diagram for Distance measures between Secondary Structures:

**Functions**

- int [vrna\\_bp\\_distance\\_pt](#) (const short \*pt1, const short \*pt2)  
*Compute the "base pair" distance between two pair tables pt1 and pt2 of secondary structures.*
- int [vrna\\_bp\\_distance](#) (const char \*str1, const char \*str2)  
*Compute the "base pair" distance between two secondary structures s1 and s2.*

**16.69.2 Function Documentation****16.69.2.1 vrna\_bp\_distance\_pt()**

```
int vrna_bp_distance_pt (
    const short * pt1,
    const short * pt2 )
#include <ViennaRNA/utils/structures.h>
```

Compute the "base pair" distance between two pair tables pt1 and pt2 of secondary structures.

The pair tables should have the same length. dist = number of base pairs in one structure but not in the other same as edit distance with open-pair close-pair as move-set

See also

[vrna\\_bp\\_distance\(\)](#)

Parameters

|            |  |
|------------|--|
| <i>pt1</i> | First structure in dot-bracket notation  |
| <i>pt2</i> | Second structure in dot-bracket notation |

Returns

The base pair distance between pt1 and pt2

**SWIG Wrapper Notes** This function is available as an overloaded method [bp\\_distance\(\)](#).

### 16.69.2.2 vrna\_bp\_distance()

```
int vrna_bp_distance (
    const char * str1,
    const char * str2 )
#include <ViennaRNA/utils/structures.h>
Compute the "base pair" distance between two secondary structures s1 and s2.
This is a wrapper around vrna\_bp\_distance\_pt\(\). The sequences should have the same length. dist = number of
base pairs in one structure but not in the other same as edit distance with open-pair close-pair as move-set
```

See also

[vrna\\_bp\\_distance\\_pt\(\)](#)

Parameters

|             |  |
|-------------|--|
| <i>str1</i> | First structure in dot-bracket notation  |
| <i>str2</i> | Second structure in dot-bracket notation |

Returns

The base pair distance between str1 and str2

**SWIG Wrapper Notes** This function is available as an overloaded method [bp\\_distance\(\)](#). Note that the SWIG wrapper takes two structure in dot-bracket notation and converts them into pair tables using [vrna\\_ptable\\_from\\_string\(\)](#). The resulting pair tables are then internally passed to [vrna\\_bp\\_distance\\_pt\(\)](#). To control which kind of matching brackets will be used during conversion, the optional argument `options` can be used. See also the description of [vrna\\_ptable\\_from\\_string\(\)](#) for available options. (default: `VRNA_BRACKETS_RND`).

## 16.70 Multiple Sequence Alignment Utilities

Functions to extract features from and to manipulate multiple sequence alignments.

### 16.70.1 Detailed Description

Functions to extract features from and to manipulate multiple sequence alignments.

Collaboration diagram for Multiple Sequence Alignment Utilities:

### Modules

- [Deprecated Interface for Multiple Sequence Alignment Utilities](#)

## Files

- file [alignments.h](#)

*Various utility- and helper-functions for sequence alignments and comparative structure prediction.*

## Data Structures

- struct [vrna\\_pinfo\\_s](#)

*A base pair info structure. [More...](#)*

## Macros

- #define **VRNA\_ALN\_DEFAULT** 0U  
*Use default alignment settings.*
- #define **VRNA\_ALN\_RNA** 1U  
*Convert to RNA alphabet.*
- #define **VRNA\_ALN\_DNA** 2U  
*Convert to DNA alphabet.*
- #define **VRNA\_ALN\_UPPERCASE** 4U  
*Convert to uppercase nucleotide letters.*
- #define **VRNA\_ALN\_LOWERCASE** 8U  
*Convert to lowercase nucleotide letters.*
- #define **VRNA\_MEASURE\_SHANNON\_ENTROPY** 1U  
*Flag indicating Shannon Entropy measure.*

## Typedefs

- typedef struct [vrna\\_pinfo\\_s](#) [vrna\\_pinfo\\_t](#)

*Typename for the base pair info representing data structure [vrna\\_pinfo\\_s](#).*

## Functions

- int [vrna\\_aln\\_mpi](#) (const char \*\*alignment)  
*Get the mean pairwise identity in steps from ?to?(ident)*
- [vrna\\_pinfo\\_t](#) \* [vrna\\_aln\\_pinfo](#) ([vrna\\_fold\\_compound\\_t](#) \*vc, const char \*structure, double threshold)  
*Retrieve an array of [vrna\\_pinfo\\_t](#) structures from precomputed pair probabilities.*
- char \*\* [vrna\\_aln\\_slice](#) (const char \*\*alignment, unsigned int i, unsigned int j)  
*Slice out a subalignment from a larger alignment.*
- void [vrna\\_aln\\_free](#) (char \*\*alignment)  
*Free memory occupied by a set of aligned sequences.*
- char \*\* [vrna\\_aln\\_uppercase](#) (const char \*\*alignment)  
*Create a copy of an alignment with only uppercase letters in the sequences.*
- char \*\* [vrna\\_aln\\_toRNA](#) (const char \*\*alignment)  
*Create a copy of an alignment where DNA alphabet is replaced by RNA alphabet.*
- char \*\* [vrna\\_aln\\_copy](#) (const char \*\*alignment, unsigned int options)  
*Make a copy of a multiple sequence alignment.*
- float \* [vrna\\_aln\\_conservation\\_struct](#) (const char \*\*alignment, const char \*structure, const [vrna\\_md\\_t](#) \*md)  
*Compute base pair conservation of a consensus structure.*
- float \* [vrna\\_aln\\_conservation\\_col](#) (const char \*\*alignment, const [vrna\\_md\\_t](#) \*md\_p, unsigned int options)  
*Compute nucleotide conservation in an alignment.*
- char \* [vrna\\_aln\\_consensus\\_sequence](#) (const char \*\*alignment, const [vrna\\_md\\_t](#) \*md\_p)  
*Compute the consensus sequence for a given multiple sequence alignment.*
- char \* [vrna\\_aln\\_consensus\\_mis](#) (const char \*\*alignment, const [vrna\\_md\\_t](#) \*md\_p)  
*Compute the Most Informative Sequence (MIS) for a given multiple sequence alignment.*

## 16.70.2 Data Structure Documentation

### 16.70.2.1 struct vrna\_pinfo\_s

A base pair info structure.

For each base pair (i,j) with i,j in [0, n-1] the structure lists:

- its probability 'p'
- an entropy-like measure for its well-definedness 'ent'
- the frequency of each type of pair in 'bp[]'
  - 'bp[0]' contains the number of non-compatible sequences
  - 'bp[1]' the number of CG pairs, etc.

#### Data Fields

- unsigned **i**  
*nucleotide position i*
- unsigned **j**  
*nucleotide position j*
- float **p**  
*Probability.*
- float **ent**  
*Pseudo entropy for  $p(i, j) = S_i + S_j - p_{ij} * \ln(p_{ij})$ .*
- short **bp** [8]  
*Frequencies of pair\_types.*
- char **comp**  
*1 iff pair is in mfe structure*

## 16.70.3 Macro Definition Documentation

### 16.70.3.1 VRNA\_MEASURE\_SHANNON\_ENTROPY

```
#define VRNA_MEASURE_SHANNON_ENTROPY 1U
#include <ViennaRNA/utils/alignments.h>
Flag indicating Shannon Entropy measure.
Shannon Entropy is defined as  $H = -\sum_c p_c \cdot \log_2 p_c$ 
```

## 16.70.4 Function Documentation

### 16.70.4.1 vrna\_aln\_mpi()

```
int vrna_aln_mpi (
    const char ** alignment )
#include <ViennaRNA/utils/alignments.h>
Get the mean pairwise identity in steps from ?to?(ident)
```

#### Parameters

|                  |                   |
|------------------|-------------------|
| <i>alignment</i> | Aligned sequences |
|------------------|-------------------|

**Returns**

The mean pairwise identity

**16.70.4.2 vrna\_aln\_pinfo()**

```
vrna_pinfo_t * vrna_aln_pinfo (
    vrna_fold_compound_t * vc,
    const char * structure,
    double threshold )
#include <ViennaRNA/utils/alignments.h>
```

Retrieve an array of `vrna_pinfo_t` structures from precomputed pair probabilities.

This array of structures contains information about positionwise pair probabilities, base pair entropy and more

**See also**

`vrna_pinfo_t`, and `vrna_pf()`

**Parameters**

|                        |  |
|------------------------|--|
| <code>vc</code>        | The <code>vrna_fold_compound_t</code> of type <code>VRNA_FC_TYPE_COMPARATIVE</code> with precomputed partition function matrices |
| <code>structure</code> | An optional structure in dot-bracket notation (Maybe NULL)   |
| <code>threshold</code> | Do not include results with pair probabilities below threshold   |

**Returns**

The `vrna_pinfo_t` array

**16.70.4.3 vrna\_aln\_slice()**

```
char ** vrna_aln_slice (
    const char ** alignment,
    unsigned int i,
    unsigned int j )
#include <ViennaRNA/utils/alignments.h>
```

Slice out a subalignment from a larger alignment.

**Note**

The user is responsible to free the memory occupied by the returned subalignment

**See also**

`vrna_aln_free()`

**Parameters**

|                        |  |
|------------------------|--|
| <code>alignment</code> | The input alignment                            |
| <code>i</code>         | The first column of the subalignment (1-based) |
| <code>j</code>         | The last column of the subalignment (1-based)  |

**Returns**

The subalignment between column  $i$  and  $j$

**16.70.4.4 vrna\_aln\_free()**

```
void vrna_aln_free (
    char ** alignment )
#include <ViennaRNA/utils/alignments.h>
Free memory occupied by a set of aligned sequences.
```

**Parameters**

|                  |                     |
|------------------|---------------------|
| <i>alignment</i> | The input alignment |
|------------------|---------------------|

**16.70.4.5 vrna\_aln\_uppercase()**

```
char ** vrna_aln_uppercase (
    const char ** alignment )
#include <ViennaRNA/utils/alignments.h>
Create a copy of an alignment with only uppercase letters in the sequences.
```

**See also**

[vrna\\_aln\\_copy](#)

**Parameters**

|                  |  |
|------------------|--|
| <i>alignment</i> | The input sequence alignment (last entry must be <i>NULL</i> terminated) |
|------------------|--|

**Returns**

A copy of the input alignment where lowercase sequence letters are replaced by uppercase letters

**16.70.4.6 vrna\_aln\_toRNA()**

```
char ** vrna_aln_toRNA (
    const char ** alignment )
#include <ViennaRNA/utils/alignments.h>
Create a copy of an alignment where DNA alphabet is replaced by RNA alphabet.
```

**See also**

[vrna\\_aln\\_copy](#)

**Parameters**

|                  |  |
|------------------|--|
| <i>alignment</i> | The input sequence alignment (last entry must be <i>NULL</i> terminated) |
|------------------|--|

**Returns**

A copy of the input alignment where DNA alphabet is replaced by RNA alphabet (T -> U)

#### 16.70.4.7 vrna\_aln\_copy()

```
char ** vrna_aln_copy (
    const char ** alignment,
    unsigned int options )
#include <ViennaRNA/utils/alignments.h>
```

Make a copy of a multiple sequence alignment.

This function allows one to create a copy of a multiple sequence alignment. The `options` parameter additionally allows for sequence manipulation, such as converting DNA to RNA alphabet, and conversion to uppercase letters.

See also

[vrna\\_aln\\_copy\(\)](#), [VRNA\\_ALN\\_RNA](#), [VRNA\\_ALN\\_UPPERCASE](#), [VRNA\\_ALN\\_DEFAULT](#)

#### Parameters

|                        |  |
|------------------------|--|
| <code>alignment</code> | The input sequence alignment (last entry must be <code>NULL</code> terminated) |
| <code>options</code>   | Option flags indicating whether the aligned sequences should be converted      |

#### Returns

A (manipulated) copy of the input alignment

#### 16.70.4.8 vrna\_aln\_conservation\_struct()

```
float * vrna_aln_conservation_struct (
    const char ** alignment,
    const char * structure,
    const vrna_md_t * md )
#include <ViennaRNA/utils/alignments.h>
```

Compute base pair conservation of a consensus structure.

This function computes the base pair conservation (fraction of canonical base pairs) of a consensus structure given a multiple sequence alignment. The base pair types that are considered canonical may be specified using the `vrna_md_t.pair` array. Passing `NULL` as parameter `md` results in default pairing rules, i.e. canonical Watson-Crick and GU Wobble pairs.

#### Parameters

|                        |  |
|------------------------|--|
| <code>alignment</code> | The input sequence alignment (last entry must be <code>NULL</code> terminated) |
| <code>structure</code> | The consensus structure in dot-bracket notation                                |
| <code>md</code>        | Model details that specify compatible base pairs (Maybe <code>NULL</code> )    |

#### Returns

A 1-based vector of base pair conservations

**SWIG Wrapper Notes** This function is available in an overloaded form where the last parameter may be omitted, indicating `md = NULL`

#### 16.70.4.9 vrna\_aln\_conservation\_col()

```
float * vrna_aln_conservation_col (
    const char ** alignment,
    const vrna_md_t * md,
    unsigned int options )
```

```
#include <ViennaRNA/utils/alignments.h>
```

Compute nucleotide conservation in an alignment.

This function computes the conservation of nucleotides in alignment columns. The simplest measure is Shannon Entropy and can be selected by passing the `VRNA_MEASURE_SHANNON_ENTROPY` flag in the `options` parameter.

#### Note

Currently, only `VRNA_MEASURE_SHANNON_ENTROPY` is supported as conservation measure.

#### See also

[VRNA\\_MEASURE\\_SHANNON\\_ENTROPY](#)

#### Parameters

|                        |  |
|------------------------|--|
| <code>alignment</code> | The input sequence alignment (last entry must be <code>NULL</code> terminated) |
| <code>md</code>        | Model details that specify known nucleotides (Maybe <code>NULL</code> )        |
| <code>options</code>   | A flag indicating which measure of conservation should be applied              |

#### Returns

A 1-based vector of column conservations

**SWIG Wrapper Notes** This function is available in an overloaded form where the last two parameters may be omitted, indicating `md = NULL`, and `options = VRNA_MEASURE_SHANNON_ENTROPY`, respectively.

#### 16.70.4.10 vrna\_aln\_consensus\_sequence()

```
char * vrna_aln_consensus_sequence (
    const char ** alignment,
    const vrna_md_t * md_p )
#include <ViennaRNA/utils/alignments.h>
Compute the consensus sequence for a given multiple sequence alignment.
```

#### Parameters

|                        |  |
|------------------------|--|
| <code>alignment</code> | The input sequence alignment (last entry must be <code>NULL</code> terminated) |
| <code>md_p</code>      | Model details that specify known nucleotides (Maybe <code>NULL</code> )        |

#### Returns

The consensus sequence of the alignment, i.e. the most frequent nucleotide for each alignment column

#### 16.70.4.11 vrna\_aln\_consensus\_mis()

```
char * vrna_aln_consensus_mis (
    const char ** alignment,
    const vrna_md_t * md_p )
#include <ViennaRNA/utils/alignments.h>
Compute the Most Informative Sequence (MIS) for a given multiple sequence alignment.
The most informative sequence (MIS) [9] displays for each alignment column the nucleotides with frequency greater
than the background frequency, projected into IUPAC notation. Columns where gaps are over-represented are in
lower case.
```

**Parameters**

|                  |  |
|------------------|--|
| <i>alignment</i> | The input sequence alignment (last entry must be <i>NULL</i> terminated) |
| <i>md_p</i>      | Model details that specify known nucleotides (Maybe <i>NULL</i> )        |

**Returns**

The most informative sequence for the alignment

## 16.71 Files and I/O

Functions to parse, write, and convert various file formats and to deal with file system related issues.

### 16.71.1 Detailed Description

Functions to parse, write, and convert various file formats and to deal with file system related issues.  
Collaboration diagram for Files and I/O:

## Modules

- [Nucleic Acid Sequences and Structures](#)

*Functions to read/write different file formats for nucleic acid sequences and secondary structures.*

- [Multiple Sequence Alignments](#)

*Functions to read/write multiple sequence alignments (MSA) in various file formats.*

- [Command Files](#)

*Functions to parse and interpret the content of [Command Files](#).*

## Files

- file [commands.h](#)

*Parse and apply different commands that alter the behavior of secondary structure prediction and evaluation.*

- file [ribo.h](#)

*Parse RiboSum Scoring Matrices for Covariance Scoring of Alignments.*

- file [file\\_formats.h](#)

*Read and write different file formats for RNA sequences, structures.*

- file [file\\_formats\\_msa.h](#)

*Functions dealing with file formats for Multiple Sequence Alignments (MSA)*

- file [utils.h](#)

*Several utilities for file handling.*

## Functions

- float \*\* [get\\_ribosum](#) (const char \*\*Alseq, int n\_seq, int length)

*Retrieve a RiboSum Scoring Matrix for a given Alignment.*

- float \*\* [readribosum](#) (char \*name)

*Read a RiboSum or other user-defined Scoring Matrix and Store into global Memory.*

- void [vrna\\_file\\_copy](#) (FILE \*from, FILE \*to)

*Inefficient 'cp'.*

- char \* [vrna\\_read\\_line](#) (FILE \*fp)

*Read a line of arbitrary length from a stream.*

- int [vrna\\_mkdir\\_p](#) (const char \*path)

*Recursively create a directory tree.*

- char \* [vrna\\_basename](#) (const char \*path)

*Extract the filename from a file path.*

- `char * vrna dirname (const char *path)`

*Extract the directory part of a file path.*

- `char * vrna_filename_sanitize (const char *name, const char *replacement)`

*Sanitize a file name.*

- `int vrna_file_exists (const char *filename)`

*Check if a file already exists in the file system.*

## 16.71.2 Function Documentation

### 16.71.2.1 vrna\_read\_line()

```
char * vrna_read_line (
    FILE * fp )
```

#include <ViennaRNA/io/utils.h>

Read a line of arbitrary length from a stream.

Returns a pointer to the resulting string. The necessary memory is allocated and should be released using `free()` when the string is no longer needed.

#### Parameters

|                 |  |
|-----------------|--|
| <code>fp</code> | A file pointer to the stream where the function should read from |
|-----------------|--|

#### Returns

A pointer to the resulting string

### 16.71.2.2 vrna\_filename\_sanitize()

```
char * vrna_filename_sanitize (
    const char * name,
    const char * replacement )
```

#include <ViennaRNA/io/utils.h>

Sanitize a file name.

Returns a new file name where all invalid characters are substituted by a replacement character. If no replacement character is supplied, invalid characters are simply removed from the filename. File names may also never exceed a length of 255 characters. Longer file names will undergo a 'smart' truncation process, where the filenames' suffix, i.e. everything after the last dot '.', is attempted to be kept intact. Hence, only the filename part before the suffix is reduced in such a way that the total filename complies to the length restriction of 255 characters. If no suffix is present or the suffix itself already exceeds the maximum length, the filename is simply truncated from the back of the string.

For now we consider the following characters invalid:

- backslash '\'
- slash '/'
- question mark '?'
- percent sign "%"
- asterisk '\*'
- colon ':'
- pipe symbol '|'

- double quote ""
- triangular brackets '<' and '>'

Furthermore, the (resulting) file name must not be a reserved file name, such as:

- ..
- ...

#### Note

This function allocates a new block of memory for the sanitized string. It also may return (a) NULL if the input is pointing to NULL, or (b) an empty string if the input only consists of invalid characters which are simply removed!

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <i>name</i>        | The input file name                |
| <i>replacement</i> | The replacement character, or NULL |

#### Returns

The sanitized file name, or NULL

### 16.71.2.3 vrna\_file\_exists()

```
int vrna_file_exists (
    const char * filename )
#include <ViennaRNA/io/utils.h>
Check if a file already exists in the file system.
```

#### Parameters

|                 |   |
|-----------------|---|
| <i>filename</i> | The name of (path to) the file to check for existence |
|-----------------|---|

#### Returns

0 if it doesn't exists, 1 otherwise

## 16.72 Nucleic Acid Sequences and Structures

Functions to read/write different file formats for nucleic acid sequences and secondary structures.

### 16.72.1 Detailed Description

Functions to read/write different file formats for nucleic acid sequences and secondary structures.  
Collaboration diagram for Nucleic Acid Sequences and Structures:

#### Files

- file [file\\_formats.h](#)  
*Read and write different file formats for RNA sequences, structures.*

#### Macros

- #define [VRNA\\_OPTION\\_MULTILINE](#) 32U

- Tell a function that an input is assumed to span several lines.
  - `#define VRNA_CONSTRAINT_MULTILINE 32U`  
*parse multiline constraint*
- ## Functions
- `void vrna_file_helixlist (const char *seq, const char *db, float energy, FILE *file)`  
*Print a secondary structure as helix list.*
  - `void vrna_file_connect (const char *seq, const char *db, float energy, const char *identifier, FILE *file)`  
*Print a secondary structure as connect table.*
  - `void vrna_file_bpseq (const char *seq, const char *db, FILE *file)`  
*Print a secondary structure in bpseq format.*
  - `void vrna_file_json (const char *seq, const char *db, double energy, const char *identifier, FILE *file)`  
*Print a secondary structure in jsonformat.*
  - `unsigned int vrna_file_fasta_read_record (char **header, char **sequence, char ***rest, FILE *file, unsigned int options)`
  - `char * vrna_extract_record_rest_structure (const char **lines, unsigned int length, unsigned int option)`  
*Extract a dot-bracket structure string from (multiline)character array.*
  - `int vrna_file_SHAPE_read (const char *file_name, int length, double default_value, char *sequence, double *values)`  
*Read data from a given SHAPE reactivity input file.*
  - `void vrna_extract_record_rest_constraint (char **cstruc, const char **lines, unsigned int option)`  
*Extract a hard constraint encoded as pseudo dot-bracket string.*
  - `unsigned int read_record (char **header, char **sequence, char ***rest, unsigned int options)`  
*Get a data record from stdin.*

### 16.72.2 Macro Definition Documentation

#### 16.72.2.1 VRNA\_OPTION\_MULTILINE

```
#define VRNA_OPTION_MULTILINE 32U
#include <ViennaRNA/io/file_formats.h>
Tell a function that an input is assumed to span several lines.
If used as input-option a function might also be returning this state telling that it has read data from multiple lines.
```

See also

[vrna\\_extract\\_record\\_rest\\_structure\(\)](#), [vrna\\_file\\_fasta\\_read\\_record\(\)](#)

#### 16.72.2.2 VRNA\_CONSTRAINT\_MULTILINE

```
#define VRNA_CONSTRAINT_MULTILINE 32U
#include <ViennaRNA/io/file_formats.h>
parse multiline constraint
```

**Deprecated** see [vrna\\_extract\\_record\\_rest\\_structure\(\)](#)

### 16.72.3 Function Documentation

### 16.72.3.1 vrna\_file\_helixlist()

```
void vrna_file_helixlist (
    const char * seq,
    const char * db,
    float energy,
    FILE * file )
#include <ViennaRNA/io/file_formats.h>
Print a secondary structure as helix list.
```

#### Parameters

|               |   |
|---------------|---|
| <i>seq</i>    | The RNA sequence  |
| <i>db</i>     | The structure in dot-bracket format   |
| <i>energy</i> | Free energy of the structure in kcal/mol  |
| <i>file</i>   | The file handle used to print to (print defaults to 'stdout' if(file == NULL) ) |

### 16.72.3.2 vrna\_file\_connect()

```
void vrna_file_connect (
    const char * seq,
    const char * db,
    float energy,
    const char * identifier,
    FILE * file )
#include <ViennaRNA/io/file_formats.h>
Print a secondary structure as connect table.
Connect table file format looks like this:
```

```
* 300 ENERGY = 7.0 example
* 1 G      0      2      22      1
* 2 G      1      3      21      2
*
```

where the headerline is followed by 6 columns with:

1. Base number: index n
2. Base (A, C, G, T, U, X)
3. Index n-1 (0 if first nucleotide)
4. Index n+1 (0 if last nucleotide)
5. Number of the base to which n is paired. No pairing is indicated by 0 (zero).
6. Natural numbering.

#### Parameters

|                   |   |
|-------------------|---|
| <i>seq</i>        | The RNA sequence  |
| <i>db</i>         | The structure in dot-bracket format   |
| <i>energy</i>     | The free energy of the structure  |
| <i>identifier</i> | An optional identifier for the sequence   |
| <i>file</i>       | The file handle used to print to (print defaults to 'stdout' if(file == NULL) ) |

### 16.72.3.3 vrna\_file\_bpseq()

```
void vrna_file_bpseq (
    const char * seq,
    const char * db,
    FILE * file )
#include <ViennaRNA/io/file_formats.h>
Print a secondary structure in bpseq format.
```

#### Parameters

|             |   |
|-------------|---|
| <i>seq</i>  | The RNA sequence  |
| <i>db</i>   | The structure in dot-bracket format   |
| <i>file</i> | The file handle used to print to (print defaults to 'stdout' if(file == NULL) ) |

### 16.72.3.4 vrna\_file\_json()

```
void vrna_file_json (
    const char * seq,
    const char * db,
    double energy,
    const char * identifier,
    FILE * file )
#include <ViennaRNA/io/file_formats.h>
Print a secondary structure in jsonformat.
```

#### Parameters

|                   |   |
|-------------------|---|
| <i>seq</i>        | The RNA sequence  |
| <i>db</i>         | The structure in dot-bracket format   |
| <i>energy</i>     | The free energy   |
| <i>identifier</i> | An identifier for the sequence  |
| <i>file</i>       | The file handle used to print to (print defaults to 'stdout' if(file == NULL) ) |

### 16.72.3.5 vrna\_file\_fasta\_read\_record()

```
unsigned int vrna_file_fasta_read_record (
    char ** header,
    char ** sequence,
    char *** rest,
    FILE * file,
    unsigned int options )
#include <ViennaRNA/io/file_formats.h>
```

@brief Get a (fasta) data set from a file or stdin

This function may be used to obtain complete datasets from a filehandle or stdin. A dataset is always defined to contain at least a sequence. If data starts with a fasta header, i.e. a line like  
 @verbatim >some header info

then `vrna_file_fasta_read_record()` will assume that the sequence that follows the header may span over several lines. To disable this behavior and to assign a single line to the argument 'sequence' one can pass `VRNA_INPUT_NO_SPAN` in the 'options' argument. If no fasta header is read in the beginning of a data block, a sequence must not span over multiple lines!

Unless the options `VRNA_INPUT_NOSKIP_COMMENTS` or `VRNA_INPUT_NOSKIP_BLANK_LINES` are passed, a sequence may be interrupted by lines starting with a comment character or empty lines.

A sequence is regarded as completely read if it was either assumed to not span over multiple lines, a secondary structure or structure constraint follows the sequence on the next line, or a new header marks the beginning of a new sequence...

All lines following the sequence (this includes comments) that do not initiate a new dataset according to the above definition are available through the line-array 'rest'. Here one can usually find the structure constraint or other information belonging to the current dataset. Filling of 'rest' may be prevented by passing `VRNA_INPUT_NO_REST` to the options argument.

#### Note

This function will exit any program with an error message if no sequence could be read!

This function is NOT threadsafe! It uses a global variable to store information about the next data block.

The main purpose of this function is to be able to easily parse blocks of data in the header of a loop where all calculations for the appropriate data is done inside the loop. The loop may be then left on certain return values, e.g.:

```
@endverbatim
char *id, *seq, **rest;
int i;
id = seq = NULL;
rest = NULL;
while(!(vrna_file_fasta_read_record(&id, &seq, &rest, NULL, 0) & (VRNA_INPUT_ERROR | VRNA_INPUT_QUIT))){
    if(id)
        printf("%s\n", id);
    printf("%s\n", seq);
    if(rest)
        for(i=0;rest[i];i++)
            printf("%s\n", rest[i]);
        free(rest[i]);
    }
    free(rest);
    free(seq);
    free(id);
}
```

In the example above, the while loop will be terminated when `vrna_file_fasta_read_record()` returns either an error, EOF, or a user initiated quit request.

As long as data is read from stdin (we are passing NULL as the file pointer), the id is printed if it is available for the current block of data. The sequence will be printed in any case and if some more lines belong to the current block of data each line will be printed as well.

#### Note

Do not forget to free the memory occupied by header, sequence and rest!

#### Parameters

|                       |  |
|-----------------------|--|
| <code>header</code>   | A pointer which will be set such that it points to the header of the record                          |
| <code>sequence</code> | A pointer which will be set such that it points to the sequence of the record                        |
| <code>rest</code>     | A pointer which will be set such that it points to an array of lines which also belong to the record |
| <code>file</code>     | A file handle to read from (if NULL, this function reads from stdin)                                 |
| <code>options</code>  | Some options which may be passed to alter the behavior of the function, use 0 for no options         |

#### Returns

A flag with information about what the function actually did read

#### 16.72.3.6 `vrna_extract_record_rest_structure()`

```
char * vrna_extract_record_rest_structure (
    const char ** lines,
```

```

    unsigned int length,
    unsigned int option )
#include <ViennaRNA/io/file_formats.h>

```

Extract a dot-bracket structure string from (multiline)character array.

This function extracts a dot-bracket structure string from the 'rest' array as returned by [vrna\\_file\\_fasta\\_read\\_record\(\)](#) and returns it. All occurrences of comments within the 'lines' array will be skipped as long as they do not break the structure string. If no structure could be read, this function returns NULL.

#### Precondition

The argument 'lines' has to be a 2-dimensional character array as obtained by [vrna\\_file\\_fasta\\_read\\_record\(\)](#)

#### See also

[vrna\\_file\\_fasta\\_read\\_record\(\)](#)

#### Parameters

|               |   |
|---------------|---|
| <i>lines</i>  | The (multiline) character array to be parsed  |
| <i>length</i> | The assumed length of the dot-bracket string (passing a value < 1 results in no length limit) |
| <i>option</i> | Some options which may be passed to alter the behavior of the function, use 0 for no options  |

#### Returns

The dot-bracket string read from lines or NULL

#### 16.72.3.7 [vrna\\_file\\_SHAPE\\_read\(\)](#)

```

int vrna_file_SHAPE_read (
    const char * file_name,
    int length,
    double default_value,
    char * sequence,
    double * values )
#include <ViennaRNA/io/file_formats.h>

```

Read data from a given SHAPE reactivity input file.

This function parses the informations from a given file and stores the result in the preallocated string sequence and the double array values.

#### Parameters

|                      |   |
|----------------------|---|
| <i>file_name</i>     | Path to the constraints file  |
| <i>length</i>        | Length of the sequence (file entries exceeding this limit will cause an error)            |
| <i>default_value</i> | Value for missing indices   |
| <i>sequence</i>      | Pointer to an array used for storing the sequence obtained from the SHAPE reactivity file |
| <i>values</i>        | Pointer to an array used for storing the values obtained from the SHAPE reactivity file   |

#### 16.72.3.8 [vrna\\_extract\\_record\\_rest\\_constraint\(\)](#)

```

void vrna_extract_record_rest_constraint (
    char ** cstruc,
    const char ** lines,
    unsigned int option )

```

```
#include <ViennaRNA/io/file_formats.h>
Extract a hard constraint encoded as pseudo dot-bracket string.
```

**Deprecated** Use [vrna\\_extract\\_record\\_rest\\_structure\(\)](#) instead!

#### Precondition

The argument 'lines' has to be a 2-dimensional character array as obtained by [vrna\\_file\\_fasta\\_read\\_record\(\)](#)

#### See also

[vrna\\_file\\_fasta\\_read\\_record\(\)](#), [VRNA\\_CONSTRAINT\\_DB\\_PIPE](#), [VRNA\\_CONSTRAINT\\_DB\\_DOT](#), [VRNA\\_CONSTRAINT\\_DB\\_BRACK](#), [VRNA\\_CONSTRAINT\\_DB\\_ANG\\_BRACK](#), [VRNA\\_CONSTRAINT\\_DB\\_RND\\_BRACK](#)

#### Parameters

|               |  |
|---------------|--|
| <i>cstruc</i> | A pointer to a character array that is used as pseudo dot-bracket output           |
| <i>lines</i>  | A 2-dimensional character array with the extension lines from the FASTA input      |
| <i>option</i> | The option flags that define the behavior and recognition pattern of this function |

#### 16.72.3.9 `read_record()`

```
unsigned int read_record (
    char ** header,
    char ** sequence,
    char *** rest,
    unsigned int options )
#include <ViennaRNA/io/file_formats.h>
Get a data record from stdin.
```

**Deprecated** This function is deprecated! Use [vrna\\_file\\_fasta\\_read\\_record\(\)](#) as a replacement.

## 16.73 Multiple Sequence Alignments

Functions to read/write multiple sequence alignments (MSA) in various file formats.

### 16.73.1 Detailed Description

Functions to read/write multiple sequence alignments (MSA) in various file formats.

Collaboration diagram for Multiple Sequence Alignments:

#### Files

- file [file\\_formats\\_msa.h](#)  
*Functions dealing with file formats for Multiple Sequence Alignments (MSA)*

#### Macros

- #define [VRNA\\_FILE\\_FORMAT\\_MSA\\_CLUSTAL](#) 1U  
*Option flag indicating ClustalW formatted files.*
- #define [VRNA\\_FILE\\_FORMAT\\_MSA\\_STOCKHOLM](#) 2U  
*Option flag indicating Stockholm 1.0 formatted files.*
- #define [VRNA\\_FILE\\_FORMAT\\_MSA\\_FASTA](#) 4U  
*Option flag indicating FASTA (Pearson) formatted files.*

- `#define VRNA_FILE_FORMAT_MSA_MAF 8U`  
*Option flag indicating MAF formatted files.*
- `#define VRNA_FILE_FORMAT_MSA_MIS 16U`  
*Option flag indicating most informative sequence (MIS) output.*
- `#define VRNA_FILE_FORMAT_MSA_DEFAULT`  
*Option flag indicating the set of default file formats.*
- `#define VRNA_FILE_FORMAT_MSA_NOCHECK 4096U`  
*Option flag to disable validation of the alignment.*
- `#define VRNA_FILE_FORMAT_MSA_UNKNOWN 8192U`  
*Return flag of `vrna_file_msa_detect_format()` to indicate unknown or malformed alignment.*
- `#define VRNA_FILE_FORMAT_MSA_APPEND 16384U`  
*Option flag indicating to append data to a multiple sequence alignment file rather than overwriting it.*
- `#define VRNA_FILE_FORMAT_MSA QUIET 32768U`  
*Option flag to suppress unnecessary spam messages on `stderr`*
- `#define VRNA_FILE_FORMAT_MSA_SILENT 65536U`  
*Option flag to completely silence any warnings on `stderr`*

## Functions

- `int vrna_file_msa_read (const char *filename, char ***names, char ***aln, char **id, char **structure, unsigned int options)`  
*Read a multiple sequence alignment from file.*
- `int vrna_file_msa_read_record (FILE *fp, char ***names, char ***aln, char **id, char **structure, unsigned int options)`  
*Read a multiple sequence alignment from file handle.*
- `unsigned int vrna_file_msa_detect_format (const char *filename, unsigned int options)`  
*Detect the format of a multiple sequence alignment file.*
- `int vrna_file_msa_write (const char *filename, const char **names, const char **aln, const char *id, const char *structure, const char *source, unsigned int options)`  
*Write multiple sequence alignment file.*

### 16.73.2 Macro Definition Documentation

#### 16.73.2.1 VRNA\_FILE\_FORMAT\_MSA\_CLUSTAL

```
#define VRNA_FILE_FORMAT_MSA_CLUSTAL 1U
#include <ViennaRNA/io/file_formats_msa.h>
Option flag indicating ClustalW formatted files.
```

See also

[vrna\\_file\\_msa\\_read\(\)](#), [vrna\\_file\\_msa\\_read\\_record\(\)](#), [vrna\\_file\\_msa\\_detect\\_format\(\)](#)

#### 16.73.2.2 VRNA\_FILE\_FORMAT\_MSA\_STOCKHOLM

```
#define VRNA_FILE_FORMAT_MSA_STOCKHOLM 2U
#include <ViennaRNA/io/file_formats_msa.h>
Option flag indicating Stockholm 1.0 formatted files.
```

See also

[vrna\\_file\\_msa\\_read\(\)](#), [vrna\\_file\\_msa\\_read\\_record\(\)](#), [vrna\\_file\\_msa\\_detect\\_format\(\)](#)

### 16.73.2.3 VRNA\_FILE\_FORMAT\_MSA\_FASTA

```
#define VRNA_FILE_FORMAT_MSA_FASTA 4U
#include <ViennaRNA/io/file_formats_msa.h>
Option flag indicating FASTA (Pearson) formatted files.
```

See also

[vrna\\_file\\_msa\\_read\(\)](#), [vrna\\_file\\_msa\\_read\\_record\(\)](#), [vrna\\_file\\_msa\\_detect\\_format\(\)](#)

### 16.73.2.4 VRNA\_FILE\_FORMAT\_MSA\_MAF

```
#define VRNA_FILE_FORMAT_MSA_MAF 8U
#include <ViennaRNA/io/file_formats_msa.h>
Option flag indicating MAF formatted files.
```

See also

[vrna\\_file\\_msa\\_read\(\)](#), [vrna\\_file\\_msa\\_read\\_record\(\)](#), [vrna\\_file\\_msa\\_detect\\_format\(\)](#)

### 16.73.2.5 VRNA\_FILE\_FORMAT\_MSA\_MIS

```
#define VRNA_FILE_FORMAT_MSA_MIS 16U
#include <ViennaRNA/io/file_formats_msa.h>
Option flag indicating most informative sequence (MIS) output.
```

The default reference sequence output for an alignment is simply a consensus sequence. This flag allows to write the most informative sequence (MIS) instead.

See also

[vrna\\_file\\_msa\\_write\(\)](#)

### 16.73.2.6 VRNA\_FILE\_FORMAT\_MSA\_DEFAULT

```
#define VRNA_FILE_FORMAT_MSA_DEFAULT
#include <ViennaRNA/io/file_formats_msa.h>
```

**Value:**

```
( \
VRNA_FILE_FORMAT_MSA_CLUSTAL \
| VRNA_FILE_FORMAT_MSA_STOCKHOLM \
| VRNA_FILE_FORMAT_MSA_FASTA \
| VRNA_FILE_FORMAT_MSA_MAF \
)
```

Option flag indicating the set of default file formats.

See also

[vrna\\_file\\_msa\\_read\(\)](#), [vrna\\_file\\_msa\\_read\\_record\(\)](#), [vrna\\_file\\_msa\\_detect\\_format\(\)](#)

### 16.73.2.7 VRNA\_FILE\_FORMAT\_MSA\_NOCHECK

```
#define VRNA_FILE_FORMAT_MSA_NOCHECK 4096U
#include <ViennaRNA/io/file_formats_msa.h>
Option flag to disable validation of the alignment.
```

See also

[vrna\\_file\\_msa\\_read\(\)](#), [vrna\\_file\\_msa\\_read\\_record\(\)](#)

### 16.73.2.8 VRNA\_FILE\_FORMAT\_MSA\_UNKNOWN

```
#define VRNA_FILE_FORMAT_MSA_UNKNOWN 8192U
#include <ViennaRNA/io/file_formats_msa.h>
Return flag of vrna_file_msa_detect_format() to indicate unknown or malformed alignment.
```

See also

[vrna\\_file\\_msa\\_detect\\_format\(\)](#)

### 16.73.2.9 VRNA\_FILE\_FORMAT\_MSA\_APPEND

```
#define VRNA_FILE_FORMAT_MSA_APPEND 16384U
#include <ViennaRNA/io/file_formats_msa.h>
Option flag indicating to append data to a multiple sequence alignment file rather than overwriting it.
```

See also

[vrna\\_file\\_msa\\_write\(\)](#)

### 16.73.2.10 VRNA\_FILE\_FORMAT\_MSA\_QUIET

```
#define VRNA_FILE_FORMAT_MSA_QUIET 32768U
#include <ViennaRNA/io/file_formats_msa.h>
Option flag to suppress unnecessary spam messages on stderr
```

See also

[vrna\\_file\\_msa\\_read\(\)](#), [vrna\\_file\\_msa\\_read\\_record\(\)](#)

### 16.73.2.11 VRNA\_FILE\_FORMAT\_MSA\_SILENT

```
#define VRNA_FILE_FORMAT_MSA_SILENT 65536U
#include <ViennaRNA/io/file_formats_msa.h>
Option flag to completely silence any warnings on stderr
```

See also

[vrna\\_file\\_msa\\_read\(\)](#), [vrna\\_file\\_msa\\_read\\_record\(\)](#)

## 16.73.3 Function Documentation

### 16.73.3.1 vrna\_file\_msa\_read()

```
int vrna_file_msa_read (
    const char * filename,
    char *** names,
    char *** aln,
    char ** id,
    char ** structure,
    unsigned int options )
#include <ViennaRNA/io/file_formats_msa.h>
```

Read a multiple sequence alignment from file.

This function reads the (first) multiple sequence alignment from an input file. The read alignment is split into the sequence id/name part and the actual sequence information and stored in memory as arrays of ids/names and sequences. If the alignment file format allows for additional information, such as an ID of the entire alignment or

consensus structure information, this data is retrieved as well and made available. The `options` parameter allows to specify the set of alignment file formats that should be used to retrieve the data. If 0 is passed as option, the list of alignment file formats defaults to `VRNA_FILE_FORMAT_MSA_DEFAULT`.

Currently, the list of parsable multiple sequence alignment file formats consists of:

- ClustalW format
- Stockholm 1.0 format
- FASTA (Pearson) format
- MAF format

#### Note

After successfully reading an alignment, this function performs a validation of the data that includes uniqueness of the sequence identifiers, and equal sequence lengths. This check can be deactivated by passing `VRNA_FILE_FORMAT_MSA_NOCHECK` in the `options` parameter.

It is the users responsibility to free any memory occupied by the output arguments `names`, `aln`, `id`, and `structure` after calling this function. The function automatically sets the latter two arguments to `NULL` in case no corresponding data could be retrieved from the input alignment.

#### See also

`vrna_file_msa_read_record()`, `VRNA_FILE_FORMAT_MSA_CLUSTAL`, `VRNA_FILE_FORMAT_MSA_STOCKHOLM`, `VRNA_FILE_FORMAT_MSA_FASTA`, `VRNA_FILE_FORMAT_MSA_MAF`, `VRNA_FILE_FORMAT_MSA_DEFAULT`, `VRNA_FILE_FORMAT_MSA_NOCHECK`

#### Parameters

|                        |   |
|------------------------|---|
| <code>filename</code>  | The name of input file that contains the alignment  |
| <code>names</code>     | An address to the pointer where sequence identifiers should be written to                                       |
| <code>aln</code>       | An address to the pointer where aligned sequences should be written to  |
| <code>id</code>        | An address to the pointer where the alignment ID should be written to (Maybe <code>NULL</code> )                |
| <code>structure</code> | An address to the pointer where consensus structure information should be written to (Maybe <code>NULL</code> ) |
| <code>options</code>   | Options to manipulate the behavior of this function   |

#### Returns

The number of sequences in the alignment, or -1 if no alignment record could be found

**SWIG Wrapper Notes** In the target scripting language, only the first and last argument, `filename` and `options`, are passed to the corresponding function. The other arguments, which serve as output in the C-library, are available as additional return values. Hence, a function call in python may look like this:

```
num_seq, names, aln, id, structure = RNA.file_msa_read("msa.stk", RNA.FILE_FORMAT_MSA_STOCKHOLM)
```

After successfully reading the first record, the variable `num_seq` contains the number of sequences in the alignment (the actual return value of the C-function), while the variables `names`, `aln`, `id`, and `structure` are lists of the sequence names and aligned sequences, as well as strings holding the alignment ID and the structure as stated in the `SS_cons` line, respectively. Note, the last two return values may be empty strings in case the alignment does not provide the required data.

This function exists as an overloaded version where the `options` parameter may be omitted! In that case, the `options` parameter defaults to `VRNA_FILE_FORMAT_MSA_STOCKHOLM`.

#### 16.73.3.2 `vrna_file_msa_read_record()`

```
int vrna_file_msa_read_record (
    FILE * fp,
```

```

    char *** names,
    char *** aln,
    char ** id,
    char ** structure,
    unsigned int options )
#include <ViennaRNA/io/file_formats_msa.h>

```

Read a multiple sequence alignment from file handle.

Similar to [vrna\\_file\\_msa\\_read\(\)](#), this function reads a multiple sequence alignment from an input file handle. Since using a file handle, this function is not limited to the first alignment record, but allows for looping over all alignments within the input.

The read alignment is split into the sequence id/name part and the actual sequence information and stored in memory as arrays of ids/names and sequences. If the alignment file format allows for additional information, such as an ID of the entire alignment or consensus structure information, this data is retrieved as well and made available. The `options` parameter allows to specify the alignment file format used to retrieve the data. A single format must be specified here, see [vrna\\_file\\_msa\\_detect\\_format\(\)](#) for helping to determine the correct MSA file format.

Currently, the list of parsable multiple sequence alignment file formats consists of:

- [ClustalW format](#)
- [Stockholm 1.0 format](#)
- [FASTA \(Pearson\) format](#)
- [MAF format](#)

#### Note

After successfully reading an alignment, this function performs a validation of the data that includes uniqueness of the sequence identifiers, and equal sequence lengths. This check can be deactivated by passing [VRNA\\_FILE\\_FORMAT\\_MSA\\_NOCHECK](#) in the `options` parameter.

It is the users responsibility to free any memory occupied by the output arguments `names`, `aln`, `id`, and `structure` after calling this function. The function automatically sets the latter two arguments to `NULL` in case no corresponding data could be retrieved from the input alignment.

#### See also

[vrna\\_file\\_msa\\_read\(\)](#), [vrna\\_file\\_msa\\_detect\\_format\(\)](#), [VRNA\\_FILE\\_FORMAT\\_MSA\\_CLUSTAL](#), [VRNA\\_FILE\\_FORMAT\\_MSA\\_FASTA](#), [VRNA\\_FILE\\_FORMAT\\_MSA\\_MAF](#), [VRNA\\_FILE\\_FORMAT\\_MSA\\_DEFAULT](#), [VRNA\\_FILE\\_FORMAT\\_MSA\\_NOCHECK](#)

#### Parameters

|                        |   |
|------------------------|---|
| <code>fp</code>        | The file pointer the data will be retrieved from  |
| <code>names</code>     | An address to the pointer where sequence identifiers should be written to                                       |
| <code>aln</code>       | An address to the pointer where aligned sequences should be written to  |
| <code>id</code>        | An address to the pointer where the alignment ID should be written to (Maybe <code>NULL</code> )                |
| <code>structure</code> | An address to the pointer where consensus structure information should be written to (Maybe <code>NULL</code> ) |
| <code>options</code>   | Options to manipulate the behavior of this function   |

#### Returns

The number of sequences in the alignment, or -1 if no alignment record could be found

**SWIG Wrapper Notes** In the target scripting language, only the first and last argument, `fp` and `options`, are passed to the corresponding function. The other arguments, which serve as output in the C-library, are available as additional return values. Hence, a function call in python may look like this:

```
f = open('msa.stk', 'r')
```

```
num_seq, names, aln, id, structure = RNA.file_msa_read_record(f, RNA.FILE_FORMAT_MSA_STOCKHOLM)
f.close()
```

After successfully reading the first record, the variable `num_seq` contains the number of sequences in the alignment (the actual return value of the C-function), while the variables `names`, `aln`, `id`, and `structure` are lists of the sequence names and aligned sequences, as well as strings holding the alignment ID and the structure as stated in the `SS_cons` line, respectively. Note, the last two return values may be empty strings in case the alignment does not provide the required data.

This function exists as an overloaded version where the `options` parameter may be omitted! In that case, the `options` parameter defaults to [VRNA\\_FILE\\_FORMAT\\_MSA\\_STOCKHOLM](#).

### 16.73.3.3 `vrna_file_msa_detect_format()`

```
unsigned int vrna_file_msa_detect_format (
    const char * filename,
    unsigned int options )
#include <ViennaRNA/io/file_formats_msa.h>
```

Detect the format of a multiple sequence alignment file.

This function attempts to determine the format of a file that supposedly contains a multiple sequence alignment (MSA). This is useful in cases where a MSA file contains more than a single record and therefore `vrna_file_msa_read()` can not be applied, since it only retrieves the first. Here, one can try to guess the correct file format using this function and then loop over the file, record by record using one of the low-level record retrieval functions for the corresponding MSA file format.

#### Note

This function parses the entire first record within the specified file. As a result, it returns [VRNA\\_FILE\\_FORMAT\\_MSA\\_UNKNOWN](#) not only if it can't detect the file's format, but also in cases where the file doesn't contain sequences!

#### See also

[vrna\\_file\\_msa\\_read\(\)](#), [vrna\\_file\\_stockholm\\_read\\_record\(\)](#), [vrna\\_file\\_clustal\\_read\\_record\(\)](#), [vrna\\_file\\_fasta\\_read\\_record\(\)](#)

#### Parameters

|                       |   |
|-----------------------|---|
| <code>filename</code> | The name of input file that contains the alignment  |
| <code>options</code>  | Options to manipulate the behavior of this function |

#### Returns

The MSA file format, or [VRNA\\_FILE\\_FORMAT\\_MSA\\_UNKNOWN](#)

**SWIG Wrapper Notes** This function exists as an overloaded version where the `options` parameter may be omitted! In that case, the `options` parameter defaults to [VRNA\\_FILE\\_FORMAT\\_MSA\\_DEFAULT](#).

### 16.73.3.4 `vrna_file_msa_write()`

```
int vrna_file_msa_write (
    const char * filename,
    const char ** names,
    const char ** aln,
    const char * id,
    const char * structure,
    const char * source,
    unsigned int options )
#include <ViennaRNA/io/file_formats_msa.h>
```

Write multiple sequence alignment file.

**Note**

Currently, we only support [Stockholm 1.0 format](#) output

**See also**

[VRNA\\_FILE\\_FORMAT\\_MSA\\_STOCKHOLM](#), [VRNA\\_FILE\\_FORMAT\\_MSA\\_APPEND](#), [VRNA\\_FILE\\_FORMAT\\_MSA\\_MIS](#)

**Parameters**

|                  |   |
|------------------|---|
| <i>filename</i>  | The output filename                                 |
| <i>names</i>     | The array of sequence names / identifies            |
| <i>aln</i>       | The array of aligned sequences                      |
| <i>id</i>        | An optional ID for the alignment                    |
| <i>structure</i> | An optional consensus structure                     |
| <i>source</i>    | A string describing the source of the alignment     |
| <i>options</i>   | Options to manipulate the behavior of this function |

**Returns**

Non-null upon successfully writing the alignment to file

**SWIG Wrapper Notes** In the target scripting language, this function exists as a set of overloaded versions, where the last four parameters may be omitted. If the *options* parameter is missing the options default to ([VRNA\\_FILE\\_FORMAT\\_MSA\\_STOCKHOLM](#) | [VRNA\\_FILE\\_FORMAT\\_MSA\\_APPEND](#)).

## 16.74 Command Files

Functions to parse and interpret the content of [Command Files](#).

### 16.74.1 Detailed Description

Functions to parse and interpret the content of [Command Files](#).

Collaboration diagram for Command Files:

## Files

- file [commands.h](#)

*Parse and apply different commands that alter the behavior of secondary structure prediction and evaluation.*

## Macros

- `#define VRNA_CMD_PARSE_HC 1U`  
*Command parse/apply flag indicating hard constraints.*
- `#define VRNA_CMD_PARSE_SC 2U`  
*Command parse/apply flag indicating soft constraints.*
- `#define VRNA_CMD_PARSE_UD 4U`  
*Command parse/apply flag indicating unstructured domains.*
- `#define VRNA_CMD_PARSE_SD 8U`  
*Command parse/apply flag indicating structured domains.*
- `#define VRNA_CMD_PARSE_DEFAULTS`  
*Command parse/apply flag indicating default set of commands.*

## Typedefs

- `typedef struct vrna_command_s *vrna_cmd_t`  
*A data structure that contains commands.*

## Functions

- `vrna_cmd_t vrna_file_commands_read (const char *filename, unsigned int options)`  
*Extract a list of commands from a command file.*
- `int vrna_file_commands_apply (vrna_fold_compound_t *vc, const char *filename, unsigned int options)`  
*Apply a list of commands from a command file.*
- `int vrna_commands_apply (vrna_fold_compound_t *vc, vrna_cmd_t commands, unsigned int options)`  
*Apply a list of commands to a vrna\_fold\_compound\_t.*
- `void vrna_commands_free (vrna_cmd_t commands)`  
*Free memory occupied by a list of commands.*

### 16.74.2 Macro Definition Documentation

#### 16.74.2.1 VRNA\_CMD\_PARSE\_HC

```
#define VRNA_CMD_PARSE_HC 1U
#include <ViennaRNA/commands.h>
Command parse/apply flag indicating hard constraints.
```

See also

[vrna\\_cmd\\_t](#), [vrna\\_file\\_commands\\_read\(\)](#), [vrna\\_file\\_commands\\_apply\(\)](#), [vrna\\_commands\\_apply\(\)](#)

#### 16.74.2.2 VRNA\_CMD\_PARSE\_SC

```
#define VRNA_CMD_PARSE_SC 2U
#include <ViennaRNA/commands.h>
Command parse/apply flag indicating soft constraints.
```

See also

[vrna\\_cmd\\_t](#), [vrna\\_file\\_commands\\_read\(\)](#), [vrna\\_file\\_commands\\_apply\(\)](#), [vrna\\_commands\\_apply\(\)](#)

#### 16.74.2.3 VRNA\_CMD\_PARSE\_UD

```
#define VRNA_CMD_PARSE_UD 4U
#include <ViennaRNA/commands.h>
Command parse/apply flag indicating unstructured domains.
```

See also

[vrna\\_cmd\\_t](#), [vrna\\_file\\_commands\\_read\(\)](#), [vrna\\_file\\_commands\\_apply\(\)](#), [vrna\\_commands\\_apply\(\)](#)

#### 16.74.2.4 VRNA\_CMD\_PARSE\_SD

```
#define VRNA_CMD_PARSE_SD 8U
#include <ViennaRNA/commands.h>
Command parse/apply flag indicating structured domains.
```

See also

[vrna\\_cmd\\_t](#), [vrna\\_file\\_commands\\_read\(\)](#), [vrna\\_file\\_commands\\_apply\(\)](#), [vrna\\_commands\\_apply\(\)](#)

#### 16.74.2.5 VRNA\_CMD\_PARSE\_DEFAULTS

```
#define VRNA_CMD_PARSE_DEFAULTS
#include <ViennaRNA/commands.h>
```

**Value:**

```
(VRNA_CMD_PARSE_HC \
| VRNA_CMD_PARSE_SC \
| VRNA_CMD_PARSE_UD \
| VRNA_CMD_PARSE_SD \
)
```

Command parse/apply flag indicating default set of commands.

See also

[vrna\\_cmd\\_t](#), [vrna\\_file\\_commands\\_read\(\)](#), [vrna\\_file\\_commands\\_apply\(\)](#), [vrna\\_commands\\_apply\(\)](#)

### 16.74.3 Function Documentation

#### 16.74.3.1 vrna\_file\_commands\_read()

```
vrna_cmd_t vrna_file_commands_read (
    const char * filename,
    unsigned int options )
#include <ViennaRNA/commands.h>
```

Extract a list of commands from a command file.

Read a list of commands specified in the input file and return them as list of abstract commands

See also

[vrna\\_commands\\_apply\(\)](#), [vrna\\_file\\_commands\\_apply\(\)](#), [vrna\\_commands\\_free\(\)](#)

#### Parameters

|                 |  |
|-----------------|--|
| <i>filename</i> | The filename   |
| <i>options</i>  | Options to limit the type of commands read from the file |

#### Returns

A list of abstract commands

#### 16.74.3.2 vrna\_file\_commands\_apply()

```
int vrna_file_commands_apply (
    vrna_fold_compound_t * vc,
    const char * filename,
    unsigned int options )
#include <ViennaRNA/commands.h>
```

Apply a list of commands from a command file.

This function is a shortcut to directly parse a commands file and apply all successfully parsed commands to a `vrna_fold_compound_t` data structure. It is the same as:

```
int r;
struct vrna_command_s *cmds;
cmds = vrna_file_commands_read(filename, options);
r = vrna_commands_apply(vc, cmds, options);
vrna_commands_free(cmds);
return r;
```

#### Parameters

|                       |   |
|-----------------------|---|
| <code>vc</code>       | The <code>vrna_fold_compound_t</code> the command list will be applied to |
| <code>filename</code> | The filename  |
| <code>options</code>  | Options to limit the type of commands read from the file                  |

#### Returns

The number of commands successfully applied

**SWIG Wrapper Notes** This function is attached as method `file_commands_apply()` to objects of type `fold_<compound>`

#### 16.74.3.3 `vrna_commands_apply()`

```
int vrna_commands_apply (
    vrna_fold_compound_t * vc,
    vrna_cmd_t commands,
    unsigned int options )
#include <ViennaRNA/commands.h>
Apply a list of commands to a vrna_fold_compound_t.
```

#### Parameters

|                       |   |
|-----------------------|---|
| <code>vc</code>       | The <code>vrna_fold_compound_t</code> the command list will be applied to |
| <code>commands</code> | The commands to apply   |
| <code>options</code>  | Options to limit the type of commands read from the file                  |

#### Returns

The number of commands successfully applied

#### 16.74.3.4 `vrna_commands_free()`

```
void vrna_commands_free (
    vrna_cmd_t commands )
#include <ViennaRNA/commands.h>
Free memory occupied by a list of commands.
Release memory occupied by a list of commands
```

#### Parameters

|                       |                                 |
|-----------------------|---------------------------------|
| <code>commands</code> | A pointer to a list of commands |
|-----------------------|---------------------------------|

## 16.75 Plotting

Functions for Creating Secondary Structure Plots, Dot-Plots, and More.

### 16.75.1 Detailed Description

Functions for Creating Secondary Structure Plots, Dot-Plots, and More.

Collaboration diagram for Plotting:

## Modules

- [Layouts and Coordinates](#)

*Functions to compute coordinate layouts for secondary structure plots.*

- [Annotation](#)

*Functions to generate annotations for Secondary Structure Plots, Dot-Plots, and Others.*

- [Alignment Plots](#)

*Functions to generate Alignment plots with annotated consensus structure.*

- [Deprecated Interface for Plotting Utilities](#)

## Files

- file [alignments.h](#)

*Various functions for plotting Sequence / Structure Alignments.*

- file [layouts.h](#)

*Secondary structure plot layout algorithms.*

- file [probabilities.h](#)

*Various functions for plotting RNA secondary structures, dot-plots and other visualizations.*

- file [structures.h](#)

*Various functions for plotting RNA secondary structures.*

- file [utils.h](#)

*Various utilities to assist in plotting secondary structures and consensus structures.*

- file [RNApuzzler.h](#)

*Implementation of the RNApuzzler RNA secondary structure layout algorithm [24].*

- file [RNAturtle.h](#)

*Implementation of the RNAturtle RNA secondary structure layout algorithm [24].*

## Data Structures

- struct [vrna\\_dotplot\\_auxdata\\_t](#)

## Functions

- int [PS\\_dot\\_plot\\_list](#) (char \*seq, char \*filename, [vrna\\_ep\\_t](#) \*pl, [vrna\\_ep\\_t](#) \*mf, char \*comment)

*Produce a postscript dot-plot from two pair lists.*

- int [PS\\_dot\\_plot](#) (char \*string, char \*file)

*Produce postscript dot-plot.*

- int [vrna\\_file\\_PS\\_rnplot](#) (const char \*seq, const char \*structure, const char \*file, [vrna\\_md\\_t](#) \*md\_p)

*Produce a secondary structure graph in PostScript and write it to 'filename'.*

- int [vrna\\_file\\_PS\\_rnplot\\_a](#) (const char \*seq, const char \*structure, const char \*file, const char \*pre, const char \*post, [vrna\\_md\\_t](#) \*md\_p)

*Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename'.*

- int [gmlRNA](#) (char \*string, char \*structure, char \*ssfile, char option)

*Produce a secondary structure graph in Graph Meta Language (gml) and write it to a file.*

- int [ssv\\_rna\\_plot](#) (char \*string, char \*structure, char \*ssfile)

- int [svg\\_rna\\_plot](#) (char \*string, char \*structure, char \*ssfile)
 

*Produce a secondary structure graph in SStructView format.*
- int [xrna\\_plot](#) (char \*string, char \*structure, char \*ssfile)
 

*Produce a secondary structure plot in SVG format and write it to a file.*
- int [PS\\_rna\\_plot](#) (char \*string, char \*structure, char \*file)
 

*Produce a secondary structure plot for further editing in XRNA.*
- int [PS\\_rna\\_plot\\_a](#) (char \*string, char \*structure, char \*file, char \*pre, char \*post)
 

*Produce a secondary structure graph in PostScript and write it to 'filename'.*
- int [PS\\_rna\\_plot\\_a\\_gquad](#) (char \*string, char \*structure, char \*ssfile, char \*pre, char \*post)
 

*Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename'.*
- int [PS\\_rna\\_plot\\_a\\_gquad](#) (char \*string, char \*structure, char \*ssfile, char \*pre, char \*post)
 

*Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename' (detect and draw g-quadruplexes)*

## 16.75.2 Data Structure Documentation

### 16.75.2.1 struct vrna\_dotplot\_auxdata\_t

Collaboration diagram for vrna\_dotplot\_auxdata\_t:

## 16.75.3 Function Documentation

### 16.75.3.1 PS\_dot\_plot\_list()

```
int PS_dot_plot_list (
    char * seq,
    char * filename,
    vrna_ep_t * pl,
    vrna_ep_t * mf,
    char * comment )
#include <ViennaRNA/plotting/probabilities.h>
```

Produce a postscript dot-plot from two pair lists.

This function reads two plist structures (e.g. base pair probabilities and a secondary structure) as produced by [assign plist from pr\(\)](#) and [assign plist from db\(\)](#) and produces a postscript "dot plot" that is written to 'filename'. Using base pair probabilities in the first and mfe structure in the second plist, the resulting "dot plot" represents each base pairing probability by a square of corresponding area in a upper triangle matrix. The lower part of the matrix contains the minimum free energy structure.

See also

[assign plist from pr\(\)](#), [assign plist from db\(\)](#)

#### Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <i>seq</i>      | The RNA sequence                     |
| <i>filename</i> | A filename for the postscript output |
| <i>pl</i>       | The base pair probability pairlist   |
| <i>mf</i>       | The mfe secondary structure pairlist |
| <i>comment</i>  | A comment                            |

#### Returns

1 if postscript was successfully written, 0 otherwise

### 16.75.3.2 PS\_dot\_plot()

```
int PS_dot_plot (
    char * string,
    char * file )
#include <ViennaRNA/plotting/probabilities.h>
```

Produce postscript dot-plot.

Wrapper to PS\_dot\_plot\_list

Reads base pair probabilities produced by [pf\\_fold\(\)](#) from the global array [pr](#) and the pair list [base\\_pair](#) produced by [fold\(\)](#) and produces a postscript "dot plot" that is written to 'filename'. The "dot plot" represents each base pairing probability by a square of corresponding area in a upper triangle matrix. The lower part of the matrix contains the minimum free energy

**Note**

DO NOT USE THIS FUNCTION ANYMORE SINCE IT IS NOT THREADSAFE

**Deprecated** This function is deprecated and will be removed soon! Use [PS\\_dot\\_plot\\_list\(\)](#) instead!

### 16.75.3.3 vrna\_file\_PS\_rnplot()

```
int vrna_file_PS_rnplot (
    const char * seq,
    const char * structure,
    const char * file,
    vrna_md_t * md_p )
#include <ViennaRNA/plotting/structures.h>
```

Produce a secondary structure graph in PostScript and write it to 'filename'.

Note that this function has changed from previous versions and now expects the structure to be plotted in dot-bracket notation as an argument. It does not make use of the global [base\\_pair](#) array anymore.

**Parameters**

|                  |  |
|------------------|--|
| <i>seq</i>       | The RNA sequence   |
| <i>structure</i> | The secondary structure in dot-bracket notation  |
| <i>file</i>      | The filename of the postscript output  |
| <i>md_p</i>      | Model parameters used to generate a commandline option string in the output (Maybe NULL) |

**Returns**

1 on success, 0 otherwise

### 16.75.3.4 vrna\_file\_PS\_rnplot\_a()

```
int vrna_file_PS_rnplot_a (
    const char * seq,
    const char * structure,
    const char * file,
    const char * pre,
    const char * post,
    vrna_md_t * md_p )
#include <ViennaRNA/plotting/structures.h>
```

Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename'. Same as [vrna\\_file\\_PS\\_rnplot\(\)](#) but adds extra PostScript macros for various annotations (see generated PS code). The 'pre' and 'post' variables contain PostScript code that is verbatim copied in the resulting PS file just before and

after the structure plot. If both arguments ('pre' and 'post') are NULL, no additional macros will be printed into the PostScript.

**Parameters**

|                  |  |
|------------------|--|
| <i>seq</i>       | The RNA sequence   |
| <i>structure</i> | The secondary structure in dot-bracket notation  |
| <i>file</i>      | The filename of the postscript output  |
| <i>pre</i>       | PostScript code to appear before the secondary structure plot                            |
| <i>post</i>      | PostScript code to appear after the secondary structure plot                             |
| <i>md_p</i>      | Model parameters used to generate a commandline option string in the output (Maybe NULL) |

**Returns**

1 on success, 0 otherwise

**16.75.3.5 gmlRNA()**

```
int gmlRNA (
    char * string,
    char * structure,
    char * ssfile,
    char option )
#include <ViennaRNA/plotting/structures.h>
```

Produce a secondary structure graph in Graph Meta Language (gml) and write it to a file.

If 'option' is an uppercase letter the RNA sequence is used to label nodes, if 'option' equals 'X' or 'x' the resulting file will contain coordinates for an initial layout of the graph.

**Parameters**

|                  |   |
|------------------|---|
| <i>string</i>    | The RNA sequence                                |
| <i>structure</i> | The secondary structure in dot-bracket notation |
| <i>ssfile</i>    | The filename of the gml output                  |
| <i>option</i>    | The option flag                                 |

**Returns**

1 on success, 0 otherwise

**16.75.3.6 ssv\_rna\_plot()**

```
int ssv_rna_plot (
    char * string,
    char * structure,
    char * ssfile )
```

```
#include <ViennaRNA/plotting/structures.h>
```

Produce a secondary structure graph in SStructView format.

Write coord file for SStructView

**Parameters**

|                  |   |
|------------------|---|
| <i>string</i>    | The RNA sequence                                |
| <i>structure</i> | The secondary structure in dot-bracket notation |
| <i>ssfile</i>    | The filename of the ssv output                  |

**Returns**

1 on success, 0 otherwise

**16.75.3.7 `svg_rna_plot()`**

```
int svg_rna_plot (
    char * string,
    char * structure,
    char * ssfile )
#include <ViennaRNA/plotting/structures.h>
Produce a secondary structure plot in SVG format and write it to a file.
```

**Parameters**

|                  |   |
|------------------|---|
| <i>string</i>    | The RNA sequence                                |
| <i>structure</i> | The secondary structure in dot-bracket notation |
| <i>ssfile</i>    | The filename of the svg output                  |

**Returns**

1 on success, 0 otherwise

**16.75.3.8 `xrna_plot()`**

```
int xrna_plot (
    char * string,
    char * structure,
    char * ssfile )
#include <ViennaRNA/plotting/structures.h>
Produce a secondary structure plot for further editing in XRNA.
```

**Parameters**

|                  |   |
|------------------|---|
| <i>string</i>    | The RNA sequence                                |
| <i>structure</i> | The secondary structure in dot-bracket notation |
| <i>ssfile</i>    | The filename of the xrna output                 |

**Returns**

1 on success, 0 otherwise

**16.75.3.9 `PS_rna_plot()`**

```
int PS_rna_plot (
    char * string,
    char * structure,
    char * file )
#include <ViennaRNA/plotting/structures.h>
Produce a secondary structure graph in PostScript and write it to 'filename'.
```

**Deprecated** Use [vrna\\_file\\_PS\\_rnaplot\(\)](#) instead!

### 16.75.3.10 PS\_rna\_plot\_a()

```
int PS_rna_plot_a (
    char * string,
    char * structure,
    char * file,
    char * pre,
    char * post )
#include <ViennaRNA/plotting/structures.h>
```

Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename'.

**Deprecated** Use [vrna\\_file\\_PS\\_rnplot\\_a\(\)](#) instead!

### 16.75.3.11 PS\_rna\_plot\_a\_gquad()

```
int PS_rna_plot_a_gquad (
    char * string,
    char * structure,
    char * ssfile,
    char * pre,
    char * post )
#include <ViennaRNA/plotting/structures.h>
```

Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename' (detect and draw g-quadruplexes)

**Deprecated** Use [vrna\\_file\\_PS\\_rnplot\\_a\(\)](#) instead!

## 16.76 Layouts and Coordinates

Functions to compute coordinate layouts for secondary structure plots.

### 16.76.1 Detailed Description

Functions to compute coordinate layouts for secondary structure plots.

Collaboration diagram for Layouts and Coordinates:

### Data Structures

- struct [vrna\\_plot\\_layout\\_s](#)
- struct [vrna\\_plot\\_options\\_puzzler\\_t](#)

*Options data structure for RNApuzzler algorithm implementation.* [More...](#)

### Macros

- #define VRNA\_PLOT\_TYPE\_SIMPLE 0  
*Definition of Plot type simple*
- #define VRNA\_PLOT\_TYPE\_NAVIEW 1  
*Definition of Plot type Naview*
- #define VRNA\_PLOT\_TYPE\_CIRCULAR 2  
*Definition of Plot type Circular*
- #define VRNA\_PLOT\_TYPE\_TURTLE 3  
*Definition of Plot type Turtle [24].*
- #define VRNA\_PLOT\_TYPE\_PUZZLER 4  
*Definition of Plot type RNApuzzler [24].*

## Typedefs

- `typedef struct vrna_plot_layout_s vrna_plot_layout_t`  
*RNA secondary structure figure layout.*

## Functions

- `vrna_plot_layout_t * vrna_plot_layout (const char *structure, unsigned int plot_type)`  
*Create a layout (coordinates, etc.) for a secondary structure plot.*
- `vrna_plot_layout_t * vrna_plot_layout_simple (const char *structure)`  
*Create a layout (coordinates, etc.) for a simple secondary structure plot.*
- `vrna_plot_layout_t * vrna_plot_layout_circular (const char *structure)`  
*Create a layout (coordinates, etc.) for a circular secondary structure plot.*
- `vrna_plot_layout_t * vrna_plot_layout_turtle (const char *structure)`  
*Create a layout (coordinates, etc.) for a secondary structure plot using the Turtle Algorithm [24].*
- `vrna_plot_layout_t * vrna_plot_layout_puzzler (const char *structure, vrna_plot_options_puzzler_t *options)`  
*Create a layout (coordinates, etc.) for a secondary structure plot using the RNApuzzler Algorithm [24].*
- `void vrna_plot_layout_free (vrna_plot_layout_t *layout)`  
*Free memory occupied by a figure layout data structure.*
- `int vrna_plot_coords (const char *structure, float **x, float **y, int plot_type)`  
*Compute nucleotide coordinates for secondary structure plot.*
- `int vrna_plot_coords_pt (const short *pt, float **x, float **y, int plot_type)`  
*Compute nucleotide coordinates for secondary structure plot.*
- `int vrna_plot_coords_simple (const char *structure, float **x, float **y)`  
*Compute nucleotide coordinates for secondary structure plot the Simple way*
- `int vrna_plot_coords_simple_pt (const short *pt, float **x, float **y)`  
*Compute nucleotide coordinates for secondary structure plot the Simple way*
- `int vrna_plot_coords_circular (const char *structure, float **x, float **y)`  
*Compute coordinates of nucleotides mapped in equal distancies onto a unit circle.*
- `int vrna_plot_coords_circular_pt (const short *pt, float **x, float **y)`  
*Compute nucleotide coordinates for a Circular Plot*
- `int vrna_plot_coords_puzzler (const char *structure, float **x, float **y, double **arc_coords, vrna_plot_options_puzzler_t *options)`  
*Compute nucleotide coordinates for secondary structure plot using the RNApuzzler algorithm [24].*
- `int vrna_plot_coords_puzzler_pt (short const *const pair_table, float **x, float **y, double **arc_coords, vrna_plot_options_puzzler_t *puzzler)`  
*Compute nucleotide coordinates for secondary structure plot using the RNApuzzler algorithm [24].*
- `vrna_plot_options_puzzler_t * vrna_plot_options_puzzler (void)`  
*Create an RNApuzzler options data structure.*
- `void vrna_plot_options_puzzler_free (vrna_plot_options_puzzler_t *options)`  
*Free memory occupied by an RNApuzzler options data structure.*
- `int vrna_plot_coords_turtle (const char *structure, float **x, float **y, double **arc_coords)`  
*Compute nucleotide coordinates for secondary structure plot using the RNAturtle algorithm [24].*
- `int vrna_plot_coords_turtle_pt (short const *const pair_table, float **x, float **y, double **arc_coords)`  
*Compute nucleotide coordinates for secondary structure plot using the RNAturtle algorithm [24].*

## 16.76.2 Data Structure Documentation

### 16.76.2.1 struct vrna\_plot\_layout\_s

### 16.76.2.2 struct vrna\_plot\_options\_puzzler\_t

Options data structure for RNApuzzler algorithm implementation.

### 16.76.3 Macro Definition Documentation

#### 16.76.3.1 VRNA\_PLOT\_TYPE\_SIMPLE

```
#define VRNA_PLOT_TYPE_SIMPLE 0
#include <ViennaRNA/plotting/layouts.h>
```

Definition of Plot type *simple*

This is the plot type definition for several RNA structure plotting functions telling them to use **Simple** plotting algorithm

See also

[rna\\_plot\\_type](#), [vrna\\_file\\_PS\\_rnplot\\_a\(\)](#), [vrna\\_file\\_PS\\_rnplot\(\)](#), [svg\\_rna\\_plot\(\)](#), [gmlRNA\(\)](#), [ssv\\_rna\\_plot\(\)](#), [xrna\\_plot\(\)](#)

#### 16.76.3.2 VRNA\_PLOT\_TYPE\_NAVIEW

```
#define VRNA_PLOT_TYPE_NAVIEW 1
#include <ViennaRNA/plotting/layouts.h>
```

Definition of Plot type *Naview*

This is the plot type definition for several RNA structure plotting functions telling them to use **Naview** plotting algorithm [5].

See also

[rna\\_plot\\_type](#), [vrna\\_file\\_PS\\_rnplot\\_a\(\)](#), [vrna\\_file\\_PS\\_rnplot\(\)](#), [svg\\_rna\\_plot\(\)](#), [gmlRNA\(\)](#), [ssv\\_rna\\_plot\(\)](#), [xrna\\_plot\(\)](#)

#### 16.76.3.3 VRNA\_PLOT\_TYPE\_CIRCULAR

```
#define VRNA_PLOT_TYPE_CIRCULAR 2
#include <ViennaRNA/plotting/layouts.h>
```

Definition of Plot type *Circular*

This is the plot type definition for several RNA structure plotting functions telling them to produce a **Circular plot**

See also

[rna\\_plot\\_type](#), [vrna\\_file\\_PS\\_rnplot\\_a\(\)](#), [vrna\\_file\\_PS\\_rnplot\(\)](#), [svg\\_rna\\_plot\(\)](#), [gmlRNA\(\)](#), [ssv\\_rna\\_plot\(\)](#), [xrna\\_plot\(\)](#)

### 16.76.4 Typedef Documentation

#### 16.76.4.1 vrna\_plot\_layout\_t

```
typedef struct vrna_plot_layout_s vrna_plot_layout_t
#include <ViennaRNA/plotting/layouts.h>
```

RNA secondary structure figure layout.

See also

[vrna\\_plot\\_layout\(\)](#), [vrna\\_plot\\_layout\\_free\(\)](#), [vrna\\_plot\\_layout\\_simple\(\)](#), [vrna\\_plot\\_layout\\_circular\(\)](#), [vrna\\_plot\\_layout\\_naview\(\)](#), [vrna\\_plot\\_layout\\_turtle\(\)](#), [vrna\\_plot\\_layout\\_puzzler\(\)](#)

### 16.76.5 Function Documentation

### 16.76.5.1 vrna\_plot\_layout()

```
vrna_plot_layout_t * vrna_plot_layout (
    const char * structure,
    unsigned int plot_type )
#include <ViennaRNA/plotting/layouts.h>
```

Create a layout (coordinates, etc.) for a secondary structure plot.

This function can be used to create a secondary structure nucleotide layout that is then further processed by an actual plotting function. The layout algorithm can be specified using the `plot_type` parameter, and the following algorithms are currently supported:

- `VRNA_PLOT_TYPE_SIMPLE`
- `VRNA_PLOT_TYPE_NAVIEW`
- `VRNA_PLOT_TYPE_CIRCULAR`
- `VRNA_PLOT_TYPE_TURTLE`
- `VRNA_PLOT_TYPE_PUZZLER`

Passing an unsupported selection leads to the default algorithm `VRNA_PLOT_TYPE_NAVIEW`

#### Note

If only X-Y coordinates of the corresponding structure layout are required, consider using `vrna_plot_coords()` instead!

#### See also

`vrna_plot_layout_free()`, `vrna_plot_layout_simple()`, `vrna_plot_layout_naview()`, `vrna_plot_layout_circular()`, `vrna_plot_layout_turtle()`, `vrna_plot_layout_puzzler()`, `vrna_plot_coords()`, `vrna_file_PS_rnplot_layout()`

#### Parameters

|                        |   |
|------------------------|---|
| <code>structure</code> | The secondary structure in dot-bracket notation |
| <code>plot_type</code> | The layout algorithm to be used                 |

#### Returns

The layout data structure for the provided secondary structure

### 16.76.5.2 vrna\_plot\_layout\_simple()

```
vrna_plot_layout_t * vrna_plot_layout_simple (
    const char * structure )
#include <ViennaRNA/plotting/layouts.h>
```

Create a layout (coordinates, etc.) for a *simple* secondary structure plot.

This function basically is a wrapper to `vrna_plot_layout()` that passes the `plot_type` `VRNA_PLOT_TYPE_SIMPLE`.

#### Note

If only X-Y coordinates of the corresponding structure layout are required, consider using `vrna_plot_coords_simple()` instead!

#### See also

`vrna_plot_layout_free()`, `vrna_plot_layout()`, `vrna_plot_layout_naview()`, `vrna_plot_layout_circular()`, `vrna_plot_layout_turtle()`, `vrna_plot_layout_puzzler()`, `vrna_plot_coords_simple()`, `vrna_file_PS_rnplot_layout()`

**Parameters**

|                        |   |
|------------------------|---|
| <code>structure</code> | The secondary structure in dot-bracket notation |
|------------------------|---|

**Returns**

The layout data structure for the provided secondary structure

### 16.76.5.3 `vrna_plot_layout_circular()`

```
vrna_plot_layout_t * vrna_plot_layout_circular (
    const char * structure )
```

```
#include <ViennaRNA/plotting/layouts.h>
```

Create a layout (coordinates, etc.) for a *circular* secondary structure plot.

This function basically is a wrapper to `vrna_plot_layout()` that passes the `plot_type` `VRNA_PLOT_TYPE_CIRCULAR`.

**Note**

If only X-Y coordinates of the corresponding structure layout are required, consider using `vrna_plot_coords_circular()` instead!

**See also**

`vrna_plot_layout_free()`, `vrna_plot_layout()`, `vrna_plot_layout_naview()`, `vrna_plot_layout_simple()`, `vrna_plot_layout_turtle()`,  
`vrna_plot_layout_puzzler()`, `vrna_plot_coords_circular()`, `vrna_file_PS_rnапlot_layout()`

**Parameters**

|                        |   |
|------------------------|---|
| <code>structure</code> | The secondary structure in dot-bracket notation |
|------------------------|---|

**Returns**

The layout data structure for the provided secondary structure

### 16.76.5.4 `vrna_plot_layout_turtle()`

```
vrna_plot_layout_t * vrna_plot_layout_turtle (
    const char * structure )
```

```
#include <ViennaRNA/plotting/layouts.h>
```

Create a layout (coordinates, etc.) for a secondary structure plot using the *Turtle Algorithm* [24].

This function basically is a wrapper to `vrna_plot_layout()` that passes the `plot_type` `VRNA_PLOT_TYPE_TURTLE`.

**Note**

If only X-Y coordinates of the corresponding structure layout are required, consider using `vrna_plot_coords_turtle()` instead!

**See also**

`vrna_plot_layout_free()`, `vrna_plot_layout()`, `vrna_plot_layout_simple()`, `vrna_plot_layout_circular()`, `vrna_plot_layout_naview()`,  
`vrna_plot_layout_puzzler()`, `vrna_plot_coords_turtle()`, `vrna_file_PS_rnапlot_layout()`

**Parameters**

|                        |   |
|------------------------|---|
| <code>structure</code> | The secondary structure in dot-bracket notation |
|------------------------|---|

**Returns**

The layout data structure for the provided secondary structure

**16.76.5.5 vrna\_plot\_layout\_puzzler()**

```
vrna_plot_layout_t * vrna_plot_layout_puzzler (
    const char * structure,
    vrna_plot_options_puzzler_t * options )
#include <ViennaRNA/plotting/layouts.h>
```

Create a layout (coordinates, etc.) for a secondary structure plot using the *RNAPuzzler Algorithm* [24].

This function basically is a wrapper to [vrna\\_plot\\_layout\(\)](#) that passes the `plot_type` `VRNA_PLOT_TYPE_PUZZLER`.

**Note**

If only X-Y coordinates of the corresponding structure layout are required, consider using [vrna\\_plot\\_coords\\_puzzler\(\)](#) instead!

**See also**

[vrna\\_plot\\_layout\\_free\(\)](#), [vrna\\_plot\\_layout\(\)](#), [vrna\\_plot\\_layout\\_simple\(\)](#), [vrna\\_plot\\_layout\\_circular\(\)](#), [vrna\\_plot\\_layout\\_naview\(\)](#), [vrna\\_plot\\_layout\\_turtle\(\)](#), [vrna\\_plot\\_coords\\_puzzler\(\)](#), [vrna\\_file\\_PS\\_rnапlot\\_layout\(\)](#)

**Parameters**

|                        |   |
|------------------------|---|
| <code>structure</code> | The secondary structure in dot-bracket notation |
|------------------------|---|

**Returns**

The layout data structure for the provided secondary structure

**16.76.5.6 vrna\_plot\_layout\_free()**

```
void vrna_plot_layout_free (
    vrna_plot_layout_t * layout )
#include <ViennaRNA/plotting/layouts.h>
```

Free memory occupied by a figure layout data structure.

**See also**

[vrna\\_plot\\_layout\\_t](#), [vrna\\_plot\\_layout\(\)](#), [vrna\\_plot\\_layout\\_simple\(\)](#), [vrna\\_plot\\_layout\\_circular\(\)](#), [vrna\\_plot\\_layout\\_naview\(\)](#), [vrna\\_plot\\_layout\\_turtle\(\)](#), [vrna\\_plot\\_layout\\_puzzler\(\)](#), [vrna\\_file\\_PS\\_rnапlot\\_layout\(\)](#)

**Parameters**

|                     |                                   |
|---------------------|-----------------------------------|
| <code>layout</code> | The layout data structure to free |
|---------------------|-----------------------------------|

**16.76.5.7 vrna\_plot\_coords()**

```
int vrna_plot_coords (
    const char * structure,
    float ** x,
    float ** y,
    int plot_type )
```

```
#include <ViennaRNA/plotting/layouts.h>
Compute nucleotide coordinates for secondary structure plot.
```

This function takes a secondary structure and computes X-Y coordinates for each nucleotide that then can be used to create a structure plot. The parameter `plot_type` is used to select the underlying layout algorithm. Currently, the following selections are provided:

- `VRNA_PLOT_TYPE_SIMPLE`
- `VRNA_PLOT_TYPE_NAVIEW`
- `VRNA_PLOT_TYPE_CIRCULAR`
- `VRNA_PLOT_TYPE_TURTLE`
- `VRNA_PLOT_TYPE_PUZZLER`

Passing an unsupported selection leads to the default algorithm `VRNA_PLOT_TYPE_NAVIEW`

Here is a simple example how to use this function, assuming variable `structure` contains a valid dot-bracket string:

```
float *x, *y;
if (vrna_plot_coords(structure, &x, &y)) {
    printf("all fine");
} else {
    printf("some failure occurred!");
}
free(x);
free(y);
```

#### Note

On success, this function allocates memory for X and Y coordinates and assigns the pointers at addresses `x` and `y` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

#### See also

`vrna_plot_coords_pt()`, `vrna_plot_coords_simple()`, `vrna_plot_coords_naview()` `vrna_plot_coords_circular()`,  
`vrna_plot_coords_turtle()`, `vrna_plot_coords_puzzler()`

#### Parameters

|                      |                        |  |
|----------------------|------------------------|--|
|                      | <code>structure</code> | The secondary structure in dot-bracket notation  |
| <code>in, out</code> | <code>x</code>         | The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure) |
| <code>in, out</code> | <code>y</code>         | The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure) |
|                      | <code>plot_type</code> | The layout algorithm to be used  |

#### Returns

The length of the structure on success, 0 otherwise

#### 16.76.5.8 `vrna_plot_coords_pt()`

```
int vrna_plot_coords_pt (
    const short * pt,
    float ** x,
    float ** y,
    int plot_type )
#include <ViennaRNA/plotting/layouts.h>
```

Compute nucleotide coordinates for secondary structure plot.

Same as `vrna_plot_coords()` but takes a pair table with the structure information as input.

**Note**

On success, this function allocates memory for X and Y coordinates and assigns the pointers at addresses `x` and `y` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

**See also**

[vrna\\_plot\\_coords\(\)](#), [vrna\\_plot\\_coords\\_simple\\_pt\(\)](#), [vrna\\_plot\\_coords\\_naview\\_pt\(\)](#) [vrna\\_plot\\_coords\\_circular\\_pt\(\)](#),  
[vrna\\_plot\\_coords\\_turtle\\_pt\(\)](#), [vrna\\_plot\\_coords\\_puzzler\\_pt\(\)](#)

**Parameters**

|         |                  |  |
|---------|------------------|--|
|         | <i>pt</i>        | The pair table that holds the secondary structure  |
| in, out | <i>x</i>         | The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure) |
| in, out | <i>y</i>         | The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure) |
|         | <i>plot_type</i> | The layout algorithm to be used  |

**Returns**

The length of the structure on success, 0 otherwise

**16.76.5.9 vrna\_plot\_coords\_simple()**

```
int vrna_plot_coords_simple (
    const char * structure,
    float ** x,
    float ** y )
#include <ViennaRNA/plotting/layouts.h>
Compute nucleotide coordinates for secondary structure plot the Simple way
This function basically is a wrapper to vrna\_plot\_coords\(\) that passes the plot_type VRNA_PLOT_TYPE_SIMPLE.
Here is a simple example how to use this function, assuming variable structure contains a valid dot-bracket
string:
float *x, *y;
if (vrna\_plot\_coords\_simple(structure, &x, &y)) {
    printf("all fine");
} else {
    printf("some failure occurred!");
}
free(x);
free(y);
```

**Note**

On success, this function allocates memory for X and Y coordinates and assigns the pointers at addresses `x` and `y` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

**See also**

[vrna\\_plot\\_coords\(\)](#), [vrna\\_plot\\_coords\\_simple\\_pt\(\)](#), [vrna\\_plot\\_coords\\_circular\(\)](#), [vrna\\_plot\\_coords\\_naview\(\)](#),  
[vrna\\_plot\\_coords\\_turtle\(\)](#), [vrna\\_plot\\_coords\\_puzzler\(\)](#)

**Parameters**

|         |                  |  |
|---------|------------------|--|
|         | <i>structure</i> | The secondary structure in dot-bracket notation  |
| in, out | <i>x</i>         | The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure) |
| in, out | <i>y</i>         | The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure) |

**Returns**

The length of the structure on success, 0 otherwise

**16.76.5.10 vrna\_plot\_coords\_simple\_pt()**

```
int vrna_plot_coords_simple_pt (
    const short * pt,
    float ** x,
    float ** y )
#include <ViennaRNA/plotting/layouts.h>
```

Compute nucleotide coordinates for secondary structure plot the *Simple way*

Same as [vrna\\_plot\\_coords\\_simple\(\)](#) but takes a pair table with the structure information as input.

**Note**

On success, this function allocates memory for X and Y coordinates and assigns the pointers at addresses `x` and `y` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

**See also**

[vrna\\_plot\\_coords\\_pt\(\)](#), [vrna\\_plot\\_coords\\_simple\(\)](#), [vrna\\_plot\\_coords\\_circular\\_pt\(\)](#), [vrna\\_plot\\_coords\\_<naview\\_pt\(\)](#), [vrna\\_plot\\_coords\\_turtle\\_pt\(\)](#), [vrna\\_plot\\_coords\\_puzzler\\_pt\(\)](#)

**Parameters**

|         |           |  |
|---------|-----------|--|
|         | <i>pt</i> | The pair table that holds the secondary structure  |
| in, out | <i>x</i>  | The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure) |
| in, out | <i>y</i>  | The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure) |

**Returns**

The length of the structure on success, 0 otherwise

**16.76.5.11 vrna\_plot\_coords\_circular()**

```
int vrna_plot_coords_circular (
    const char * structure,
    float ** x,
    float ** y )
#include <ViennaRNA/plotting/layouts.h>
```

Compute coordinates of nucleotides mapped in equal distancies onto a unit circle.

This function basically is a wrapper to [vrna\\_plot\\_coords\(\)](#) that passes the `plot_type` [VRNA\\_PLOT\\_TYPE\\_CIRCULAR](#).

In order to draw nice arcs using quadratic bezier curves that connect base pairs one may calculate a second tangential point  $P^t$  in addition to the actual  $R^2$  coordinates. the simplest way to do so may be to compute a radius scaling factor  $rs$  in the interval  $[0, 1]$  that weights the proportion of base pair span to the actual length of the sequence. This scaling factor can then be used to calculate the coordinates for  $P^t$ , i.e.

$$P_x^t[i] = X[i] * rs$$

and

$$P_y^t[i] = Y[i] * rs$$

**Note**

On success, this function allocates memory for X and Y coordinates and assigns the pointers at addresses `x` and `y` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

**See also**

[vrna\\_plot\\_coords\(\)](#), [vrna\\_plot\\_coords\\_circular\\_pt\(\)](#), [vrna\\_plot\\_coords\\_simple\(\)](#), [vrna\\_plot\\_coords\\_naview\(\)](#), [vrna\\_plot\\_coords\\_turtle\(\)](#), [vrna\\_plot\\_coords\\_puzzler\(\)](#)

**Parameters**

|         |                        |  |
|---------|------------------------|--|
|         | <code>structure</code> | The secondary structure in dot-bracket notation  |
| in, out | <code>x</code>         | The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure) |
| in, out | <code>y</code>         | The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure) |

**Returns**

The length of the structure on success, 0 otherwise

**16.76.5.12 vrna\_plot\_coords\_circular\_pt()**

```
int vrna_plot_coords_circular_pt (
    const short * pt,
    float ** x,
    float ** y )
#include <ViennaRNA/plotting/layouts.h>
```

Compute nucleotide coordinates for a *Circular Plot*

Same as [vrna\\_plot\\_coords\\_circular\(\)](#) but takes a pair table with the structure information as input.

**Note**

On success, this function allocates memory for X and Y coordinates and assigns the pointers at addresses `x` and `y` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

**See also**

[vrna\\_plot\\_coords\\_pt\(\)](#), [vrna\\_plot\\_coords\\_circular\(\)](#), [vrna\\_plot\\_coords\\_simple\\_pt\(\)](#), [vrna\\_plot\\_coords\\_naview\\_pt\(\)](#), [vrna\\_plot\\_coords\\_turtle\\_pt\(\)](#), [vrna\\_plot\\_coords\\_puzzler\\_pt\(\)](#)

**Parameters**

|         |                 |  |
|---------|-----------------|--|
|         | <code>pt</code> | The pair table that holds the secondary structure  |
| in, out | <code>x</code>  | The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure) |
| in, out | <code>y</code>  | The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure) |

**Returns**

The length of the structure on success, 0 otherwise

**16.76.5.13 vrna\_plot\_coords\_puzzler()**

```
int vrna_plot_coords_puzzler (
```

```

    const char * structure,
    float ** x,
    float ** y,
    double ** arc_coords,
    vrna_plot_options_puzzler_t * options )
#include <ViennaRNA/plotting/RNApuzzler/RNApuzzler.h>
Compute nucleotide coordinates for secondary structure plot using the RNApuzzler algorithm [24].
This function basically is a wrapper to vrna_plot_coords() that passes the plot_type VRNA_PLOT_TYPE_PUZZLER.
Here is a simple example how to use this function, assuming variable structure contains a valid dot-bracket
string and using the default options (options = NULL):
float **x, **y;
double **arcs;
if (vrna_plot_coords_puzzler(structure, &x, &y, &arcs, NULL)) {
    printf("all fine");
} else {
    printf("some failure occurred!");
}
free(x);
free(y);
free(arcs);

```

**Note**

On success, this function allocates memory for X, Y and arc coordinates and assigns the pointers at addresses x, y and arc\_coords to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

**See also**

[vrna\\_plot\\_coords\(\)](#), [vrna\\_plot\\_coords\\_puzzler\\_pt\(\)](#), [vrna\\_plot\\_coords\\_circular\(\)](#), [vrna\\_plot\\_coords\\_simple\(\)](#),  
[vrna\\_plot\\_coords\\_turtle\(\)](#), [vrna\\_plot\\_coords\\_naview\(\)](#), [vrna\\_plot\\_options\\_puzzler\(\)](#)

**Parameters**

|        |                   |  |
|--------|-------------------|--|
|        | <i>structure</i>  | The secondary structure in dot-bracket notation  |
| in,out | <i>x</i>          | The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure)               |
| in,out | <i>y</i>          | The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure)               |
| in,out | <i>arc_coords</i> | The address of a pointer that will hold arc coordinates (pointer will point to memory, or NULL on failure) |
|        | <i>options</i>    | The options for the RNApuzzler algorithm (or NULL)   |

**Returns**

The length of the structure on success, 0 otherwise

**16.76.5.14 vrna\_plot\_coords\_puzzler\_pt()**

```

int vrna_plot_coords_puzzler_pt (
    short const *const pair_table,
    float ** x,
    float ** y,
    double ** arc_coords,
    vrna_plot_options_puzzler_t * puzzler )
#include <ViennaRNA/plotting/RNApuzzler/RNApuzzler.h>
Compute nucleotide coordinates for secondary structure plot using the RNApuzzler algorithm [24].
Same as vrna\_plot\_coords\_puzzler\(\) but takes a pair table with the structure information as input.

```

**Note**

On success, this function allocates memory for X, Y and arc coordinates and assigns the pointers at addresses `x`, `y` and `arc_coords` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

**See also**

[vrna\\_plot\\_coords\\_pt\(\)](#), [vrna\\_plot\\_coords\\_puzzler\(\)](#), [vrna\\_plot\\_coords\\_circular\\_pt\(\)](#), [vrna\\_plot\\_coords\\_simple\\_pt\(\)](#),  
[vrna\\_plot\\_coords\\_turtle\\_pt\(\)](#), [vrna\\_plot\\_coords\\_naview\\_pt\(\)](#)

**Parameters**

|         |                   |  |
|---------|-------------------|--|
|         | <i>pt</i>         | The pair table that holds the secondary structure  |
| in, out | <i>x</i>          | The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure)               |
| in, out | <i>y</i>          | The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure)               |
| in, out | <i>arc_coords</i> | The address of a pointer that will hold arc coordinates (pointer will point to memory, or NULL on failure) |
|         | <i>options</i>    | The options for the RNApuzzler algorithm (or NULL)   |

**Returns**

The length of the structure on success, 0 otherwise

**16.76.5.15 vrna\_plot\_options\_puzzler()**

```
vrna_plot_options_puzzler_t * vrna_plot_options_puzzler (
    void )
#include <ViennaRNA/plotting/RNApuzzler/RNApuzzler.h>
Create an RNApuzzler options data structure.
```

**See also**

[vrna\\_plot\\_options\\_puzzler\\_free\(\)](#), [vrna\\_plot\\_coords\\_puzzler\(\)](#), [vrna\\_plot\\_coords\\_puzzler\\_pt\(\)](#), [vrna\\_plot\\_layout\\_puzzler\(\)](#)

**Returns**

An RNApuzzler options data structure with default settings

**16.76.5.16 vrna\_plot\_options\_puzzler\_free()**

```
void vrna_plot_options_puzzler_free (
    vrna_plot_options_puzzler_t * options )
#include <ViennaRNA/plotting/RNApuzzler/RNApuzzler.h>
Free memory occupied by an RNApuzzler options data structure.
```

**See also**

[vrna\\_plot\\_options\\_puzzler\(\)](#), [vrna\\_plot\\_coords\\_puzzler\(\)](#), [vrna\\_plot\\_coords\\_puzzler\\_pt\(\)](#), [vrna\\_plot\\_layout\\_puzzler\(\)](#)

**Parameters**

|                |   |
|----------------|---|
| <i>options</i> | A pointer to the options data structure to free |
|----------------|---|

### 16.76.5.17 vrna\_plot\_coords\_turtle()

```
int vrna_plot_coords_turtle (
    const char * structure,
    float ** x,
    float ** y,
    double ** arc_coords )
#include <ViennaRNA/plotting/RNApuzzler/RNAturtle.h>
Compute nucleotide coordinates for secondary structure plot using the RNAturtle algorithm [24].
This function basically is a wrapper to vrna\_plot\_coords\(\) that passes the plot_type VRNA_PLOT_TYPE_TURTLE.
Here is a simple example how to use this function, assuming variable structure contains a valid dot-bracket
string:
float *x, *y;
double *arcs;
if (vrna\_plot\_coords\_turtle(structure, &x, &y, &arcs)) {
    printf("all fine");
} else {
    printf("some failure occurred!");
}
free(x);
free(y);
free(arcs);
```

#### Note

On success, this function allocates memory for X, Y and arc coordinates and assigns the pointers at addresses `x`, `y` and `arc_coords` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

#### See also

[vrna\\_plot\\_coords\(\)](#), [vrna\\_plot\\_coords\\_turtle\\_pt\(\)](#), [vrna\\_plot\\_coords\\_circular\(\)](#), [vrna\\_plot\\_coords\\_simple\(\)](#),  
[vrna\\_plot\\_coords\\_naview\(\)](#), [vrna\\_plot\\_coords\\_puzzler\(\)](#)

#### Parameters

|                      |                         |  |
|----------------------|-------------------------|--|
|                      | <code>structure</code>  | The secondary structure in dot-bracket notation  |
| <code>in, out</code> | <code>x</code>          | The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure)               |
| <code>in, out</code> | <code>y</code>          | The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure)               |
| <code>in, out</code> | <code>arc_coords</code> | The address of a pointer that will hold arc coordinates (pointer will point to memory, or NULL on failure) |

#### Returns

The length of the structure on success, 0 otherwise

### 16.76.5.18 vrna\_plot\_coords\_turtle\_pt()

```
int vrna_plot_coords_turtle_pt (
    short const *const pair_table,
    float ** x,
    float ** y,
    double ** arc_coords )
#include <ViennaRNA/plotting/RNApuzzler/RNAturtle.h>
Compute nucleotide coordinates for secondary structure plot using the RNAturtle algorithm [24].
Same as vrna\_plot\_coords\_turtle\(\) but takes a pair table with the structure information as input.
```

**Note**

On success, this function allocates memory for X, Y and arc coordinates and assigns the pointers at addresses `x`, `y` and `arc_coords` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

**See also**

[vrna\\_plot\\_coords\\_pt\(\)](#), [vrna\\_plot\\_coords\\_turtle\(\)](#), [vrna\\_plot\\_coords\\_circular\\_pt\(\)](#), [vrna\\_plot\\_coords\\_simple\\_pt\(\)](#),  
[vrna\\_plot\\_coords\\_puzzler\\_pt\(\)](#), [vrna\\_plot\\_coords\\_naview\\_pt\(\)](#)

**Parameters**

|        |                         |  |
|--------|-------------------------|--|
|        | <code>pt</code>         | The pair table that holds the secondary structure  |
| in,out | <code>x</code>          | The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure)               |
| in,out | <code>y</code>          | The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure)               |
| in,out | <code>arc_coords</code> | The address of a pointer that will hold arc coordinates (pointer will point to memory, or NULL on failure) |

**Returns**

The length of the structure on success, 0 otherwise

## 16.77 Annotation

Functions to generate annotations for Secondary Structure Plots, Dot-Plots, and Others.

### 16.77.1 Detailed Description

Functions to generate annotations for Secondary Structure Plots, Dot-Plots, and Others.

Collaboration diagram for Annotation:

## Functions

- `char ** vrna_annotation_covar_db (const char **alignment, const char *structure, vrna_md_t *md_p)`  
*Produce covariance annotation for an alignment given a secondary structure.*
- `vrna_cpair_t * vrna_annotation_covar_pairs (const char **alignment, vrna_ep_t *pl, vrna_ep_t *mfel, double threshold, vrna_md_t *md)`  
*Produce covariance annotation for an alignment given a set of base pairs.*

## 16.78 Alignment Plots

Functions to generate Alignment plots with annotated consensus structure.

### 16.78.1 Detailed Description

Functions to generate Alignment plots with annotated consensus structure.

Collaboration diagram for Alignment Plots:

## Functions

- `int vrna_file_PS_aln (const char *filename, const char **seqs, const char **names, const char *structure, unsigned int columns)`

*Create an annotated PostScript alignment plot.*

- int [vrna\\_file\\_PS\\_aln\\_slice](#) (const char \*filename, const char \*\*seqs, const char \*\*names, const char \*structure, unsigned int start, unsigned int end, int offset, unsigned int columns)

*Create an annotated PostScript alignment plot.*

## 16.78.2 Function Documentation

### 16.78.2.1 [vrna\\_file\\_PS\\_aln\(\)](#)

```
int vrna_file_PS_aln (
    const char * filename,
    const char ** seqs,
    const char ** names,
    const char * structure,
    unsigned int columns )
#include <ViennaRNA/plotting	alignments.h>
```

Create an annotated PostScript alignment plot.

See also

[vrna\\_file\\_PS\\_aln\\_slice\(\)](#)

#### Parameters

|                  |   |
|------------------|---|
| <i>filename</i>  | The output file name  |
| <i>seqs</i>      | The aligned sequences   |
| <i>names</i>     | The names of the sequences  |
| <i>structure</i> | The consensus structure in dot-bracket notation   |
| <i>columns</i>   | The number of columns before the alignment is wrapped as a new block (a value of 0 indicates no wrapping) |

**SWIG Wrapper Notes** This function is available as overloaded function `file_PS_aln()` with three additional parameters `start`, `end`, and `offset` before the `columns` argument. Thus, it resembles the [vrna\\_file\\_PS\\_aln\\_slice\(\)](#) function. The last four arguments may be omitted, indicating the default of `start = 0`, `end = 0`, `offset = 0`, and `columns = 60`.

### 16.78.2.2 [vrna\\_file\\_PS\\_aln\\_slice\(\)](#)

```
int vrna_file_PS_aln_slice (
    const char * filename,
    const char ** seqs,
    const char ** names,
    const char * structure,
    unsigned int start,
    unsigned int end,
    int offset,
    unsigned int columns )
#include <ViennaRNA/plotting	alignments.h>
```

Create an annotated PostScript alignment plot.

Similar to [vrna\\_file\\_PS\\_aln\(\)](#) but allows the user to print a particular slice of the alignment by specifying a `start` and `end` position. The additional `offset` parameter allows for adjusting the alignment position ruler value.

See also

[vrna\\_file\\_PS\\_aln\\_slice\(\)](#)

#### Parameters

|                  |   |
|------------------|---|
| <i>filename</i>  | The output file name  |
| <i>seqs</i>      | The aligned sequences   |
| <i>names</i>     | The names of the sequences  |
| <i>structure</i> | The consensus structure in dot-bracket notation   |
| <i>start</i>     | The start of the alignment slice (a value of 0 indicates the first position of the alignment, i.e. no slicing at 5' side) |
| <i>end</i>       | The end of the alignment slice (a value of 0 indicates the last position of the alignment, i.e. no slicing at 3' side)    |
| <i>offset</i>    | The alignment coordinate offset for the position ruler.   |
| <i>columns</i>   | The number of columns before the alignment is wrapped as a new block (a value of 0 indicates no wrapping)                 |

**SWIG Wrapper Notes** This function is available as overloaded function `file_PS_aln()` where the last four parameter may be omitted, indicating `start = 0`, `end = 0`, `offset = 0`, and `columns = 60`.

## 16.79 Search Algorithms

Implementations of various search algorithms to detect strings of objects within other strings of objects.

### 16.79.1 Detailed Description

Implementations of various search algorithms to detect strings of objects within other strings of objects.  
Collaboration diagram for Search Algorithms:

#### Files

- file [BoyerMoore.h](#)

*Variants of the Boyer-Moore string search algorithm.*

#### Functions

- `const unsigned int * vrna_search_BMH_num (const unsigned int *needle, size_t needle_size, const unsigned int *haystack, size_t haystack_size, size_t start, size_t *badchars, unsigned char cyclic)`

*Search for a string of elements in a larger string of elements using the Boyer-Moore-Horspool algorithm.*
- `const char * vrna_search_BMH (const char *needle, size_t needle_size, const char *haystack, size_t haystack_size, size_t start, size_t *badchars, unsigned char cyclic)`

*Search for an ASCII pattern within a larger ASCII string using the Boyer-Moore-Horspool algorithm.*
- `size_t * vrna_search_BM_BCT_num (const unsigned int *pattern, size_t pattern_size, unsigned int num_max)`

*Retrieve a Boyer-Moore Bad Character Table for a pattern of elements represented by natural numbers.*
- `size_t * vrna_search_BM_BCT (const char *pattern)`

*Retrieve a Boyer-Moore Bad Character Table for a NULL-terminated pattern of ASCII characters.*

### 16.79.2 Function Documentation

### 16.79.2.1 vrna\_search\_BMH\_num()

```
const unsigned int * vrna_search_BMH_num (
    const unsigned int * needle,
    size_t needle_size,
    const unsigned int * haystack,
    size_t haystack_size,
    size_t start,
    size_t * badchars,
    unsigned char cyclic )
#include <ViennaRNA/search/BoyerMoore.h>
```

Search for a string of elements in a larger string of elements using the Boyer-Moore-Horspool algorithm. To speed-up subsequent searches with this function, the Bad Character Table should be precomputed and passed as argument `badchars`.

#### See also

[vrna\\_search\\_BM\\_BCT\\_num\(\)](#), [vrna\\_search\\_BMH\(\)](#)

#### Parameters

|                            |  |
|----------------------------|--|
| <code>needle</code>        | The pattern of object representations to search for  |
| <code>needle_size</code>   | The size (length) of the pattern provided in <code>needle</code>   |
| <code>haystack</code>      | The string of objects the search will be performed on  |
| <code>haystack_size</code> | The size (length) of the <code>haystack</code> string  |
| <code>start</code>         | The position within <code>haystack</code> where to start the search  |
| <code>badchars</code>      | A pre-computed Bad Character Table obtained from <a href="#">vrna_search_BM_BCT_num()</a> (If NULL, a Bad Character Table will be generated automatically) |
| <code>cyclic</code>        | Allow for cyclic matches if non-zero, stop search at end of <code>haystack</code> otherwise  |

#### Returns

A pointer to the first occurrence of `needle` within `haystack` after position `start`

### 16.79.2.2 vrna\_search\_BMH()

```
const char * vrna_search_BMH (
    const char * needle,
    size_t needle_size,
    const char * haystack,
    size_t haystack_size,
    size_t start,
    size_t * badchars,
    unsigned char cyclic )
#include <ViennaRNA/search/BoyerMoore.h>
```

Search for an ASCII pattern within a larger ASCII string using the Boyer-Moore-Horspool algorithm. To speed-up subsequent searches with this function, the Bad Character Table should be precomputed and passed as argument `badchars`. Furthermore, both, the lengths of `needle` and the length of `haystack` should be pre-computed and must be passed along with each call.

#### See also

[vrna\\_search\\_BM\\_BCT\(\)](#), [vrna\\_search\\_BMH\\_num\(\)](#)

**Parameters**

|                      |  |
|----------------------|--|
| <i>needle</i>        | The NULL-terminated ASCII pattern to search for  |
| <i>needle_size</i>   | The size (length) of the pattern provided in <i>needle</i>   |
| <i>haystack</i>      | The NULL-terminated ASCII string of the search will be performed on  |
| <i>haystack_size</i> | The size (length) of the <i>haystack</i> string  |
| <i>start</i>         | The position within <i>haystack</i> where to start the search  |
| <i>badchars</i>      | A pre-computed Bad Character Table obtained from <a href="#">vrna_search_BM_BCT()</a> (If NULL, a Bad Character Table will be generated automatically) |
| <i>cyclic</i>        | Allow for cyclic matches if non-zero, stop search at end of haystack otherwise   |

**Returns**

A pointer to the first occurrence of *needle* within *haystack* after position *start*

**16.79.2.3 vrna\_search\_BM\_BCT\_num()**

```
size_t * vrna_search_BM_BCT_num (
    const unsigned int * pattern,
    size_t pattern_size,
    unsigned int num_max )
#include <ViennaRNA/search/BoyerMoore.h>
```

Retrieve a Boyer-Moore Bad Character Table for a pattern of elements represented by natural numbers.

**Note**

We store the maximum number representation of an element *num\_max* at position 0. So the actual bad character table *T* starts at *T[1]* for an element represented by number 0.

**See also**

[vrna\\_search\\_BMH\\_num\(\)](#), [vrna\\_search\\_BM\\_BCT\(\)](#)

**Parameters**

|                     |  |
|---------------------|--|
| <i>pattern</i>      | The pattern of element representations used in the subsequent search           |
| <i>pattern_size</i> | The size (length) of the pattern provided in <i>pattern</i>                    |
| <i>num_max</i>      | The maximum number representation of an element, i.e. the size of the alphabet |

**Returns**

A Bad Character Table for use in our Boyer-Moore search algorithm implementation(s)

**16.79.2.4 vrna\_search\_BM\_BCT()**

```
size_t * vrna_search_BM_BCT (
    const char * pattern )
#include <ViennaRNA/search/BoyerMoore.h>
```

Retrieve a Boyer-Moore Bad Character Table for a NULL-terminated pattern of ASCII characters.

**Note**

We store the maximum number representation of an element, i.e. 127 at position 0. So the actual bad character table *T* starts at *T[1]* for an element represented by ASCII code 0.

See also

[vrna\\_search\\_BMH\(\)](#), [vrna\\_search\\_BM\\_BCT\\_num\(\)](#)

#### Parameters

|                |   |
|----------------|---|
| <i>pattern</i> | The NULL-terminated pattern of ASCII characters used in the subsequent search |
|----------------|---|

#### Returns

A Bad Character Table for use in our Boyer-Moore search algorithm implementation(s)

## 16.80 Combinatorics Algorithms

Implementations to solve various combinatorial aspects for strings of objects.

### 16.80.1 Detailed Description

Implementations to solve various combinatorial aspects for strings of objects.

Collaboration diagram for Combinatorics Algorithms:

#### Files

- file [combinatorics.h](#)

*Various implementations that deal with combinatorial aspects of objects.*

#### Functions

- unsigned int \*\* [vrna\\_enumerate\\_necklaces](#) (const unsigned int \*type\_counts)  
*Enumerate all necklaces with fixed content.*
- unsigned int [vrna\\_rotational\\_symmetry\\_num](#) (const unsigned int \*string, size\_t string\_length)  
*Determine the order of rotational symmetry for a string of objects represented by natural numbers.*
- unsigned int [vrna\\_rotational\\_symmetry\\_pos\\_num](#) (const unsigned int \*string, size\_t string\_length, unsigned int \*\*positions)  
*Determine the order of rotational symmetry for a string of objects represented by natural numbers.*
- unsigned int [vrna\\_rotational\\_symmetry](#) (const char \*string)  
*Determine the order of rotational symmetry for a NULL-terminated string of ASCII characters.*
- unsigned int [vrna\\_rotational\\_symmetry\\_pos](#) (const char \*string, unsigned int \*\*positions)  
*Determine the order of rotational symmetry for a NULL-terminated string of ASCII characters.*
- unsigned int [vrna\\_rotational\\_symmetry\\_db](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, const char \*structure)  
*Determine the order of rotational symmetry for a dot-bracket structure.*
- unsigned int [vrna\\_rotational\\_symmetry\\_db\\_pos](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, const char \*structure, unsigned int \*\*positions)  
*Determine the order of rotational symmetry for a dot-bracket structure.*
- unsigned int \*\* [vrna\\_n\\_multichoose\\_k](#) (size\_t n, size\_t k)  
*Obtain a list of k-combinations with repetition (n multichoose k)*
- unsigned int \* [vrna\\_boustrophedon](#) (size\_t start, size\_t end)  
*Generate a sequence of Boustrophedon distributed numbers.*
- unsigned int [vrna\\_boustrophedon\\_pos](#) (size\_t start, size\_t end, size\_t pos)  
*Obtain the i-th element in a Boustrophedon distributed interval of natural numbers.*

### 16.80.2 Function Documentation

### 16.80.2.1 vrna\_enumerate\_necklaces()

```
unsigned int ** vrna_enumerate_necklaces (
    const unsigned int * type_counts )
#include <ViennaRNA/combinatorics.h>
```

Enumerate all necklaces with fixed content.

This function implements *A fast algorithm to generate necklaces with fixed content* as published by Joe Sawada in 2003 [20].

The function receives a list of counts (the elements on the necklace) for each type of object within a necklace. The list starts at index 0 and ends with an entry that has a count of 0. The algorithm then enumerates all non-cyclic permutations of the content, returned as a list of necklaces. This list, again, is zero-terminated, i.e. the last entry of the list is a NULL pointer.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>type_counts</code> | A 0-terminated list of entity counts |
|--------------------------|--------------------------------------|

#### Returns

A list of all non-cyclic permutations of the entities

**SWIG Wrapper Notes** This function is available as global function `enumerate_necklaces()` which accepts lists input, an produces list of lists output.

### 16.80.2.2 vrna\_rotational\_symmetry\_num()

```
unsigned int vrna_rotational_symmetry_num (
    const unsigned int * string,
    size_t string_length )
#include <ViennaRNA/combinatorics.h>
```

Determine the order of rotational symmetry for a string of objects represented by natural numbers.

The algorithm applies a fast search of the provided string within itself, assuming the end of the string wraps around to connect with its start. For example, a string of the form 011011 has rotational symmetry of order 2

This is a simplified version of `vrna_rotational_symmetry_pos_num()` that may be useful if one is only interested in the degree of rotational symmetry but not the actual set of rotational symmetric strings.

#### See also

[vrna\\_rotational\\_symmetry\\_pos\\_num\(\)](#), [vrna\\_rotationa\\_symmetry\(\)](#)

#### Parameters

|                            |   |
|----------------------------|---|
| <code>string</code>        | The string of elements encoded as natural numbers |
| <code>string_length</code> | The length of the string                          |

#### Returns

The order of rotational symmetry

**SWIG Wrapper Notes** This function is available as global function `rotational_symmetry()`. See [vrna\\_rotational\\_symmetry\\_pos\(\)](#) for details. Note, that in the target language the length of the list `string` is always known a-priori, so the parameter `string_length` must be omitted.

### 16.80.2.3 vrna\_rotational\_symmetry\_pos\_num()

```
unsigned int vrna_rotational_symmetry_pos_num (
```

```

    const unsigned int * string,
    size_t string_length,
    unsigned int ** positions )
#include <ViennaRNA/combinatorics.h>

```

Determine the order of rotational symmetry for a string of objects represented by natural numbers.

The algorithm applies a fast search of the provided string within itself, assuming the end of the string wraps around to connect with its start. For example, a string of the form 011011 has rotational symmetry of order 2

If the argument `positions` is not NULL, the function stores an array of string start positions for rotational shifts that map the string back onto itself. This array has length of order of rotational symmetry, i.e. the number returned by this function. The first element `positions[0]` always contains a shift value of 0 representing the trivial rotation.

#### Note

Do not forget to release the memory occupied by `positions` after a successful execution of this function.

#### See also

[vrna\\_rotational\\_symmetry\\_num\(\)](#), [vrna\\_rotational\\_symmetry\(\)](#), [vrna\\_rotational\\_symmetry\\_pos\(\)](#)

#### Parameters

|                            |   |
|----------------------------|---|
| <code>string</code>        | The string of elements encoded as natural numbers   |
| <code>string_length</code> | The length of the string  |
| <code>positions</code>     | A pointer to an (undefined) list of alternative string start positions that lead to an identity mapping (may be NULL) |

#### Returns

The order of rotational symmetry

**SWIG Wrapper Notes** This function is available as global function `rotational_symmetry()`. See [vrna\\_rotational\\_symmetry\\_pos\(\)](#) for details. Note, that in the target language the length of the list `string` is always known a-priori, so the parameter `string_length` must be omitted.

#### 16.80.2.4 vrna\_rotational\_symmetry()

```

unsigned int vrna_rotational_symmetry (
    const char * string )
#include <ViennaRNA/combinatorics.h>

```

Determine the order of rotational symmetry for a NULL-terminated string of ASCII characters.

The algorithm applies a fast search of the provided string within itself, assuming the end of the string wraps around to connect with its start. For example, a string of the form AABAAB has rotational symmetry of order 2

This is a simplified version of [vrna\\_rotational\\_symmetry\\_pos\(\)](#) that may be useful if one is only interested in the degree of rotational symmetry but not the actual set of rotational symmetric strings.

#### See also

[vrna\\_rotational\\_symmetry\\_pos\(\)](#), [vrna\\_rotationa\\_symmetry\\_num\(\)](#)

#### Parameters

|                     |  |
|---------------------|--|
| <code>string</code> | A NULL-terminated string of characters |
|---------------------|--|

**Returns**

The order of rotational symmetry

**SWIG Wrapper Notes** This function is available as global function **rotational\_symmetry()**. See [vrna\\_rotational\\_symmetry\\_pos\(\)](#) for details.

**16.80.2.5 vrna\_rotational\_symmetry\_pos()**

```
unsigned int vrna_rotational_symmetry_pos (
    const char * string,
    unsigned int ** positions )
#include <ViennaRNA/combinatorics.h>
```

Determine the order of rotational symmetry for a NULL-terminated string of ASCII characters.

The algorithm applies a fast search of the provided string within itself, assuming the end of the string wraps around to connect with its start. For example, a string of the form AABAAB has rotational symmetry of order 2

If the argument `positions` is not NULL, the function stores an array of string start positions for rotational shifts that map the string back onto itself. This array has length of order of rotational symmetry, i.e. the number returned by this function. The first element `positions[0]` always contains a shift value of 0 representing the trivial rotation.

**Note**

Do not forget to release the memory occupied by `positions` after a successful execution of this function.

**See also**

[vrna\\_rotational\\_symmetry\(\)](#), [vrna\\_rotational\\_symmetry\\_num\(\)](#), [vrna\\_rotational\\_symmetry\\_num\\_pos\(\)](#)

**Parameters**

|                        |   |
|------------------------|---|
| <code>string</code>    | A NULL-terminated string of characters  |
| <code>positions</code> | A pointer to an (undefined) list of alternative string start positions that lead to an identity mapping (may be NULL) |

**Returns**

The order of rotational symmetry

**SWIG Wrapper Notes** This function is available as overloaded global function **rotational\_symmetry()**. It merges the functionalities of [vrna\\_rotational\\_symmetry\(\)](#), [vrna\\_rotational\\_symmetry\\_pos\(\)](#), [vrna\\_rotational\\_symmetry\\_num\(\)](#), and [vrna\\_rotational\\_symmetry\\_pos\\_num\(\)](#). In contrast to our C-implementation, this function doesn't return the order of rotational symmetry as a single value, but returns a list of cyclic permutation shifts that result in a rotationally symmetric string. The length of the list then determines the order of rotational symmetry.

**16.80.2.6 vrna\_rotational\_symmetry\_db()**

```
unsigned int vrna_rotational_symmetry_db (
    vrna_fold_compound_t * fc,
    const char * structure )
#include <ViennaRNA/combinatorics.h>
```

Determine the order of rotational symmetry for a dot-bracket structure.

Given a (permutation of multiple) RNA strand(s) and a particular secondary structure in dot-bracket notation, compute the degree of rotational symmetry. In case there is only a single linear RNA strand, the structure always has degree 1, as there are no rotational symmetries due to the direction of the nucleic acid sequence and the fixed positions of 5' and 3' ends. However, for circular RNAs, rotational symmetries might arise if the sequence consists of a concatenation of  $k$  identical subsequences.

This is a simplified version of [vrna\\_rotational\\_symmetry\\_db\\_pos\(\)](#) that may be useful if one is only interested in the degree of rotational symmetry but not the actual set of rotational symmetric strings.

#### See also

[vrna\\_rotational\\_symmetry\\_db\\_pos\(\)](#), [vrna\\_rotational\\_symmetry\(\)](#), [vrna\\_rotational\\_symmetry\\_num\(\)](#)

#### Parameters

|                  |   |
|------------------|---|
| <i>fc</i>        | A fold_compound data structure containing the nucleic acid sequence(s), their order, and model settings |
| <i>structure</i> | The dot-bracket structure the degree of rotational symmetry is checked for                              |

#### Returns

The degree of rotational symmetry of the *structure* (0 in case of any errors)

**SWIG Wrapper Notes** This function is attached as method **rotational\_symmetry\_db()** to objects of type `fold_compound` (i.e. [vrna\\_fold\\_compound\\_t](#)). See [vrna\\_rotational\\_symmetry\\_db\\_pos\(\)](#) for details.

#### 16.80.2.7 vrna\_rotational\_symmetry\_db\_pos()

```
unsigned int vrna_rotational_symmetry_db_pos (
    vrna_fold_compound_t * fc,
    const char * structure,
    unsigned int ** positions )
#include <ViennaRNA/combinatorics.h>
```

Determine the order of rotational symmetry for a dot-bracket structure.

Given a (permutation of multiple) RNA strand(s) and a particular secondary structure in dot-bracket notation, compute the degree of rotational symmetry. In case there is only a single linear RNA strand, the structure always has degree 1, as there are no rotational symmetries due to the direction of the nucleic acid sequence and the fixed positions of 5' and 3' ends. However, for circular RNAs, rotational symmetries might arise if the sequence consists of a concatenation of *k* identical subsequences.

If the argument *positions* is not NULL, the function stores an array of string start positions for rotational shifts that map the string back onto itself. This array has length of order of rotational symmetry, i.e. the number returned by this function. The first element *positions*[0] always contains a shift value of 0 representing the trivial rotation.

#### Note

Do not forget to release the memory occupied by *positions* after a successful execution of this function.

#### See also

[vrna\\_rotational\\_symmetry\\_db\(\)](#), [vrna\\_rotational\\_symmetry\\_pos\(\)](#), [vrna\\_rotational\\_symmetry\\_pos\\_num\(\)](#)

#### Parameters

|                  |   |
|------------------|---|
| <i>fc</i>        | A fold_compound data structure containing the nucleic acid sequence(s), their order, and model settings               |
| <i>structure</i> | The dot-bracket structure the degree of rotational symmetry is checked for  |
| <i>positions</i> | A pointer to an (undefined) list of alternative string start positions that lead to an identity mapping (may be NULL) |

**Returns**

The degree of rotational symmetry of the structure (0 in case of any errors)

**SWIG Wrapper Notes** This function is attached as method `rotational_symmetry_db()` to objects of type `fold->_compound` (i.e. `vrna_fold_compound_t`). Thus, the first argument must be omitted. In contrast to our C-implementation, this function doesn't simply return the order of rotational symmetry of the secondary structure, but returns the list position of cyclic permutation shifts that result in a rotationally symmetric structure. The length of the list then determines the order of rotational symmetry.

**16.80.2.8 vrna\_n\_multichoose\_k()**

```
unsigned int ** vrna_n_multichoose_k (
    size_t n,
    size_t k )
#include <ViennaRNA/combinatorics.h>
```

Obtain a list of k-combinations with repetition ( $n$  multichoose  $k$ )

This function compiles a list of k-combinations, or k-multicombination, i.e. a list of multisubsets of size  $k$  from a set of integer values from 0 to  $n - 1$ . For that purpose, we enumerate  $n + k - 1$  choose  $k$  and decrease each index position  $i$  by  $i$  to obtain  $n$  multichoose  $k$ .

**Parameters**

|          |   |
|----------|---|
| <i>n</i> | Maximum number to choose from (interval of integers from 0 to $n - 1$ ) |
| <i>k</i> | Number of elements to choose, i.e. size of each multisubset             |

**Returns**

A list of lists of elements of combinations (last entry is terminated by **NULL**)

**16.80.2.9 vrna\_boustrophedon()**

```
unsigned int vrna_boustrophedon (
    size_t start,
    size_t end )
#include <ViennaRNA/combinatorics.h>
```

Generate a sequence of Boustrophedon distributed numbers.

This function generates a sequence of positive natural numbers within the interval  $[start, end]$  in a Boustrophedon fashion. That is, the numbers  $start, \dots, end$  in the resulting list are alternating between left and right ends of the interval while progressing to the inside, i.e. the list consists of a sequence of natural numbers of the form:

$$start, end, start + 1, end - 1, start + 2, end - 2, \dots$$

The resulting list is 1-based and contains the length of the sequence of numbers at its 0-th position.

Upon failure, the function returns **NULL**

**See also**

[vrna\\_boustrophedon\\_pos\(\)](#)

**Parameters**

|              |  |
|--------------|--|
| <i>start</i> | The first number of the list (left side of the interval) |
| <i>end</i>   | The last number of the list (right side of the interval) |

**Returns**

A list of alternating numbers from the interval  $[start, end]$  (or **NULL** on error)

**SWIG Wrapper Notes** This function is available as overloaded global function **boustrophedon()**.

**16.80.2.10 vrna\_boustrophedon\_pos()**

```
unsigned int vrna_boustrophedon_pos (
    size_t start,
    size_t end,
    size_t pos )
#include <ViennaRNA/combinatorics.h>
Obtain the i-th element in a Boustrophedon distributed interval of natural numbers.
```

**See also**

[vrna\\_boustrophedon\(\)](#)

**Parameters**

|              |   |
|--------------|---|
| <i>start</i> | The first number of the list (left side of the interval)                        |
| <i>end</i>   | The last number of the list (right side of the interval)                        |
| <i>pos</i>   | The index of the number within the Boustrophedon distributed sequence (1-based) |

**Returns**

The  $pos$ -th element in the Boustrophedon distributed sequence of natural numbers of the interval

**SWIG Wrapper Notes** This function is available as overloaded global function **boustrophedon()**. Omitting the *pos* argument yields the entire sequence from *start* to *end*.

## 16.81 (Abstract) Data Structures

All datastructures and typedefs shared among the ViennaRNA Package can be found here.

### 16.81.1 Detailed Description

All datastructures and typedefs shared among the ViennaRNA Package can be found here.

Collaboration diagram for (Abstract) Data Structures:

### Modules

- [The Fold Compound](#)

*This module provides interfaces that deal with the most basic data structure used in structure predicting and energy evaluating function of the RNAlib.*

- [The Dynamic Programming Matrices](#)

*This module provides interfaces that deal with creation and destruction of dynamic programming matrices used within the RNAlib.*

- [Hash Tables](#)

*Various implementations of hash table functions.*

- [Heaps](#)

*Interface for an abstract implementation of a heap data structure.*

- [Buffers](#)

*Functions that provide dynamically buffered stream-like data structures.*

## Files

- file [dp\\_matrices.h](#)  
*Functions to deal with standard dynamic programming (DP) matrices.*
- file [basic.h](#)  
*Various data structures and pre-processor macros.*

## Data Structures

- struct [vrna\\_basepair\\_s](#)  
*Base pair data structure used in subopt.c. [More...](#)*
- struct [vrna\\_cpair\\_s](#)  
*this datastructure is used as input parameter in functions of PS\_dot.c [More...](#)*
- struct [vrna\\_color\\_s](#)
- struct [vrna\\_data\\_linear\\_s](#)
- struct [vrna\\_sect\\_s](#)  
*Stack of partial structures for backtracking. [More...](#)*
- struct [vrna\\_bp\\_stack\\_s](#)  
*Base pair stack element. [More...](#)*
- struct [pu\\_contrib](#)  
*contributions to p\_u [More...](#)*
- struct [interact](#)  
*interaction data structure for RNAup [More...](#)*
- struct [pu\\_out](#)  
*Collection of all free\_energy of beeing unpaired values for output. [More...](#)*
- struct [constrain](#)  
*constraints for cofolding [More...](#)*
- struct [duplexT](#)  
*Data structure for RNAduplex. [More...](#)*
- struct [node](#)  
*Data structure for RNAsnoop (fold energy list) [More...](#)*
- struct [snoopT](#)  
*Data structure for RNAsnoop. [More...](#)*
- struct [dupVar](#)  
*Data structure used in RNApkplex. [More...](#)*

## Typedefs

- typedef struct [vrna\\_basepair\\_s](#) **vrna\_basepair\_t**  
*Typename for the base pair repesenting data structure [vrna\\_basepair\\_s](#).*
- typedef struct [vrna\\_elem\\_prob\\_s](#) **vrna\_plist\_t**  
*Typename for the base pair list repesenting data structure [vrna\\_elem\\_prob\\_s](#).*
- typedef struct [vrna\\_bp\\_stack\\_s](#) **vrna\_bp\_stack\_t**  
*Typename for the base pair stack repesenting data structure [vrna\\_bp\\_stack\\_s](#).*
- typedef struct [vrna\\_cpair\\_s](#) **vrna\_cpair\_t**  
*Typename for data structure [vrna\\_cpair\\_s](#).*
- typedef struct [vrna\\_sect\\_s](#) **vrna\_sect\_t**  
*Typename for stack of partial structures [vrna\\_sect\\_s](#).*
- typedef double **FLT\_OR\_DBL**  
*Typename for floating point number in partition function computations.*
- typedef struct [vrna\\_basepair\\_s](#) **PAIR**  
*Old typename of [vrna\\_basepair\\_s](#).*

- **typedef struct vrna\_elem\_prob\_s plist**  
*Old typename of vrna\_elem\_prob\_s.*
- **typedef struct vrna\_cpair\_s cpair**  
*Old typename of vrna\_cpair\_s.*
- **typedef struct vrna\_sect\_s sect**  
*Old typename of vrna\_sect\_s.*
- **typedef struct vrna\_bp\_stack\_s bondT**  
*Old typename of vrna\_bp\_stack\_s.*
- **typedef struct pu\_contrib pu\_contrib**  
*contributions to p\_u*
- **typedef struct interact interact**  
*interaction data structure for RNAup*
- **typedef struct pu\_out pu\_out**  
*Collection of all free\_energy of being unpaired values for output.*
- **typedef struct constrain constrain**  
*constraints for cofolding*
- **typedef struct node folden**  
*Data structure for RNAsnoop (fold energy list)*
- **typedef struct dupVar dupVar**  
*Data structure used in RNApkplex.*

## Functions

- **void vrna\_C11\_features (void)**  
*Dummy symbol to check whether the library was build using C11/C++11 features.*

### 16.81.2 Data Structure Documentation

#### 16.81.2.1 struct vrna\_basepair\_s

Base pair data structure used in subopt.c.

#### 16.81.2.2 struct vrna\_cpair\_s

this datastructure is used as input parameter in functions of PS\_dot.c

#### 16.81.2.3 struct vrna\_color\_s

#### 16.81.2.4 struct vrna\_data\_linear\_s

Collaboration diagram for vrna\_data\_linear\_s:

#### 16.81.2.5 struct vrna\_sect\_s

Stack of partial structures for backtracking.

#### 16.81.2.6 struct vrna\_bp\_stack\_s

Base pair stack element.

#### 16.81.2.7 struct pu\_contrib

contributions to p\_u

**Data Fields**

- double \*\* **H**  
*hairpin loops*
- double \*\* **I**  
*interior loops*
- double \*\* **M**  
*multi loops*
- double \*\* **E**  
*exterior loop*
- int **length**  
*length of the input sequence*
- int **w**  
*longest unpaired region*

**16.81.2.8 struct interact**

interaction data structure for RNAup

**Data Fields**

- double \* **Pi**  
*probabilities of interaction*
- double \* **Gi**  
*free energies of interaction*
- double **Gikjl**  
*full free energy for interaction between [k,j] k< i in longer seq and [j,l] j< l in shorter seq*
- double **Gikjl\_wo**  
*Gikjl without contributions for prob\_unpaired.*
- int **i**  
*k < i in longer seq*
- int **k**  
*k < i in longer seq*
- int **j**  
*j < l in shorter seq*
- int **l**  
*j < l in shorter seq*
- int **length**  
*length of longer sequence*

**16.81.2.9 struct pu\_out**

Collection of all free\_energy of beeing unpaired values for output.

**Data Fields**

- int **len**  
*sequence length*
- int **u\_vals**  
*number of different -u values*
- int **contribs**  
*[-c "SHIME"]*

- char \*\* **header**  
*header line*
- double \*\* **u\_values**  
*(the -u values \* [-c "SHIME"]) \* seq len*

### 16.81.2.10 struct constrain

constraints for cofolding

### 16.81.2.11 struct duplexT

Data structure for RNAduplex.

### 16.81.2.12 struct node

Data structure for RNAsnoop (fold energy list)

Collaboration diagram for node:

### 16.81.2.13 struct snoopT

Data structure for RNAsnoop.

### 16.81.2.14 struct dupVar

Data structure used in RNApkplex.

## 16.81.3 Typedef Documentation

### 16.81.3.1 PAIR

```
typedef struct vrna_basepair_s PAIR
#include <ViennaRNA/datastructures/basic.h>
Old typename of vrna_basepair_s.
```

**Deprecated** Use [vrna\\_basepair\\_t](#) instead!

### 16.81.3.2 plist

```
typedef struct vrna_elem_prob_s plist
#include <ViennaRNA/datastructures/basic.h>
Old typename of vrna_elem_prob_s.
```

**Deprecated** Use [vrna\\_ep\\_t](#) or [vrna\\_elem\\_prob\\_s](#) instead!

### 16.81.3.3 cpair

```
typedef struct vrna_cpair_s cpair
#include <ViennaRNA/datastructures/basic.h>
Old typename of vrna_cpair_s.
```

**Deprecated** Use [vrna\\_cpair\\_t](#) instead!

### 16.81.3.4 sect

```
typedef struct vrna_sect_s sect
#include <ViennaRNA/datastructures/basic.h>
Old typename of vrna_sect_s.
```

**Deprecated** Use `vrna_sect_t` instead!

### 16.81.3.5 bondT

```
typedef struct vrna_bp_stack_s bondT
#include <ViennaRNA/datastructures/basic.h>
Old typename of vrna_bp_stack_s.
```

**Deprecated** Use `vrna_bp_stack_t` instead!

## 16.81.4 Function Documentation

### 16.81.4.1 vrna\_C11\_features()

```
void vrna_C11_features (
    void )
#include <ViennaRNA/datastructures/basic.h>
```

Dummy symbol to check whether the library was build using C11/C++11 features.

By default, several data structures of our new v3.0 API use C11/C++11 features, such as unnamed unions, unnamed structs. However, these features can be deactivated at compile time to allow building the library and executables with compilers that do not support these features.

Now, the problem arises that once our static library is compiled and a third-party application is supposed to link against it, it needs to know, at compile time, how to correctly address particular data structures. This is usually implicitly taken care of through the API exposed in our header files. Unfortunately, we had some preprocessor directives in our header files that changed the API depending on the capabilities of the compiler the third-party application is build with. This in turn prohibited the use of an RNAlib compiled without C11/C++11 support in a program that compiles/links with enabled C11/C++11 support and vice-versa.

Therefore, we introduce this dummy symbol which can be used to check, whether the static library was build with C11/C++11 features.

#### Note

If the symbol is present, the library was build with enabled C11/C++11 features support and no action is required. However, if the symbol is missing in RNAlib  $\geq 2.2.9$ , programs that link to RNAlib must define a pre-processor identifier `VRNA_DISABLE_C11_FEATURES` before including any ViennaRNA Package header file, for instance by adding a `CPPFLAG`

```
CPPFLAGS+=-DVRNA_DISABLE_C11_FEATURES
```

#### Since

v2.2.9

## 16.82 Messages

Functions to print various kind of messages.

### 16.82.1 Detailed Description

Functions to print various kind of messages.

Collaboration diagram for Messages:

## Functions

- void [vrna\\_message\\_error](#) (const char \*format,...)  
*Print an error message and die.*
- void [vrna\\_message\\_verror](#) (const char \*format, va\_list args)  
*Print an error message and die.*
- void [vrna\\_message\\_warning](#) (const char \*format,...)  
*Print a warning message.*
- void [vrna\\_message\\_vwarning](#) (const char \*format, va\_list args)  
*Print a warning message.*
- void [vrna\\_message\\_info](#) (FILE \*fp, const char \*format,...)  
*Print an info message.*
- void [vrna\\_message\\_vinfo](#) (FILE \*fp, const char \*format, va\_list args)  
*Print an info message.*
- void [vrna\\_message\\_input\\_seq\\_simple](#) (void)  
*Print a line to stdout that asks for an input sequence.*
- void [vrna\\_message\\_input\\_seq](#) (const char \*s)  
*Print a line with a user defined string and a ruler to stdout.*

### 16.82.2 Function Documentation

#### 16.82.2.1 [vrna\\_message\\_error\(\)](#)

```
void vrna_message_error (
    const char * format,
    ...
)
#include <ViennaRNA/utils/basic.h>
```

Print an error message and die.

This function is a wrapper to `fprintf(stderr, ...)` that puts a capital **ERROR:** in front of the message and then exits the calling program.

See also

[vrna\\_message\\_verror\(\)](#), [vrna\\_message\\_warning\(\)](#), [vrna\\_message\\_info\(\)](#)

#### Parameters

|               |   |
|---------------|---|
| <i>format</i> | The error message to be printed                     |
| ...           | Optional arguments for the formatted message string |

#### 16.82.2.2 [vrna\\_message\\_verror\(\)](#)

```
void vrna_message_verror (
    const char * format,
    va_list args )
#include <ViennaRNA/utils/basic.h>
```

Print an error message and die.

This function is a wrapper to `vfprintf(stderr, ...)` that puts a capital **ERROR:** in front of the message and then exits the calling program.

## See also

[vrna\\_message\\_error\(\)](#), [vrna\\_message\\_warning\(\)](#), [vrna\\_message\\_info\(\)](#)

## Parameters

|               |  |
|---------------|--|
| <i>format</i> | The error message to be printed                    |
| <i>args</i>   | The argument list for the formatted message string |

**16.82.2.3 vrna\_message\_warning()**

```
void vrna_message_warning (
    const char * format,
    ...
)
```

#include <ViennaRNA/utils/basic.h>

Print a warning message.

This function is a wrapper to *fprintf(stderr, ...)* that puts a capital **WARNING:** in front of the message.

## See also

[vrna\\_message\\_vwarning\(\)](#), [vrna\\_message\\_error\(\)](#), [vrna\\_message\\_info\(\)](#)

## Parameters

|               |   |
|---------------|---|
| <i>format</i> | The warning message to be printed                   |
| ...           | Optional arguments for the formatted message string |

**16.82.2.4 vrna\_message\_vwarning()**

```
void vrna_message_vwarning (
    const char * format,
    va_list args
)
```

#include <ViennaRNA/utils/basic.h>

Print a warning message.

This function is a wrapper to *fprintf(stderr, ...)* that puts a capital **WARNING:** in front of the message.

## See also

[vrna\\_message\\_vwarning\(\)](#), [vrna\\_message\\_error\(\)](#), [vrna\\_message\\_info\(\)](#)

## Parameters

|               |  |
|---------------|--|
| <i>format</i> | The warning message to be printed                  |
| <i>args</i>   | The argument list for the formatted message string |

**16.82.2.5 vrna\_message\_info()**

```
void vrna_message_info (
    FILE * fp,
    const char * format,
    ...
)
```

```
#include <ViennaRNA/utils/basic.h>
Print an info message.
This function is a wrapper to fprintf(...).
```

See also

[vrna\\_message\\_vinfo\(\)](#), [vrna\\_message\\_error\(\)](#), [vrna\\_message\\_warning\(\)](#)

#### Parameters

|               |   |
|---------------|---|
| <i>fp</i>     | The file pointer where the message is printed to    |
| <i>format</i> | The warning message to be printed                   |
| ...           | Optional arguments for the formatted message string |

#### 16.82.2.6 vrna\_message\_vinfo()

```
void vrna_message_vinfo (
    FILE * fp,
    const char * format,
    va_list args)
#include <ViennaRNA/utils/basic.h>
Print an info message.
This function is a wrapper to fprintf(...).
```

See also

[vrna\\_message\\_vinfo\(\)](#), [vrna\\_message\\_error\(\)](#), [vrna\\_message\\_warning\(\)](#)

#### Parameters

|               |  |
|---------------|--|
| <i>fp</i>     | The file pointer where the message is printed to   |
| <i>format</i> | The info message to be printed                     |
| <i>args</i>   | The argument list for the formatted message string |

#### 16.82.2.7 vrna\_message\_input\_seq\_simple()

```
void vrna_message_input_seq_simple (
    void )
#include <ViennaRNA/utils/basic.h>
Print a line to stdout that asks for an input sequence.
There will also be a ruler (scale line) printed that helps orientation of the sequence positions
```

#### 16.82.2.8 vrna\_message\_input\_seq()

```
void vrna_message_input_seq (
    const char * s)
#include <ViennaRNA/utils/basic.h>
Print a line with a user defined string and a ruler to stdout.
(usually this is used to ask for user input) There will also be a ruler (scale line) printed that helps orientation of the sequence positions
```

#### Parameters

|          |   |
|----------|---|
| <i>s</i> | A user defined string that will be printed to <i>stdout</i> |
|----------|---|

## 16.83 Unit Conversion

Functions to convert between various physical units.

### 16.83.1 Detailed Description

Functions to convert between various physical units.

Collaboration diagram for Unit Conversion:

#### Files

- file `units.h`

*Physical Units and Functions to convert them into each other.*

#### Enumerations

- enum `vrna_unit_energy_e` {
   
VRNA\_UNIT\_J , VRNA\_UNIT\_KJ , VRNA\_UNIT\_CAL\_IT , VRNA\_UNIT\_DACAL\_IT ,
   
VRNA\_UNIT\_KCAL\_IT , VRNA\_UNIT\_CAL , VRNA\_UNIT\_DACAL , VRNA\_UNIT\_KCAL ,
   
VRNA\_UNIT\_G\_TNT , VRNA\_UNIT\_KG\_TNT , VRNA\_UNIT\_T\_TNT , VRNA\_UNIT\_EV ,
   
VRNA\_UNIT\_WH , VRNA\_UNIT\_KWH }
   
*Energy / Work Units.*
- enum `vrna_unit_temperature_e` {
   
VRNA\_UNIT\_K , VRNA\_UNIT\_DEG\_C , VRNA\_UNIT\_DEG\_F , VRNA\_UNIT\_DEG\_R ,
   
VRNA\_UNIT\_DEG\_N , VRNA\_UNIT\_DEG\_DE , VRNA\_UNIT\_DEG\_RE , VRNA\_UNIT\_DEG\_RO }
   
*Temperature Units.*

#### Functions

- double `vrna_convert_energy` (double energy, `vrna_unit_energy_e` from, `vrna_unit_energy_e` to)
   
*Convert between energy / work units.*
- double `vrna_convert_temperature` (double temp, `vrna_unit_temperature_e` from, `vrna_unit_temperature_e` to)
   
*Convert between temperature units.*
- int `vrna_convert_kcal_to_dcal` (double energy)
   
*Convert floating point energy value into integer representation.*
- double `vrna_convert_dcal_to_kcal` (int energy)
   
*Convert an integer representation of free energy in deka-cal/mol to kcal/mol.*

### 16.83.2 Enumeration Type Documentation

#### 16.83.2.1 `vrna_unit_energy_e`

```
enum vrna_unit_energy_e
#include <ViennaRNA/utils/units.h>
Energy / Work Units.
```

See also

`vrna_convert_energy()`

Enumerator

|                          |  |
|--------------------------|--|
| <code>VRNA_UNIT_J</code> | Joule ( $1 \text{ J} = 1 \text{ kg} \cdot \text{m}^2\text{s}^{-2}$ ) |
|--------------------------|--|

Enumerator

|                    |  |
|--------------------|--|
| VRNA_UNIT_KJ       | Kilojoule ( $1 \text{ kJ} = 1,000 \text{ J}$ )   |
| VRNA_UNIT_CAL_IT   | Calorie (International (Steam) Table, $1 \text{ cal}_{IT} = 4.1868 \text{ J}$ )  |
| VRNA_UNIT_DACAL_IT | Decacalorie (International (Steam) Table, $1 \text{ dacal}_{IT} = 10 \text{ cal}_{IT} = 41.868 \text{ J}$ )              |
| VRNA_UNIT_KCAL_IT  | Kilocalorie (International (Steam) Table, $1 \text{ kcal}_{IT} = 4.1868 \text{ kJ}$ )                                    |
| VRNA_UNIT_CAL      | Calorie (Thermochemical, $1 \text{ cal}_{th} = 4.184 \text{ J}$ )  |
| VRNA_UNIT_DACAL    | Decacalorie (Thermochemical, $1 \text{ dacal}_{th} = 10 \text{ cal}_{th} = 41.84 \text{ J}$ )                            |
| VRNA_UNIT_KCAL     | Kilocalorie (Thermochemical, $1 \text{ kcal}_{th} = 4.184 \text{ kJ}$ )  |
| VRNA_UNIT_G_TNT    | g TNT ( $1 \text{ g TNT} = 1,000 \text{ cal}_{th} = 4,184 \text{ J}$ )   |
| VRNA_UNIT_KG_TNT   | kg TNT ( $1 \text{ kg TNT} = 1,000 \text{ kcal}_{th} = 4,184 \text{ kJ}$ )   |
| VRNA_UNIT_T_TNT    | ton TNT ( $1 \text{ t TNT} = 1,000,000 \text{ kcal}_{th} = 4,184 \text{ MJ}$ )   |
| VRNA_UNIT_EV       | Electronvolt ( $1 \text{ eV} = 1.602176565 \times 10^{-19} \text{ J}$ )  |
| VRNA_UNIT_WH       | Watt hour ( $1 \text{ W} \cdot \text{h} = 1 \text{ W} \cdot 3,600 \text{ s} = 3,600 \text{ J} = 3.6 \text{ kJ}$ )        |
| VRNA_UNIT_KWH      | Kilowatt hour ( $1 \text{ kW} \cdot \text{h} = 1 \text{ kW} \cdot 3,600 \text{ s} = 3,600 \text{ kJ} = 3.6 \text{ MJ}$ ) |

### 16.83.2.2 vrna\_unit\_temperature\_e

```
enum vrna_unit_temperature_e
#include <ViennaRNA/utils/units.h>
Temperature Units.
```

See also

[vrna\\_convert\\_temperature\(\)](#)

Enumerator

|                  |  |
|------------------|--|
| VRNA_UNIT_K      | Kelvin (K)   |
| VRNA_UNIT_DEG_C  | Degree Celcius ( $^{\circ}\text{C}$ ) ( $[^{\circ}\text{C}] = [K] - 273.15$ )                              |
| VRNA_UNIT_DEG_F  | Degree Fahrenheit ( $^{\circ}\text{F}$ ) ( $[^{\circ}\text{F}] = [K] \times \frac{9}{5} - 459.67$ )        |
| VRNA_UNIT_DEG_R  | Degree Rankine ( $^{\circ}\text{R}$ ) ( $[^{\circ}\text{R}] = [K] \times \frac{9}{5}$ )                    |
| VRNA_UNIT_DEG_N  | Degree Newton ( $^{\circ}\text{N}$ ) ( $[^{\circ}\text{N}] = ([K] - 273.15) \times \frac{33}{100}$ )       |
| VRNA_UNIT_DEG_DE | Degree Delisle ( $^{\circ}\text{De}$ ) ( $[^{\circ}\text{De}] = (373.15 - [K]) \times \frac{3}{2}$ )       |
| VRNA_UNIT_DEG_RE | Degree Réaumur ( $^{\circ}\text{Ré}$ ) ( $[^{\circ}\text{Ré}] = ([K] - 273.15) \times \frac{4}{5}$ )       |
| VRNA_UNIT_DEG_RO | Degree Rømer ( $^{\circ}\text{Rø}$ ) ( $[^{\circ}\text{Rø}] = ([K] - 273.15) \times \frac{21}{40} + 7.5$ ) |

### 16.83.3 Function Documentation

#### 16.83.3.1 vrna\_convert\_energy()

```
double vrna_convert_energy (
    double energy,
    vrna_unit_energy_e from,
    vrna_unit_energy_e to )
#include <ViennaRNA/utils/units.h>
```

Convert between energy / work units.

See also

[vrna\\_unit\\_energy\\_e](#)

Parameters

|               |                    |
|---------------|--------------------|
| <i>energy</i> | Input energy value |
| <i>from</i>   | Input unit         |
| <i>to</i>     | Output unit        |

Returns

Energy value in Output unit

### 16.83.3.2 vrna\_convert\_temperature()

```
double vrna_convert_temperature (
    double temp,
    vrna_unit_temperature_e from,
    vrna_unit_temperature_e to )
#include <ViennaRNA/utils/units.h>
Convert between temperature units.
```

See also

[vrna\\_unit\\_temperature\\_e](#)

Parameters

|             |                         |
|-------------|-------------------------|
| <i>temp</i> | Input temperature value |
| <i>from</i> | Input unit              |
| <i>to</i>   | Output unit             |

Returns

Temperature value in Output unit

### 16.83.3.3 vrna\_convert\_kcal\_to\_dcal()

```
int vrna_convert_kcal_to_dcal (
    double energy )
#include <ViennaRNA/utils/units.h>
Convert floating point energy value into integer representation.
This function converts a floating point value in kcal/mol into its corresponding deka-cal/mol integer representation
as used throughout RNAlib.
```

See also

[vrna\\_convert\\_dcal\\_to\\_kcal\(\)](#)

**Parameters**

|                     |                              |
|---------------------|------------------------------|
| <code>energy</code> | The energy value in kcal/mol |
|---------------------|------------------------------|

**Returns**

The energy value in deka-cal/mol

**16.83.3.4 `vrna_convert_dcal_to_kcal()`**

```
double vrna_convert_dcal_to_kcal (
    int energy )
#include <ViennaRNA/utils/units.h>
```

Convert an integer representation of free energy in deka-cal/mol to kcal/mol.

This function converts a free energy value given as integer in deka-cal/mol into the corresponding floating point number in kcal/mol

**See also**

[vrna\\_convert\\_kcal\\_to\\_dcal\(\)](#)

**Parameters**

|                     |                            |
|---------------------|----------------------------|
| <code>energy</code> | The energy in deka-cal/mol |
|---------------------|----------------------------|

**Returns**

The energy in kcal/mol

## 16.84 The Fold Compound

This module provides interfaces that deal with the most basic data structure used in structure predicting and energy evaluating function of the RNAlib.

### 16.84.1 Detailed Description

This module provides interfaces that deal with the most basic data structure used in structure predicting and energy evaluating function of the RNAlib.

Throughout the entire RNAlib, the [vrna\\_fold\\_compound\\_t](#), is used to group information and data that is required for structure prediction and energy evaluation. Here, you'll find interface functions to create, modify, and delete [vrna\\_fold\\_compound\\_t](#) data structures. Collaboration diagram for The Fold Compound:

### Files

- file [fold\\_compound.h](#)  
*The Basic Fold Compound API.*

### Data Structures

- struct [vrna\\_fc\\_s](#)  
*The most basic data structure required by many functions throughout the RNAlib.* [More...](#)

### Macros

- #define [VRNA\\_STATUS\\_MFE\\_PRE](#) (unsigned char)1

- `#define VRNA_STATUS_MFE_POST` (unsigned char)2  
*Status message indicating that MFE computations are about to begin.*
- `#define VRNA_STATUS_PF_POST` (unsigned char)3  
*Status message indicating that MFE computations are finished.*
- `#define VRNA_STATUS_PF_PRE` (unsigned char)4  
*Status message indicating that Partition function computations are about to begin.*
- `#define VRNA_STATUS_PF_POST` (unsigned char)4  
*Status message indicating that Partition function computations are finished.*
- `#define VRNA_OPTION_DEFAULT` 0U  
*Option flag to specify default settings/requirements.*
- `#define VRNA_OPTION_MFE` 1U  
*Option flag to specify requirement of Minimum Free Energy (MFE) DP matrices and corresponding set of energy parameters.*
- `#define VRNA_OPTION_PF` 2U  
*Option flag to specify requirement of Partition Function (PF) DP matrices and corresponding set of Boltzmann factors.*
- `#define VRNA_OPTION_HYBRID` 4U  
*Option flag to specify requirement of dimer DP matrices.*
- `#define VRNA_OPTION_EVAL_ONLY` 8U  
*Option flag to specify that neither MFE, nor PF DP matrices are required.*
- `#define VRNA_OPTION_WINDOW` 16U  
*Option flag to specify requirement of DP matrices for local folding approaches.*

## Typedefs

- `typedef struct vrna_fc_s vrna_fold_compound_t`  
*Typename for the fold\_compound data structure `vrna_fc_s`.*
- `typedef void() vrna_callback_free_auxdata(void *data)`  
*Callback to free memory allocated for auxiliary user-provided data.*
- `typedef void() vrna_callback_recursion_status(unsigned char status, void *data)`  
*Callback to perform specific user-defined actions before, or after recursive computations.*

## Enumerations

- `enum vrna_fc_type_e { VRNA_FC_TYPE_SINGLE , VRNA_FC_TYPE_COMPARATIVE }`  
*An enumerator that is used to specify the type of a `vrna_fold_compound_t`.*

## Functions

- `vrna_fold_compound_t * vrna_fold_compound (const char *sequence, const vrna_md_t *md_p, unsigned int options)`  
*Retrieve a `vrna_fold_compound_t` data structure for single sequences and hybridizing sequences.*
- `vrna_fold_compound_t * vrna_fold_compound_comparative (const char **sequences, vrna_md_t *md_p, unsigned int options)`  
*Retrieve a `vrna_fold_compound_t` data structure for sequence alignments.*
- `void vrna_fold_compound_free (vrna_fold_compound_t *fc)`  
*Free memory occupied by a `vrna_fold_compound_t`.*
- `void vrna_fold_compound_add_auxdata (vrna_fold_compound_t *fc, void *data, vrna_callback_free_auxdata *f)`  
*Add auxiliary data to the `vrna_fold_compound_t`.*
- `void vrna_fold_compound_add_callback (vrna_fold_compound_t *fc, vrna_callback_recursion_status *f)`  
*Add a recursion status callback to the `vrna_fold_compound_t`.*

## 16.84.2 Data Structure Documentation

### 16.84.2.1 struct vrna\_fc\_s

The most basic data structure required by many functions throughout the RNAlib.

#### Note

Please read the documentation of this data structure carefully! Some attributes are only available for specific types this data structure can adopt.

#### Warning

Reading/Writing from/to attributes that are not within the scope of the current type usually result in undefined behavior!

#### See also

[vrna\\_fold\\_compound\\_t.type](#), [vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\\_compound\\_comparative\(\)](#), [vrna\\_fold\\_compound\\_free\(\)](#),  
[VRNA\\_FC\\_TYPE\\_SINGLE](#), [VRNA\\_FC\\_TYPE\\_COMPARATIVE](#)

**SWIG Wrapper Notes** This data structure is wrapped as an object **fold\_compound** with several related functions attached as methods.

A new **fold\_compound** can be obtained by calling one of its constructors:

- *fold\_compound(seq)* – Initialize with a single sequence, or two concatenated sequences separated by an ampersand character '&' (for cofolding)
- *fold\_compound(aln)* – Initialize with a sequence alignment *aln* stored as a list of sequences (with gap characters)

The resulting object has a list of attached methods which in most cases directly correspond to functions that mainly operate on the corresponding C data structure:

- *type()* – Get the type of the *fold\_compound* (See [vrna\\_fc\\_type\\_e](#))
- *length()* – Get the length of the sequence(s) or alignment stored within the *fold\_compound*

Collaboration diagram for vrna\_fc\_s:

#### Data Fields

##### Common data fields

- const [vrna\\_fc\\_type\\_e](#) **type**  
*The type of the vrna\_fold\_compound\_t.*
- unsigned int **length**  
*The length of the sequence (or sequence alignment)*
- int **cutpoint**  
*The position of the (cofold) cutpoint within the provided sequence. If there is no cutpoint, this field will be set to -1.*
- unsigned int \* **strand\_number**  
*The strand number a particular nucleotide is associated with.*
- unsigned int \* **strand\_order**  
*The strand order, i.e. permutation of current concatenated sequence.*
- unsigned int \* **strand\_order\_uniq**  
*The strand order array where identical sequences have the same ID.*
- unsigned int \* **strand\_start**  
*The start position of a particular strand within the current concatenated sequence.*
- unsigned int \* **strand\_end**  
*The end (last) position of a particular strand within the current concatenated sequence.*

- **unsigned int strands**  
*Number of interacting strands.*
- **vrna\_seq\_t \* nucleotides**  
*Set of nucleotide sequences.*
- **vrna\_msa\_t \* alignment**  
*Set of alignments.*
- **vrna\_hc\_t \* hc**  
*The hard constraints data structure used for structure prediction.*
- **vrna\_mx\_mfe\_t \* matrices**  
*The MFE DP matrices.*
- **vrna\_mx\_pf\_t \* exp\_matrices**  
*The PF DP matrices*
- **vrna\_param\_t \* params**  
*The precomputed free energy contributions for each type of loop.*
- **vrna\_exp\_param\_t \* exp\_params**  
*The precomputed free energy contributions as Boltzmann factors*
- **int \* iindx**  
*DP matrix accessor*
- **int \* jindx**  
*DP matrix accessor*

### User-defined data fields

- **vrna\_callback\_recursion\_status \* stat\_cb**  
*Recursion status callback (usually called just before, and after recursive computations in the library).*
- **void \* auxdata**  
*A pointer to auxiliary, user-defined data.*
- **vrna\_callback\_free\_auxdata \* free\_auxdata**  
*A callback to free auxiliary user data whenever the fold\_compound itself is free'd.*

### Secondary Structure Decomposition (grammar) related data fields

- **vrna\_sd\_t \* domains\_struct**  
*Additional structured domains.*
- **vrna\_ud\_t \* domains\_up**  
*Additional unstructured domains.*
- **vrna\_gr\_aux\_t \* aux\_grammar**  
*Additional decomposition grammar rules.*

### Data fields available for single/hybrid structure prediction

#### Data fields for consensus structure prediction

#### Additional data fields for Distance Class Partitioning

*These data fields are typically populated with meaningful data only if used in the context of Distance Class Partitioning*

- **unsigned int maxD1**  
*Maximum allowed base pair distance to first reference.*
- **unsigned int maxD2**  
*Maximum allowed base pair distance to second reference.*
- **short \* reference\_pt1**  
*A pairtable of the first reference structure.*
- **short \* reference\_pt2**  
*A pairtable of the second reference structure.*
- **unsigned int \* referenceBPs1**

- **unsigned int \* referenceBPs2**  
*Matrix containing number of basepairs of reference structure1 in interval [i,j].*
- **unsigned int \* bpdist**  
*Matrix containing number of basepairs of reference structure2 in interval [i,j].*
- **unsigned int \* mm1**  
*Matrix containing base pair distance of reference structure 1 and 2 on interval [i,j].*
- **unsigned int \* mm2**  
*Maximum matching matrix, reference struct 1 disallowed.*
- **unsigned int \* mm2**  
*Maximum matching matrix, reference struct 2 disallowed.*

### Additional data fields for local folding

These data fields are typically populated with meaningful data only if used in the context of local folding

- **int window\_size**  
*window size for local folding sliding window approach*
- **char \*\* ptype\_local**  
*Pair type array (for local folding)*
- **vrna\_zsc\_dat\_t zscore\_data**  
*Data structure with settings for z-score computations.*

#### 16.84.2.1.1 Field Documentation

##### 16.84.2.1.1.1 **type** const vrna\_fc\_type\_e vrna\_fc\_s::type

The type of the [vrna\\_fold\\_compound\\_t](#).

Currently possible values are [VRNA\\_FC\\_TYPE\\_SINGLE](#), and [VRNA\\_FC\\_TYPE\\_COMPARATIVE](#)

##### Warning

Do not edit this attribute, it will be automatically set by the corresponding get() methods for the [vrna\\_fold\\_compound\\_t](#). The value specified in this attribute dictates the set of other attributes to use within this data structure.

##### 16.84.2.1.1.2 **stat\_cb** vrna\_callback\_recursion\_status\* vrna\_fc\_s::stat\_cb

Recursion status callback (usually called just before, and after recursive computations in the library).

##### See also

[vrna\\_callback\\_recursion\\_status\(\)](#), [vrna\\_fold\\_compound\\_add\\_callback\(\)](#)

##### 16.84.2.1.1.3 **auxdata** void\* vrna\_fc\_s::auxdata

A pointer to auxiliary, user-defined data.

##### See also

[vrna\\_fold\\_compound\\_add\\_auxdata\(\)](#), [vrna\\_fold\\_compound\\_t.free\\_auxdata](#)

##### 16.84.2.1.1.4 **free\_auxdata** vrna\_callback\_free\_auxdata\* vrna\_fc\_s::free\_auxdata

A callback to free auxiliary user data whenever the fold\_compound itself is free'd.

##### See also

[vrna\\_fold\\_compound\\_t.auxdata](#), [vrna\\_callback\\_free\\_auxdata\(\)](#)

**16.84.2.1.1.5 sequence** `char* vrna_fc_s::sequence`  
The input sequence string.

**Warning**

Only available if

`type==VRNA_FC_TYPE_SINGLE`

**16.84.2.1.1.6 sequence\_encoding** `short* vrna_fc_s::sequence_encoding`  
Numerical encoding of the input sequence.

**See also**

[vrna\\_sequence\\_encode\(\)](#)

**Warning**

Only available if

`type==VRNA_FC_TYPE_SINGLE`

**16.84.2.1.1.7 ptype** `char* vrna_fc_s::ptype`

Pair type array.

Contains the numerical encoding of the pair type for each pair (i,j) used in MFE, Partition function and Evaluation computations.

**Note**

This array is always indexed via jindx, in contrast to previously different indexing between mfe and pf variants!

**Warning**

Only available if

`type==VRNA_FC_TYPE_SINGLE`

**See also**

[vrna\\_idx\\_col\\_wise\(\)](#), [vrna\\_ptypes\(\)](#)

**16.84.2.1.1.8 ptype\_pf\_compat** `char* vrna_fc_s::ptype_pf_compat`  
ptype array indexed via iindx

**Deprecated** This attribute will vanish in the future! It's meant for backward compatibility only!

**Warning**

Only available if

`type==VRNA_FC_TYPE_SINGLE`

**16.84.2.1.1.9 sc** `vrna_sc_t* vrna_fc_s::sc`

The soft constraints for usage in structure prediction and evaluation.

**Warning**

Only available if

`type==VRNA_FC_TYPE_SINGLE`

**16.84.2.1.1.10 sequences** `char** vrna_fc_s::sequences`  
The aligned sequences.

**Note**

The end of the alignment is indicated by a NULL pointer in the second dimension

**Warning**

Only available if

`type==VRNA_FC_TYPE_COMPARATIVE`

**16.84.2.1.1.11 n\_seq** `unsigned int vrna_fc_s::n_seq`  
The number of sequences in the alignment.

**Warning**

Only available if

`type==VRNA_FC_TYPE_COMPARATIVE`

**16.84.2.1.1.12 cons\_seq** `char* vrna_fc_s::cons_seq`  
The consensus sequence of the aligned sequences.

**Warning**

Only available if

`type==VRNA_FC_TYPE_COMPARATIVE`

**16.84.2.1.1.13 S\_cons** `short* vrna_fc_s::S_cons`  
Numerical encoding of the consensus sequence.

**Warning**

Only available if

`type==VRNA_FC_TYPE_COMPARATIVE`

**16.84.2.1.1.14 S** `short** vrna_fc_s::S`  
Numerical encoding of the sequences in the alignment.

**Warning**

Only available if

`type==VRNA_FC_TYPE_COMPARATIVE`

**16.84.2.1.1.15 S5** `short** vrna_fc_s::S5`  
`S5[s][i]` holds next base 5' of i in sequence s.

**Warning**

Only available if

`type==VRNA_FC_TYPE_COMPARATIVE`

**16.84.2.1.1.16 S3** short\*\* vrna\_fc\_s::S3  
Sl[s][i] holds next base 3' of i in sequence s.

Warning

Only available if

```
type==VRNA_FC_TYPE_COMPARATIVE
```

**16.84.2.1.1.17 pscore** int\* vrna\_fc\_s::pscore

Precomputed array of pair types expressed as pairing scores.

Warning

Only available if

```
type==VRNA_FC_TYPE_COMPARATIVE
```

**16.84.2.1.1.18 pscore\_local** int\*\* vrna\_fc\_s::pscore\_local

Precomputed array of pair types expressed as pairing scores.

Warning

Only available if

```
type==VRNA_FC_TYPE_COMPARATIVE
```

**16.84.2.1.1.19 pscore\_pf\_compat** short\* vrna\_fc\_s::pscore\_pf\_compat

Precomputed array of pair types expressed as pairing scores indexed via iindx.

**Deprecated** This attribute will vanish in the future!

Warning

Only available if

```
type==VRNA_FC_TYPE_COMPARATIVE
```

**16.84.2.1.1.20 scs** vrna\_sc\_t\*\* vrna\_fc\_s::scs

A set of soft constraints (for each sequence in the alignment)

Warning

Only available if

```
type==VRNA_FC_TYPE_COMPARATIVE
```

## 16.84.3 Macro Definition Documentation

### 16.84.3.1 VRNA\_STATUS\_MFE\_PRE

```
#define VRNA_STATUS_MFE_PRE (unsigned char)1
#include <ViennaRNA/fold_compound.h>
```

Status message indicating that MFE computations are about to begin.

See also

[vrna\\_fold\\_compound\\_t.stat\\_cb](#), [vrna\\_callback\\_recursion\\_status\(\)](#), [vrna\\_mfe\(\)](#), [vrna\\_fold\(\)](#), [vrna\\_circfold\(\)](#), [vrna\\_alifold\(\)](#), [vrna\\_circalifold\(\)](#), [vrna\\_cofold\(\)](#)

### 16.84.3.2 VRNA\_STATUS\_MFE\_POST

```
#define VRNA_STATUS_MFE_POST (unsigned char)2
#include <ViennaRNA/fold_compound.h>
Status message indicating that MFE computations are finished.
```

See also

[vrna\\_fold\\_compound\\_t.stat\\_cb](#), [vrna\\_callback\\_recursion\\_status\(\)](#), [vrna\\_mfe\(\)](#), [vrna\\_fold\(\)](#), [vrna\\_circfold\(\)](#), [vrna\\_alifold\(\)](#), [vrna\\_circalifold\(\)](#), [vrna\\_cofold\(\)](#)

### 16.84.3.3 VRNA\_STATUS\_PF\_PRE

```
#define VRNA_STATUS_PF_PRE (unsigned char)3
#include <ViennaRNA/fold_compound.h>
Status message indicating that Partition function computations are about to begin.
```

See also

[vrna\\_fold\\_compound\\_t.stat\\_cb](#), [vrna\\_callback\\_recursion\\_status\(\)](#), [vrna\\_pf\(\)](#)

### 16.84.3.4 VRNA\_STATUS\_PF\_POST

```
#define VRNA_STATUS_PF_POST (unsigned char)4
#include <ViennaRNA/fold_compound.h>
Status message indicating that Partition function computations are finished.
```

See also

[vrna\\_fold\\_compound\\_t.stat\\_cb](#), [vrna\\_callback\\_recursion\\_status\(\)](#), [vrna\\_pf\(\)](#)

### 16.84.3.5 VRNA\_OPTION\_MFE

```
#define VRNA_OPTION_MFE 1U
#include <ViennaRNA/fold_compound.h>
Option flag to specify requirement of Minimum Free Energy (MFE) DP matrices and corresponding set of energy parameters.
```

See also

[vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\\_compound\\_comparative\(\)](#), [VRNA\\_OPTION\\_EVAL\\_ONLY](#)

### 16.84.3.6 VRNA\_OPTION\_PF

```
#define VRNA_OPTION_PF 2U
#include <ViennaRNA/fold_compound.h>
Option flag to specify requirement of Partition Function (PF) DP matrices and corresponding set of Boltzmann factors.
```

See also

[vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\\_compound\\_comparative\(\)](#), [VRNA\\_OPTION\\_EVAL\\_ONLY](#)

### 16.84.3.7 VRNA\_OPTION\_EVAL\_ONLY

```
#define VRNA_OPTION_EVAL_ONLY 8U
#include <ViennaRNA/fold_compound.h>
```

Option flag to specify that neither MFE, nor PF DP matrices are required.

Use this flag in conjunction with [VRNA\\_OPTION\\_MFE](#), and [VRNA\\_OPTION\\_PF](#) to save memory for a [vrna\\_fold\\_compound\\_t](#) obtained from [vrna\\_fold\\_compound\(\)](#), or [vrna\\_fold\\_compound\\_comparative\(\)](#) in cases where only energy evaluation but no structure prediction is required.

See also

[vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\\_compound\\_comparative\(\)](#), [vrna\\_eval\\_structure\(\)](#)

## 16.84.4 Typedef Documentation

### 16.84.4.1 vrna\_callback\_free\_auxdata

```
typedef void() vrna_callback_free_auxdata(void *data)
#include <ViennaRNA/fold_compound.h>
```

Callback to free memory allocated for auxiliary user-provided data.

This type of user-implemented function usually deletes auxiliary data structures. The user must take care to free all the memory occupied by the data structure passed.

**Notes on Callback Functions** This callback is supposed to free memory occupied by an auxiliary data structure. It will be called when the [vrna\\_fold\\_compound\\_t](#) is erased from memory through a call to [vrna\\_fold\\_compound\\_free\(\)](#) and will be passed the address of memory previously bound to the [vrna\\_fold\\_compound\\_t](#) via [vrna\\_fold\\_compound\\_add\\_auxdata\(\)](#).

See also

[vrna\\_fold\\_compound\\_add\\_auxdata\(\)](#), [vrna\\_fold\\_compound\\_free\(\)](#), [vrna\\_fold\\_compound\\_add\\_callback\(\)](#)

#### Parameters

|             |                                  |
|-------------|----------------------------------|
| <i>data</i> | The data that needs to be free'd |
|-------------|----------------------------------|

### 16.84.4.2 vrna\_callback\_recursion\_status

```
typedef void() vrna_callback_recursion_status(unsigned char status, void *data)
#include <ViennaRNA/fold_compound.h>
```

Callback to perform specific user-defined actions before, or after recursive computations.

**Notes on Callback Functions** This function will be called to notify a third-party implementation about the status of a currently ongoing recursion. The purpose of this callback mechanism is to provide users with a simple way to ensure pre- and post conditions for auxiliary mechanisms attached to our implementations.

See also

[vrna\\_fold\\_compound\\_add\\_auxdata\(\)](#), [vrna\\_fold\\_compound\\_add\\_callback\(\)](#), [vrna\\_mfe\(\)](#), [vrna\\_pf\(\)](#), [VRNA\\_STATUS\\_MFE\\_PRE](#), [VRNA\\_STATUS\\_MFE\\_POST](#), [VRNA\\_STATUS\\_PF\\_PRE](#), [VRNA\\_STATUS\\_PF\\_POST](#)

## Parameters

|                     |  |
|---------------------|--|
| <code>status</code> | The status indicator   |
| <code>data</code>   | The data structure that was assigned with <a href="#">vrna_fold_compound_add_auxdata()</a> |

## 16.84.5 Enumeration Type Documentation

### 16.84.5.1 `vrna_fc_type_e`

```
enum vrna_fc_type_e
#include <ViennaRNA/fold_compound.h>
```

An enumerator that is used to specify the type of a [vrna\\_fold\\_compound\\_t](#).

#### Enumerator

|                                       |   |
|---------------------------------------|---|
| <code>VRNA_FC_TYPE_SINGLE</code>      | Type is suitable for single, and hybridizing sequences                    |
| <code>VRNA_FC_TYPE_COMPARATIVE</code> | Type is suitable for sequence alignments (consensus structure prediction) |

## 16.84.6 Function Documentation

### 16.84.6.1 `vrna_fold_compound()`

```
vrna_fold_compound_t * vrna_fold_compound (
    const char * sequence,
    const vrna_md_t * md_p,
    unsigned int options )
#include <ViennaRNA/fold_compound.h>
```

Retrieve a [vrna\\_fold\\_compound\\_t](#) data structure for single sequences and hybridizing sequences. This function provides an easy interface to obtain a prefilled [vrna\\_fold\\_compound\\_t](#) by passing a single sequence, or two concatenated sequences as input. For the latter, sequences need to be separated by an '&' character like this:

```
char *sequence = "GGGG&CCCC";
```

The optional parameter `md_p` can be used to specify the model details for successive computations based on the content of the generated [vrna\\_fold\\_compound\\_t](#). Passing NULL will instruct the function to use default model details. The third parameter `options` may be used to specify dynamic programming (DP) matrix requirements.

#### Options

- [VRNA\\_OPTION\\_DEFAULT](#) - Option flag to specify default settings/requirements.
- [VRNA\\_OPTION\\_MFE](#) - Option flag to specify requirement of Minimum Free Energy (MFE) DP matrices and corresponding set of energy parameters.
- [VRNA\\_OPTION\\_PF](#) - Option flag to specify requirement of Partition Function (PF) DP matrices and corresponding set of Boltzmann factors.
- [VRNA\\_OPTION\\_WINDOW](#) - Option flag to specify requirement of DP matrices for local folding approaches.

The above options may be OR-ed together.

If you just need the folding compound serving as a container for your data, you can simply pass [VRNA\\_OPTION\\_DEFAULT](#) to the `option` parameter. This creates a [vrna\\_fold\\_compound\\_t](#) without DP matrices, thus saving memory. Subsequent calls of any structure prediction function will then take care of allocating the memory required for the DP matrices. If you only intend to evaluate structures instead of actually predicting

them, you may use the `VRNA_OPTION_EVAL_ONLY` macro. This will seriously speedup the creation of the `vrna_fold_compound_t`.

#### Note

The sequence string must be uppercase, and should contain only RNA (resp. DNA) alphabet depending on what energy parameter set is used

#### See also

`vrna_fold_compound_free()`, `vrna_fold_compound_comparative()`, `vrna_md_t`

#### Parameters

|                       |  |
|-----------------------|--|
| <code>sequence</code> | A single sequence, or two concatenated sequences seperated by an '&' character |
| <code>md_p</code>     | An optional set of model details   |
| <code>options</code>  | The options for DP matrices memory allocation                                  |

#### Returns

A prefilled `vrna_fold_compound_t` ready to be used for computations (may be `NULL` on error)

#### 16.84.6.2 `vrna_fold_compound_comparative()`

```
vrna_fold_compound_t * vrna_fold_compound_comparative (
    const char ** sequences,
    vrna_md_t * md_p,
    unsigned int options )
#include <ViennaRNA/fold_compound.h>
```

Retrieve a `vrna_fold_compound_t` data structure for sequence alignments.

This function provides an easy interface to obtain a prefilled `vrna_fold_compound_t` by passing an alignment of sequences.

The optional parameter `md_p` can be used to specify the model details for successive computations based on the content of the generated `vrna_fold_compound_t`. Passing `NULL` will instruct the function to use default model details. The third parameter `options` may be used to specify dynamic programming (DP) matrix requirements.

#### Options

- `VRNA_OPTION_DEFAULT` - Option flag to specify default settings/requirements.
- `VRNA_OPTION_MFE` - Option flag to specify requirement of Minimum Free Energy (MFE) DP matrices and corresponding set of energy parameters.
- `VRNA_OPTION_PF` - Option flag to specify requirement of Partition Function (PF) DP matrices and corresponding set of Boltzmann factors.
- `VRNA_OPTION_WINDOW` - Option flag to specify requirement of DP matrices for local folding approaches.

The above options may be OR-ed together.

If you just need the folding compound serving as a container for your data, you can simply pass `VRNA_OPTION_DEFAULT` to the `option` parameter. This creates a `vrna_fold_compound_t` without DP matrices, thus saving memory. Subsequent calls of any structure prediction function will then take care of allocating the memory required for the DP matrices. If you only intend to evaluate structures instead of actually predicting them, you may use the `VRNA_OPTION_EVAL_ONLY` macro. This will seriously speedup the creation of the `vrna_fold_compound_t`.

**Note**

The sequence strings must be uppercase, and should contain only RNA (resp. DNA) alphabet including gap characters depending on what energy parameter set is used.

**See also**

[vrna\\_fold\\_compound\\_free\(\)](#), [vrna\\_fold\\_compound\(\)](#), [vrna\\_md\\_t](#), [VRNA\\_OPTION\\_MFE](#), [VRNA\\_OPTION\\_PF](#), [VRNA\\_OPTION\\_EVAL\\_ONLY](#), [read\\_clustal\(\)](#)

**Parameters**

|                  |   |
|------------------|---|
| <i>sequences</i> | A sequence alignment including 'gap' characters |
| <i>md_p</i>      | An optional set of model details                |
| <i>options</i>   | The options for DP matrices memory allocation   |

**Returns**

A prefilled [vrna\\_fold\\_compound\\_t](#) ready to be used for computations (may be `NULL` on error)

**16.84.6.3 vrna\_fold\_compound\_free()**

```
void vrna_fold_compound_free (
    vrna_fold_compound_t * fc )
#include <ViennaRNA/fold_compound.h>
Free memory occupied by a vrna\_fold\_compound\_t.
```

**See also**

[vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\\_compound\\_comparative\(\)](#), [vrna\\_mx\\_mfe\\_free\(\)](#), [vrna\\_mx\\_pf\\_free\(\)](#)

**Parameters**

|           |   |
|-----------|---|
| <i>fc</i> | The <a href="#">vrna_fold_compound_t</a> that is to be erased from memory |
|-----------|---|

**16.84.6.4 vrna\_fold\_compound\_add\_auxdata()**

```
void vrna_fold_compound_add_auxdata (
    vrna_fold_compound_t * fc,
    void * data,
    vrna_callback_free_auxdata * f )
#include <ViennaRNA/fold_compound.h>
Add auxiliary data to the vrna\_fold\_compound\_t.
```

This function allows one to bind arbitrary data to a [vrna\\_fold\\_compound\\_t](#) which may later on be used by one of the callback functions, e.g. [vrna\\_callback\\_recursion\\_status\(\)](#). To allow for proper cleanup of the memory occupied by this auxiliary data, the user may also provide a pointer to a cleanup function that free's the corresponding memory. This function will be called automatically when the [vrna\\_fold\\_compound\\_t](#) is free'd with [vrna\\_fold\\_compound\\_free\(\)](#).

**Note**

Before attaching the arbitrary data pointer, this function will call the [vrna\\_callback\\_free\\_auxdata\(\)](#) on any pre-existing data that is already attached.

See also

[vrna\\_callback\\_free\\_auxdata\(\)](#)

Parameters

|             |   |
|-------------|---|
| <i>fc</i>   | The fold_compound the arbitrary data pointer should be associated with                |
| <i>data</i> | A pointer to an arbitrary data structure  |
| <i>f</i>    | A pointer to function that free's memory occupied by the arbitrary data (May be NULL) |

#### 16.84.6.5 vrna\_fold\_compound\_add\_callback()

```
void vrna_fold_compound_add_callback (
    vrna_fold_compound_t * fc,
    vrna_callback_recursion_status * f )
#include <ViennaRNA/fold_compound.h>
```

Add a recursion status callback to the [vrna\\_fold\\_compound\\_t](#).

Binding a recursion status callback function to a [vrna\\_fold\\_compound\\_t](#) allows one to perform arbitrary operations just before, or after an actual recursive computations, e.g. MFE prediction, is performed by the RNAlib. The callback function will be provided with a pointer to its [vrna\\_fold\\_compound\\_t](#), and a status message. Hence, it has complete access to all variables that influence the recursive computations.

See also

[vrna\\_callback\\_recursion\\_status\(\)](#), [vrna\\_fold\\_compound\\_t](#), [VRNA\\_STATUS\\_MFE\\_PRE](#), [VRNA\\_STATUS\\_MFE\\_POST](#),  
[VRNA\\_STATUS\\_PF\\_PRE](#), [VRNA\\_STATUS\\_PF\\_POST](#)

Parameters

|           |   |
|-----------|---|
| <i>fc</i> | The fold_compound the callback function should be attached to |
| <i>f</i>  | The pointer to the recursion status callback function         |

## 16.85 The Dynamic Programming Matrices

This module provides interfaces that deal with creation and destruction of dynamic programming matrices used within the RNAlib.

### 16.85.1 Detailed Description

This module provides interfaces that deal with creation and destruction of dynamic programming matrices used within the RNAlib.

Collaboration diagram for The Dynamic Programming Matrices:

### Data Structures

- struct [vrna\\_mx\\_mfe\\_s](#)

Minimum Free Energy (MFE) Dynamic Programming (DP) matrices data structure required within the [vrna\\_fold\\_compound\\_t](#). More...
- struct [vrna\\_mx\\_pf\\_s](#)

Partition function (PF) Dynamic Programming (DP) matrices data structure required within the [vrna\\_fold\\_compound\\_t](#). More...

## Typedefs

- `typedef struct vrna_mx_mfe_s vrna_mx_mfe_t`  
*Typename for the Minimum Free Energy (MFE) DP matrices data structure `vrna_mx_mfe_s`.*
- `typedef struct vrna_mx_pf_s vrna_mx_pf_t`  
*Typename for the Partition Function (PF) DP matrices data structure `vrna_mx_pf_s`.*

## Enumerations

- enum `vrna_mx_type_e { VRNA_MX_DEFAULT , VRNA_MX_WINDOW , VRNA_MX_2DFOLD }`  
*An enumerator that is used to specify the type of a polymorphic Dynamic Programming (DP) matrix data structure.*

## Functions

- int `vrna_mx_add (vrna_fold_compound_t *vc, vrna_mx_type_e type, unsigned int options)`  
*Add Dynamic Programming (DP) matrices (allocate memory)*
- void `vrna_mx_mfe_free (vrna_fold_compound_t *vc)`  
*Free memory occupied by the Minimum Free Energy (MFE) Dynamic Programming (DP) matrices.*
- void `vrna_mx_pf_free (vrna_fold_compound_t *vc)`  
*Free memory occupied by the Partition Function (PF) Dynamic Programming (DP) matrices.*

### 16.85.2 Data Structure Documentation

#### 16.85.2.1 struct vrna\_mx\_mfe\_s

Minimum Free Energy (MFE) Dynamic Programming (DP) matrices data structure required within the `vrna_fold_compound_t`.

#### Data Fields

##### Common fields for MFE matrices

- const `vrna_mx_type_e type`
- unsigned int `length`  
*Length of the sequence, therefore an indicator of the size of the DP matrices.*
- unsigned int `strands`

##### Default DP matrices

###### Note

*These data fields are available if*  
`vrna_mx_mfe_t.type == VRNA_MX_DEFAULT`

##### Local Folding DP matrices using window approach

###### Note

*These data fields are available if*  
`vrna_mx_mfe_t.type == VRNA_MX_WINDOW`

##### Distance Class DP matrices

###### Note

*These data fields are available if*  
`vrna_mx_mfe_t.type == VRNA_MX_2DFOLD`

#### 16.85.2.1.1 Field Documentation

##### 16.85.2.1.1.1 type const vrna\_mx\_type\_e vrna\_mx\_mfe\_s::type

Type of the DP matrices

**16.85.2.1.1.2 strands** `unsigned int vrna_mx_mfe_s::strands`  
 Number of strands

### 16.85.2.2 struct vrna\_mx\_pf\_s

Partition function (PF) Dynamic Programming (DP) matrices data structure required within the `vrna_fold_compound_t`.

#### Data Fields

##### Common fields for DP matrices

- `const vrna_mx_type_e type`
- `unsigned int length`
- `FLT_OR_DBL * scale`
- `FLT_OR_DBL * expMLbase`

##### Default PF matrices

###### Note

*These data fields are available if*  
`vrna_mx_pf_t.type == VRNA_MX_DEFAULT`

##### Local Folding DP matrices using window approach

###### Note

*These data fields are available if*  
`vrna_mx_mfe_t.type == VRNA_MX_WINDOW`

##### Distance Class DP matrices

###### Note

*These data fields are available if*  
`vrna_mx_pf_t.type == VRNA_MX_2DFOLD`

### 16.85.2.2.1 Field Documentation

**16.85.2.2.1.1 type** `const vrna_mx_type_e vrna_mx_pf_s::type`  
 Type of the DP matrices

**16.85.2.2.1.2 length** `unsigned int vrna_mx_pf_s::length`  
 Size of the DP matrices (i.e. sequence length)

**16.85.2.2.1.3 scale** `FLT_OR_DBL* vrna_mx_pf_s::scale`  
 Boltzmann factor scaling

**16.85.2.2.1.4 expMLbase** `FLT_OR_DBL* vrna_mx_pf_s::expMLbase`  
 Boltzmann factors for unpaired bases in multibranch loop

### 16.85.3 Enumeration Type Documentation

#### 16.85.3.1 vrna\_mx\_type\_e

```
enum vrna_mx_type_e
#include <ViennaRNA/dp_matrices.h>
```

An enumerator that is used to specify the type of a polymorphic Dynamic Programming (DP) matrix data structure.

See also

`vrna_mx_mfe_t, vrna_mx_pf_t`

## Enumerator

|                 |   |
|-----------------|---|
| VRNA_MX_DEFAULT | Default DP matrices.  |
| VRNA_MX_WINDOW  | DP matrices suitable for local structure prediction using window approach.<br><br>See also<br><br><a href="#">vrna_mfe_window()</a> , <a href="#">vrna_mfe_window_zscore()</a> , <a href="#">pfl_fold()</a> |
| VRNA_MX_2DFOLD  | DP matrices suitable for distance class partitioned structure prediction.<br><br>See also<br><br><a href="#">vrna_mfe_TwoD()</a> , <a href="#">vrna_pf_TwoD()</a>   |

**16.85.4 Function Documentation****16.85.4.1 vrna\_mx\_add()**

```
int vrna_mx_add (
    vrna_fold_compound_t * vc,
    vrna_mx_type_e type,
    unsigned int options )
#include <ViennaRNA/dp_matrices.h>
```

Add Dynamic Programming (DP) matrices (allocate memory)

This function adds DP matrices of a specific type to the provided `vrna_fold_compound_t`, such that successive DP recursion can be applied. The function caller has to specify which type of DP matrix is requested, see `vrna_mx_type_e`, and what kind of recursive algorithm will be applied later on, using the parameters `type`, and `options`, respectively. For the latter, Minimum free energy (MFE), and Partition function (PF) computations are distinguished. A third option that may be passed is `VRNA_OPTION_HYBRID`, indicating that auxiliary DP arrays are required for RNA-RNA interaction prediction.

**Note**

Usually, there is no need to call this function, since the constructors of `vrna_fold_compound_t` are handling all the DP matrix memory allocation.

**See also**

[vrna\\_mx\\_mfe\\_add\(\)](#), [vrna\\_mx\\_pf\\_add\(\)](#), [vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\\_compound\\_comparative\(\)](#), [vrna\\_fold\\_compound\\_free\(\)](#), [vrna\\_mx\\_pf\\_free\(\)](#), [vrna\\_mx\\_mfe\\_free\(\)](#), [vrna\\_mx\\_type\\_e](#), [VRNA\\_OPTION\\_MFE](#), [VRNA\\_OPTION\\_PF](#), [VRNA\\_OPTION\\_HYBRID](#), [VRNA\\_OPTION\\_EVAL\\_ONLY](#)

**Parameters**

|                      |   |
|----------------------|---|
| <code>vc</code>      | The <code>vrna_fold_compound_t</code> that holds pointers to the DP matrices                            |
| <code>type</code>    | The type of DP matrices requested   |
| <code>options</code> | Option flags that specify the kind of DP matrices, such as MFE or PF arrays, and auxiliary requirements |

**Returns**

1 if DP matrices were properly allocated and attached, 0 otherwise

**16.85.4.2 vrna\_mx\_mfe\_free()**

```
void vrna_mx_mfe_free (
    vrna_fold_compound_t * vc )
#include <ViennaRNA/dp_matrices.h>
```

Free memory occupied by the Minimum Free Energy (MFE) Dynamic Programming (DP) matrices.

**See also**

[vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\\_compound\\_comparative\(\)](#), [vrna\\_fold\\_compound\\_free\(\)](#), [vrna\\_mx\\_pf\\_free\(\)](#)

**Parameters**

|    |  |
|----|--|
| vc | The <a href="#">vrna_fold_compound_t</a> storing the MFE DP matrices that are to be erased from memory |
|----|--|

**16.85.4.3 vrna\_mx\_pf\_free()**

```
void vrna_mx_pf_free (
    vrna_fold_compound_t * vc )
#include <ViennaRNA/dp_matrices.h>
```

Free memory occupied by the Partition Function (PF) Dynamic Programming (DP) matrices.

**See also**

[vrna\\_fold\\_compound\(\)](#), [vrna\\_fold\\_compound\\_comparative\(\)](#), [vrna\\_fold\\_compound\\_free\(\)](#), [vrna\\_mx\\_mfe\\_free\(\)](#)

**Parameters**

|    |   |
|----|---|
| vc | The <a href="#">vrna_fold_compound_t</a> storing the PF DP matrices that are to be erased from memory |
|----|---|

## 16.86 Hash Tables

Various implementations of hash table functions.

### 16.86.1 Detailed Description

Various implementations of hash table functions.

Hash tables are common data structures that allow for fast random access to the data that is stored within.

Here, we provide an abstract implementation of a hash table interface and a concrete implementation for pairs of secondary structure and corresponding free energy value. Collaboration diagram for Hash Tables:

### Files

- file [hash\\_tables.h](#)

*Implementations of hash table functions.*

### Data Structures

- struct [vrna\\_ht\\_entry\\_db\\_t](#)

*Default hash table entry.* [More...](#)

## Abstract interface

- `typedef struct vrna_hash_table_s *vrna_hash_table_t`  
*A hash table object.*
- `typedef int() vrna_callback_ht_compare_entries(void *x, void *y)`  
*Callback function to compare two hash table entries.*
- `typedef unsigned int() vrna_callback_ht_hash_function(void *x, unsigned long hashtable_size)`  
*Callback function to generate a hash key, i.e. hash function.*
- `typedef int() vrna_callback_ht_free_entry(void *x)`  
*Callback function to free a hash table entry.*
- `vrna_hash_table_t vrna_ht_init(unsigned int b, vrna_callback_ht_compare_entries *compare_function, vrna_callback_ht_hash_function *hash_function, vrna_callback_ht_free_entry *free_hash_entry)`  
*Get an initialized hash table.*
- `unsigned long vrna_ht_size(vrna_hash_table_t ht)`  
*Get the size of the hash table.*
- `unsigned long vrna_ht_collisions(struct vrna_hash_table_s *ht)`  
*Get the number of collisions in the hash table.*
- `void *vrna_ht_get(vrna_hash_table_t ht, void *x)`  
*Get an element from the hash table.*
- `int vrna_ht_insert(vrna_hash_table_t ht, void *x)`  
*Insert an object into a hash table.*
- `void vrna_ht_remove(vrna_hash_table_t ht, void *x)`  
*Remove an object from the hash table.*
- `void vrna_ht_clear(vrna_hash_table_t ht)`  
*Clear the hash table.*
- `void vrna_ht_free(vrna_hash_table_t ht)`  
*Free all memory occupied by the hash table.*

## Dot-Bracket / Free Energy entries

- `int vrna_ht_db_comp(void **x, void **y)`  
*Default hash table entry comparison.*
- `unsigned int vrna_ht_db_hash_func(void *x, unsigned long hashtable_size)`  
*Default hash function.*
- `int vrna_ht_db_free_entry(void *hash_entry)`  
*Default function to free memory occupied by a hash entry.*

## 16.86.2 Data Structure Documentation

### 16.86.2.1 struct vrna\_ht\_entry\_db\_t

Default hash table entry.

See also

[vrna\\_ht\\_init\(\)](#), [vrna\\_ht\\_db\\_comp\(\)](#), [vrna\\_ht\\_db\\_hash\\_func\(\)](#), [vrna\\_ht\\_db\\_free\\_entry\(\)](#)

#### Data Fields

- `char *structure`
- `float energy`

#### 16.86.2.1.1 Field Documentation

**16.86.2.1.1.1 structure** `char* vrna_ht_entry_db_t::structure`  
A secondary structure in dot-bracket notation

**16.86.2.1.1.2 energy** `float vrna_ht_entry_db_t::energy`  
The free energy of structure

### 16.86.3 Typedef Documentation

#### 16.86.3.1 vrna\_hash\_table\_t

```
typedef struct vrna_hash_table_s* vrna_hash_table_t
#include <ViennaRNA/datastructures/hash_tables.h>
```

A hash table object.

See also

[vrna\\_ht\\_init\(\)](#), [vrna\\_ht\\_free\(\)](#)

#### 16.86.3.2 vrna\_callback\_ht\_compare\_entries

```
typedef int() vrna_callback_ht_compare_entries(void *x, void *y)
#include <ViennaRNA/datastructures/hash_tables.h>
```

Callback function to compare two hash table entries.

See also

[vrna\\_ht\\_init\(\)](#), [vrna\\_ht\\_db\\_comp\(\)](#)

Parameters

|                |                    |
|----------------|--------------------|
| <code>x</code> | A hash table entry |
| <code>y</code> | A hash table entry |

Returns

-1 if `x` is smaller, +1 if `x` is larger than `y`. 0 if `x == y`

#### 16.86.3.3 vrna\_callback\_ht\_hash\_function

```
typedef unsigned int() vrna_callback_ht_hash_function(void *x, unsigned long hashtable_size)
#include <ViennaRNA/datastructures/hash_tables.h>
```

Callback function to generate a hash key, i.e. hash function.

See also

[vrna\\_ht\\_init\(\)](#), [vrna\\_ht\\_db\\_hash\\_func\(\)](#)

Parameters

|                             |                            |
|-----------------------------|----------------------------|
| <code>x</code>              | A hash table entry         |
| <code>hashtable_size</code> | The size of the hash table |

**Returns**

The hash table key for entry  $x$

**16.86.3.4 vrna\_callback\_ht\_free\_entry**

```
typedef int() vrna_callback_ht_free_entry(void **x)
#include <ViennaRNA/datastructures/hash_tables.h>
Callback function to free a hash table entry.
```

**See also**

[vrna\\_ht\\_init\(\)](#), [vrna\\_ht\\_db\\_free\\_entry\(\)](#)

**Parameters**

|     |                    |
|-----|--------------------|
| $x$ | A hash table entry |
|-----|--------------------|

**Returns**

0 on success

**16.86.4 Function Documentation****16.86.4.1 vrna\_ht\_init()**

```
vrna_hash_table_t vrna_ht_init (
    unsigned int b,
    vrna_callback_ht_compare_entries * compare_function,
    vrna_callback_ht_hash_function * hash_function,
    vrna_callback_ht_free_entry * free_hash_entry )
#include <ViennaRNA/datastructures/hash_tables.h>
Get an initialized hash table.
```

This function returns a ready-to-use hash table with pre-allocated memory for a particular number of entries.

**Note**

If all function pointers are `NULL`, this function initializes the hash table with *default functions*, i.e.

- [vrna\\_ht\\_db\\_comp\(\)](#) for the `compare_function`,
- [vrna\\_ht\\_db\\_hash\\_func\(\)](#) for the `hash_function`, and
- [vrna\\_ht\\_db\\_free\\_entry\(\)](#) for the `free_hash_entry`

arguments.

**Warning**

If `hash_bits` is larger than 27 you have to compile it with the flag `gcc -fno-tree-vectorize`.

**Parameters**

|                               |   |
|-------------------------------|---|
| $b$                           | Number of bits for the hash table. This determines the size ( $2^b - 1$ ).                  |
| <code>compare_function</code> | A function pointer to compare any two entries in the hash table (may be <code>NULL</code> ) |
| <code>hash_function</code>    | A function pointer to retrieve the hash value of any entry (may be <code>NULL</code> )      |
| <code>free_hash_entry</code>  | A function pointer to free the memory occupied by any entry (may be <code>NULL</code> )     |

**Returns**

An initialized, empty hash table, or NULL on any error

**16.86.4.2 vrna\_ht\_size()**

```
unsigned long vrna_ht_size (
    vrna_hash_table_t ht )
#include <ViennaRNA/datastructures/hash_tables.h>
Get the size of the hash table.
```

**Parameters**

|           |                |
|-----------|----------------|
| <i>ht</i> | The hash table |
|-----------|----------------|

**Returns**

The size of the hash table, i.e. the maximum number of entries

**16.86.4.3 vrna\_ht\_collisions()**

```
unsigned long vrna_ht_collisions (
    struct vrna_hash_table_s * ht )
#include <ViennaRNA/datastructures/hash_tables.h>
Get the number of collisions in the hash table.
```

**Parameters**

|           |                |
|-----------|----------------|
| <i>ht</i> | The hash table |
|-----------|----------------|

**Returns**

The number of collisions in the hash table

**16.86.4.4 vrna\_ht\_get()**

```
void * vrna_ht_get (
    vrna_hash_table_t ht,
    void * x )
#include <ViennaRNA/datastructures/hash_tables.h>
Get an element from the hash table.
```

This function takes an object *x* and performs a look-up whether the object is stored within the hash table *ht*. If the object is already stored in *ht*, the function simply returns the entry, otherwise it returns NULL.

**See also**

[vrna\\_ht\\_insert\(\)](#), [vrna\\_hash\\_delete\(\)](#), [vrna\\_ht\\_init\(\)](#)

**Parameters**

|           |                           |
|-----------|---------------------------|
| <i>ht</i> | The hash table            |
| <i>x</i>  | The hash entry to look-up |

**Returns**

The entry *x* if it is stored in *ht*, NULL otherwise

**16.86.4.5 vrna\_ht\_insert()**

```
int vrna_ht_insert (
    vrna_hash_table_t ht,
    void * x )
#include <ViennaRNA/datastructures/hash_tables.h>
Insert an object into a hash table.
Writes the pointer to your hash entry into the table.
```

**Warning**

In case of collisions, this function simply increments the hash key until a free entry in the hash table is found.

**See also**

[vrna\\_ht\\_init\(\)](#), [vrna\\_hash\\_delete\(\)](#), [vrna\\_ht\\_clear\(\)](#)

**Parameters**

|           |                |
|-----------|----------------|
| <i>ht</i> | The hash table |
| <i>x</i>  | The hash entry |

**Returns**

0 on success, 1 if the value is already in the hash table, -1 on error.

**16.86.4.6 vrna\_ht\_remove()**

```
void vrna_ht_remove (
    vrna_hash_table_t ht,
    void * x )
#include <ViennaRNA/datastructures/hash_tables.h>
Remove an object from the hash table.
Deletes the pointer to your hash entry from the table.
```

**Note**

This function doesn't free any memory occupied by the hash entry.

**Parameters**

|           |                |
|-----------|----------------|
| <i>ht</i> | The hash table |
| <i>x</i>  | The hash entry |

**16.86.4.7 vrna\_ht\_clear()**

```
void vrna_ht_clear (
    vrna_hash_table_t ht )
#include <ViennaRNA/datastructures/hash_tables.h>
```

Clear the hash table.

This function removes all entries from the hash table and automatically free's the memory occupied by each entry using the bound [vrna\\_callback\\_ht\\_free\\_entry\(\)](#) function.

See also

[vrna\\_ht\\_free\(\)](#), [vrna\\_ht\\_init\(\)](#)

Parameters

|           |                |
|-----------|----------------|
| <i>ht</i> | The hash table |
|-----------|----------------|

#### 16.86.4.8 vrna\_ht\_free()

```
void vrna_ht_free (
    vrna_hash_table_t ht )
#include <ViennaRNA/datastructures/hash_tables.h>
```

Free all memory occupied by the hash table.

This function removes all entries from the hash table by calling the [vrna\\_callback\\_ht\\_free\\_entry\(\)](#) function for each entry. Finally, the memory occupied by the hash table itself is free'd as well.

Parameters

|           |                |
|-----------|----------------|
| <i>ht</i> | The hash table |
|-----------|----------------|

#### 16.86.4.9 vrna\_ht\_db\_comp()

```
int vrna_ht_db_comp (
    void * x,
    void * y )
#include <ViennaRNA/datastructures/hash_tables.h>
```

Default hash table entry comparison.

This is the default comparison function for hash table entries. It assumes the both entries *x* and *y* are of type [vrna\\_ht\\_entry\\_db\\_t](#) and compares the `structure` attribute of both entries

See also

[vrna\\_ht\\_entry\\_db\\_t](#), [vrna\\_ht\\_init\(\)](#), [vrna\\_ht\\_db\\_hash\\_func\(\)](#), [vrna\\_ht\\_db\\_free\\_entry\(\)](#)

Parameters

|          |   |
|----------|---|
| <i>x</i> | A hash table entry of type <a href="#">vrna_ht_entry_db_t</a> |
| <i>y</i> | A hash table entry of type <a href="#">vrna_ht_entry_db_t</a> |

Returns

-1 if *x* is smaller, +1 if *x* is larger than *y*. 0 if both are equal.

#### 16.86.4.10 vrna\_ht\_db\_hash\_func()

```
unsigned int vrna_ht_db_hash_func (
    void * x,
    unsigned long hashtable_size )
```

```
#include <ViennaRNA/datastructures/hash_tables.h>
```

Default hash function.

This is the default hash function for hash table insertion/lookup. It assumes that entries are of type `vrna_ht_entry_db_t` and uses the Bob Jenkins 1996 mix function to create a hash key from the `structure` attribute of the hash entry.

See also

[vrna\\_ht\\_entry\\_db\\_t](#), [vrna\\_ht\\_init\(\)](#), [vrna\\_ht\\_db\\_comp\(\)](#), [vrna\\_ht\\_db\\_free\\_entry\(\)](#)

Parameters

|                             |   |
|-----------------------------|---|
| <code>x</code>              | A hash table entry to compute the key for |
| <code>hashtable_size</code> | The size of the hash table                |

Returns

The hash key for entry `x`

#### 16.86.4.11 `vrna_ht_db_free_entry()`

```
int vrna_ht_db_free_entry (
    void * hash_entry )
```

```
#include <ViennaRNA/datastructures/hash_tables.h>
```

Default function to free memory occupied by a hash entry.

This function assumes that hash entries are of type `vrna_ht_entry_db_t` and free's the memory occupied by that entry.

See also

[vrna\\_ht\\_entry\\_db\\_t](#), [vrna\\_ht\\_init\(\)](#), [vrna\\_ht\\_db\\_comp\(\)](#), [vrna\\_ht\\_db\\_hash\\_func\(\)](#)

Parameters

|                         |                                      |
|-------------------------|--------------------------------------|
| <code>hash_entry</code> | The hash entry to remove from memory |
|-------------------------|--------------------------------------|

Returns

0 on success

## 16.87 Heaps

Interface for an abstract implementation of a heap data structure.

### 16.87.1 Detailed Description

Interface for an abstract implementation of a heap data structure.

Collaboration diagram for Heaps:

### Files

- file `heap.h`

*Implementation of an abstract heap data structure.*

## Typedefs

- `typedef struct vrna_heap_s * vrna_heap_t`  
*An abstract heap data structure.*
- `typedef int() vrna_callback_heap_cmp(const void *a, const void *b, void *data)`  
*Heap compare function prototype.*
- `typedef size_t() vrna_callback_heap_get_pos(const void *a, void *data)`  
*Retrieve the position of a particular heap entry within the heap.*
- `typedef void() vrna_callback_heap_set_pos(const void *a, size_t pos, void *data)`  
*Store the position of a particular heap entry within the heap.*

## Functions

- `vrna_heap_t vrna_heap_init (size_t n, vrna_callback_heap_cmp *cmp, vrna_callback_heap_get_pos *get_entry_pos, vrna_callback_heap_set_pos *set_entry_pos, void *data)`  
*Initialize a heap data structure.*
- `void vrna_heap_free (vrna_heap_t h)`  
*Free memory occupied by a heap data structure.*
- `size_t vrna_heap_size (struct vrna_heap_s *h)`  
*Get the size of a heap data structure, i.e. the number of stored elements.*
- `void vrna_heap_insert (vrna_heap_t h, void *v)`  
*Insert an element into the heap.*
- `void * vrna_heap_pop (vrna_heap_t h)`  
*Pop (remove and return) the object at the root of the heap.*
- `const void * vrna_heap_top (vrna_heap_t h)`  
*Get the object at the root of the heap.*
- `void * vrna_heap_remove (vrna_heap_t h, const void *v)`  
*Remove an arbitrary element within the heap.*
- `void * vrna_heap_update (vrna_heap_t h, void *v)`  
*Update an arbitrary element within the heap.*

### 16.87.2 Typedef Documentation

#### 16.87.2.1 vrna\_heap\_t

```
typedef struct vrna_heap_s* vrna_heap_t
#include <ViennaRNA/datastructures/heap.h>
```

An abstract heap data structure.

#### See also

[vrna\\_heap\\_init\(\)](#), [vrna\\_heap\\_free\(\)](#), [vrna\\_heap\\_insert\(\)](#), [vrna\\_heap\\_pop\(\)](#), [vrna\\_heap\\_top\(\)](#), [vrna\\_heap\\_remove\(\)](#), [vrna\\_heap\\_update\(\)](#)

#### 16.87.2.2 vrna\_callback\_heap\_cmp

```
typedef int() vrna_callback_heap_cmp(const void *a, const void *b, void *data)
#include <ViennaRNA/datastructures/heap.h>
```

Heap compare function prototype.

Use this prototype to design the compare function for the heap implementation. The arbitrary data pointer `data` may be used to get access to further information required to actually compare the two values `a` and `b`.

**Note**

The heap implementation acts as a *min-heap*, therefore, the minimum element will be present at the heap's root. In case a *max-heap* is required, simply reverse the logic of this compare function.

**Parameters**

|             |   |
|-------------|---|
| <i>a</i>    | The first object to compare   |
| <i>b</i>    | The second object to compare  |
| <i>data</i> | An arbitrary data pointer passed through from the heap implementation |

**Returns**

A value less than zero if  $a < b$ , a value greater than zero if  $a > b$ , and 0 otherwise

**16.87.2.3 vrna\_callback\_heap\_get\_pos**

```
typedef size_t() vrna_callback_heap_get_pos(const void *a, void *data)
#include <ViennaRNA/datastructures/heap.h>
```

Retrieve the position of a particular heap entry within the heap.

**Parameters**

|             |   |
|-------------|---|
| <i>a</i>    | The object to look-up within the heap                                 |
| <i>data</i> | An arbitrary data pointer passed through from the heap implementation |

**Returns**

The position of the element *a* within the heap, or 0 if it is not in the heap

**16.87.2.4 vrna\_callback\_heap\_set\_pos**

```
typedef void() vrna_callback_heap_set_pos(const void *a, size_t pos, void *data)
#include <ViennaRNA/datastructures/heap.h>
```

Store the position of a particular heap entry within the heap.

**Parameters**

|             |  |
|-------------|--|
| <i>a</i>    | The object whose position shall be stored                                      |
| <i>pos</i>  | The current position of <i>a</i> within the heap, or 0 if <i>a</i> was deleted |
| <i>data</i> | An arbitrary data pointer passed through from the heap implementation          |

**16.87.3 Function Documentation****16.87.3.1 vrna\_heap\_init()**

```
vrna_heap_t vrna_heap_init (
    size_t n,
    vrna_callback_heap_cmp * cmp,
    vrna_callback_heap_get_pos * get_entry_pos,
    vrna_callback_heap_set_pos * set_entry_pos,
    void * data )
#include <ViennaRNA/datastructures/heap.h>
```

Initialize a heap data structure.

This function initializes a heap data structure. The implementation is based on a *min-heap*, i.e. the minimal element is located at the root of the heap. However, by reversing the logic of the compare function, one can easily transform this into a *max-heap* implementation.

Beside the regular operations on a heap data structure, we implement removal and update of arbitrary elements within the heap. For that purpose, however, one requires a reverse-index lookup system that, (i) for a given element stores the current position in the heap, and (ii) allows for fast lookup of an elements current position within the heap. The corresponding getter- and setter- functions may be provided through the arguments `get_entry_pos` and `set_entry_pos`, respectively.

Sometimes, it is difficult to simply compare two data structures without any context. Therefore, the compare function is provided with a user-defined data pointer that can hold any context required.

#### Warning

If any of the arguments `get_entry_pos` or `set_entry_pos` is `NULL`, the operations `vrna_heap_update()` and `vrna_heap_remove()` won't work.

#### See also

[vrna\\_heap\\_free\(\)](#), [vrna\\_heap\\_insert\(\)](#), [vrna\\_heap\\_pop\(\)](#), [vrna\\_heap\\_top\(\)](#), [vrna\\_heap\\_remove\(\)](#), [vrna\\_heap\\_update\(\)](#), [vrna\\_heap\\_t](#), [vrna\\_callback\\_heap\\_cmp](#), [vrna\\_callback\\_heap\\_get\\_pos](#), [vrna\\_callback\\_heap\\_set\\_pos](#)

#### Parameters

|                            |   |
|----------------------------|---|
| <code>n</code>             | The initial size of the heap, i.e. the number of elements to store  |
| <code>cmp</code>           | The address of a compare function that will be used to fullfill the partial order requirement   |
| <code>get_entry_pos</code> | The address of a function that retrieves the position of an element within the heap (or <code>NULL</code> )   |
| <code>set_entry_pos</code> | The address of a function that stores the position of an element within the heap (or <code>NULL</code> )  |
| <code>data</code>          | An arbitrary data pointer passed through to the compare function <code>cmp</code> , and the set/get functions <code>get_entry_pos</code> / <code>set_entry_pos</code> |

#### Returns

An initialized heap data structure, or `NULL` on error

#### 16.87.3.2 vrna\_heap\_free()

```
void vrna_heap_free (
    vrna_heap_t h )
#include <ViennaRNA/datastructures/heap.h>
Free memory occupied by a heap data structure.
```

#### See also

[vrna\\_heap\\_init\(\)](#)

#### Parameters

|                |                                |
|----------------|--------------------------------|
| <code>h</code> | The heap that should be free'd |
|----------------|--------------------------------|

#### 16.87.3.3 vrna\_heap\_size()

```
size_t vrna_heap_size (
    struct vrna_heap_s * h )
```

```
#include <ViennaRNA/datastructures/heap.h>
Get the size of a heap data structure, i.e. the number of stored elements.
```

#### Parameters

|          |                         |
|----------|-------------------------|
| <i>h</i> | The heap data structure |
|----------|-------------------------|

#### Returns

The number of elements currently stored in the heap, or 0 upon any error

### 16.87.3.4 vrna\_heap\_insert()

```
void vrna_heap_insert (
    vrna_heap_t h,
    void * v )
#include <ViennaRNA/datastructures/heap.h>
Insert an element into the heap.
```

#### See also

[vrna\\_heap\\_init\(\)](#), [vrna\\_heap\\_pop\(\)](#), [vrna\\_heap\\_top\(\)](#), [vrna\\_heap\\_free\(\)](#), [vrna\\_heap\\_remove\(\)](#), [vrna\\_heap\\_update\(\)](#)

#### Parameters

|          |  |
|----------|--|
| <i>h</i> | The heap data structure  |
| <i>v</i> | A pointer to the object that is about to be inserted into the heap |

### 16.87.3.5 vrna\_heap\_pop()

```
void * vrna_heap_pop (
    vrna_heap_t h )
#include <ViennaRNA/datastructures/heap.h>
Pop (remove and return) the object at the root of the heap.
This function removes the root from the heap and returns it to the caller.
```

#### See also

[vrna\\_heap\\_init\(\)](#), [vrna\\_heap\\_top\(\)](#), [vrna\\_heap\\_insert\(\)](#), [vrna\\_heap\\_free\(\)](#), [vrna\\_heap\\_remove\(\)](#), [vrna\\_heap\\_update\(\)](#)

#### Parameters

|          |                         |
|----------|-------------------------|
| <i>h</i> | The heap data structure |
|----------|-------------------------|

#### Returns

The object at the root of the heap, i.e. the minimal element (or NULL if (a) the heap is empty or (b) any error occurred)

### 16.87.3.6 vrna\_heap\_top()

```
const void * vrna_heap_top (
    vrna_heap_t h )
```

```
#include <ViennaRNA/datastructures/heap.h>
Get the object at the root of the heap.
```

See also

[vrna\\_heap\\_init\(\)](#), [vrna\\_heap\\_pop\(\)](#), [vrna\\_heap\\_insert\(\)](#), [vrna\\_heap\\_free\(\)](#) [vrna\\_heap\\_remove\(\)](#), [vrna\\_heap\\_update\(\)](#)

#### Parameters

|          |                         |
|----------|-------------------------|
| <i>h</i> | The heap data structure |
|----------|-------------------------|

#### Returns

The object at the root of the heap, i.e. the minimal element (or NULL if (a) the heap is empty or (b) any error occurred)

### 16.87.3.7 vrna\_heap\_remove()

```
void * vrna_heap_remove (
    vrna_heap_t h,
    const void * v )
#include <ViennaRNA/datastructures/heap.h>
```

Remove an arbitrary element within the heap.

See also

[vrna\\_heap\\_init\(\)](#), [vrna\\_callback\\_heap\\_get\\_pos\(\)](#), [vrna\\_callback\\_heap\\_set\\_pos\(\)](#), [vrna\\_heap\\_pop\(\)](#), [vrna\\_heap\\_free\(\)](#)

#### Warning

This function won't work if the heap was not properly initialized with callback functions for fast reverse-index mapping!

#### Parameters

|          |                                    |
|----------|------------------------------------|
| <i>h</i> | The heap data structure            |
| <i>v</i> | The object to remove from the heap |

#### Returns

The object that was removed from the heap (or NULL if (a) it wasn't found or (b) any error occurred)

### 16.87.3.8 vrna\_heap\_update()

```
void * vrna_heap_update (
    vrna_heap_t h,
    void * v )
#include <ViennaRNA/datastructures/heap.h>
```

Update an arbitrary element within the heap.

#### Note

If the object that is to be updated is not currently stored in the heap, it will be inserted. In this case, the function returns NULL.

**Warning**

This function won't work if the heap was not properly initialized with callback functions for fast reverse-index mapping!

**See also**

[vrna\\_heap\\_init\(\)](#), [vrna\\_callback\\_heap\\_get\\_pos\(\)](#), [vrna\\_callback\\_heap\\_set\\_pos\(\)](#) [vrna\\_heap\\_pop\(\)](#), [vrna\\_heap\\_remove\(\)](#), [vrna\\_heap\\_free\(\)](#)

**Parameters**

|          |                         |
|----------|-------------------------|
| <i>h</i> | The heap data structure |
| <i>v</i> | The object to update    |

**Returns**

The 'previous' object within the heap that now got replaced by *v* (or NULL if (a) it wasn't found or (b) any error occurred)

## 16.88 Buffers

Functions that provide dynamically buffered stream-like data structures.

### 16.88.1 Detailed Description

Functions that provide dynamically buffered stream-like data structures.

Collaboration diagram for Buffers:

### Files

- file [char\\_stream.h](#)  
*Implementation of a dynamic, buffered character stream.*
- file [stream\\_output.h](#)  
*An implementation of a buffered, ordered stream output data structure.*

### Typedefs

- **typedef struct vrna\_ordered\_stream\_s \* vrna\_ostream\_t**  
*An ordered output stream structure with unordered insert capabilities.*
- **typedef void() vrna\_callback\_stream\_output**(void \*auxdata, unsigned int i, void \*data)  
*Ordered stream processing callback.*

### Functions

- **vrna\_cstr\_t vrna\_cstr**(size\_t size, FILE \*output)  
*Create a dynamic char \* stream data structure.*
- **void vrna\_cstr\_discard**(struct vrna\_cstr\_s \*buf)  
*Discard the current content of the dynamic char \* stream data structure.*
- **void vrna\_cstr\_free**(vrna\_cstr\_t buf)  
*Free the memory occupied by a dynamic char \* stream data structure.*
- **void vrna\_cstr\_close**(vrna\_cstr\_t buf)  
*Free the memory occupied by a dynamic char \* stream and close the output stream.*
- **void vrna\_cstr\_fflush**(struct vrna\_cstr\_s \*buf)

- Flush the dynamic `char *` output stream.
- `vrna_ostream_t vrna_ostream_init (vrna_callback_stream_output *output, void *auxdata)`  
Get an initialized ordered output stream.
- `void vrna_ostream_free (vrna_ostream_t dat)`  
Free an initialized ordered output stream.
- `void vrna_ostream_request (vrna_ostream_t dat, unsigned int num)`  
Request index in ordered output stream.
- `void vrna_ostream_provide (vrna_ostream_t dat, unsigned int i, void *data)`  
Provide output stream data for a particular index.

## 16.88.2 Typedef Documentation

### 16.88.2.1 `vrna_callback_stream_output`

```
typedef void() vrna_callback_stream_output(void *auxdata, unsigned int i, void *data)
#include <ViennaRNA/datastructures/stream_output.h>
Ordered stream processing callback.
This callback will be processed in sequential order as soon as sequential data in the output stream becomes available.
```

#### Note

The callback must also release the memory occupied by the data passed since the stream will lose any reference to it after the callback has been executed.

#### Parameters

|                      |   |
|----------------------|---|
| <code>auxdata</code> | A shared pointer for all calls, as provided by the second argument to <a href="#">vrna_ostream_init()</a> |
| <code>i</code>       | The index number of the data passed to <code>data</code>  |
| <code>data</code>    | A block of data ready for processing  |

## 16.88.3 Function Documentation

### 16.88.3.1 `vrna_cstr()`

```
vrna_cstr_t vrna_cstr (
    size_t size,
    FILE * output )
#include <ViennaRNA/datastructures/char_stream.h>
Create a dynamic char * stream data structure.
```

#### See also

[vrna\\_cstr\\_free\(\)](#), [vrna\\_cstr\\_close\(\)](#), [vrna\\_cstr\\_fflush\(\)](#), [vrna\\_cstr\\_discard\(\)](#), [vrna\\_cstr\\_printf\(\)](#)

#### Parameters

|                     |   |
|---------------------|---|
| <code>size</code>   | The initial size of the buffer in characters  |
| <code>output</code> | An optional output file stream handle that is used to write the collected data to (defaults to <code>stdout</code> if <code>NULL</code> ) |

### 16.88.3.2 vrna\_cstr\_discard()

```
void vrna_cstr_discard (
    struct vrna_cstr_s * buf )
#include <ViennaRNA/datastructures/char_stream.h>
Discard the current content of the dynamic char * stream data structure.
```

See also

[vrna\\_cstr\\_free\(\)](#), [vrna\\_cstr\\_close\(\)](#), [vrna\\_cstr\\_fflush\(\)](#), [vrna\\_cstr\\_printf\(\)](#)

Parameters

|            |  |
|------------|--|
| <i>buf</i> | The dynamic char * stream data structure to free |
|------------|--|

### 16.88.3.3 vrna\_cstr\_free()

```
void vrna_cstr_free (
    vrna_cstr_t buf )
#include <ViennaRNA/datastructures/char_stream.h>
Free the memory occupied by a dynamic char * stream data structure.
This function first flushes any remaining character data within the stream and then free's the memory occupied by the data structure.
```

See also

[vrna\\_cstr\\_close\(\)](#), [vrna\\_cstr\\_fflush\(\)](#), [vrna\\_cstr\(\)](#)

Parameters

|            |  |
|------------|--|
| <i>buf</i> | The dynamic char * stream data structure to free |
|------------|--|

### 16.88.3.4 vrna\_cstr\_close()

```
void vrna_cstr_close (
    vrna_cstr_t buf )
#include <ViennaRNA/datastructures/char_stream.h>
Free the memory occupied by a dynamic char * stream and close the output stream.
This function first flushes any remaining character data within the stream then closes the attached output file stream (if any), and finally free's the memory occupied by the data structure.
```

See also

[vrna\\_cstr\\_free\(\)](#), [vrna\\_cstr\\_fflush\(\)](#), [vrna\\_cstr\(\)](#)

Parameters

|            |  |
|------------|--|
| <i>buf</i> | The dynamic char * stream data structure to free |
|------------|--|

### 16.88.3.5 vrna\_cstr\_fflush()

```
void vrna_cstr_fflush (
    struct vrna_cstr_s * buf )
```

```
#include <ViennaRNA/datastructures/char_stream.h>
Flush the dynamic char * output stream.
This function flushes the collected char * stream, either by writing to the attached file handle, or simply by writing to
stdout if no file handle has been attached upon construction using vrna_cstr().
```

**Postcondition**

The stream buffer is empty after execution of this function

**See also**

[vrna\\_cstr\(\)](#), [vrna\\_cstr\\_close\(\)](#), [vrna\\_cstr\\_free\(\)](#)

**Parameters**

|            |   |
|------------|---|
| <i>buf</i> | The dynamic char * stream data structure to flush |
|------------|---|

**16.88.3.6 vrna\_ostream\_init()**

```
vrna_ostream_t vrna_ostream_init (
    vrna_callback_stream_output * output,
    void * auxdata )
#include <ViennaRNA/datastructures/stream_output.h>
Get an initialized ordered output stream.
```

**See also**

[vrna\\_ostream\\_free\(\)](#), [vrna\\_ostream\\_request\(\)](#), [vrna\\_ostream\\_provide\(\)](#)

**Parameters**

|                |  |
|----------------|--|
| <i>output</i>  | A callback function that processes and releases data in the stream                 |
| <i>auxdata</i> | A pointer to auxiliary data passed as first argument to the <i>output</i> callback |

**Returns**

An initialized ordered output stream

**16.88.3.7 vrna\_ostream\_free()**

```
void vrna_ostream_free (
    vrna_ostream_t dat )
#include <ViennaRNA/datastructures/stream_output.h>
Free an initialized ordered output stream.
```

**See also**

[vrna\\_ostream\\_init\(\)](#)

**Parameters**

|            |  |
|------------|--|
| <i>dat</i> | The output stream for which occupied memory should be free'd |
|------------|--|

### 16.88.3.8 vrna\_ostream\_request()

```
void vrna_ostream_request (
    vrna_ostream_t dat,
    unsigned int num )
#include <ViennaRNA/datastructures/stream_output.h>
```

Request index in ordered output stream.

This function must be called prior to [vrna\\_ostream\\_provide\(\)](#) to indicate that data associated with a certain index number is expected to be inserted into the stream in the future.

See also

[vrna\\_ostream\\_init\(\)](#), [vrna\\_ostream\\_provide\(\)](#), [vrna\\_ostream\\_free\(\)](#)

#### Parameters

|            |  |
|------------|--|
| <i>dat</i> | The output stream for which the index is requested |
| <i>num</i> | The index to request data for                      |

### 16.88.3.9 vrna\_ostream\_provide()

```
void vrna_ostream_provide (
    vrna_ostream_t dat,
    unsigned int i,
    void * data )
#include <ViennaRNA/datastructures/stream_output.h>
```

Provide output stream data for a particular index.

#### Precondition

The index data is provided for must have been requested using [vrna\\_ostream\\_request\(\)](#) beforehand.

See also

[vrna\\_ostream\\_request\(\)](#)

#### Parameters

|             |  |
|-------------|--|
| <i>dat</i>  | The output stream for which data is provided |
| <i>i</i>    | The index of the provided data               |
| <i>data</i> | The data provided                            |

## 16.89 Deprecated Interface for Global MFE Prediction

### 16.89.1 Detailed Description

Collaboration diagram for Deprecated Interface for Global MFE Prediction:

#### Files

- file [alifold.h](#)  
*Functions for comparative structure prediction using RNA sequence alignments.*
- file [cofold.h](#)  
*MFE implementations for RNA-RNA interaction.*

- file `fold.h`  
*MFE calculations for single RNA sequences.*

## Functions

- float `cofold` (const char \*sequence, char \*structure)  
*Compute the minimum free energy of two interacting RNA molecules.*
- float `cofold_par` (const char \*string, char \*structure, `vrna_param_t` \*parameters, int is\_constrained)  
*Compute the minimum free energy of two interacting RNA molecules.*
- void `free_co_arrays` (void)  
*Free memory occupied by `cofold()`.*
- void `update_cofold_params` (void)  
*Recalculate parameters.*
- void `update_cofold_params_par` (`vrna_param_t` \*parameters)  
*Recalculate parameters.*
- void `export_cofold_arrays_gq` (int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*fc\_p, int \*\*ggg\_p, int \*\*indx\_p, char \*\*ptype\_p)  
*Export the arrays of partition function cofold (with gquadruplex support)*
- void `export_cofold_arrays` (int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*fc\_p, int \*\*indx\_p, char \*\*ptype\_p)  
*Export the arrays of partition function cofold.*
- void `initialize_cofold` (int length)  
*Compute minimum free energy and an appropriate secondary structure of an RNA sequence.*
- float `fold` (const char \*sequence, char \*structure)  
*Compute minimum free energy and an appropriate secondary structure of an RNA sequence.*
- float `circfold` (const char \*sequence, char \*structure)  
*Compute minimum free energy and an appropriate secondary structure of a circular RNA sequence.*
- void `free_arrays` (void)  
*Free arrays for mfe folding.*
- void `update_fold_params` (void)  
*Recalculate energy parameters.*
- void `update_fold_params_par` (`vrna_param_t` \*parameters)  
*Recalculate energy parameters.*
- void `export_fold_arrays` (int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*indx\_p, char \*\*ptype\_p)  
• void `export_fold_arrays_par` (int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*indx\_p, char \*\*ptype\_p, `vrna_param_t` \*\*P\_p)  
• void `export_circfold_arrays` (int \*Fc\_p, int \*FcH\_p, int \*FcI\_p, int \*FcM\_p, int \*\*fM2\_p, int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*indx\_p, char \*\*ptype\_p)  
• void `export_circfold_arrays_par` (int \*Fc\_p, int \*FcH\_p, int \*FcI\_p, int \*FcM\_p, int \*\*fM2\_p, int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*indx\_p, char \*\*ptype\_p, `vrna_param_t` \*\*P\_p)  
• int `LoopEnergy` (int n1, int n2, int type, int type\_2, int si1, int sj1, int sp1, int sq1)  
• int `HairpinE` (int size, int type, int si1, int sj1, const char \*string)  
• void `initialize_fold` (int length)
- float `alifold` (const char \*\*strings, char \*structure)  
*Compute MFE and according consensus structure of an alignment of sequences.*
- float `circalifold` (const char \*\*strings, char \*structure)  
*Compute MFE and according structure of an alignment of sequences assuming the sequences are circular instead of linear.*
- void `free_alifold_arrays` (void)  
*Free the memory occupied by MFE alifold functions.*

## 16.89.2 Function Documentation

### 16.89.2.1 alifold()

```
float alifold (
    const char ** strings,
    char * structure )
#include <ViennaRNA/alifold.h>
```

Compute MFE and according consensus structure of an alignment of sequences.

This function predicts the consensus structure for the aligned 'sequences' and returns the minimum free energy; the mfe structure in bracket notation is returned in 'structure'.

Sufficient space must be allocated for 'structure' before calling [alifold\(\)](#).

**Deprecated** Usage of this function is discouraged! Use [vrna\\_alifold\(\)](#), or [vrna\\_mfe\(\)](#) instead!

#### See also

[vrna\\_alifold\(\)](#), [vrna\\_mfe\(\)](#)

#### Parameters

|                  |   |
|------------------|---|
| <i>strings</i>   | A pointer to a NULL terminated array of character arrays  |
| <i>structure</i> | A pointer to a character array that may contain a constraining consensus structure (will be overwritten by a consensus structure that exhibits the MFE) |

#### Returns

The free energy score in kcal/mol

### 16.89.2.2 cofold()

```
float cofold (
    const char * sequence,
    char * structure )
#include <ViennaRNA/cofold.h>
```

Compute the minimum free energy of two interacting RNA molecules.

The code is analog to the [fold\(\)](#) function. If *cut\_point* == -1 results should be the same as with [fold\(\)](#).

**Deprecated** use [vrna\\_mfe\\_dimer\(\)](#) instead

#### Parameters

|                  |   |
|------------------|---|
| <i>sequence</i>  | The two sequences concatenated                            |
| <i>structure</i> | Will hold the bracket dot structure of the dimer molecule |

#### Returns

minimum free energy of the structure

### 16.89.2.3 cofold\_par()

```
float cofold_par (
```

```

    const char * string,
    char * structure,
    vrna_param_t * parameters,
    int is_constrained )
#include <ViennaRNA/cofold.h>
Compute the minimum free energy of two interacting RNA molecules.
```

**Deprecated** use [vrna\\_mfe\\_dimer\(\)](#) instead

#### 16.89.2.4 free\_co\_arrays()

```

void free_co_arrays (
    void )
#include <ViennaRNA/cofold.h>
Free memory occupied by cofold()
```

**Deprecated** This function will only free memory allocated by a prior call of [cofold\(\)](#) or [cofold\\_par\(\)](#). See [vrna\\_mfe\\_dimer\(\)](#) for how to use the new API

##### Note

folding matrices now reside in the fold compound, and should be free'd there

##### See also

[vrna\\_fc\\_destroy\(\)](#), [vrna\\_mfe\\_dimer\(\)](#)

#### 16.89.2.5 update\_cofold\_params()

```

void update_cofold_params (
    void )
#include <ViennaRNA/cofold.h>
Recalculate parameters.
```

**Deprecated** See [vrna\\_params\\_subst\(\)](#) for an alternative using the new API

#### 16.89.2.6 update\_cofold\_params\_par()

```

void update_cofold_params_par (
    vrna_param_t * parameters )
#include <ViennaRNA/cofold.h>
Recalculate parameters.
```

**Deprecated** See [vrna\\_params\\_subst\(\)](#) for an alternative using the new API

#### 16.89.2.7 export\_cofold\_arrays\_gq()

```

void export_cofold_arrays_gq (
    int ** f5_p,
    int ** c_p,
    int ** fML_p,
    int ** fMI_p,
    int ** fc_p,
```

```

    int ** ggg_p,
    int ** indx_p,
    char ** ptype_p )
#include <ViennaRNA/cofold.h>
Export the arrays of partition function cofold (with gquadruplex support)
Export the cofold arrays for use e.g. in the concentration Computations or suboptimal secondary structure back-
tracking

```

**Deprecated** folding matrices now reside within the fold compound. Thus, this function will only work in conjunction with a prior call to [cofold\(\)](#) or [cofold\\_par\(\)](#)

#### See also

[vrna\\_mfe\\_dimer\(\)](#) for the new API

#### Parameters

|                            |   |
|----------------------------|---|
| <i>f5_p</i>                | A pointer to the 'f5' array, i.e. array containing best free energy in interval [1,j]   |
| <i>c_p</i>                 | A pointer to the 'c' array, i.e. array containing best free energy in interval [i,j] given that i pairs with j                        |
| <i>fML_p</i>               | A pointer to the 'M' array, i.e. array containing best free energy in interval [i,j] for any multiloop segment with at least one stem |
| <i>fM1_p</i>               | A pointer to the 'M1' array, i.e. array containing best free energy in interval [i,j] for multiloop segment with exactly one stem     |
| <i>fc_p</i>                | A pointer to the 'fc' array, i.e. array ...   |
| <i>ggg_p</i>               | A pointer to the 'ggg' array, i.e. array containing best free energy of a gquadruplex delimited by [i,j]                              |
| <i>indx_p</i>              | A pointer to the indexing array used for accessing the energy matrices  |
| <i>ptype<sub>↔</sub>_p</i> | A pointer to the ptype array containing the base pair types for each possibility (i,j)  |

#### 16.89.2.8 `export_cofold_arrays()`

```

void export_cofold_arrays (
    int ** f5_p,
    int ** c_p,
    int ** fML_p,
    int ** fM1_p,
    int ** fc_p,
    int ** indx_p,
    char ** ptype_p )
#include <ViennaRNA/cofold.h>
Export the arrays of partition function cofold.
Export the cofold arrays for use e.g. in the concentration Computations or suboptimal secondary structure back-
tracking

```

**Deprecated** folding matrices now reside within the [vrna\\_fold\\_compound\\_t](#). Thus, this function will only work in conjunction with a prior call to the deprecated functions [cofold\(\)](#) or [cofold\\_par\(\)](#)

#### See also

[vrna\\_mfe\\_dimer\(\)](#) for the new API

#### Parameters

|             |   |
|-------------|---|
| <i>f5_p</i> | A pointer to the 'f5' array, i.e. array containing best free energy in interval [1,j] |
|-------------|---|

## Parameters

|                 |   |
|-----------------|---|
| <i>c_p</i>      | A pointer to the 'c' array, i.e. array containing best free energy in interval [i,j] given that i pairs with j                        |
| <i>fML_p</i>    | A pointer to the 'M' array, i.e. array containing best free energy in interval [i,j] for any multiloop segment with at least one stem |
| <i>fM1_p</i>    | A pointer to the 'M1' array, i.e. array containing best free energy in interval [i,j] for multiloop segment with exactly one stem     |
| <i>fc_p</i>     | A pointer to the 'fc' array, i.e. array ...   |
| <i>indx_p</i>   | A pointer to the indexing array used for accessing the energy matrices  |
| <i>ptype←_p</i> | A pointer to the ptype array containing the base pair types for each possibility (i,j)  |

### 16.89.2.9 initialize\_cofold()

```
void initialize_cofold (
    int length )
#include <ViennaRNA/cofold.h>
allocate arrays for folding
```

**Deprecated** {This function is obsolete and will be removed soon!}

### 16.89.2.10 fold\_par()

```
float fold_par (
    const char * sequence,
    char * structure,
    vrna_param_t * parameters,
    int is_constrained,
    int is_circular )
#include <ViennaRNA/fold.h>
```

Compute minimum free energy and an appropriate secondary structure of an RNA sequence.

The first parameter given, the RNA sequence, must be *uppercase* and should only contain an alphabet  $\Sigma$  that is understood by the RNAlib

(e.g.  $\Sigma = \{A, U, C, G\}$ )

The second parameter, *structure*, must always point to an allocated block of memory with a size of at least `strlen(sequence) + 1`

If the third parameter is NULL, global model detail settings are assumed for the folding recursions. Otherwise, the provided parameters are used.

The fourth parameter indicates whether a secondary structure constraint in enhanced dot-bracket notation is passed through the structure parameter or not. If so, the characters " | x < >" are recognized to mark bases that are paired, unpaired, paired upstream, or downstream, respectively. Matching brackets " ( ) " denote base pairs, dots " ." are used for unconstrained bases.

To indicate that the RNA sequence is circular and thus has to be post-processed, set the last parameter to non-zero. After a successful call of `fold_par()`, a backtracked secondary structure (in dot-bracket notation) that exhibits the minimum of free energy will be written to the memory *structure* is pointing to. The function returns the minimum of free energy for any fold of the sequence given.

#### Note

OpenMP: Passing NULL to the 'parameters' argument involves access to several global model detail variables and thus is not to be considered threadsafe

**Deprecated** use `vrna_mfe()` instead!

## See also

[vrna\\_mfe\(\)](#), [fold\(\)](#), [circfold\(\)](#), [vrna\\_md\\_t](#), [set\\_energy\\_model\(\)](#), [get\\_scaled\\_parameters\(\)](#)

## Parameters

|                       |   |
|-----------------------|---|
| <i>sequence</i>       | RNA sequence  |
| <i>structure</i>      | A pointer to the character array where the secondary structure in dot-bracket notation will be written to                           |
| <i>parameters</i>     | A data structure containing the pre-scaled energy contributions and the model details. (NULL may be passed, see OpenMP notes above) |
| <i>is_constrained</i> | Switch to indicate that a structure constraint is passed via the structure argument (0==off)  |
| <i>is_circular</i>    | Switch to (de-)activate post-processing steps in case RNA sequence is circular (0==off)   |

## Returns

the minimum free energy (MFE) in kcal/mol

**16.89.2.11 fold()**

```
float fold (
    const char * sequence,
    char * structure )
#include <ViennaRNA/fold.h>
Compute minimum free energy and an appropriate secondary structure of an RNA sequence.
This function essentially does the same thing as fold\_par\(\). However, it takes its model details, i.e. temperature,
dangles, tetra\_loop, noGU, no\_closingGU, fold\_constrained, noLonelyPairs from the current global settings within
the library
```

**Deprecated** use [vrna\\_fold\(\)](#), or [vrna\\_mfe\(\)](#) instead!

## See also

[fold\\_par\(\)](#), [circfold\(\)](#)

## Parameters

|                  |   |
|------------------|---|
| <i>sequence</i>  | RNA sequence  |
| <i>structure</i> | A pointer to the character array where the secondary structure in dot-bracket notation will be written to |

## Returns

the minimum free energy (MFE) in kcal/mol

**16.89.2.12 circfold()**

```
float circfold (
    const char * sequence,
    char * structure )
#include <ViennaRNA/fold.h>
Compute minimum free energy and an appropriate secondary structure of a circular RNA sequence.
This function essentially does the same thing as fold\_par\(\). However, it takes its model details, i.e. temperature,
dangles, tetra\_loop, noGU, no\_closingGU, fold\_constrained, noLonelyPairs from the current global settings within
the library
```

**Deprecated** Use [vrna\\_circfold\(\)](#), or [vrna\\_mfe\(\)](#) instead!

See also

[fold\\_par\(\)](#), [circfold\(\)](#)

Parameters

|                  |   |
|------------------|---|
| <i>sequence</i>  | RNA sequence  |
| <i>structure</i> | A pointer to the character array where the secondary structure in dot-bracket notation will be written to |

Returns

the minimum free energy (MFE) in kcal/mol

#### 16.89.2.13 free\_arrays()

```
void free_arrays (
    void )
#include <ViennaRNA/fold.h>
Free arrays for mfe folding.
```

**Deprecated** See [vrna\\_fold\(\)](#), [vrna\\_circfold\(\)](#), or [vrna\\_mfe\(\)](#) and [vrna\\_fold\\_compound\\_t](#) for the usage of the new API!

#### 16.89.2.14 update\_fold\_params()

```
void update_fold_params (
    void )
#include <ViennaRNA/fold.h>
Recalculate energy parameters.
```

**Deprecated** For non-default model settings use the new API with [vrna\\_params\\_subst\(\)](#) and [vrna\\_mfe\(\)](#) instead!

#### 16.89.2.15 update\_fold\_params\_par()

```
void update_fold_params_par (
    vrna_param_t * parameters )
#include <ViennaRNA/fold.h>
Recalculate energy parameters.
```

**Deprecated** For non-default model settings use the new API with [vrna\\_params\\_subst\(\)](#) and [vrna\\_mfe\(\)](#) instead!

#### 16.89.2.16 export\_fold\_arrays()

```
void export_fold_arrays (
    int ** f5_p,
    int ** c_p,
    int ** fML_p,
    int ** fM1_p,
    int ** idx_p,
    char ** ptype_p )
#include <ViennaRNA/fold.h>
```

**Deprecated** See [vrna\\_mfe\(\)](#) and [vrna\\_fold\\_compound\\_t](#) for the usage of the new API!

#### 16.89.2.17 `export_fold_arrays_par()`

```
void export_fold_arrays_par (
    int ** f5_p,
    int ** c_p,
    int ** fML_p,
    int ** fM1_p,
    int ** indx_p,
    char ** ptype_p,
    vrna_param_t ** P_p )
#include <ViennaRNA/fold.h>
```

**Deprecated** See [vrna\\_mfe\(\)](#) and [vrna\\_fold\\_compound\\_t](#) for the usage of the new API!

#### 16.89.2.18 `export_circfold_arrays()`

```
void export_circfold_arrays (
    int * Fc_p,
    int * FcH_p,
    int * FcI_p,
    int * FcM_p,
    int ** fM2_p,
    int ** f5_p,
    int ** c_p,
    int ** fML_p,
    int ** fM1_p,
    int ** indx_p,
    char ** ptype_p )
#include <ViennaRNA/fold.h>
```

**Deprecated** See [vrna\\_mfe\(\)](#) and [vrna\\_fold\\_compound\\_t](#) for the usage of the new API!

#### 16.89.2.19 `export_circfold_arrays_par()`

```
void export_circfold_arrays_par (
    int * Fc_p,
    int * FcH_p,
    int * FcI_p,
    int * FcM_p,
    int ** fM2_p,
    int ** f5_p,
    int ** c_p,
    int ** fML_p,
    int ** fM1_p,
    int ** indx_p,
    char ** ptype_p,
    vrna_param_t ** P_p )
#include <ViennaRNA/fold.h>
```

**Deprecated** See [vrna\\_mfe\(\)](#) and [vrna\\_fold\\_compound\\_t](#) for the usage of the new API!

### 16.89.2.20 LoopEnergy()

```
int LoopEnergy (
    int n1,
    int n2,
    int type,
    int type_2,
    int sil,
    int sjl,
    int spl,
    int sql )
#include <ViennaRNA/fold.h>
```

**Deprecated** {This function is deprecated and will be removed soon. Use [E\\_IntLoop\(\)](#) instead!}

### 16.89.2.21 HairpinE()

```
int HairpinE (
    int size,
    int type,
    int sil,
    int sjl,
    const char * string )
#include <ViennaRNA/fold.h>
```

**Deprecated** {This function is deprecated and will be removed soon. Use [E\\_Hairpin\(\)](#) instead!}

### 16.89.2.22 initialize\_fold()

```
void initialize_fold (
    int length )
#include <ViennaRNA/fold.h>
Allocate arrays for folding
```

**Deprecated** See [vrna\\_mfe\(\)](#) and [vrna\\_fold\\_compound\\_t](#) for the usage of the new API!

### 16.89.2.23 circalifold()

```
float circalifold (
    const char ** strings,
    char * structure )
#include <ViennaRNA/alifold.h>
Compute MFE and according structure of an alignment of sequences assuming the sequences are circular instead of linear.
```

**Deprecated** Usage of this function is discouraged! Use [vrna\\_alicircfold\(\)](#), and [vrna\\_mfe\(\)](#) instead!

#### See also

[vrna\\_alicircfold\(\)](#), [vrna\\_alifold\(\)](#), [vrna\\_mfe\(\)](#)

#### Parameters

|                  |   |
|------------------|---|
| <i>strings</i>   | A pointer to a NULL terminated array of character arrays  |
| <i>structure</i> | A pointer to a character array that may contain a constraining consensus structure (will be overwritten by a consensus structure that exhibits the MFE) |

Generated by Doxygen

**Returns**

The free energy score in kcal/mol

**16.89.2.24 free\_alifold\_arrays()**

```
void free_alifold_arrays (
    void )
#include <ViennaRNA/alifold.h>
Free the memory occupied by MFE alifold functions.
```

**Deprecated** Usage of this function is discouraged! It only affects memory being free'd that was allocated by an old API function before. Release of memory occupied by the newly introduced [vrna\\_fold\\_compound\\_t](#) is handled by [vrna\\_fold\\_compound\\_free\(\)](#)

**See also**

[vrna\\_fold\\_compound\\_free\(\)](#)

**16.90 Deprecated Interface for Local (Sliding Window) MFE Prediction****16.90.1 Detailed Description**

Collaboration diagram for Deprecated Interface for Local (Sliding Window) MFE Prediction:

**Files**

- file [Lfold.h](#)  
*Functions for locally optimal MFE structure prediction.*

**Functions**

- float [Lfold](#) (const char \*string, const char \*structure, int maxdist)  
*The local analog to [fold\(\)](#).*
- float [Lfoldz](#) (const char \*string, const char \*structure, int maxdist, int zsc, double min\_z)

**16.90.2 Function Documentation****16.90.2.1 Lfold()**

```
float Lfold (
    const char * string,
    const char * structure,
    int maxdist )
#include <ViennaRNA/Lfold.h>
```

The local analog to [fold\(\)](#).

Computes the minimum free energy structure including only base pairs with a span smaller than 'maxdist'

**Deprecated** Use [vrna\\_mfe\\_window\(\)](#) instead!

### 16.90.2.2 Lfoldz()

```
float Lfoldz (
    const char * string,
    const char * structure,
    int maxdist,
    int zsc,
    double min_z )
#include <ViennaRNA/Lfold.h>
```

**Deprecated** Use [vrna\\_mfe\\_window\\_zscore\(\)](#) instead!

## 16.91 Deprecated Interface for Global Partition Function Computation

### 16.91.1 Detailed Description

Collaboration diagram for Deprecated Interface for Global Partition Function Computation:

### Files

- file [part\\_func\\_co.h](#)

*Partition function for two RNA sequences.*

### Functions

- float [pf\\_fold\\_par](#) (const char \*sequence, char \*structure, [vrna\\_exp\\_param\\_t](#) \*parameters, int calculate\_←  
bppm, int is\_constrained, int is\_circular)  
*Compute the partition function  $Q$  for a given RNA sequence.*
- float [pf\\_fold](#) (const char \*sequence, char \*structure)  
*Compute the partition function  $Q$  of an RNA sequence.*
- float [pf\\_circ\\_fold](#) (const char \*sequence, char \*structure)  
*Compute the partition function of a circular RNA sequence.*
- void [free\\_pf\\_arrays](#) (void)  
*Free arrays for the partition function recursions.*
- void [update\\_pf\\_params](#) (int length)  
*Recalculate energy parameters.*
- void [update\\_pf\\_params\\_par](#) (int length, [vrna\\_exp\\_param\\_t](#) \*parameters)  
*Recalculate energy parameters.*
- [FLT\\_OR\\_DBL](#) \* [export\\_bppm](#) (void)  
*Get a pointer to the base pair probability array.*
- int [get\\_pf\\_arrays](#) (short \*\*S\_p, short \*\*S1\_p, char \*\*ptype\_p, [FLT\\_OR\\_DBL](#) \*\*qb\_p, [FLT\\_OR\\_DBL](#) \*\*qm←  
\_p, [FLT\\_OR\\_DBL](#) \*\*q1k\_p, [FLT\\_OR\\_DBL](#) \*\*qln\_p)  
*Get the pointers to (almost) all relevant computation arrays used in partition function computation.*
- double [get\\_subseq\\_F](#) (int i, int j)  
*Get the free energy of a subsequence from the  $q[]$  array.*
- double [mean\\_bp\\_distance](#) (int length)  
*Get the mean base pair distance of the last partition function computation.*
- double [mean\\_bp\\_distance\\_pr](#) (int length, [FLT\\_OR\\_DBL](#) \*pr)  
*Get the mean base pair distance in the thermodynamic ensemble.*
- [vrna\\_ep\\_t](#) \* [stackProb](#) (double cutoff)  
*Get the probability of stacks.*
- void [init\\_pf\\_fold](#) (int length)  
*Allocate space for [pf\\_fold\(\)](#)*
- [vrna\\_dimer\\_pf\\_t](#) [co\\_pf\\_fold](#) (char \*sequence, char \*structure)

- Calculate partition function and base pair probabilities.*
- `vrna_dimer_pf_t co_pf_fold_par` (char \*sequence, char \*structure, `vrna_exp_param_t` \*parameters, int calculate\_bppm, int is\_constrained)
 

*Calculate partition function and base pair probabilities.*
  - void `compute_probabilities` (double FAB, double FEA, double FEB, `vrna_ep_t` \*prAB, `vrna_ep_t` \*prA, `vrna_ep_t` \*prB, int Alength)
 

*Compute Boltzmann probabilities of dimerization without homodimers.*
  - void `init_co_pf_fold` (int length)
    - `FLT_OR_DBL * export_co_bppm` (void)
 

*Get a pointer to the base pair probability array.*
    - void `free_co_pf_arrays` (void)
 

*Free the memory occupied by `co_pf_fold()`*
    - void `update_co_pf_params` (int length)
 

*Recalculate energy parameters.*
    - void `update_co_pf_params_par` (int length, `vrna_exp_param_t` \*parameters)
 

*Recalculate energy parameters.*
    - void `assign_plist_from_db` (`vrna_ep_t` \*\*pl, const char \*struc, float pr)
 

*Create a `vrna_ep_t` from a dot-bracket string.*
    - void `assign_plist_from_pr` (`vrna_ep_t` \*\*pl, `FLT_OR_DBL` \*probs, int length, double cutoff)
 

*Create a `vrna_ep_t` from a probability matrix.*
    - float `alipf_fold_par` (const char \*\*sequences, char \*structure, `vrna_ep_t` \*\*pl, `vrna_exp_param_t` \*parameters, int calculate\_bppm, int is\_constrained, int is\_circular)
 

*The partition function version of `alifold()` works in analogy to `pf_fold()`. Pair probabilities and information about sequence covariations are returned via the 'pi' variable as a list of `vrna_pinfo_t` structs. The list is terminated by the first entry with  $pi.i = 0$ .*
    - float `alipf_circ_fold` (const char \*\*sequences, char \*structure, `vrna_ep_t` \*\*pl)
 

*Get a pointer to the base pair probability array.*
    - void `free_alipf_arrays` (void)
 

*Free the memory occupied by folding matrices allocated by `alipf_fold`, `alipf_circ_fold`, etc.*
    - char \* `alipbacktrack` (double \*prob)
 

*Sample a consensus secondary structure from the Boltzmann ensemble according its probability.*
    - int `get_alipf_arrays` (short \*\*\*S\_p, short \*\*\*S5\_p, short \*\*\*S3\_p, unsigned short \*\*\*a2s\_p, char \*\*\*Ss←\_p, `FLT_OR_DBL` \*\*qb\_p, `FLT_OR_DBL` \*\*qm\_p, `FLT_OR_DBL` \*\*q1k\_p, `FLT_OR_DBL` \*\*qln\_p, short \*\*pscore)
 

*Get pointers to (almost) all relevant arrays used in alifold's partition function computation.*

## 16.91.2 Function Documentation

### 16.91.2.1 alipf\_fold\_par()

```
float alipf_fold_par (
    const char ** sequences,
    char * structure,
    vrna_ep_t ** pl,
    vrna_exp_param_t * parameters,
    int calculate_bppm,
    int is_constrained,
    int is_circular )
#include <ViennaRNA/alifold.h>
```

**Deprecated** Use `vrna_pf()` instead

**Parameters**

|                       |  |
|-----------------------|--|
| <i>sequences</i>      |  |
| <i>structure</i>      |  |
| <i>pl</i>             |  |
| <i>parameters</i>     |  |
| <i>calculate_bppm</i> |  |
| <i>is_constrained</i> |  |
| <i>is_circular</i>    |  |

**Returns****16.91.2.2 pf\_fold\_par()**

```
float pf_fold_par (
    const char * sequence,
    char * structure,
    vrna_exp_param_t * parameters,
    int calculate_bppm,
    int is_constrained,
    int is_circular )
#include <ViennaRNA/part_func.h>
```

Compute the partition function  $Q$  for a given RNA sequence.

If *structure* is not a NULL pointer on input, it contains on return a string consisting of the letters ". , | { } ( )" denoting bases that are essentially unpaired, weakly paired, strongly paired without preference, weakly upstream (downstream) paired, or strongly up- (down-)stream paired bases, respectively. If *fold\_constrained* is not 0, the *structure* string is interpreted on input as a list of constraints for the folding. The character "x" marks bases that must be unpaired, matching brackets "( )" denote base pairs, all other characters are ignored. Any pairs conflicting with the constraint will be forbidden. This is usually sufficient to ensure the constraints are honored. If the parameter *calculate\_bppm* is set to 0 base pairing probabilities will not be computed (saving CPU time), otherwise after calculations took place *pr* will contain the probability that bases *i* and *j* pair.

**Deprecated** Use [vrna\\_pf\(\)](#) instead

**Note**

The global array *pr* is deprecated and the user who wants the calculated base pair probabilities for further computations is advised to use the function [export\\_bppm\(\)](#)

**Postcondition**

After successful run the hidden folding matrices are filled with the appropriate Boltzmann factors. Depending on whether the global variable [do\\_backtrack](#) was set the base pair probabilities are already computed and may be accessed for further usage via the [export\\_bppm\(\)](#) function. A call of [free\\_pf\\_arrays\(\)](#) will free all memory allocated by this function. Successive calls will first free previously allocated memory before starting the computation.

**See also**

[vrna\\_pf\(\)](#), [bppm\\_to\\_structure\(\)](#), [export\\_bppm\(\)](#), [vrna\\_exp\\_params\(\)](#), [free\\_pf\\_arrays\(\)](#)

**Parameters**

|    |                 |                        |
|----|-----------------|------------------------|
| in | <i>sequence</i> | The RNA sequence input |
|----|-----------------|------------------------|

## Parameters

|         |                       |   |
|---------|-----------------------|---|
| in, out | <i>structure</i>      | A pointer to a char array where a base pair probability information can be stored in a pseudo-dot-bracket notation (may be NULL, too) |
| in      | <i>parameters</i>     | Data structure containing the precalculated Boltzmann factors   |
| in      | <i>calculate_bppm</i> | Switch to Base pair probability calculations on/off (0==off)  |
| in      | <i>is_constrained</i> | Switch to indicate that a structure constraint is passed via the structure argument (0==off)  |
| in      | <i>is_circular</i>    | Switch to (de-)activate postprocessing steps in case RNA sequence is circular (0==off)  |

## Returns

The ensemble free energy  $G = -RT \cdot \log(Q)$  in kcal/mol

### 16.91.2.3 pf\_fold()

```
float pf_fold (
    const char * sequence,
    char * structure )
#include <ViennaRNA/part_func.h>
```

Compute the partition function  $Q$  of an RNA sequence.

If *structure* is not a NULL pointer on input, it contains on return a string consisting of the letters ". , | { } ( ) " denoting bases that are essentially unpaired, weakly paired, strongly paired without preference, weakly upstream (downstream) paired, or strongly up- (down-)stream paired bases, respectively. If *fold\_constrained* is not 0, the *structure* string is interpreted on input as a list of constraints for the folding. The character "x" marks bases that must be unpaired, matching brackets "( )" denote base pairs, all other characters are ignored. Any pairs conflicting with the constraint will be forbidden. This is usually sufficient to ensure the constraints are honored. If *do\_backtrack* has been set to 0 base pairing probabilities will not be computed (saving CPU time), otherwise *pr* will contain the probability that bases *i* and *j* pair.

#### Note

The global array *pr* is deprecated and the user who wants the calculated base pair probabilities for further computations is advised to use the function [export\\_bppm\(\)](#).

**OpenMP:** This function is not entirely threadsafe. While the recursions are working on their own copies of data the model details for the recursions are determined from the global settings just before entering the recursions. Consider using [pf\\_fold\\_par\(\)](#) for a really threadsafe implementation.

#### Precondition

This function takes its model details from the global variables provided in *RNAlib*

#### Postcondition

After successful run the hidden folding matrices are filled with the appropriate Boltzmann factors. Depending on whether the global variable *do\_backtrack* was set the base pair probabilities are already computed and may be accessed for further usage via the [export\\_bppm\(\)](#) function. A call of [free\\_pf\\_arrays\(\)](#) will free all memory allocated by this function. Successive calls will first free previously allocated memory before starting the computation.

#### See also

[pf\\_fold\\_par\(\)](#), [pf\\_circ\\_fold\(\)](#), [bppm\\_to\\_structure\(\)](#), [export\\_bppm\(\)](#)

**Parameters**

|                  |   |
|------------------|---|
| <i>sequence</i>  | The RNA sequence input  |
| <i>structure</i> | A pointer to a char array where a base pair probability information can be stored in a pseudo-dot-bracket notation (may be NULL, too) |

**Returns**

The ensemble free energy  $G = -RT \cdot \log(Q)$  in kcal/mol

**16.91.2.4 pf\_circ\_fold()**

```
float pf_circ_fold (
    const char * sequence,
    char * structure )
#include <ViennaRNA/part_func.h>
```

Compute the partition function of a circular RNA sequence.

**Note**

The global array `pr` is deprecated and the user who wants the calculated base pair probabilities for further computations is advised to use the function `export_bppm()`.

**OpenMP:** This function is not entirely threadsafe. While the recursions are working on their own copies of data the model details for the recursions are determined from the global settings just before entering the recursions. Consider using `pf_fold_par()` for a really threadsafe implementation.

**Precondition**

This function takes its model details from the global variables provided in *RNALib*

**Postcondition**

After successful run the hidden folding matrices are filled with the appropriate Boltzmann factors. Depending on whether the global variable `do_backtrack` was set the base pair probabilities are already computed and may be accessed for further usage via the `export_bppm()` function. A call of `free_pf_arrays()` will free all memory allocated by this function. Successive calls will first free previously allocated memory before starting the computation.

**See also**

[vrna\\_pf\(\)](#)

**Deprecated** Use [vrna\\_pf\(\)](#) instead!

**Parameters**

|               |                  |   |
|---------------|------------------|---|
| <i>in</i>     | <i>sequence</i>  | The RNA sequence input  |
| <i>in,out</i> | <i>structure</i> | A pointer to a char array where a base pair probability information can be stored in a pseudo-dot-bracket notation (may be NULL, too) |

**Returns**

The ensemble free energy  $G = -RT \cdot \log(Q)$  in kcal/mol

### 16.91.2.5 free\_pf\_arrays()

```
void free_pf_arrays (
    void )
#include <ViennaRNA/part_func.h>
```

Free arrays for the partition function recursions.

Call this function if you want to free all allocated memory associated with the partition function forward recursion.

#### Note

Successive calls of [pf\\_fold\(\)](#), [pf\\_circ\\_fold\(\)](#) already check if they should free any memory from a previous run.

#### OpenMP notice:

This function should be called before leaving a thread in order to avoid leaking memory

**Deprecated** See [vrna\\_fold\\_compound\\_t](#) and its related functions for how to free memory occupied by the dynamic programming matrices

#### Postcondition

All memory allocated by [pf\\_fold\\_par\(\)](#), [pf\\_fold\(\)](#) or [pf\\_circ\\_fold\(\)](#) will be free'd

#### See also

[pf\\_fold\\_par\(\)](#), [pf\\_fold\(\)](#), [pf\\_circ\\_fold\(\)](#)

### 16.91.2.6 update\_pf\_params()

```
void update_pf_params (
    int length )
#include <ViennaRNA/part_func.h>
```

Recalculate energy parameters.

Call this function to recalculate the pair matrix and energy parameters after a change in folding parameters like temperature

**Deprecated** Use [vrna\\_exp\\_params\\_subst\(\)](#) instead

### 16.91.2.7 update\_pf\_params\_par()

```
void update_pf_params_par (
    int length,
    vrna_exp_param_t * parameters )
#include <ViennaRNA/part_func.h>
```

Recalculate energy parameters.

**Deprecated** Use [vrna\\_exp\\_params\\_subst\(\)](#) instead

### 16.91.2.8 export\_bppm()

```
FLOAT_OR_DOUBLE * export_bppm (
    void )
#include <ViennaRNA/part_func.h>
```

Get a pointer to the base pair probability array.

Accessing the base pair probabilities for a pair (i,j) is achieved by

```
FLOAT_OR_DOUBLE *pr = export_bppm();
pr[ij] = pr[iindex[i]-j];
```

**Precondition**

Call [pf\\_fold\\_par\(\)](#), [pf\\_fold\(\)](#) or [pf\\_circ\\_fold\(\)](#) first to fill the base pair probability array

**See also**

[pf\\_fold\(\)](#), [pf\\_circ\\_fold\(\)](#), [vrna\\_idx\\_row\\_wise\(\)](#)

**Returns**

A pointer to the base pair probability array

**16.91.2.9 get\_pf\_arrays()**

```
int get_pf_arrays (
    short ** S_p,
    short ** S1_p,
    char ** ptype_p,
    FLT_OR_DBL ** qb_p,
    FLT_OR_DBL ** qm_p,
    FLT_OR_DBL ** q1k_p,
    FLT_OR_DBL ** qln_p )
#include <ViennaRNA/part_func.h>
```

Get the pointers to (almost) all relevant computation arrays used in partition function computation.

**Precondition**

In order to assign meaningful pointers, you have to call [pf\\_fold\\_par\(\)](#) or [pf\\_fold\(\)](#) first!

**See also**

[pf\\_fold\\_par\(\)](#), [pf\\_fold\(\)](#), [pf\\_circ\\_fold\(\)](#)

**Parameters**

|     |                |  |
|-----|----------------|--|
| out | <i>S_p</i>     | A pointer to the 'S' array (integer representation of nucleotides)           |
| out | <i>S1_p</i>    | A pointer to the 'S1' array (2nd integer representation of nucleotides)      |
| out | <i>ptype_p</i> | A pointer to the pair type matrix  |
| out | <i>qb_p</i>    | A pointer to the Q <sup>B</sup> matrix                                       |
| out | <i>qm_p</i>    | A pointer to the Q <sup>M</sup> matrix                                       |
| out | <i>q1k_p</i>   | A pointer to the 5' slice of the Q matrix ( <i>q1k(k)</i> = <i>Q(1, k)</i> ) |
| out | <i>qln_p</i>   | A pointer to the 3' slice of the Q matrix ( <i>qln(l)</i> = <i>Q(l, n)</i> ) |

**Returns**

Non Zero if everything went fine, 0 otherwise

**16.91.2.10 mean\_bp\_distance()**

```
double mean_bp_distance (
    int length )
```

```
#include <ViennaRNA/part_func.h>
```

Get the mean base pair distance of the last partition function computation.

**Deprecated** Use `vrna_mean_bp_distance()` or `vrna_mean_bp_distance_pr()` instead!

#### See also

`vrna_mean_bp_distance()`, `vrna_mean_bp_distance_pr()`

#### Parameters

|                     |  |
|---------------------|--|
| <code>length</code> |  |
|---------------------|--|

#### Returns

mean base pair distance in thermodynamic ensemble

### 16.91.2.11 `mean_bp_distance_pr()`

```
double mean_bp_distance_pr (
    int length,
    FLT_OR_DBL * pr )
#include <ViennaRNA/part_func.h>
Get the mean base pair distance in the thermodynamic ensemble.
This is a threadsafe implementation of mean_bp_dist() !
< $d$ > =  $\sum_{a,b} p_a p_b d(S_a, S_b)$ 
this can be computed from the pair probs  $p_{ij}$  as
< $d$ > =  $\sum_{ij} p_{ij}(1 - p_{ij})$ 
```

**Deprecated** Use `vrna_mean_bp_distance()` or `vrna_mean_bp_distance_pr()` instead!

#### Parameters

|                     |   |
|---------------------|---|
| <code>length</code> | The length of the sequence                        |
| <code>pr</code>     | The matrix containing the base pair probabilities |

#### Returns

The mean pair distance of the structure ensemble

### 16.91.2.12 `stackProb()`

```
vrna_ep_t * stackProb (
    double cutoff )
#include <ViennaRNA/part_func.h>
Get the probability of stacks.
```

**Deprecated** Use `vrna_stack_prob()` instead!

### 16.91.2.13 `init_pf_fold()`

```
void init_pf_fold (
    int length )
#include <ViennaRNA/part_func.h>
Allocate space for pf_fold()
```

**Deprecated** This function is obsolete and will be removed soon!

#### 16.91.2.14 co\_pf\_fold()

```
vrna_dimer_pf_t co_pf_fold (
    char * sequence,
    char * structure )
#include <ViennaRNA/part_func_co.h>
```

Calculate partition function and base pair probabilities.

This is the cofold partition function folding. The second molecule starts at the `cut_point` nucleotide.

##### Note

OpenMP: Since this function relies on the global parameters `do_backtrack`, `dangles`, `temperature` and `pf_scale` it is not threadsafe according to concurrent changes in these variables! Use `co_pf_fold_par()` instead to circumvent this issue.

**Deprecated** {Use `vrna_pf_dimer()` instead!}

##### Parameters

|                        |  |
|------------------------|--|
| <code>sequence</code>  | Concatenated RNA sequences             |
| <code>structure</code> | Will hold the structure or constraints |

##### Returns

`vrna_dimer_pf_t` structure containing a set of energies needed for concentration computations.

#### 16.91.2.15 co\_pf\_fold\_par()

```
vrna_dimer_pf_t co_pf_fold_par (
    char * sequence,
    char * structure,
    vrna_exp_param_t * parameters,
    int calculate_bppm,
    int is_constrained )
#include <ViennaRNA/part_func_co.h>
```

Calculate partition function and base pair probabilities.

This is the cofold partition function folding. The second molecule starts at the `cut_point` nucleotide.

**Deprecated** Use `vrna_pf_dimer()` instead!

##### See also

`get_boltzmann_factors()`, `co_pf_fold()`

##### Parameters

|                             |  |
|-----------------------------|--|
| <code>sequence</code>       | Concatenated RNA sequences   |
| <code>structure</code>      | Pointer to the structure constraint  |
| <code>parameters</code>     | Data structure containing the precalculated Boltzmann factors                                |
| <code>calculate_bppm</code> | Switch to turn Base pair probability calculations on/off (0==off)                            |
| <code>is_constrained</code> | Switch to indicate that a structure constraint is passed via the structure argument (0==off) |

**Returns**

`vrna_dimer_pf_t` structure containing a set of energies needed for concentration computations.

**16.91.2.16 compute\_probabilities()**

```
void compute_probabilities (
    double FAB,
    double FEA,
    double FEB,
    vrna_ep_t * prAB,
    vrna_ep_t * prA,
    vrna_ep_t * prB,
    int Alength )
```

```
#include <ViennaRNA/part_func_co.h>
```

Compute Boltzmann probabilities of dimerization without homodimers.

Given the pair probabilities and free energies (in the null model) for a dimer AB and the two constituent monomers A and B, compute the conditional pair probabilities given that a dimer AB actually forms. Null model pair probabilities are given as a list as produced by [assign plist from pr\(\)](#), the dimer probabilities 'prAB' are modified in place.

**Deprecated** { Use [vrna\\_pf\\_dimer\\_probs\(\)](#) instead!}

**Parameters**

|                |                              |
|----------------|------------------------------|
| <i>FAB</i>     | free energy of dimer AB      |
| <i>FEA</i>     | free energy of monomer A     |
| <i>FEB</i>     | free energy of monomer B     |
| <i>prAB</i>    | pair probabilities for dimer |
| <i>prA</i>     | pair probabilities monomer   |
| <i>prB</i>     | pair probabilities monomer   |
| <i>Alength</i> | Length of molecule A         |

**16.91.2.17 init\_co\_pf\_fold()**

```
void init_co_pf_fold (
    int length )
#include <ViennaRNA/part_func_co.h>
```

DO NOT USE THIS FUNCTION ANYMORE

**Deprecated** { This function is deprecated and will be removed soon!}

**16.91.2.18 export\_co\_bppm()**

```
FLT_OR_DBL * export_co_bppm (
    void )
#include <ViennaRNA/part_func_co.h>
```

Get a pointer to the base pair probability array.

Accessing the base pair probabilities for a pair (i,j) is achieved by

```
FLT_OR_DBL *pr = export_bppm(); pr_ij = pr[iindx[i]-j];
```

**Deprecated** This function is deprecated and will be removed soon! The base pair probability array is available through the [vrna\\_fold\\_compound\\_t](#) data structure, and its associated [vrna\\_mx\\_pf\\_t](#) member.

See also

[vrna\\_idx\\_row\\_wise\(\)](#)

Returns

A pointer to the base pair probability array

#### 16.91.2.19 free\_co\_pf\_arrays()

```
void free_co_pf_arrays (
    void )
#include <ViennaRNA/part_func_co.h>
Free the memory occupied by co\_pf\_fold\(\)
```

**Deprecated** This function will be removed for the new API soon! See [vrna\\_pf\\_dimer\(\)](#), [vrna\\_fold\\_compound\(\)](#), and [vrna\\_fold\\_compound\\_free\(\)](#) for an alternative

#### 16.91.2.20 update\_co\_pf\_params()

```
void update_co_pf_params (
    int length )
#include <ViennaRNA/part_func_co.h>
Recalculate energy parameters.
This function recalculates all energy parameters given the current model settings.
```

**Deprecated** Use [vrna\\_exp\\_params\\_subst\(\)](#) instead!

Parameters

|               |                                    |
|---------------|------------------------------------|
| <i>length</i> | Length of the current RNA sequence |
|---------------|------------------------------------|

#### 16.91.2.21 update\_co\_pf\_params\_par()

```
void update_co_pf_params_par (
    int length,
    vrna_exp_param_t * parameters )
#include <ViennaRNA/part_func_co.h>
Recalculate energy parameters.
```

This function recalculates all energy parameters given the current model settings. Its second argument can either be NULL or a data structure containing the precomputed Boltzmann factors. In the first scenario, the necessary data structure will be created automatically according to the current global model settings, i.e. this mode might not be threadsafe. However, if the provided data structure is not NULL, threadsafety for the model parameters [dangles](#), [pf\\_scale](#) and [temperature](#) is regained, since their values are taken from this data structure during subsequent calculations.

**Deprecated** Use [vrna\\_exp\\_params\\_subst\(\)](#) instead!

Parameters

|                   |   |
|-------------------|---|
| <i>length</i>     | Length of the current RNA sequence                          |
| <i>parameters</i> | data structure containing the precomputed Boltzmann factors |

### 16.91.2.22 assign\_plist\_from\_db()

```
void assign_plist_from_db (
    vrna_ep_t ** pl,
    const char * struc,
    float pr )
#include <ViennaRNA/utils/structures.h>
```

Create a `vrna_ep_t` from a dot-bracket string.

The dot-bracket string is parsed and for each base pair an entry in the plist is created. The probability of each pair in the list is set by a function parameter.

The end of the plist is marked by sequence positions i as well as j equal to 0. This condition should be used to stop looping over its entries

**Deprecated** Use `vrna_plist()` instead

#### Parameters

|                    |   |
|--------------------|---|
| <code>pl</code>    | A pointer to the <code>vrna_ep_t</code> that is to be created |
| <code>struc</code> | The secondary structure in dot-bracket notation               |
| <code>pr</code>    | The probability for each base pair                            |

### 16.91.2.23 assign\_plist\_from\_pr()

```
void assign_plist_from_pr (
    vrna_ep_t ** pl,
    FLT_OR_DBL * probs,
    int length,
    double cutoff )
#include <ViennaRNA/utils/structures.h>
```

Create a `vrna_ep_t` from a probability matrix.

The probability matrix given is parsed and all pair probabilities above the given threshold are used to create an entry in the plist

The end of the plist is marked by sequence positions i as well as j equal to 0. This condition should be used to stop looping over its entries

#### Note

This function is threadsafe

**Deprecated** Use `vrna_plist_from_probs()` instead!

#### Parameters

|                  |                     |   |
|------------------|---------------------|---|
| <code>out</code> | <code>pl</code>     | A pointer to the <code>vrna_ep_t</code> that is to be created |
| <code>in</code>  | <code>probs</code>  | The probability matrix used for creating the plist            |
| <code>in</code>  | <code>length</code> | The length of the RNA sequence                                |
| <code>in</code>  | <code>cutoff</code> | The cutoff value  |

### 16.91.2.24 alipf\_fold()

```
float alipf_fold (
```

```

    const char ** sequences,
    char * structure,
    vrna_ep_t ** pl )
#include <ViennaRNA/alifold.h>

```

The partition function version of `alifold()` works in analogy to `pf_fold()`. Pair probabilities and information about sequence covariations are returned via the 'pi' variable as a list of `vrna_pinfo_t` structs. The list is terminated by the first entry with `pi.i = 0`.

**Deprecated** Use `vrna_pf()` instead

#### Parameters

|                        |  |
|------------------------|--|
| <code>sequences</code> |  |
| <code>structure</code> |  |
| <code>pl</code>        |  |

#### Returns

#### 16.91.2.25 alipf\_circ\_fold()

```

float alipf_circ_fold (
    const char ** sequences,
    char * structure,
    vrna_ep_t ** pl )
#include <ViennaRNA/alifold.h>

```

**Deprecated** Use `vrna_pf()` instead

#### Parameters

|                        |  |
|------------------------|--|
| <code>sequences</code> |  |
| <code>structure</code> |  |
| <code>pl</code>        |  |

#### Returns

#### 16.91.2.26 export\_ali\_bppm()

```

FLT_OR_DBL * export_ali_bppm (
    void )
#include <ViennaRNA/alifold.h>
Get a pointer to the base pair probability array.
Accessing the base pair probabilities for a pair (i,j) is achieved by
FLT_OR_DBL *pr = export_bppm(); pr_ij = pr[iindx[i]-j];

```

**Deprecated** Usage of this function is discouraged! The new `vrna_fold_compound_t` allows direct access to the folding matrices, including the pair probabilities! The pair probability array returned here reflects the one of the latest call to `vrna_pf()`, or any of the old API calls for consensus structure partition function folding.

## See also

[vrna\\_fold\\_compound\\_t](#), [vrna\\_fold\\_compound\\_comparative\(\)](#), and [vrna\\_pf\(\)](#)

## Returns

A pointer to the base pair probability array

**16.91.2.27 free\_alipf\_arrays()**

```
void free_alipf_arrays (
    void )
```

```
#include <ViennaRNA/alifold.h>
```

Free the memory occupied by folding matrices allocated by alipf\_fold, alipf\_circ\_fold, etc.

**Deprecated** Usage of this function is discouraged! This function only free's memory allocated by old API function calls. Memory allocated by any of the new API calls (starting with vrna\_) will be not affected!

## See also

[vrna\\_fold\\_compound\\_t](#), [vrna\\_vrna\\_fold\\_compound\\_free\(\)](#)

**16.91.2.28 alipbacktrack()**

```
char * alipbacktrack (
    double * prob )
```

```
#include <ViennaRNA/alifold.h>
```

Sample a consensus secondary structure from the Boltzmann ensemble according its probability.

**Deprecated** Use [vrna\\_pbacktrack\(\)](#) instead!

## Parameters

|             |                         |
|-------------|-------------------------|
| <i>prob</i> | to be described (berni) |
|-------------|-------------------------|

## Returns

A sampled consensus secondary structure in dot-bracket notation

**16.91.2.29 get\_alipf\_arrays()**

```
int get_alipf_arrays (
    short *** S_p,
    short *** S5_p,
    short *** S3_p,
    unsigned short *** a2s_p,
    char *** Ss_p,
    FLT_OR_DBL ** qb_p,
    FLT_OR_DBL ** qm_p,
    FLT_OR_DBL ** qlk_p,
    FLT_OR_DBL ** qln_p,
    short ** pscore )
```

```
#include <ViennaRNA/alifold.h>
```

Get pointers to (almost) all relevant arrays used in alifold's partition function computation.

**Note**

To obtain meaningful pointers, call `alipf_fold` first!

**See also**

`pf_alifold()`, `alipf_circ_fold()`

**Deprecated** It is discouraged to use this function! The new `vrna_fold_compound_t` allows direct access to all necessary consensus structure prediction related variables!

**See also**

`vrna_fold_compound_t`, `vrna_fold_compound_comparative()`, `vrna_pf()`

**Parameters**

|                     |  |
|---------------------|--|
| <code>S_p</code>    | A pointer to the 'S' array (integer representation of nucleotides)   |
| <code>S5_p</code>   | A pointer to the 'S5' array  |
| <code>S3_p</code>   | A pointer to the 'S3' array  |
| <code>a2s_p</code>  | A pointer to the alignment-column to sequence position mapping array |
| <code>Ss_p</code>   | A pointer to the 'Ss' array  |
| <code>qb_p</code>   | A pointer to the $Q^B$ matrix  |
| <code>qm_p</code>   | A pointer to the $Q^M$ matrix  |
| <code>q1k_p</code>  | A pointer to the 5' slice of the Q matrix ( $q1k(k) = Q(1, k)$ )     |
| <code>qln_p</code>  | A pointer to the 3' slice of the Q matrix ( $qln(l) = Q(l, n)$ )     |
| <code>pscore</code> | A pointer to the start of a pscore list                              |

**Returns**

Non Zero if everything went fine, 0 otherwise

## 16.92 Deprecated Interface for Local (Sliding Window) Partition Function Computation

### 16.92.1 Detailed Description

Collaboration diagram for Deprecated Interface for Local (Sliding Window) Partition Function Computation:

### Files

- file `LPfold.h`

*Partition function and equilibrium probability implementation for the sliding window algorithm.*

### Functions

- void `update_pf_paramsLP` (int length)
- `vrna_ep_t * pfl_fold` (char \*sequence, int winSize, int pairSize, float cutoffb, double \*\*pU, `vrna_ep_t` \*\*dpp2, FILE \*pUfp, FILE \*spup)
 

*Compute partition functions for locally stable secondary structures.*
- `vrna_ep_t * pfl_fold_par` (char \*sequence, int winSize, int pairSize, float cutoffb, double \*\*pU, `vrna_ep_t` \*\*dpp2, FILE \*pUfp, FILE \*spup, `vrna_exp_param_t` \*parameters)

- Compute partition functions for locally stable secondary structures.*
- void **putoutpU\_prob** (double \*\*pU, int length, int ulength, FILE \*fp, int energies)  
*Writes the unpaired probabilities (pU) or opening energies into a file.*
  - void **putoutpU\_prob\_bin** (double \*\*pU, int length, int ulength, FILE \*fp, int energies)  
*Writes the unpaired probabilities (pU) or opening energies into a binary file.*

## 16.92.2 Function Documentation

### 16.92.2.1 update\_pf\_paramsLP()

```
void update_pf_paramsLP (
    int length )
#include <ViennaRNA/LPfold.h>
```

#### Parameters

|        |  |
|--------|--|
| length |  |
|--------|--|

### 16.92.2.2 pfl\_fold()

```
vrna_ep_t * pfl_fold (
    char * sequence,
    int winSize,
    int pairSize,
    float cutoffb,
    double ** pU,
    vrna_ep_t ** dpp2,
    FILE * pUfp,
    FILE * spup )
#include <ViennaRNA/LPfold.h>
```

Compute partition functions for locally stable secondary structures.

pfl\_fold computes partition functions for every window of size 'winSize' possible in a RNA molecule, allowing only pairs with a span smaller than 'pairSize'. It returns the mean pair probabilities averaged over all windows containing the pair in 'pl'. 'winSize' should always be  $\geq$  'pairSize'. Note that in contrast to [Lfold\(\)](#), bases outside of the window do not influence the structure at all. Only probabilities higher than 'cutoffb' are kept.

If 'pU' is supplied (i.e is not the NULL pointer), [pfl\\_fold\(\)](#) will also compute the mean probability that regions of length 'u' and smaller are unpaired. The parameter 'u' is supplied in 'pup[0][0]'. On return the 'pup' array will contain these probabilities, with the entry on 'pup[x][y]' containing the mean probability that x and the y-1 preceding bases are unpaired. The 'pU' array needs to be large enough to hold n+1 float\* entries, where n is the sequence length.  
If an array dpp2 is supplied, the probability of base pair (i,j) given that there already exists a base pair (i+1,j-1) is also computed and saved in this array. If pUfp is given (i.e. not NULL), pU is not saved but put out imediately. If spup is given (i.e. is not NULL), the pair probabilities in pl are not saved but put out imediately.

#### Parameters

|          |  |
|----------|--|
| sequence | RNA sequence                             |
| winSize  | size of the window                       |
| pairSize | maximum size of base pair                |
| cutoffb  | cutoffb for base pairs                   |
| pU       | array holding all unpaired probabilities |
| dpp2     | array of dependent pair probabilities    |
| pUfp     | file pointer for pU                      |
| spup     | file pointer for pair probabilities      |

**Returns**

list of pair probabilities

**16.92.2.3 putoutpU\_prob()**

```
void putoutpU_prob (
    double ** pU,
    int length,
    int ulength,
    FILE * fp,
    int energies )
#include <ViennaRNA/LPfold.h>
```

Writes the unpaired probabilities (pU) or opening energies into a file.

Can write either the unpaired probabilities (accessibilities) pU or the opening energies -log(pU)kT into a file

**Parameters**

|                 |                                       |
|-----------------|---------------------------------------|
| <i>pU</i>       | pair probabilities                    |
| <i>length</i>   | length of RNA sequence                |
| <i>ulength</i>  | maximum length of unpaired stretch    |
| <i>fp</i>       | file pointer of destination file      |
| <i>energies</i> | switch to put out as opening energies |

**16.92.2.4 putoutpU\_prob\_bin()**

```
void putoutpU_prob_bin (
    double ** pU,
    int length,
    int ulength,
    FILE * fp,
    int energies )
#include <ViennaRNA/LPfold.h>
```

Writes the unpaired probabilities (pU) or opening energies into a binary file.

Can write either the unpaired probabilities (accessibilities) pU or the opening energies -log(pU)kT into a file

**Parameters**

|                 |                                       |
|-----------------|---------------------------------------|
| <i>pU</i>       | pair probabilities                    |
| <i>length</i>   | length of RNA sequence                |
| <i>ulength</i>  | maximum length of unpaired stretch    |
| <i>fp</i>       | file pointer of destination file      |
| <i>energies</i> | switch to put out as opening energies |

**16.93 Deprecated Interface for Stochastic Backtracking****16.93.1 Detailed Description**

Collaboration diagram for Deprecated Interface for Stochastic Backtracking:

## Functions

- `char * pbacktrack (char *sequence)`  
*Sample a secondary structure from the Boltzmann ensemble according its probability.*
- `char * pbacktrack5 (char *sequence, int length)`  
*Sample a sub-structure from the Boltzmann ensemble according its probability.*
- `char * pbacktrack_circ (char *sequence)`  
*Sample a secondary structure of a circular RNA from the Boltzmann ensemble according its probability.*

## Variables

- `int st_back`  
*Flag indicating that auxiliary arrays are needed throughout the computations. This is essential for stochastic backtracking.*

### 16.93.2 Function Documentation

#### 16.93.2.1 pbacktrack()

```
char * pbacktrack (
    char * sequence )
#include <ViennaRNA/part_func.h>
Sample a secondary structure from the Boltzmann ensemble according its probability.
```

##### Precondition

`st_back` has to be set to 1 before calling `pf_fold()` or `pf_fold_par()`  
`pf_fold_par()` or `pf_fold()` have to be called first to fill the partition function matrices

##### Parameters

|                       |                  |
|-----------------------|------------------|
| <code>sequence</code> | The RNA sequence |
|-----------------------|------------------|

##### Returns

A sampled secondary structure in dot-bracket notation

#### 16.93.2.2 pbacktrack\_circ()

```
char * pbacktrack_circ (
    char * sequence )
#include <ViennaRNA/part_func.h>
Sample a secondary structure of a circular RNA from the Boltzmann ensemble according its probability.
This function does the same as pbacktrack() but assumes the RNA molecule to be circular
```

##### Precondition

`st_back` has to be set to 1 before calling `pf_fold()` or `pf_fold_par()`  
`pf_fold_par()` or `pf_circ_fold()` have to be called first to fill the partition function matrices

**Deprecated** Use `vrna_pbacktrack()` instead.

**Parameters**

|                       |                  |
|-----------------------|------------------|
| <code>sequence</code> | The RNA sequence |
|-----------------------|------------------|

**Returns**

A sampled secondary structure in dot-bracket notation

### 16.93.3 Variable Documentation

#### 16.93.3.1 `st_back`

```
int st_back [extern]
#include <ViennaRNA/part_func.h>
```

Flag indicating that auxiliary arrays are needed throughout the computations. This is essential for stochastic backtracking.

Set this variable to 1 prior to a call of `pf_fold()` to ensure that all matrices needed for stochastic backtracking are filled in the forward recursions

**Deprecated** set the `uniq_ML` flag in `vrna_md_t` before passing it to `vrna_fold_compound()`.

**See also**

`pbacktrack()`, `pbacktrack_circ`

## 16.94 Deprecated Interface for Multiple Sequence Alignment Utilities

### 16.94.1 Detailed Description

Collaboration diagram for Deprecated Interface for Multiple Sequence Alignment Utilities:

#### Typedefs

- `typedef struct vrna_pinfo_s pair_info`  
*Old typename of `vrna_pinfo_s`.*

#### Functions

- `int get_mpi (char *Alseq[], int n_seq, int length, int *mini)`  
*Get the mean pairwise identity in steps from ?to?(ident)*
- `void encode_ali_sequence (const char *sequence, short *S, short *S5, short *S3, char *ss, unsigned short *as, int circ)`  
*Get arrays with encoded sequence of the alignment.*
- `void alloc_sequence_arrays (const char **sequences, short ***S, short ***S5, short ***S3, unsigned short ***a2s, char ***Ss, int circ)`  
*Allocate memory for sequence array used to deal with aligned sequences.*
- `void free_sequence_arrays (unsigned int n_seq, short ***S, short ***S5, short ***S3, unsigned short ***a2s, char ***Ss)`  
*Free the memory of the sequence arrays used to deal with aligned sequences.*

### 16.94.2 Typedef Documentation

### 16.94.2.1 pair\_info

```
typedef struct vrna_pinfo_s pair_info
#include <ViennaRNA/utils/alignments.h>
Old typename of vrna_pinfo_s.
```

**Deprecated** Use [vrna\\_pinfo\\_t](#) instead!

## 16.94.3 Function Documentation

### 16.94.3.1 get\_mpi()

```
int get_mpi (
    char * Alseq[],
    int n_seq,
    int length,
    int * mini )
#include <ViennaRNA/utils/alignments.h>
Get the mean pairwise identity in steps from ?to?(ident)
```

**Deprecated** Use [vrna\\_aln\\_mpi\(\)](#) as a replacement

#### Parameters

|               |  |
|---------------|--|
| <i>Alseq</i>  |  |
| <i>n_seq</i>  | The number of sequences in the alignment |
| <i>length</i> | The length of the alignment              |
| <i>mini</i>   |  |

#### Returns

The mean pairwise identity

### 16.94.3.2 encode\_ali\_sequence()

```
void encode_ali_sequence (
    const char * sequence,
    short * S,
    short * s5,
    short * s3,
    char * ss,
    unsigned short * as,
    int circ )
#include <ViennaRNA/utils/alignments.h>
Get arrays with encoded sequence of the alignment.
this function assumes that in S, S5, s3, ss and as enough space is already allocated (size must be at least sequence length+2)
```

#### Parameters

|                 |   |
|-----------------|---|
| <i>sequence</i> | The gapped sequence from the alignment                                  |
| <i>S</i>        | pointer to an array that holds encoded sequence                         |
| <i>s5</i>       | pointer to an array that holds the next base 5' of alignment position i |
| <i>s3</i>       | pointer to an array that holds the next base 3' of alignment position i |

## Parameters

|             |  |
|-------------|--|
| <i>ss</i>   |  |
| <i>as</i>   |  |
| <i>circ</i> | assume the molecules to be circular instead of linear (circ=0) |

**16.94.3.3 alloc\_sequence\_arrays()**

```
void alloc_sequence_arrays (
    const char ** sequences,
    short *** S,
    short *** S5,
    short *** S3,
    unsigned short *** a2s,
    char *** Ss,
    int circ )
```

```
#include <ViennaRNA/utils/alignments.h>
```

Allocate memory for sequence array used to deal with aligned sequences.

Note that these arrays will also be initialized according to the sequence alignment given

## See also

[free\\_sequence\\_arrays\(\)](#)

## Parameters

|                  |  |
|------------------|--|
| <i>sequences</i> | The aligned sequences  |
| <i>S</i>         | A pointer to the array of encoded sequences  |
| <i>S5</i>        | A pointer to the array that contains the next 5' nucleotide of a sequence position |
| <i>S3</i>        | A pointer to the array that contains the next 3' nucleotide of a sequence position |
| <i>a2s</i>       | A pointer to the array that contains the alignment to sequence position mapping    |
| <i>Ss</i>        | A pointer to the array that contains the ungapped sequence                         |
| <i>circ</i>      | assume the molecules to be circular instead of linear (circ=0)                     |

**16.94.3.4 free\_sequence\_arrays()**

```
void free_sequence_arrays (
    unsigned int n_seq,
    short *** S,
    short *** S5,
    short *** S3,
    unsigned short *** a2s,
    char *** Ss )
```

```
#include <ViennaRNA/utils/alignments.h>
```

Free the memory of the sequence arrays used to deal with aligned sequences.

This function frees the memory previously allocated with [alloc\\_sequence\\_arrays\(\)](#)

## See also

[alloc\\_sequence\\_arrays\(\)](#)

## Parameters

|                    |  |
|--------------------|--|
| <code>n_seq</code> | The number of aligned sequences  |
| <code>S</code>     | A pointer to the array of encoded sequences  |
| <code>S5</code>    | A pointer to the array that contains the next 5' nucleotide of a sequence position |
| <code>S3</code>    | A pointer to the array that contains the next 3' nucleotide of a sequence position |
| <code>a2s</code>   | A pointer to the array that contains the alignment to sequence position mapping    |
| <code>Ss</code>    | A pointer to the array that contains the ungapped sequence                         |

## 16.95 Deprecated Interface for Secondary Structure Utilities

### 16.95.1 Detailed Description

Collaboration diagram for Deprecated Interface for Secondary Structure Utilities:

#### Files

- file [RNAsruct.h](#)

*Parsing and Coarse Graining of Structures.*

#### Functions

- `char * b2HIT (const char *structure)`

*Converts the full structure from bracket notation to the HIT notation including root.*
- `char * b2C (const char *structure)`

*Converts the full structure from bracket notation to the a coarse grained notation using the 'H' 'B' 'I' 'M' and 'R' identifiers.*
- `char * b2Shapiro (const char *structure)`

*Converts the full structure from bracket notation to the weighted coarse grained notation using the 'H' 'B' 'I' 'M' 'S' 'E' and 'R' identifiers.*
- `char * add_root (const char *structure)`

*Adds a root to an un-rooted tree in any except bracket notation.*
- `char * expand_Shapiro (const char *coarse)`

*Inserts missing 'S' identifiers in unweighted coarse grained structures as obtained from `b2C()`.*
- `char * expand_Full (const char *structure)`

*Convert the full structure from bracket notation to the expanded notation including root.*
- `char * unexpand_Full (const char *ffull)`

*Restores the bracket notation from an expanded full or HIT tree, that is any tree using only identifiers 'U' 'P' and 'R'.*
- `char * unweight (const char *wcoarse)`

*Strip weights from any weighted tree.*
- `void unexpand_aligned_F (char *align[2])`

*Converts two aligned structures in expanded notation.*
- `void parse_structure (const char *structure)`

*Collects a statistic of structure elements of the full structure in bracket notation.*
- `char * pack_structure (const char *struc)`

*Pack secondary secondary structure, 5:1 compression using base 3 encoding.*
- `char * unpack_structure (const char *packed)`

*Unpack secondary structure previously packed with `pack_structure()`*
- `short * make_pair_table (const char *structure)`

*Create a pair table of a secondary structure.*
- `short * copy_pair_table (const short *pt)`

- short \* **alimake\_pair\_table** (const char \*structure)
  - Get an exact copy of a pair table.
- short \* **make\_pair\_table\_snoop** (const char \*structure)
- int **bp\_distance** (const char \*str1, const char \*str2)
  - Compute the "base pair" distance between two secondary structures s1 and s2.
- unsigned int \* **make\_referenceBP\_array** (short \*reference\_pt, unsigned int turn)
  - Make a reference base pair count matrix.
- unsigned int \* **compute\_BPdifferences** (short \*pt1, short \*pt2, unsigned int turn)
  - Make a reference base pair distance matrix.
- void **parenthesis\_structure** (char \*structure, vrna\_bp\_stack\_t \*bp, int length)
  - Create a dot-backet/parenthesis structure from backtracking stack.
- void **parenthesis\_zuker** (char \*structure, vrna\_bp\_stack\_t \*bp, int length)
  - Create a dot-backet/parenthesis structure from backtracking stack obtained by zuker suboptimal calculation in cofold.c.
- void **bppm\_to\_structure** (char \*structure, FLT\_OR\_DBLE \*pr, unsigned int length)
  - Create a dot-bracket like structure string from base pair probability matrix.
- char **bppm\_symbol** (const float \*x)
  - Get a pseudo dot bracket notation for a given probability information.

## Variables

- int **loop\_size** [2000]
  - contains a list of all loop sizes. `loop_size[0]` contains the number of external bases.
- int **helix\_size** [2000]
  - contains a list of all stack sizes.
- int **loop\_degree** [2000]
  - contains the corresponding list of loop degrees.
- int **loops**
  - contains the number of loops ( and therefore of stacks ).
- int **unpaired**
  - contains the number of unpaired bases.
- int **pairs**
  - contains the number of base pairs in the last parsed structure.

## 16.95.2 Function Documentation

### 16.95.2.1 b2HIT()

```
char * b2HIT (
    const char * structure )
#include <ViennaRNA/RNAstruct.h>
Converts the full structure from bracket notation to the HIT notation including root.
```

**Deprecated** See [vrna\\_db\\_to\\_tree\\_string\(\)](#) and [VRNA\\_STRUCTURE\\_TREE\\_HIT](#) for a replacement

#### Parameters

|                        |                      |
|------------------------|----------------------|
| <code>structure</code> | <input type="text"/> |
|------------------------|----------------------|

Returns

### 16.95.2.2 b2C()

```
char * b2C (
    const char * structure )
#include <ViennaRNA/RNAstruct.h>
```

Converts the full structure from bracket notation to the a coarse grained notation using the 'H' 'B' 'I' 'M' and 'R' identifiers.

**Deprecated** See [vrna\\_db\\_to\\_tree\\_string\(\)](#) and [VRNA\\_STRUCTURE\\_TREE\\_SHAPIRO\\_SHORT](#) for a replacement

Parameters

|                  |                      |
|------------------|----------------------|
| <i>structure</i> | <input type="text"/> |
|------------------|----------------------|

Returns

### 16.95.2.3 b2Shapiro()

```
char * b2Shapiro (
    const char * structure )
#include <ViennaRNA/RNAstruct.h>
```

Converts the full structure from bracket notation to the *weighted* coarse grained notation using the 'H' 'B' 'I' 'M' 'S' 'E' and 'R' identifiers.

**Deprecated** See [vrna\\_db\\_to\\_tree\\_string\(\)](#) and [VRNA\\_STRUCTURE\\_TREE\\_SHAPIRO\\_WEIGHT](#) for a replacement

Parameters

|                  |                      |
|------------------|----------------------|
| <i>structure</i> | <input type="text"/> |
|------------------|----------------------|

Returns

### 16.95.2.4 add\_root()

```
char * add_root (
    const char * structure )
#include <ViennaRNA/RNAstruct.h>
```

Adds a root to an un-rooted tree in any except bracket notation.

Parameters

|                  |                      |
|------------------|----------------------|
| <i>structure</i> | <input type="text"/> |
|------------------|----------------------|

Returns

#### 16.95.2.5 expand\_Shapiro()

```
char * expand_Shapiro (
    const char * coarse )
```

```
#include <ViennaRNA/RNAstruct.h>
```

Inserts missing 'S' identifiers in unweighted coarse grained structures as obtained from [b2C\(\)](#).

Parameters

|        |                      |
|--------|----------------------|
| coarse | <input type="text"/> |
|--------|----------------------|

Returns

#### 16.95.2.6 expand\_Full()

```
char * expand_Full (
    const char * structure )
```

```
#include <ViennaRNA/RNAstruct.h>
```

Convert the full structure from bracket notation to the expanded notation including root.

Parameters

|           |                      |
|-----------|----------------------|
| structure | <input type="text"/> |
|-----------|----------------------|

Returns

#### 16.95.2.7 unexpand\_Full()

```
char * unexpand_Full (
    const char * ffull )
```

```
#include <ViennaRNA/RNAstruct.h>
```

Restores the bracket notation from an expanded full or HIT tree, that is any tree using only identifiers 'U' 'P' and 'R'.

Parameters

|       |                      |
|-------|----------------------|
| ffull | <input type="text"/> |
|-------|----------------------|

Returns

#### 16.95.2.8 unweight()

```
char * unweight (
    const char * wcoarse )
```

```
#include <ViennaRNA/RNAstruct.h>
Strip weights from any weighted tree.
```

#### Parameters

|         |                          |
|---------|--------------------------|
| wcoarse | <input type="checkbox"/> |
|---------|--------------------------|

#### Returns

### 16.95.2.9 unexpand\_aligned\_F()

```
void unexpand_aligned_F (
    char * align[2] )
#include <ViennaRNA/RNAstruct.h>
Converts two aligned structures in expanded notation.
Takes two aligned structures as produced by tree_edit_distance() function back to bracket notation with '_' as the gap character. The result overwrites the input.
```

#### Parameters

|       |                          |
|-------|--------------------------|
| align | <input type="checkbox"/> |
|-------|--------------------------|

### 16.95.2.10 parse\_structure()

```
void parse_structure (
    const char * structure )
#include <ViennaRNA/RNAstruct.h>
Collects a statistic of structure elements of the full structure in bracket notation.
The function writes to the following global variables: loop_size, loop_degree, helix_size, loops, pairs, unpaired
```

#### Parameters

|           |                          |
|-----------|--------------------------|
| structure | <input type="checkbox"/> |
|-----------|--------------------------|

### 16.95.2.11 pack\_structure()

```
char * pack_structure (
    const char * struc )
#include <ViennaRNA/utils/structures.h>
Pack secondary secondary structure, 5:1 compression using base 3 encoding.
Returns a binary string encoding of the secondary structure using a 5:1 compression scheme. The string is NULL terminated and can therefore be used with standard string functions such as strcmp(). Useful for programs that need to keep many structures in memory.
```

**Deprecated** Use [vrna\\_db\\_pack\(\)](#) as a replacement

#### Parameters

|       |   |
|-------|---|
| struc | The secondary structure in dot-bracket notation |
|-------|---|

**Returns**

The binary encoded structure

**16.95.2.12 `unpack_structure()`**

```
char * unpack_structure (
    const char * packed )
#include <ViennaRNA/utils/structures.h>
Unpack secondary structure previously packed with pack\_structure\(\)
Translate a compressed binary string produced by pack\_structure\(\) back into the familiar dot-bracket notation.
```

**Deprecated** Use [vrna\\_db\\_unpack\(\)](#) as a replacement

**Parameters**

|               |   |
|---------------|---|
| <i>packed</i> | The binary encoded packed secondary structure |
|---------------|---|

**Returns**

The unpacked secondary structure in dot-bracket notation

**16.95.2.13 `make_pair_table()`**

```
short * make_pair_table (
    const char * structure )
#include <ViennaRNA/utils/structures.h>
Create a pair table of a secondary structure.
Returns a newly allocated table, such that table[i]=j if (i,j) pair or 0 if i is unpaired, table[0] contains the length of the structure.
```

**Deprecated** Use [vrna\\_ptable\(\)](#) instead

**Parameters**

|                  |   |
|------------------|---|
| <i>structure</i> | The secondary structure in dot-bracket notation |
|------------------|---|

**Returns**

A pointer to the created pair\_table

**16.95.2.14 `copy_pair_table()`**

```
short * copy_pair_table (
    const short * pt )
#include <ViennaRNA/utils/structures.h>
Get an exact copy of a pair table.
```

**Deprecated** Use [vrna\\_ptable\\_copy\(\)](#) instead

**Parameters**

|           |                             |
|-----------|-----------------------------|
| <i>pt</i> | The pair table to be copied |
|-----------|-----------------------------|

**Returns**

A pointer to the copy of 'pt'

**16.95.2.15 alimake\_pair\_table()**

```
short * alimake_pair_table (
    const char * structure )
#include <ViennaRNA/utils/structures.h>
Pair table for snoop align
```

**Deprecated** Use [vrna\\_pt\\_ali\\_get\(\)](#) instead!

**16.95.2.16 make\_pair\_table\_snoop()**

```
short * make_pair_table_snoop (
    const char * structure )
#include <ViennaRNA/utils/structures.h>
returns a newly allocated table, such that: table[i]=j if (i,j) pair or 0 if i is unpaired, table[0] contains the length of the structure. The special pseudoknotted H/ACA-mRNA structure is taken into account.
```

**Deprecated** Use [vrna\\_pt\\_snoop\\_get\(\)](#) instead!

**16.95.2.17 bp\_distance()**

```
int bp_distance (
    const char * str1,
    const char * str2 )
#include <ViennaRNA/utils/structures.h>
Compute the "base pair" distance between two secondary structures s1 and s2.
The sequences should have the same length. dist = number of base pairs in one structure but not in the other same as edit distance with open-pair close-pair as move-set
```

**Deprecated** Use [vrna\\_bp\\_distance](#) instead

**Parameters**

|             |  |
|-------------|--|
| <i>str1</i> | First structure in dot-bracket notation  |
| <i>str2</i> | Second structure in dot-bracket notation |

**Returns**

The base pair distance between str1 and str2

**16.95.2.18 make\_referenceBP\_array()**

```
unsigned int * make_referenceBP_array (
```

```

    short * reference_pt,
    unsigned int turn )
#include <ViennaRNA/utils/structures.h>
Make a reference base pair count matrix.
Get an upper triangular matrix containing the number of basepairs of a reference structure for each interval [i,j] with i<j. Access it via iindx!!!

```

**Deprecated** Use `vrna_refBPCnt_matrix()` instead

#### 16.95.2.19 compute\_BPdifferences()

```

unsigned int * compute_BPdifferences (
    short * pt1,
    short * pt2,
    unsigned int turn )
#include <ViennaRNA/utils/structures.h>
Make a reference base pair distance matrix.
Get an upper triangular matrix containing the base pair distance of two reference structures for each interval [i,j] with i<j. Access it via iindx!!!

```

**Deprecated** Use `vrna_refBPdist_matrix()` instead

#### 16.95.2.20 parenthesis\_structure()

```

void parenthesis_structure (
    char * structure,
    vrna_bp_stack_t * bp,
    int length )
#include <ViennaRNA/utils/structures.h>
Create a dot-bracket/parenthesis structure from backtracking stack.

```

**Deprecated** use `vrna_parenthesis_structure()` instead

##### Note

This function is threadsafe

#### 16.95.2.21 parenthesis\_zuker()

```

void parenthesis_zuker (
    char * structure,
    vrna_bp_stack_t * bp,
    int length )
#include <ViennaRNA/utils/structures.h>
Create a dot-bracket/parenthesis structure from backtracking stack obtained by zuker suboptimal calculation in cofold.c.

```

**Deprecated** use `vrna_parenthesis_zuker` instead

##### Note

This function is threadsafe

### 16.95.2.22 bppm\_to\_structure()

```
void bppm_to_structure (
    char * structure,
    FLT_OR_DBL * pr,
    unsigned int length )
#include <ViennaRNA/utils/structures.h>
```

Create a dot-bracket like structure string from base pair probability matrix.

**Deprecated** Use [vrna\\_db\\_from\\_probs\(\)](#) instead!

### 16.95.2.23 bppm\_symbol()

```
char bppm_symbol (
    const float * x )
#include <ViennaRNA/utils/structures.h>
```

Get a pseudo dot bracket notation for a given probability information.

**Deprecated** Use [vrna\\_bpp\\_symbol\(\)](#) instead!

## 16.96 Deprecated Interface for Plotting Utilities

### 16.96.1 Detailed Description

Collaboration diagram for Deprecated Interface for Plotting Utilities:

### Data Structures

- struct [COORDINATE](#)

*this is a workarround for the SWIG Perl Wrapper RNA plot function that returns an array of type [COORDINATE](#) More...*

### Functions

- int [PS\\_color\\_aln](#) (const char \*structure, const char \*filename, const char \*seqs[], const char \*names[])
 

*Produce PostScript sequence alignment color-annotated by consensus structure.*
- int [aliPS\\_color\\_aln](#) (const char \*structure, const char \*filename, const char \*seqs[], const char \*names[])
 

*PS\_color\_aln for duplexes.*
- int [simple\\_xy\\_coordinates](#) (short \*pair\_table, float \*X, float \*Y)
 

*Calculate nucleotide coordinates for secondary structure plot the Simple way*
- int [simple\\_circplot\\_coordinates](#) (short \*pair\_table, float \*x, float \*y)
 

*Calculate nucleotide coordinates for Circular Plot*

### Variables

- int [rna\\_plot\\_type](#)

*Switch for changing the secondary structure layout algorithm.*

### 16.96.2 Data Structure Documentation

#### 16.96.2.1 struct COORDINATE

this is a workarround for the SWIG Perl Wrapper RNA plot function that returns an array of type [COORDINATE](#)

### 16.96.3 Function Documentation

### 16.96.3.1 PS\_color\_aln()

```
int PS_color_aln (
    const char * structure,
    const char * filename,
    const char * seqs[],
    const char * names[] )
#include <ViennaRNA/plotting/alignments.h>
Produce PostScript sequence alignment color-annotated by consensus structure.
```

**Deprecated** Use [vrna\\_file\\_PS\\_aln\(\)](#) instead!

### 16.96.3.2 aliPS\_color\_aln()

```
int aliPS_color_aln (
    const char * structure,
    const char * filename,
    const char * seqs[],
    const char * names[] )
#include <ViennaRNA/plotting/alignments.h>
PS_color_aln for duplexes.
```

**Deprecated** Use [vrna\\_file\\_PS\\_aln\(\)](#) instead!

### 16.96.3.3 simple\_xy\_coordinates()

```
int simple_xy_coordinates (
    short * pair_table,
    float * X,
    float * Y )
#include <ViennaRNA/plotting/layouts.h>
Calculate nucleotide coordinates for secondary structure plot the Simple way
```

**See also**

[make\\_pair\\_table\(\)](#), [rna\\_plot\\_type](#), [simple\\_circplot\\_coordinates\(\)](#), [naview\\_xy\\_coordinates\(\)](#), [vrna\\_file\\_PS\\_rnaplot\\_a\(\)](#), [vrna\\_file\\_PS\\_rnaplot](#), [svg\\_rna\\_plot\(\)](#)

**Deprecated** Consider switching to [vrna\\_plot\\_coords\\_simple\\_pt\(\)](#) instead!

#### Parameters

|                   |   |
|-------------------|---|
| <i>pair_table</i> | The pair table of the secondary structure                                   |
| <i>X</i>          | a pointer to an array with enough allocated space to hold the x coordinates |
| <i>Y</i>          | a pointer to an array with enough allocated space to hold the y coordinates |

#### Returns

length of sequence on success, 0 otherwise

### 16.96.3.4 simple\_circplot\_coordinates()

```
int simple_circplot_coordinates (
    short * pair_table,
```

```

    float * x,
    float * y )
#include <ViennaRNA/plotting/layouts.h>
Calculate nucleotide coordinates for Circular Plot

```

This function calculates the coordinates of nucleotides mapped in equal distancies onto a unit circle.

#### Note

In order to draw nice arcs using quadratic bezier curves that connect base pairs one may calculate a second tangential point  $P^t$  in addition to the actual  $R^2$  coordinates. the simplest way to do so may be to compute a radius scaling factor  $rs$  in the interval  $[0, 1]$  that weights the proportion of base pair span to the actual length of the sequence. This scaling factor can then be used to calculate the coordinates for  $P^t$ , i.e.  $P_x^t[i] = X[i] * rs$  and  $P_y^t[i] = Y[i] * rs$ .

#### See also

[make\\_pair\\_table\(\)](#), [rna\\_plot\\_type](#), [simple\\_xy\\_coordinates\(\)](#), [naview\\_xy\\_coordinates\(\)](#), [vrna\\_file\\_PS\\_rnplot\\_a\(\)](#), [vrna\\_file\\_PS\\_rnplot](#), [svg\\_rna\\_plot\(\)](#)

**Deprecated** Consider switching to [vrna\\_plot\\_coords\\_circular\\_pt\(\)](#) instead!

#### Parameters

|                   |   |
|-------------------|---|
| <i>pair_table</i> | The pair table of the secondary structure                                   |
| <i>x</i>          | a pointer to an array with enough allocated space to hold the x coordinates |
| <i>y</i>          | a pointer to an array with enough allocated space to hold the y coordinates |

#### Returns

length of sequence on success, 0 otherwise

### 16.96.4 Variable Documentation

#### 16.96.4.1 rna\_plot\_type

```

int rna_plot_type [extern]
#include <ViennaRNA/plotting/layouts.h>
Switch for changing the secondary structure layout algorithm.

```

Current possibility are 0 for a simple radial drawing or 1 for the modified radial drawing taken from the *naview* program of [5].

#### Note

To provide thread safety please do not rely on this global variable in future implementations but pass a plot type flag directly to the function that decides which layout algorithm it may use!

#### See also

[VRNA\\_PLOT\\_TYPE\\_SIMPLE](#), [VRNA\\_PLOT\\_TYPE\\_NAVIEW](#), [VRNA\\_PLOT\\_TYPE\\_CIRCULAR](#)

## 16.97 Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers

### 16.97.1 Detailed Description

Collaboration diagram for Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers:

## Typedefs

- `typedef struct vrna_path_s path_t`

*Old typename of vrna\_path\_s.*

## Functions

- `int find_saddle (const char *seq, const char *s1, const char *s2, int width)`  
*Find energy of a saddle point between 2 structures (search only direct path)*
- `void free_path (vrna_path_t *path)`  
*Free memory allocated by get\_path() function.*
- `vrna_path_t * get_path (const char *seq, const char *s1, const char *s2, int width)`  
*Find refolding path between 2 structures (search only direct path)*

### 16.97.2 Typedef Documentation

#### 16.97.2.1 path\_t

```
typedef struct vrna_path_s path_t
#include <ViennaRNA/landscape/paths.h>
Old typename of vrna_path_s.
```

**Deprecated** Use `vrna_path_t` instead!

### 16.97.3 Function Documentation

#### 16.97.3.1 find\_saddle()

```
int find_saddle (
    const char * seq,
    const char * s1,
    const char * s2,
    int width )
#include <ViennaRNA/landscape/findpath.h>
Find energy of a saddle point between 2 structures (search only direct path)
```

**Deprecated** Use `vrna_path_findpath_saddle()` instead!

#### Parameters

|                    |  |
|--------------------|--|
| <code>seq</code>   | RNA sequence   |
| <code>s1</code>    | A pointer to the character array where the first secondary structure in dot-bracket notation will be written to  |
| <code>s2</code>    | A pointer to the character array where the second secondary structure in dot-bracket notation will be written to |
| <code>width</code> | integer how many strutures are being kept during the search  |

#### Returns

the saddle energy in 10cal/mol

### 16.97.3.2 free\_path()

```
void free_path (
    vrna_path_t * path )
#include <ViennaRNA/landscape/findpath.h>
Free memory allocated by get\_path\(\) function.
```

**Deprecated** Use [vrna\\_path\\_free\(\)](#) instead!

#### Parameters

|             |                               |
|-------------|-------------------------------|
| <i>path</i> | pointer to memory to be freed |
|-------------|-------------------------------|

### 16.97.3.3 get\_path()

```
vrna_path_t * get_path (
    const char * seq,
    const char * s1,
    const char * s2,
    int width )
#include <ViennaRNA/landscape/findpath.h>
Find refolding path between 2 structures (search only direct path)
```

**Deprecated** Use [vrna\\_path\\_findpath\(\)](#) instead!

#### Parameters

|              |  |
|--------------|--|
| <i>seq</i>   | RNA sequence   |
| <i>s1</i>    | A pointer to the character array where the first secondary structure in dot-bracket notation will be written to  |
| <i>s2</i>    | A pointer to the character array where the second secondary structure in dot-bracket notation will be written to |
| <i>width</i> | integer how many strutures are being kept during the search  |

#### Returns

direct refolding path between two structures

## Chapter 17

# Data Structure Documentation

### 17.1 \_struct\_en Struct Reference

Data structure for [energy\\_of\\_move\(\)](#)

#### 17.1.1 Detailed Description

Data structure for [energy\\_of\\_move\(\)](#)

The documentation for this struct was generated from the following file:

- ViennaRNA/move\_set.h

### 17.2 LIST Struct Reference

Collaboration diagram for LIST:

The documentation for this struct was generated from the following file:

- ViennaRNA/datastructures/lists.h

### 17.3 LST\_BUCKET Struct Reference

Collaboration diagram for LST\_BUCKET:

The documentation for this struct was generated from the following file:

- ViennaRNA/datastructures/lists.h

### 17.4 Postorder\_list Struct Reference

Postorder data structure.

#### 17.4.1 Detailed Description

Postorder data structure.

The documentation for this struct was generated from the following file:

- ViennaRNA/dist\_vars.h

### 17.5 swString Struct Reference

Some other data structure.

### 17.5.1 Detailed Description

Some other data structure.

The documentation for this struct was generated from the following file:

- ViennaRNA/[dist\\_vars.h](#)

## 17.6 Tree Struct Reference

[Tree](#) data structure.

Collaboration diagram for Tree:

### 17.6.1 Detailed Description

[Tree](#) data structure.

The documentation for this struct was generated from the following file:

- ViennaRNA/[dist\\_vars.h](#)

## 17.7 TwoDpfold\_vars Struct Reference

Variables compound for 2Dfold partition function folding.

Collaboration diagram for TwoDpfold\_vars:

### Data Fields

- **char \* ptype**  
*Precomputed array of pair types.*
- **char \* sequence**  
*The input sequence*
- **short \* S1**  
*The input sequences in numeric form.*
- **unsigned int maxD1**  
*Maximum allowed base pair distance to first reference.*
- **unsigned int maxD2**  
*Maximum allowed base pair distance to second reference.*
- **int \* my\_iindx**  
*Index for moving in quadratic distancy dimensions.*
- **int \* jindx**  
*Index for moving in the triangular matrix qm1.*
- **unsigned int \* referenceBPs1**  
*Matrix containing number of basepairs of reference structure1 in interval [i,j].*
- **unsigned int \* referenceBPs2**  
*Matrix containing number of basepairs of reference structure2 in interval [i,j].*
- **unsigned int \* bpdist**  
*Matrix containing base pair distance of reference structure 1 and 2 on interval [i,j].*
- **unsigned int \* mm1**  
*Maximum matching matrix, reference struct 1 disallowed.*
- **unsigned int \* mm2**  
*Maximum matching matrix, reference struct 2 disallowed.*

### 17.7.1 Detailed Description

Variables compound for 2Dfold partition function folding.

**Deprecated** This data structure will be removed from the library soon! Use [vrna\\_fold\\_compound\\_t](#) and the corresponding functions [vrna\\_fold\\_compound\\_TwoD\(\)](#), [vrna\\_pf\\_TwoD\(\)](#), and [vrna\\_fold\\_compound\\_free\(\)](#) instead!

The documentation for this struct was generated from the following file:

- ViennaRNA/[2Dpfold.h](#)

## 17.8 vrna\_dimer\_conc\_s Struct Reference

Data structure for concentration dependency computations.

### Data Fields

- double **Ac\_start**  
*start concentration A*
- double **Bc\_start**  
*start concentration B*
- double **ABc**  
*End concentration AB.*

### 17.8.1 Detailed Description

Data structure for concentration dependency computations.

The documentation for this struct was generated from the following file:

- ViennaRNA/[concentrations.h](#)

## 17.9 vrna\_sc\_bp\_storage\_t Struct Reference

A base pair constraint.

### 17.9.1 Detailed Description

A base pair constraint.

The documentation for this struct was generated from the following file:

- ViennaRNA/constraints/[soft.h](#)

## 17.10 vrna\_structured\_domains\_s Struct Reference

The documentation for this struct was generated from the following file:

- ViennaRNA/[structured\\_domains.h](#)

## 17.11 vrna\_subopt\_sol\_s Struct Reference

Solution element from subopt.c.

### Data Fields

- float **energy**  
*Free Energy of structure in kcal/mol.*
- char \* **structure**  
*Structure in dot-bracket notation.*

### 17.11.1 Detailed Description

Solution element from subopt.c.

The documentation for this struct was generated from the following file:

- ViennaRNA/[subopt.h](#)

## 17.12 vrna\_unstructured\_domain\_motif\_s Struct Reference

The documentation for this struct was generated from the following file:

- ViennaRNA/[unstructured\\_domains.h](#)

# Chapter 18

## File Documentation

### 18.1 ViennaRNA/2Dfold.h File Reference

MFE structures for base pair distance classes.  
Include dependency graph for 2Dfold.h:

### 18.2 2Dfold.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_TWO_D_FOLD_H
2 #define VIENNA_RNA_PACKAGE_TWO_D_FOLD_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 # define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 # define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 # define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
35 #include <ViennaRNA/fold_compound.h>
36 #include <ViennaRNA/datastructures/basic.h>
37 #include <ViennaRNA/params/basic.h>
38
53 typedef struct vrna_sol_TwoD_t {
54     int k;
55     int l;
56     float en;
57     char *s;
58 } vrna_sol_TwoD_t;
59
60
88 vrna_sol_TwoD_t *
89 vrna_mfe_TwoD(vrna_fold_compound_t *vc,
90                 int             distance1,
91                 int             distance2);
92
93
112 char *
113 vrna_backtrack5_TwoD(vrna_fold_compound_t *vc,
114                         int             k,
115                         int             l,
116                         unsigned int    j);
117
118
119 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
120
121 #define TwoDfold_solution     vrna_sol_TwoD_t      /* restore compatibility of struct rename */
122
130 typedef struct TwoDfold_vars {
131     vrna_param_t      *P;
132     int               do_backtrack;
133     char              *ptype;
134     char              *sequence;
135     short             *S, *S1;
136     unsigned int      maxD1;
137     unsigned int      maxD2;
```

```

140     unsigned int          *mml;
141     unsigned int          *mm2;
142     int                  *my_iindx;
143     double               temperature;
144
145     unsigned int          *referenceBPs1;
146     unsigned int          *referenceBPs2;
147     unsigned int          *bpdist;
148     short                *reference_pt1;
149     short                *reference_pt2;
150     int                  circ;
151     int                  dangles;
152     unsigned int          seq_length;
153
154     int                  ***E_F5;
155     int                  ***E_F3;
156     int                  ***E_C;
157     int                  ***E_M;
158     int                  ***E_M1;
159     int                  ***E_M2;
160
161     int                  **E_Fc;
162     int                  **E_FcH;
163     int                  **E_FcI;
164     int                  **E_FcM;
165
166     int                  **l_min_values;
167     int                  **l_max_values;
168     int                  *k_min_values;
169     int                  *k_max_values;
170
171     int                  **l_min_values_m;
172     int                  **l_max_values_m;
173     int                  *k_min_values_m;
174     int                  *k_max_values_m;
175
176     int                  **l_min_values_m1;
177     int                  **l_max_values_m1;
178     int                  *k_min_values_m1;
179     int                  *k_max_values_m1;
180
181     int                  **l_min_values_f;
182     int                  **l_max_values_f;
183     int                  *k_min_values_f;
184     int                  *k_max_values_f;
185
186     int                  **l_min_values_f3;
187     int                  **l_max_values_f3;
188     int                  *k_min_values_f3;
189     int                  *k_max_values_f3;
190
191     int                  **l_min_values_m2;
192     int                  **l_max_values_m2;
193     int                  *k_min_values_m2;
194     int                  *k_max_values_m2;
195
196     int                  *l_min_values_fc;
197     int                  *l_max_values_fc;
198     int                  k_min_values_fc;
199     int                  k_max_values_fc;
200
201     int                  *l_min_values_fcH;
202     int                  *l_max_values_fcH;
203     int                  k_min_values_fcH;
204     int                  k_max_values_fcH;
205
206     int                  *l_min_values_fcI;
207     int                  *l_max_values_fcI;
208     int                  k_min_values_fcI;
209     int                  k_max_values_fcI;
210
211     int                  *l_min_values_fcM;
212     int                  *l_max_values_fcM;
213     int                  k_min_values_fcM;
214     int                  k_max_values_fcM;
215
216     /* auxilary arrays for remaining set of coarse graining (k,l) > (k_max, l_max) */
217     int                  *E_F5_rem;
218     int                  *E_F3_rem;
219     int                  *E_C_rem;
220     int                  *E_M_rem;
221     int                  *E_M1_rem;
222     int                  *E_M2_rem;
223
224     int                  E_Fc_rem;
225     int                  E_FcH_rem;
226     int                  E_FcI_rem;
227
228
229

```

```

230     int             E_FcM_rem;
231
232 #ifdef COUNT_STATES
233     unsigned long   ***N_F5;
234     unsigned long   ***N_C;
235     unsigned long   ***N_M;
236     unsigned long   ***N_M1;
237 #endif
238
239     vrna_fold_compound_t *compatibility;
240 } TwoDfold_vars;
241
242 DEPRECATED(TwoDfold_vars *
243             get_TwoDfold_variables(const char *seq,
244                                     const char *structure1,
245                                     const char *structure2,
246                                     int         circ),
247             "Use the new API and corresponding functions vrna_fold_compound_TwoD(), etc. instead");
248
249 DEPRECATED(void
250             destroy_TwoDfold_variables(TwoDfold_vars *our_variables),
251             "Use the new API and vrna_fold_compound_free() instead");
252
253 DEPRECATED(TwoDfold_solution *
254             TwoDfoldList(TwoDfold_vars *vars,
255                         int          distance1,
256                         int          distance2),
257             "Use the new API and vrna_mfe_TwoD() instead");
258
259 DEPRECATED(char *TwoDfold_backtrack_f5(unsigned int j,
260                                         int          k,
261                                         int          l,
262                                         TwoDfold_vars *vars),
263             "Use the new API and vrna_backtrack5_TwoD() instead");
264
265 DEPRECATED(TwoDfold_solution **TwoDfold(TwoDfold_vars *our_variables,
266                                         int          distance1,
267                                         int          distance2),
268             "Use the new API and vrna_mfe_TwoD() instead");
269
270 #endif
271
272 #endif

```

## 18.3 ViennaRNA/2Dpfold.h File Reference

Partition function implementations for base pair distance classes.

Include dependency graph for 2Dpfold.h:

### Data Structures

- struct [vrna\\_sol\\_TwoD\\_pf\\_t](#)  
*Solution element returned from [vrna\\_pf\\_TwoD\(\)](#) More...*
- struct [TwoDfold\\_vars](#)  
*Variables compound for 2Dfold partition function folding.*

### Typedefs

- typedef struct [vrna\\_sol\\_TwoD\\_pf\\_t](#) [vrna\\_sol\\_TwoD\\_pf\\_t](#)  
*Solution element returned from [vrna\\_pf\\_TwoD\(\)](#)*

### Functions

- [vrna\\_sol\\_TwoD\\_pf\\_t \\* vrna\\_pf\\_TwoD \(vrna\\_fold\\_compound\\_t \\*vc, int maxDistance1, int maxDistance2\)](#)  
*Compute the partition function for all distance classes.*
- [char \\* vrna\\_pbacktrack\\_TwoD \(vrna\\_fold\\_compound\\_t \\*vc, int d1, int d2\)](#)  
*Sample secondary structure representatives from a set of distance classes according to their Boltzmann probability.*
- [char \\* vrna\\_pbacktrack5\\_TwoD \(vrna\\_fold\\_compound\\_t \\*vc, int d1, int d2, unsigned int length\)](#)  
*Sample secondary structure representatives with a specified length from a set of distance classes according to their Boltzmann probability.*

- `TwoDpfold_vars * get_TwoDpfold_variables (const char *seq, const char *structure1, char *structure2, int circ)`  
*Get a datastructure containing all necessary attributes and global folding switches.*
- `void destroy_TwoDpfold_variables (TwoDpfold_vars *vars)`  
*Free all memory occupied by a `TwoDpfold_vars` datastructure.*
- `vrna_sol_TwoD_pf_t * TwoDpfoldList (TwoDpfold_vars *vars, int maxDistance1, int maxDistance2)`  
*Compute the partition function for all distance classes.*
- `char * TwoDpfold_pbacktrack (TwoDpfold_vars *vars, int d1, int d2)`  
*Sample secondary structure representatives from a set of distance classes according to their Boltzmann probability.*
- `char * TwoDpfold_pbacktrack5 (TwoDpfold_vars *vars, int d1, int d2, unsigned int length)`  
*Sample secondary structure representatives with a specified length from a set of distance classes according to their Boltzmann probability.*

### 18.3.1 Detailed Description

Partition function implementations for base pair distance classes.

### 18.3.2 Function Documentation

#### 18.3.2.1 `get_TwoDpfold_variables()`

```
TwoDpfold_vars * get_TwoDpfold_variables (
    const char * seq,
    const char * structure1,
    char * structure2,
    int circ )
```

Get a datastructure containing all necessary attributes and global folding switches.

This function prepares all necessary attributes and matrices etc which are needed for a call of `TwoDpfold()`. A snapshot of all current global model switches (dangles, temperature and so on) is done and stored in the returned datastructure. Additionally, all matrices that will hold the partition function values are prepared.

**Deprecated** Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_<-compound_TwoD()`, `vrna_pf_TwoD()`, and `vrna_fold_compound_free()` instead!

#### Parameters

|                         |  |
|-------------------------|--|
| <code>seq</code>        | the RNA sequence in uppercase format with letters from the alphabet {AUCG} |
| <code>structure1</code> | the first reference structure in dot-bracket notation                      |
| <code>structure2</code> | the second reference structure in dot-bracket notation                     |
| <code>circ</code>       | a switch indicating if the sequence is linear (0) or circular (1)          |

#### Returns

the datastructure containing all necessary partition function attributes

#### 18.3.2.2 `destroy_TwoDpfold_variables()`

```
void destroy_TwoDpfold_variables (
    TwoDpfold_vars * vars )
```

Free all memory occupied by a `TwoDpfold_vars` datastructure.

This function free's all memory occupied by a datastructure obtained from `get_TwoDpfold_variables()` or `get-<-TwoDpfold_variables_from_MFE()`

**Deprecated** Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_<compound_TwoD()`, `vrna_pf_TwoD()`, and `vrna_fold_compound_free()` instead!

#### See also

[get\\_TwoDpfold\\_variables\(\)](#), [get\\_TwoDpfold\\_variables\\_from\\_MFE\(\)](#)

#### Parameters

|                   |                                |
|-------------------|--------------------------------|
| <code>vars</code> | the datastructure to be free'd |
|-------------------|--------------------------------|

### 18.3.2.3 TwoDpfoldList()

```
vrna_sol_TwoD_pf_t * TwoDpfoldList (
    TwoDpfold_vars * vars,
    int maxDistance1,
    int maxDistance2 )
```

Compute the partition function for all distance classes.

This function computes the partition functions for all distance classes according the two reference structures specified in the datastructure 'vars'. Similar to `TwoDfold()` the arguments `maxDistance1` and `maxDistance2` specify the maximum distance to both reference structures. A value of '-1' in either of them makes the appropriate distance restrictionless, i.e. all basepair distancies to the reference are taken into account during computation. In case there is a restriction, the returned solution contains an entry where the attribute `k=l=-1` contains the partition function for all structures exceeding the restriction. A values of `INF` in the attribute 'k' of the returned list denotes the end of the list

**Deprecated** Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_<compound_TwoD()`, `vrna_pf_TwoD()`, and `vrna_fold_compound_free()` instead!

#### See also

[get\\_TwoDpfold\\_variables\(\)](#), [destroy\\_TwoDpfold\\_variables\(\)](#), [vrna\\_sol\\_TwoD\\_pf\\_t](#)

#### Parameters

|                           |  |
|---------------------------|--|
| <code>vars</code>         | the datastructure containing all necessary folding attributes and matrices |
| <code>maxDistance1</code> | the maximum basepair distance to reference1 (may be -1)                    |
| <code>maxDistance2</code> | the maximum basepair distance to reference2 (may be -1)                    |

#### Returns

a list of partition funtions for the appropriate distance classes

### 18.3.2.4 TwoDpfold\_pbacktrack()

```
char * TwoDpfold_pbacktrack (
    TwoDpfold_vars * vars,
    int d1,
    int d2 )
```

Sample secondary structure representatives from a set of distance classes according to their Boltzmann probability. If the argument 'd1' is set to '-1', the structure will be backtracked in the distance class where all structures exceeding the maximum basepair distance to either of the references reside.

**Precondition**

The argument 'vars' must contain precalculated partition function matrices, i.e. a call to TwoDpfold() preceding this function is mandatory!

**Deprecated** Use the new API that relies on [vrna\\_fold\\_compound\\_t](#) and the corresponding functions [vrna\\_fold\\_compound\\_TwoD\(\)](#), [vrna\\_pf\\_TwoD\(\)](#), [vrna\\_pbacktrack\\_TwoD\(\)](#), and [vrna\\_fold\\_compound\\_free\(\)](#) instead!

**See also**

[TwoDpfold\(\)](#)

**Parameters**

|    |             |  |
|----|-------------|--|
| in | <i>vars</i> | the datastructure containing all necessary folding attributes and matrices |
| in | <i>d1</i>   | the distance to reference1 (may be -1)                                     |
| in | <i>d2</i>   | the distance to reference2   |

**Returns**

A sampled secondary structure in dot-bracket notation

**18.3.2.5 TwoDpfold\_pbacktrack5()**

```
char * TwoDpfold_pbacktrack5 (
    TwoDpfold_vars * vars,
    int d1,
    int d2,
    unsigned int length )
```

Sample secondary structure representatives with a specified length from a set of distance classes according to their Boltzmann probability.

This function does essentially the same as [TwoDpfold\\_pbacktrack\(\)](#) with the only difference that partial structures, i.e. structures beginning from the 5' end with a specified length of the sequence, are backtracked

**Note**

This function does not work (since it makes no sense) for circular RNA sequences!

**Precondition**

The argument 'vars' must contain precalculated partition function matrices, i.e. a call to TwoDpfold() preceding this function is mandatory!

**Deprecated** Use the new API that relies on [vrna\\_fold\\_compound\\_t](#) and the corresponding functions [vrna\\_fold\\_compound\\_TwoD\(\)](#), [vrna\\_pf\\_TwoD\(\)](#), [vrna\\_pbacktrack5\\_TwoD\(\)](#), and [vrna\\_fold\\_compound\\_free\(\)](#) instead!

**See also**

[TwoDpfold\\_pbacktrack\(\)](#), [TwoDpfold\(\)](#)

**Parameters**

|    |               |  |
|----|---------------|--|
| in | <i>vars</i>   | the datastructure containing all necessary folding attributes and matrices |
| in | <i>d1</i>     | the distance to reference1 (may be -1)                                     |
| in | <i>d2</i>     | the distance to reference2   |
| in | <i>length</i> | the length of the structure beginning from the 5' end                      |

**Returns**

A sampled secondary structure in dot-bracket notation

## 18.4 2Dpfold.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_TWO_D_PF_FOLD_H
2 #define VIENNA_RNA_PACKAGE_TWO_D_PF_FOLD_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 # define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 # define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 # define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
32 #include <ViennaRNA/fold_compound.h>
33 #include <ViennaRNA/datastructures/basic.h>
34 #include <ViennaRNA/fold_compound.h>
35 #include <ViennaRNA/params/basic.h>
36
49 typedef struct vrna_sol_TwoD_pf_t {
50     int          k;
51     int          l;
52     FLT_OR_DBL q;
53 } vrna_sol_TwoD_pf_t;
54
77 vrna_sol_TwoD_pf_t *
78 vrna_pf_TwoD(vrna_fold_compound_t *vc,
79                 int           maxDistance1,
80                 int           maxDistance2);
81
82 /* End of group kl_neighborhood_pf */
83
108 char *
109 vrna_pbacktrack_TwoD(vrna_fold_compound_t *vc,
110                         int           d1,
111                         int           d2);
112
113
133 char *
134 vrna_pbacktrack5_TwoD(vrna_fold_compound_t *vc,
135                         int           d1,
136                         int           d2,
137                         unsigned int length);
138
139 /* End of group kl_neighborhood_stochbt */
140
141
144 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
145
146 #define TwoDpfold_solution      vrna_sol_TwoD_pf_t      /* restore compatibility of struct rename */
147
155 typedef struct {
156     unsigned int      alloc;
157     char            *ptype;
158     char            *sequence;
159     short           *S, *S1;
160     unsigned int     maxD1;
161     unsigned int     maxD2;
162     double          temperature; /* temperature in last call to scale_pf_params */
163     double          init_temp;   /* temperature in last call to scale_pf_params */
164     FLT_OR_DBL      *scale;
165     FLT_OR_DBL      pf_scale;
166     vrna_exp_param_t *pf_params; /* holds all [unscaled] pf parameters */
167
168     int             *my_iindx;
169     int             *jindx;
170     short           *reference_pt1;
171     short           *reference_pt2;
172
173     unsigned int    *referenceBPs1;
174     unsigned int    *referenceBPs2;
175     unsigned int    *bpdist;
176     unsigned int    *mml;
177     unsigned int    *mm2;
178     int             circ;
179     int             dangles;
180     unsigned int    seq_length;
181
182
183
184

```

```

185     FLT_OR_DBL      ***Q;
186     FLT_OR_DBL      ***Q_B;
187     FLT_OR_DBL      ***Q_M;
188     FLT_OR_DBL      ***Q_M1;
189     FLT_OR_DBL      ***Q_M2;
190
191
192     FLT_OR_DBL      **Q_c;
193     FLT_OR_DBL      **Q_ch;
194     FLT_OR_DBL      **Q_ci;
195     FLT_OR_DBL      **Q_cm;
196
197     int             **l_min_values;
198     int             **l_max_values;
199     int             *k_min_values;
200     int             *k_max_values;
201
202     int             **l_min_values_b;
203     int             **l_max_values_b;
204     int             *k_min_values_b;
205     int             *k_max_values_b;
206
207     int             **l_min_values_m;
208     int             **l_max_values_m;
209     int             *k_min_values_m;
210     int             *k_max_values_m;
211
212     int             **l_min_values_m1;
213     int             **l_max_values_m1;
214     int             *k_min_values_m1;
215     int             *k_max_values_m1;
216
217     int             **l_min_values_m2;
218     int             **l_max_values_m2;
219     int             *k_min_values_m2;
220     int             *k_max_values_m2;
221
222     int             *l_min_values_qc;
223     int             *l_max_values_qc;
224     int             k_min_values_qc;
225     int             k_max_values_qc;
226
227     int             *l_min_values_qch;
228     int             *l_max_values_qch;
229     int             k_min_values_qch;
230     int             k_max_values_qch;
231
232     int             *l_min_values_qci;
233     int             *l_max_values_qci;
234     int             k_min_values_qci;
235     int             k_max_values_qci;
236
237     int             *l_min_values_qcm;
238     int             *l_max_values_qcm;
239     int             k_min_values_qcm;
240     int             k_max_values_qcm;
241
242 /* auxiliary arrays for remaining set of coarse graining (k,l) > (k_max, l_max) */
243     FLT_OR_DBL      *Q_rem;
244     FLT_OR_DBL      *Q_B_rem;
245     FLT_OR_DBL      *Q_M_rem;
246     FLT_OR_DBL      *Q_M1_rem;
247     FLT_OR_DBL      *Q_M2_rem;
248
249     FLT_OR_DBL      Q_c_rem;
250     FLT_OR_DBL      Q_ch_rem;
251     FLT_OR_DBL      Q_ci_rem;
252     FLT_OR_DBL      Q_cm_rem;
253
254     vrna_fold_compound_t *compatibility;
255 } TwoDpfold_vars;
256
275 DEPRECATED(TwoDpfold_vars *
276             get_TwoDpfold_variables(const char *seq,
277                                     const char *structure1,
278                                     char       *structure2,
279                                     int        circ),
280             "Use the new API and vrna_fold_compound_TwoD() instead");
281
295 DEPRECATED(void
296             destroy_TwoDpfold_variables(TwoDpfold_vars *vars),
297             "Use the new API and vrna_fold_compound_free() instead");
298
323 DEPRECATED(TwoDpfold_solution *
324             TwoDpfoldList(TwoDpfold_vars *vars,
325                           int          maxDistance1,
326                           int          maxDistance2),

```

```

327     "Use the new API and vrna_pf_TwoD() instead");
328
329 DEPRECATED(char *
330             TwoDpfold_pbacktrack(TwoDpfold_vars *vars,
331                         int d1,
332                         int d2),
333     "Use the new API and vrna_pbacktrack_TwoD() instead");
334
335 DEPRECATED(char *
336             TwoDpfold_pbacktrack5(TwoDpfold_vars *vars,
337                         int d1,
338                         int d2,
339                         unsigned int length),
340     "Use the new API and vrna_pbacktrack5_TwoD() instead");
341
342 DEPRECATED(FLT_OR_DBL **TwoDpfold(TwoDpfold_vars *our_variables,
343                                     int maxDistance1,
344                                     int maxDistance2),
345     "Use the new API and vrna_pf_TwoD() instead");
346
347 DEPRECATED(FLT_OR_DBL **TwoDpfold_circ(TwoDpfold_vars *our_variables,
348                                         int maxDistance1,
349                                         int maxDistance2),
350     "Use the new API and vrna_pf_TwoD() instead");
351
352 #endif
353
354 #endif

```

## 18.5 ali\_plex.h

```

1 #ifndef VIENNA_RNA_PACKAGE_ALI_PLEX_H
2 #define VIENNA_RNA_PACKAGE_ALI_PLEX_H
3
4 #include <ViennaRNA/datastructures/basic.h>
5
6 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
7
11 duplexT **aliLduplexfold(const char *s1[],
12                           const char *s2[],
13                           const int threshold,
14                           const int extension_cost,
15                           const int alignment_length,
16                           const int delta,
17                           const int fast,
18                           const int il_a,
19                           const int il_b,
20                           const int b_a,
21                           const int b_b);
22
23
27 duplexT **aliLduplexfold_XS(const char *s1[],
28                             const char *s2[],
29                             const int **access_s1,
30                             const int **access_s2,
31                             const int threshold,
32                             const int alignment_length,
33                             const int delta,
34                             const int fast,
35                             const int il_a,
36                             const int il_b,
37                             const int b_a,
38                             const int b_b);
39
40
41 /*
42  * extern duplexT aliduplexfold(const char *s1[], const char *s2[], const int extension_cost);
43  * extern duplexT aliduplexfold_XS(const char *s1[], const char *s2[], const int **access_s1,
44  * const int **access_s2, const int i_pos, const int j_pos, const int threshold);
45 */
46 #endif
47
48 #endif

```

## 18.6 ViennaRNA/alifold.h File Reference

Functions for comparative structure prediction using RNA sequence alignments.  
Include dependency graph for alifold.h:

## Functions

- float `energy_of_alistruct` (const char \*\*sequences, const char \*structure, int n\_seq, float \*energy)
 

*Calculate the free energy of a consensus structure given a set of aligned sequences.*
- void `update_alifold_params` (void)
 

*Update the energy parameters for alifold function.*
  
- float `alifold` (const char \*\*strings, char \*structure)
 

*Compute MFE and according consensus structure of an alignment of sequences.*
- float `circalifold` (const char \*\*strings, char \*structure)
 

*Compute MFE and according structure of an alignment of sequences assuming the sequences are circular instead of linear.*
- void `free_alifold_arrays` (void)
 

*Free the memory occupied by MFE alifold functions.*
  
- float `alipf_fold_par` (const char \*\*sequences, char \*structure, `vrna_ep_t` \*\*pl, `vrna_exp_param_t` \*parameters, int calculate\_bppm, int is\_constrained, int is\_circular)
 

*The partition function version of `alifold()` works in analogy to `pf_fold()`. Pair probabilities and information about sequence covariations are returned via the 'pi' variable as a list of `vrna_pinfo_t` structs. The list is terminated by the first entry with  $pi.i = 0$ .*
- float `alipf_fold` (const char \*\*sequences, char \*structure, `vrna_ep_t` \*\*pl)
 

*Get a pointer to the base pair probability array.*
- void `free_alipf_arrays` (void)
 

*Free the memory occupied by folding matrices allocated by `alipf_fold`, `alipf_circ_fold`, etc.*
- char \* `alipbacktrack` (double \*prob)
 

*Sample a consensus secondary structure from the Boltzmann ensemble according its probability.*
- int `get_alipf_arrays` (short \*\*\*S\_p, short \*\*\*S5\_p, short \*\*\*S3\_p, unsigned short \*\*\*a2s\_p, char \*\*\*Ss\_p, `FLT_OR_DBL` \*\*qb\_p, `FLT_OR_DBL` \*\*qm\_p, `FLT_OR_DBL` \*\*q1k\_p, `FLT_OR_DBL` \*\*qln\_p, short \*\*pscore)
 

*Get pointers to (almost) all relevant arrays used in alifold's partition function computation.*

## Variables

- double `cv_fact`

*This variable controls the weight of the covariance term in the energy function of alignment folding algorithms.*
- double `nc_fact`

*This variable controls the magnitude of the penalty for non-compatible sequences in the covariance term of alignment folding algorithms.*

### 18.6.1 Detailed Description

Functions for comparative structure prediction using RNA sequence alignments.

### 18.6.2 Function Documentation

### 18.6.2.1 energy\_of\_alistruct()

```
float energy_of_alistruct (
    const char ** sequences,
    const char * structure,
    int n_seq,
    float * energy )
```

Calculate the free energy of a consensus structure given a set of aligned sequences.

**Deprecated** Usage of this function is discouraged! Use [vrna\\_eval\\_structure\(\)](#), and [vrna\\_eval\\_covar\\_structure\(\)](#) instead!

#### Parameters

|                  |  |
|------------------|--|
| <i>sequences</i> | The NULL terminated array of sequences   |
| <i>structure</i> | The consensus structure  |
| <i>n_seq</i>     | The number of sequences in the alignment   |
| <i>energy</i>    | A pointer to an array of at least two floats that will hold the free energies (energy[0] will contain the free energy, energy[1] will be filled with the covariance energy term) |

#### Returns

free energy in kcal/mol

### 18.6.2.2 update\_alifold\_params()

```
void update_alifold_params (
    void )
```

Update the energy parameters for alifold function.

Call this to recalculate the pair matrix and energy parameters after a change in folding parameters like [temperature](#)

**Deprecated** Usage of this function is discouraged! The new API uses [vrna\\_fold\\_compound\\_t](#) to lump all folding related necessities together, including the energy parameters. Use [vrna\\_update\\_fold\\_params\(\)](#) to update the energy parameters within a [vrna\\_fold\\_compound\\_t](#).

## 18.6.3 Variable Documentation

### 18.6.3.1 cv\_fact

```
double cv_fact [extern]
```

This variable controls the weight of the covariance term in the energy function of alignment folding algorithms.

**Deprecated** See [vrna\\_md\\_t.cv\\_fact](#), and [vrna\\_mfe\(\)](#) to avoid using global variables

Default is 1.

### 18.6.3.2 nc\_fact

```
double nc_fact [extern]
```

This variable controls the magnitude of the penalty for non-compatible sequences in the covariance term of alignment folding algorithms.

**Deprecated** See [vrna\\_md\\_t.nc\\_fact](#), and [vrna\\_mfe\(\)](#) to avoid using global variables

Default is 1.

## 18.7 alifold.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_ALIFOLD_H
2 #define VIENNA_RNA_PACKAGE_ALIFOLD_H
3
4 #include <ViennaRNA/datastructures/basic.h>
5 #include <ViennaRNA/params/basic.h>
6 #include <ViennaRNA/ribo.h>
7 #include <ViennaRNA/mfe.h>
8 #include <ViennaRNA/part_func.h>
9 #include <ViennaRNA/utils/alignments.h>
10 #include <ViennaRNA/utils/structures.h>
11 #include <ViennaRNA/boltzmann_sampling.h>
12
13 #ifdef VRNA_WARN_DEPRECATED
14 # if defined(__clang__)
15 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
16 # elif defined(__GNUC__)
17 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
18 # else
19 #  define DEPRECATED(func, msg) func
20 # endif
21 #else
22 # define DEPRECATED(func, msg) func
23 #endif
24
32 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
33
34 /*
35 ##### DEPRECATED FUNCTIONS #####
36 # DEPRECATED FUNCTIONS
37 ##### DEPRECATED FUNCTIONS #####
38 */
39
63 DEPRECATED(float alifold( const char **strings, char *structure),
64             "Use vrna_alifold() or vrna_mfe() instead");
65
77 DEPRECATED(float circalifold( const char **strings, char *structure),
78             "Use vrna_alicircfold() or vrna_mfe() instead");
79
91 DEPRECATED(void free_alifold_arrays(void),
92             "This function is obsolete");
93
94 /* End group mfe_global_DEPRECATED */
112 DEPRECATED(float energy_of-alistruct(const char **sequences, const char *structure, int n_seq, float
*energy),
113             "Use vrna_eval_structure() and vrna_eval_covar_structure() instead");
114
115 DEPRECATED(float energy_of_ali_gquad_structure(const char **sequences, const char *structure, int n_seq,
float *energy),
116             "Use vrna_eval_structure() and vrna_eval_covar_structure() instead");
117
128 DEPRECATED(extern double cv_fact,
129             "Use the cv_fact attribute of the vrna_md_t datastructure instead");
140 DEPRECATED(extern double nc_fact,
141             "Use the nc_fact attribute of the vrna_md_t datastructure instead");
142
162 DEPRECATED(float alipf_fold_par( const char **sequences,
163             char *structure,
164             vrna_ep_t **pl,
165             vrna_exp_param_t *parameters,
166             int calculate_bppm,
167             int is_constrained,
168             int is_circular),
169             "Use vrna_pf_alifold() or vrna_pf() instead");
170
187 DEPRECATED(float alipf_fold( const char **sequences, char *structure, vrna_ep_t **pl),
188             "Use vrna_pf_alifold() or vrna_pf() instead");
189
200 DEPRECATED(float alipf_circ_fold(const char **sequences, char *structure, vrna_ep_t **pl),
201             "Use vrna_pf_circalifold() or vrna_pf() instead");
202
203
220 DEPRECATED(FLT_OR_DBL *export_ali_bppm(void),
221             "Use the new API with vrna_fold_compound_t datastructure instead");
222
233 DEPRECATED(void free_alipf_arrays(void),
234             "This function is obsolete");
235
244 DEPRECATED(char *alipbacktrack(double *prob),
245             "Use the new API and vrna_pbacktrack() instead");
246
271 DEPRECATED(int get_alipf_arrays(short ***S_p,
272             short ***S5_p,
273             short ***S3_p,

```

```

274         unsigned short ***a2s_p,
275         char ***Ss_p,
276         FLT_OR_DBL **qb_p,
277         FLT_OR_DBL **qm_p,
278         FLT_OR_DBL **q1k_p,
279         FLT_OR_DBL **qln_p,
280         short **pscore),
281     "Use the new API with vrna_fold_compound_t datastructure instead");
282
283
284 /* End group part_func_global_deprecated */
285 DEPRECATED(void update_alifold_params(void),
286             "Use the new API with vrna_fold_compound_t datastructure instead");
287
288 #endif
289
290 #endif
291
292 #endif

```

## 18.8 ViennaRNA/aln\_util.h File Reference

Use [ViennaRNA/utils/alignments.h](#) instead.

Include dependency graph for aln\_util.h:

### 18.8.1 Detailed Description

Use [ViennaRNA/utils/alignments.h](#) instead.

**Deprecated** Use [ViennaRNA/utils/alignments.h](#) instead

## 18.9 aln\_util.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_ALN_UTILS_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_ALN_UTILS_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 # ifdef VRNA_WARN_DEPRECATED
6 # warning "Including deprecated header file <ViennaRNA/aln_util.h>! Use <ViennaRNA/utils/alignments.h>
7           instead!"
8 # endif
9 #include <ViennaRNA/utils/alignments.h>
10#endif
11
12#endif

```

## 18.10 ViennaRNA/alphabet.h File Reference

Functions to process, convert, and generally handle different nucleotide and/or base pair alphabets.

Include dependency graph for alphabet.h: This graph shows which files directly or indirectly include this file:

### Functions

- `char * vrna_ptypes (const short *S, vrna_md_t *md)`

*Get an array of the numerical encoding for each possible base pair (i,j)*
- `short * vrna_seq_encode (const char *sequence, vrna_md_t *md)`

*Get a numerical representation of the nucleotide sequence.*
- `short * vrna_seq_encode_simple (const char *sequence, vrna_md_t *md)`

*Get a numerical representation of the nucleotide sequence (simple version)*
- `int vrna_nucleotide_encode (char c, vrna_md_t *md)`

*Encode a nucleotide character to numerical value.*
- `char vrna_nucleotide_decode (int enc, vrna_md_t *md)`

*Decode a numerical representation of a nucleotide back into nucleotide alphabet.*

### **18.10.1 Detailed Description**

Functions to process, convert, and generally handle different nucleotide and/or base pair alphabets.

,

## 18.11 alphabet.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_ALPHABET_H
2 #define VIENNA_RNA_PACKAGE_ALPHABET_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 # define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 # define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 # define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
16 #include <ViennaRNA/fold_compound.h>
17 #include <ViennaRNA/model.h>
18
19 unsigned int
20 vrna_sequence_length_max(unsigned int options);
21
22 int
23 vrna_nucleotide_IUPAC_identity(char a,
24                                 char b);
25
26 void
27 vrna_ptypes_prepare(vrna_fold_compound_t *fc,
28                      unsigned int options);
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
```

```

131         char *ptype);
132
133
134 unsigned int
135 vrna_get_ptype_window(int    i,
136                         int    j,
137                         char   **ptype);
138
139
140 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
141
142 DEPRECATED(char *get_ptypes(const short   *s,
143                             vrna_md_t      *md,
144                             unsigned int   idx_type),
145
146             "Use vrna_pyptes() instead");
147
148#endif
149
150#endif
151#endif
152
153#endif

```

## 18.12 ViennaRNA/boltzmann\_sampling.h File Reference

Boltzmann Sampling of secondary structures from the ensemble.

Include dependency graph for boltzmann\_sampling.h: This graph shows which files directly or indirectly include this file:

### Macros

- `#define VRNA_PBACKTRACK_DEFAULT 0`  
*Boltzmann sampling flag indicating default backtracing mode.*
- `#define VRNA_PBACKTRACK_NON_REDUNDANT 1`  
*Boltzmann sampling flag indicating non-redundant backtracing mode.*

### Typedefs

- `typedef void() vrna_boltzmann_sampling_callback(const char *structure, void *data)`  
*Callback for Boltzmann sampling.*
- `typedef struct vrna_pbacktrack_memory_s * vrna_pbacktrack_mem_t`  
*Boltzmann sampling memory data structure.*

### Functions

- `char * vrna_pbacktrack5 (vrna_fold_compound_t *fc, unsigned int length)`  
*Sample a secondary structure of a subsequence from the Boltzmann ensemble according its probability.*
- `char ** vrna_pbacktrack5_num (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int length, unsigned int options)`  
*Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.*
- `unsigned int vrna_pbacktrack5_cb (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int length, vrna_boltzmann_sampling_callback *cb, void *data, unsigned int options)`  
*Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.*
- `char ** vrna_pbacktrack5_resume (vrna_fold_compound_t *vc, unsigned int num_samples, unsigned int length, vrna_pbacktrack_mem_t *nr_mem, unsigned int options)`  
*Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.*
- `unsigned int vrna_pbacktrack5_resume_cb (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int length, vrna_boltzmann_sampling_callback *cb, void *data, vrna_pbacktrack_mem_t *nr_mem, unsigned int options)`  
*Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.*

- `char * vrna_pbacktrack (vrna_fold_compound_t *fc)`  
*Sample a secondary structure from the Boltzmann ensemble according its probability.*
- `char ** vrna_pbacktrack_num (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int options)`  
*Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.*
- `unsigned int vrna_pbacktrack_cb (vrna_fold_compound_t *fc, unsigned int num_samples, vrna_boltzmann_sampling_callback *cb, void *data, unsigned int options)`  
*Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.*
- `char ** vrna_pbacktrack_resume (vrna_fold_compound_t *fc, unsigned int num_samples, vrna_pbacktrack_mem_t *nr_mem, unsigned int options)`  
*Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.*
- `unsigned int vrna_pbacktrack_resume_cb (vrna_fold_compound_t *fc, unsigned int num_samples, vrna_boltzmann_sampling_callback *cb, void *data, vrna_pbacktrack_mem_t *nr_mem, unsigned int options)`  
*Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.*
- `char * vrna_pbacktrack_sub (vrna_fold_compound_t *fc, unsigned int start, unsigned int end)`  
*Sample a secondary structure of a subsequence from the Boltzmann ensemble according its probability.*
- `char ** vrna_pbacktrack_sub_num (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int start, unsigned int end, unsigned int options)`  
*Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.*
- `unsigned int vrna_pbacktrack_sub_cb (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int start, unsigned int end, vrna_boltzmann_sampling_callback *cb, void *data, unsigned int options)`  
*Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.*
- `char ** vrna_pbacktrack_sub_resume (vrna_fold_compound_t *vc, unsigned int num_samples, unsigned int start, unsigned int end, vrna_pbacktrack_mem_t *nr_mem, unsigned int options)`  
*Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.*
- `unsigned int vrna_pbacktrack_sub_resume_cb (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int start, unsigned int end, vrna_boltzmann_sampling_callback *bs_cb, void *data, vrna_pbacktrack_mem_t *nr_mem, unsigned int options)`  
*Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.*
- `void vrna_pbacktrack_mem_free (vrna_pbacktrack_mem_t s)`  
*Release memory occupied by a Boltzmann sampling memory data structure.*

### 18.12.1 Detailed Description

Boltzmann Sampling of secondary structures from the ensemble.

A.k.a. Stochastic backtracking

## 18.13 boltzmann\_sampling.h

Go to the documentation of this file.

```

1 #ifndef VIENNA_RNA_PACKAGE_BOLTZMANN_SAMPLING_H
2 #define VIENNA_RNA_PACKAGE_BOLTZMANN_SAMPLING_H
3
28 #define VRNA_PBACKTRACK_DEFAULT          0
29
40 #define VRNA_PBACKTRACK_NON_REDUNDANT    1
41
57 typedef void (vrna_boltzmann_sampling_callback)(const char *structure,
58                                     void           *data);
59
75 typedef struct vrna_pbacktrack_memory_s *vrna_pbacktrack_mem_t;
76
77 #include <ViennaRNA/fold_compound.h>
78
108 char *
109 vrna_pbacktrack5(vrna_fold_compound_t *fc,
```

```
110             unsigned int          length);
111
112
157 char **
158 vrna_pbacktrack5_num(vrna_fold_compound_t *fc,
159                         unsigned int      num_samples,
160                         unsigned int      length,
161                         unsigned int      options);
162
163
213 unsigned int
214 vrna_pbacktrack5_cb(vrna_fold_compound_t           *fc,
215                         unsigned int      num_samples,
216                         unsigned int      length,
217                         vrna_boltzmann_sampling_callback *cb,
218                         void              *data,
219                         unsigned int      options);
220
221
296 char **
297 vrna_pbacktrack5_resume(vrna_fold_compound_t     *vc,
298                           unsigned int      num_samples,
299                           unsigned int      length,
300                           vrna_pbacktrack_mem_t *nr_mem,
301                           unsigned int      options);
302
303
381 unsigned int
382 vrna_pbacktrack5_resume_cb(vrna_fold_compound_t   *fc,
383                             unsigned int      num_samples,
384                             unsigned int      length,
385                             vrna_boltzmann_sampling_callback *cb,
386                             void              *data,
387                             vrna_pbacktrack_mem_t *nr_mem,
388                             unsigned int      options);
389
390
418 char *
419 vrna_pbacktrack(vrna_fold_compound_t *fc);
420
421
464 char **
465 vrna_pbacktrack_num(vrna_fold_compound_t *fc,
466                         unsigned int      num_samples,
467                         unsigned int      options);
468
469
517 unsigned int
518 vrna_pbacktrack_cb(vrna_fold_compound_t           *fc,
519                         unsigned int      num_samples,
520                         vrna_boltzmann_sampling_callback *cb,
521                         void              *data,
522                         unsigned int      options);
523
524
595 char **
596 vrna_pbacktrack_resume(vrna_fold_compound_t    *fc,
597                           unsigned int      num_samples,
598                           vrna_pbacktrack_mem_t *nr_mem,
599                           unsigned int      options);
600
601
675 unsigned int
676 vrna_pbacktrack_resume_cb(vrna_fold_compound_t   *fc,
677                             unsigned int      num_samples,
678                             vrna_boltzmann_sampling_callback *cb,
679                             void              *data,
680                             vrna_pbacktrack_mem_t *nr_mem,
681                             unsigned int      options);
682
683
684
685
686
687
688
689
722 char *
723 vrna_pbacktrack_sub(vrna_fold_compound_t *fc,
724                         unsigned int      start,
725                         unsigned int      end);
726
727
773 char **
774 vrna_pbacktrack_sub_num(vrna_fold_compound_t *fc,
775                           unsigned int      num_samples,
776                           unsigned int      start,
```

```

777             unsigned int      end,
778             unsigned int      options);
779
780
831 unsigned int
832 vrna_pbacktrack_sub_cb(vrna_fold_compound_t           *fc,
833                           unsigned int      num_samples,
834                           unsigned int      start,
835                           unsigned int      end,
836                           vrna_boltzmann_sampling_callback *cb,
837                           void              *data,
838                           unsigned int      options);
839
840
916 char **
917 vrna_pbacktrack_sub_resume(vrna_fold_compound_t      *vc,
918                             unsigned int      num_samples,
919                             unsigned int      start,
920                             unsigned int      end,
921                             vrna_pbacktrack_mem_t *nr_mem,
922                             unsigned int      options);
923
924
1003 unsigned int
1004 vrna_pbacktrack_sub_resume_cb(vrna_fold_compound_t      *fc,
1005                               unsigned int      num_samples,
1006                               unsigned int      start,
1007                               unsigned int      end,
1008                               vrna_boltzmann_sampling_callback *bs_cb,
1009                               void              *data,
1010                               vrna_pbacktrack_mem_t *nr_mem,
1011                               unsigned int      options);
1012
1013
1022 void
1023 vrna_pbacktrack_mem_free(vrna_pbacktrack_mem_t s);
1024
1025
1029 #endif

```

## 18.14 ViennaRNA/centroid.h File Reference

Centroid structure computation.

Include dependency graph for centroid.h: This graph shows which files directly or indirectly include this file:

### Functions

- `char * vrna_centroid (vrna_fold_compound_t *vc, double *dist)`  
*Get the centroid structure of the ensemble.*
- `char * vrna_centroid_from plist (int length, double *dist, vrna_ep_t *pl)`  
*Get the centroid structure of the ensemble.*
- `char * vrna_centroid_from_probs (int length, double *dist, FLT_OR_DBL *probs)`  
*Get the centroid structure of the ensemble.*
- `char * get_centroid_struct_pl (int length, double *dist, vrna_ep_t *pl)`  
*Get the centroid structure of the ensemble.*
- `char * get_centroid_struct_pr (int length, double *dist, FLT_OR_DBL *pr)`  
*Get the centroid structure of the ensemble.*

### 18.14.1 Detailed Description

Centroid structure computation.

### 18.14.2 Function Documentation

### 18.14.2.1 `get_centroid_struct_pl()`

```
char * get_centroid_struct_pl (
    int length,
    double * dist,
    vrna_ep_t * pl )
```

Get the centroid structure of the ensemble.

**Deprecated** This function was renamed to `vrna_centroid_from plist()`

### 18.14.2.2 `get_centroid_struct_pr()`

```
char * get_centroid_struct_pr (
    int length,
    double * dist,
    FLT_OR_DBL * pr )
```

Get the centroid structure of the ensemble.

**Deprecated** This function was renamed to `vrna_centroid_from_probs()`

## 18.15 centroid.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_CENTROID_H
2 #define VIENNA_RNA_PACKAGE_CENTROID_H
3
4 #include <ViennaRNA/datastructures/basic.h>
5 #include <ViennaRNA/fold_compound.h>
6 #include <ViennaRNA/utils/structures.h>
7
8 #ifdef VRNA_WARN_DEPRECATED
9 # if defined(__clang__)
10 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
11 # elif defined(__GNUC__)
12 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
13 # else
14 #  define DEPRECATED(func, msg) func
15 # endif
16 #else
17 # define DEPRECATED(func, msg) func
18 #endif
19
39 char *
40 vrna_centroid(vrna_fold_compound_t *vc,
41                 double           *dist);
42
43
60 char *
61 vrna_centroid_from plist(int      length,
62                         double   *dist,
63                         vrna_ep_t *pl);
64
65
82 char *
83 vrna_centroid_from_probs(int      length,
84                           double   *dist,
85                           FLT_OR_DBL *probs);
86
87
88 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
89
95 DEPRECATED(char *get_centroid_struct_pl(int      length,
96                                         double   *dist,
97                                         vrna_ep_t *pl),
98         "Use vrna_centroid_from plist() instead");
99
105 DEPRECATED(char *get_centroid_struct_pr(int      length,
106                                         double   *dist,
107                                         FLT_OR_DBL *pr),
108         "Use vrna_centroid_from_probs() instead");
109
110#endif
111
112#endif
```

## 18.16 ViennaRNA/char\_stream.h File Reference

Use [ViennaRNA/datastructures/char\\_stream.h](#) instead.

Include dependency graph for char\_stream.h:

### 18.16.1 Detailed Description

Use [ViennaRNA/datastructures/char\\_stream.h](#) instead.

**Deprecated** Use [ViennaRNA/datastructures/char\\_stream.h](#) instead

## 18.17 char\_stream.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_DATA_STRUCTURES_CHAR_STREAM_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_DATA_STRUCTURES_CHAR_STREAM_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 # warning "Including deprecated header file <ViennaRNA/char_stream.h>! Use
13 # <ViennaRNA/datastructures/char_stream.h> instead!"
14 #endif
15 #include <ViennaRNA/datastructures/char_stream.h>
16 #endif
17 #endif

```

## 18.18 ViennaRNA/datastructures/char\_stream.h File Reference

Implementation of a dynamic, buffered character stream.

Include dependency graph for char\_stream.h: This graph shows which files directly or indirectly include this file:

### Functions

- vrna\_cstr\_t [vrna\\_cstr](#) (size\_t size, FILE \*output)  
*Create a dynamic char \* stream data structure.*
- void [vrna\\_cstr\\_discard](#) (struct vrna\_cstr\_s \*buf)  
*Discard the current content of the dynamic char \* stream data structure.*
- void [vrna\\_cstr\\_free](#) (vrna\_cstr\_t buf)  
*Free the memory occupied by a dynamic char \* stream data structure.*
- void [vrna\\_cstr\\_close](#) (vrna\_cstr\_t buf)  
*Free the memory occupied by a dynamic char \* stream and close the output stream.*
- void [vrna\\_cstr\\_fflush](#) (struct vrna\_cstr\_s \*buf)  
*Flush the dynamic char \* output stream.*

### 18.18.1 Detailed Description

Implementation of a dynamic, buffered character stream.

,

## 18.19 char\_stream.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_CHAR_STREAM_H
2 #define VIENNA_RNA_PACKAGE_CHAR_STREAM_H
3
16 #include <stdarg.h>
17 #include <stdio.h>
18
19 /* below is our own implementation of a dynamic char * stream */
20 typedef struct vrna_cstr_s *vrna_cstr_t;
21

```

```
30 vrna_cstr_t
31 vrna_cstr(size_t size,
32             FILE *output);
33
34
35 void
36 vrna_cstr_discard(struct vrna_cstr_s *buf);
37
38 void
39 vrna_cstr_free(vrna_cstr_t buf);
40
41
42 void
43 vrna_cstr_close(vrna_cstr_t buf);
44
45
46 void
47 vrna_cstr_fflush(struct vrna_cstr_s *buf);
48
49
50
51 const char *
52 vrna_cstr_string(vrna_cstr_t buf);
53
54
55 int
56 vrna_cstr_vprintf(vrna_cstr_t buf,
57                     const char *format,
58                     va_list args);
59
60
61 int
62 vrna_cstr_printf(vrna_cstr_t buf,
63                   const char *format,
64                   ...);
65
66
67 void
68 vrna_cstr_message_info(vrna_cstr_t buf,
69                         const char *format,
70                         ...);
71
72
73 void
74 vrna_cstr_message_vinfo(vrna_cstr_t buf,
75                          const char *format,
76                          va_list args);
77
78
79 void
80 vrna_cstr_message_warning(struct vrna_cstr_s *buf,
81                            const char *format,
82                            ...);
83
84
85 void
86 vrna_cstr_message_vwarning(struct vrna_cstr_s *buf,
87                            const char *format,
88                            va_list args);
89
90
91
92 void
93 vrna_cstr_print_fasta_header(vrna_cstr_t buf,
94                               const char *head);
95
96
97 void
98 vrna_cstr_printf_structure(struct vrna_cstr_s *buf,
99                           const char *structure,
100                          const char *format,
101                          ...);
102
103
104 void
105 vrna_cstr_vprintf_structure(struct vrna_cstr_s *buf,
106                             const char *structure,
107                             const char *format,
108                             va_list args);
109
110
111 void
112 vrna_cstr_printf_comment(struct vrna_cstr_s *buf,
113                           const char *format,
114                           ...);
115
116
117 void
118 vrna_cstr_vprintf_comment(struct vrna_cstr_s *buf,
119                           const char *format,
120                           va_list args);
121
122
123 void
124 vrna_cstr_vprintf_comment(struct vrna_cstr_s *buf,
125                           const char *format,
126                           va_list args);
127
128
129 void
130 vrna_cstr_vprintf_comment(struct vrna_cstr_s *buf,
131                           const char *format,
132                           va_list args);
133
134
135 void
136 vrna_cstr_vprintf_comment(struct vrna_cstr_s *buf,
137                           const char *format,
138                           va_list args);
139
140
141 void
142 vrna_cstr_vprintf_comment(struct vrna_cstr_s *buf,
143                           const char *format,
144                           va_list args);
145
146
147 void
148 vrna_cstr_vprintf_comment(struct vrna_cstr_s *buf,
149                           const char *format,
150                           va_list args);
151
152
153 void
154 vrna_cstr_vprintf_comment(struct vrna_cstr_s *buf,
155                           const char *format,
156                           ...);
157
```

```
158 vrna_cstr_vprintf_comment(struct vrna_cstr_s *buf,
159                         const char      *format,
160                         va_list        args);
161
162
163 void
164 vrna_cstr_printf_thead(struct vrna_cstr_s *buf,
165                         const char      *format,
166                         ...);
167
168
169 void
170 vrna_cstr_vprintf_thead(struct vrna_cstr_s *buf,
171                         const char      *format,
172                         va_list        args);
173
174
175 void
176 vrna_cstr_printf_tbody(struct vrna_cstr_s *buf,
177                         const char      *format,
178                         ...);
179
180
181 void
182 vrna_cstr_vprintf_tbody(struct vrna_cstr_s *buf,
183                         const char      *format,
184                         va_list        args);
185
186
187 void
188 vrna_cstr_print_eval_sd_corr(struct vrna_cstr_s *buf);
189
190
191 void
192 vrna_cstr_print_eval_ext_loop(struct vrna_cstr_s *buf,
193                               int             energy);
194
195
196 void
197 vrna_cstr_print_eval_hp_loop(struct vrna_cstr_s *buf,
198                               int             i,
199                               int             j,
200                               char            si,
201                               char            sj,
202                               int             energy);
203
204
205 void
206 vrna_cstr_print_eval_hp_loop_revert(struct vrna_cstr_s *buf,
207                                       int             i,
208                                       int             j,
209                                       char            si,
210                                       char            sj,
211                                       int             energy);
212
213
214 void
215 vrna_cstr_print_eval_int_loop(struct vrna_cstr_s *buf,
216                               int             i,
217                               int             j,
218                               char            si,
219                               char            sj,
220                               int             k,
221                               int             l,
222                               char            sk,
223                               char            sl,
224                               int             energy);
225
226
227 void
228 vrna_cstr_print_eval_int_loop_revert(struct vrna_cstr_s *buf,
229                                       int             i,
230                                       int             j,
231                                       char            si,
232                                       char            sj,
233                                       int             k,
234                                       int             l,
235                                       char            sk,
236                                       char            sl,
237                                       int             energy);
238
239
240 void
241 vrna_cstr_print_eval_mb_loop(struct vrna_cstr_s *buf,
242                               int             i,
243                               int             j,
244                               char            si,
```

```

245     char          sj,
246     int          energy);
247
248
249 void
250 vrna_cstr_print_eval_mb_loop_revert(struct vrna_cstr_s *buf,
251           int          i,
252           int          j,
253           char         si,
254           char         sj,
255           int          energy);
256
257
258 void
259 vrna_cstr_print_eval_gquad(struct vrna_cstr_s *buf,
260           int          i,
261           int          L,
262           int          l[3],
263           int          energy);
264
265
270 #endif

```

## 18.20 ViennaRNA/cofold.h File Reference

MFE implementations for RNA-RNA interaction.

Include dependency graph for cofold.h:

### Functions

- float [cofold](#) (const char \*sequence, char \*structure)  
*Compute the minimum free energy of two interacting RNA molecules.*
- float [cofold\\_par](#) (const char \*string, char \*structure, [vrna\\_param\\_t](#) \*parameters, int is\_constrained)  
*Compute the minimum free energy of two interacting RNA molecules.*
- void [free\\_co\\_arrays](#) (void)  
*Free memory occupied by [cofold\(\)](#)*
- void [update\\_cofold\\_params](#) (void)  
*Recalculate parameters.*
- void [update\\_cofold\\_params\\_par](#) ([vrna\\_param\\_t](#) \*parameters)  
*Recalculate parameters.*
- void [export\\_cofold\\_arrays\\_gq](#) (int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*fc\_p, int \*\*ggg\_p, int \*\*indx\_p, char \*\*ptype\_p)  
*Export the arrays of partition function cofold (with gquadruplex support)*
- void [export\\_cofold\\_arrays](#) (int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*fc\_p, int \*\*indx\_p, char \*\*ptype\_p)  
*Export the arrays of partition function cofold.*
- void [initialize\\_cofold](#) (int length)

### 18.20.1 Detailed Description

MFE implementations for RNA-RNA interaction.

## 18.21 cofold.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_COFOLD_H
2 #define VIENNA_RNA_PACKAGE_COFOLD_H
3
4 #include <ViennaRNA/datastructures/basic.h>
5 #include <ViennaRNA/params/basic.h>
6 #include <ViennaRNA/mfe.h>
7
8 #ifdef VRNA_WARN_DEPRECATED
9 # if defined(__clang__)
10 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))

```

```

11 # elif defined(__GNUC__)
12 # define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
13 # else
14 # define DEPRECATED(func, msg) func
15 # endif
16 #else
17 # define DEPRECATED(func, msg) func
18 #endif
19
26 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
27
42 DEPRECATED(float
43         cofold(const char *sequence,
44                 char *structure),
45         "Use vrna_cofold() instead");
46
54 DEPRECATED(float
55         cofold_par(const char *string,
56                     char *structure,
57                     vrna_param_t *parameters,
58                     int is_constrained),
59         "Use the new API and vrna_mfe_dimer() instead");
60
72 DEPRECATED(void
73         free_co_arrays(void),
74         "This function is obsolete");
75
82 DEPRECATED(void
83         update_cofold_params(void),
84         "This function is obsolete");
85
92 DEPRECATED(void
93         update_cofold_params_par(vrna_param_t *parameters),
94         "Use the new API with vrna_fold_compound_t instead");
95
96
118 DEPRECATED(void
119         export_cofold_arrays_gq(int **f5_p,
120                             int **c_p,
121                             int **fML_p,
122                             int **fM1_p,
123                             int **fc_p,
124                             int **ggg_p,
125                             int **indx_p,
126                             char **ptype_p),
127         "Use the new API with vrna_fold_compound_t instead");
128
149 DEPRECATED(void
150         export_cofold_arrays(int **f5_p,
151                             int **c_p,
152                             int **fML_p,
153                             int **fM1_p,
154                             int **fc_p,
155                             int **indx_p,
156                             char **ptype_p),
157         "Use the new API with vrna_fold_compound_t instead");
158
159
166 DEPRECATED(void
167         initialize_cofold(int length),
168         "This function is obsolete");
169
170 #endif
171
172 #endif

```

## 18.22 ViennaRNA/combinatorics.h File Reference

Various implementations that deal with combinatorial aspects of objects.

Include dependency graph for combinatorics.h:

### Functions

- `unsigned int ** vrna_enumerate_necklaces (const unsigned int *type_counts)`  
*Enumerate all necklaces with fixed content.*
- `unsigned int vrna_rotational_symmetry_num (const unsigned int *string, size_t string_length)`  
*Determine the order of rotational symmetry for a string of objects represented by natural numbers.*
- `unsigned int vrna_rotational_symmetry_pos_num (const unsigned int *string, size_t string_length, unsigned int **positions)`

*Determine the order of rotational symmetry for a string of objects represented by natural numbers.*

- `unsigned int vrna_rotational_symmetry (const char *string)`

*Determine the order of rotational symmetry for a NULL-terminated string of ASCII characters.*

- `unsigned int vrna_rotational_symmetry_pos (const char *string, unsigned int **positions)`

*Determine the order of rotational symmetry for a NULL-terminated string of ASCII characters.*

- `unsigned int vrna_rotational_symmetry_db (vrna_fold_compound_t *fc, const char *structure)`

*Determine the order of rotational symmetry for a dot-bracket structure.*

- `unsigned int vrna_rotational_symmetry_db_pos (vrna_fold_compound_t *fc, const char *structure, unsigned int **positions)`

*Determine the order of rotational symmetry for a dot-bracket structure.*

- `unsigned int ** vrna_n_multichoose_k (size_t n, size_t k)`

*Obtain a list of k-combinations with repetition ( $n$  multichoose  $k$ )*

- `unsigned int * vrna_boustrophedon (size_t start, size_t end)`

*Generate a sequence of Boustrophedon distributed numbers.*

- `unsigned int vrna_boustrophedon_pos (size_t start, size_t end, size_t pos)`

*Obtain the  $i$ -th element in a Boustrophedon distributed interval of natural numbers.*

### 18.22.1 Detailed Description

Various implementations that deal with combinatorial aspects of objects.

,

## 18.23 combinatorics.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_COMBINATORICS_H
2 #define VIENNA_RNA_PACKAGE_COMBINATORICS_H
3
10 #include <ViennaRNA/fold_compound.h>
11
33 unsigned int **
34 vrna_enumerate_necklaces(const unsigned int *type_counts);
35
36
56 unsigned int
57 vrna_rotational_symmetry_num(const unsigned int *string,
58                               size_t                      string_length);
59
60
87 unsigned int
88 vrna_rotational_symmetry_pos_num(const unsigned int *string,
89                                   size_t                      string_length,
90                                   unsigned int                **positions);
91
92
110 unsigned int
111 vrna_rotational_symmetry(const char *string);
112
113
138 unsigned int
139 vrna_rotational_symmetry_pos(const char   *string,
140                               unsigned int **positions);
141
142
165 unsigned int
166 vrna_rotational_symmetry_db(vrna_fold_compound_t  *fc,
167                               const char           *structure);
168
169
200 unsigned int
201 vrna_rotational_symmetry_db_pos(vrna_fold_compound_t  *fc,
202                                   const char           *structure,
203                                   unsigned int                **positions);
204
205
218 unsigned int **
219 vrna_n_multichoose_k(size_t  n,
220                       size_t  k);
221
222
245 unsigned int *

```

```

246 vrna_boustrophedon(size_t start,
247             size_t end);
248
249
261 unsigned int
262 vrna_boustrophedon_pos(size_t start,
263             size_t end,
264             size_t pos);
265
266
270 #endif

```

## 18.24 ViennaRNA/commands.h File Reference

Parse and apply different commands that alter the behavior of secondary structure prediction and evaluation.  
Include dependency graph for commands.h:

### Macros

- `#define VRNA_CMD_PARSE_HC 1U`  
*Command parse/apply flag indicating hard constraints.*
- `#define VRNA_CMD_PARSE_SC 2U`  
*Command parse/apply flag indicating soft constraints.*
- `#define VRNA_CMD_PARSE_UD 4U`  
*Command parse/apply flag indicating unstructured domains.*
- `#define VRNA_CMD_PARSE_SD 8U`  
*Command parse/apply flag indicating structured domains.*
- `#define VRNA_CMD_PARSE_DEFAULTS`  
*Command parse/apply flag indicating default set of commands.*

### Typedefs

- `typedef struct vrna_command_s *vrna_cmd_t`  
*A data structure that contains commands.*

### Functions

- `vrna_cmd_t vrna_file_commands_read (const char *filename, unsigned int options)`  
*Extract a list of commands from a command file.*
- `int vrna_file_commands_apply (vrna_fold_compound_t *vc, const char *filename, unsigned int options)`  
*Apply a list of commands from a command file.*
- `int vrna_commands_apply (vrna_fold_compound_t *vc, vrna_cmd_t commands, unsigned int options)`  
*Apply a list of commands to a vrna\_fold\_compound\_t.*
- `void vrna_commands_free (vrna_cmd_t commands)`  
*Free memory occupied by a list of commands.*

### 18.24.1 Detailed Description

Parse and apply different commands that alter the behavior of secondary structure prediction and evaluation.

, ,

## 18.25 commands.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_COMMANDS_H
2 #define VIENNA_RNA_PACKAGE_COMMANDS_H
3
18 typedef struct vrna_command_s *vrna_cmd_t;
19

```

```

20
21 #include <ViennaRNA/fold_compound.h>
22
27 #define VRNA_CMD_PARSE_HC      1U
32 #define VRNA_CMD_PARSE_SC      2U
37 #define VRNA_CMD_PARSE_UD      4U
42 #define VRNA_CMD_PARSE_SD      8U
47 #define VRNA_CMD_PARSE_DEFAULTS (VRNA_CMD_PARSE_HC \
48                                | VRNA_CMD_PARSE_SC \
49                                | VRNA_CMD_PARSE_UD \
50                                | VRNA_CMD_PARSE_SD \
51                                )
52
53 #define VRNA_CMD_PARSE_SILENT 16U
54
68 vrna_cmd_t
69 vrna_file_commands_read(const char *filename,
70                          unsigned int options);
71
72
86 int
87 vrna_file_commands_apply(vrna_fold_compound_t *vc,
88                           const char *filename,
89                           unsigned int options);
90
91
100 int
101 vrna_commands_apply(vrna_fold_compound_t *vc,
102                       vrna_cmd_t commands,
103                       unsigned int options);
104
105
112 void
113 vrna_commands_free(vrna_cmd_t commands);
114
115
120 #endif

```

## 18.26 ViennaRNA/concentrations.h File Reference

Concentration computations for RNA-RNA interactions.

Include dependency graph for concentrations.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [vrna\\_dimer\\_conc\\_s](#)  
*Data structure for concentration dependency computations.*

### Functions

- [vrna\\_dimer\\_conc\\_t \\* get\\_concentrations](#) (double FEAB, double FEAA, double FEBB, double FEA, double FEB, double \*startconc)  
*Given two start monomer concentrations a and b, compute the concentrations in thermodynamic equilibrium of all dimers and the monomers.*
- [typedef struct vrna\\_dimer\\_conc\\_s vrna\\_dimer\\_conc\\_t](#)  
*Typename for the data structure that stores the dimer concentrations, [vrna\\_dimer\\_conc\\_s](#), as required by [vrna\\_pf\\_dimer\\_concentration\(\)](#)*
- [typedef struct vrna\\_dimer\\_conc\\_s ConcEnt](#)  
*Backward compatibility typedef for [vrna\\_dimer\\_conc\\_s](#).*
- [vrna\\_dimer\\_conc\\_t \\* vrna\\_pf\\_dimer\\_concentrations](#) (double FcAB, double FcAA, double FcBB, double FEA, double FEB, const double \*startconc, const [vrna\\_exp\\_param\\_t](#) \*exp\_params)  
*Given two start monomer concentrations a and b, compute the concentrations in thermodynamic equilibrium of all dimers and the monomers.*

### 18.26.1 Detailed Description

Concentration computations for RNA-RNA interactions.

## 18.26.2 Function Documentation

### 18.26.2.1 get\_concentrations()

```
vrna_dimer_conc_t * get_concentrations (
    double FEAB,
    double FEAA,
    double FEBB,
    double FEA,
    double FEB,
    double * startconc )
```

Given two start monomer concentrations a and b, compute the concentrations in thermodynamic equilibrium of all dimers and the monomers.

This function takes an array 'startconc' of input concentrations with alternating entries for the initial concentrations of molecules A and B (terminated by two zeroes), then computes the resulting equilibrium concentrations from the free energies for the dimers. Dimer free energies should be the dimer-only free energies, i.e. the FcAB entries from the `vrna_dimer_pf_t` struct.

**Deprecated** { Use `vrna_pf_dimer_concentrations()` instead!}

#### Parameters

|                        |   |
|------------------------|---|
| <code>FEAB</code>      | Free energy of AB dimer (FcAB entry)                                  |
| <code>FEAA</code>      | Free energy of AA dimer (FcAB entry)                                  |
| <code>FEBB</code>      | Free energy of BB dimer (FcAB entry)                                  |
| <code>FEA</code>       | Free energy of monomer A  |
| <code>FEB</code>       | Free energy of monomer B  |
| <code>startconc</code> | List of start concentrations [a0],[b0],[a1],[b1],...,[an][bn],[0],[0] |

#### Returns

`vrna_dimer_conc_t` array containing the equilibrium energies and start concentrations

## 18.27 concentrations.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_CONCENTRATIONS_H
2 #define VIENNA_RNA_PACKAGE_CONCENTRATIONS_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 # define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 # define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 # define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
28 typedef struct vrna_dimer_conc_s vrna_dimer_conc_t;
29
30
31 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
32
36 typedef struct vrna_dimer_conc_s ConcEnt;
37
38 #endif
39
40 #include <ViennaRNA/params/basic.h>
41
45 struct vrna_dimer_conc_s {
46     double Ac_start;
```

```

47     double  Bc_start;
48     double  ABC;
49     double  AAC;
50     double  BBC;
51     double  Ac;
52     double  Bc;
53 };
54
55
75 vrna_dimer_conc_t *vrna_pf_dimer_concentrations(double
76                                     double
77                                     double
78                                     double
79                                     double
80                                     const double
81                                     const vrna_exp_param_t *startconc,
82                                     *exp_params);
83
84 double *
85 vrna_equilibrium_constants(const double      *dG_complexes,
86                           const double      *dG_strands,
87                           const unsigned int **A,
88                           double            kT,
89                           size_t             strands,
90                           size_t             complexes);
91
95 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
96
97 /*
98 ##### DEPRECATED FUNCTIONS #####
99 #endif
100 ##### DEPRECATED FUNCTIONS #####
101 */
102
123 DEPRECATED(vrna_dimer_conc_t *get_concentrations(double FEAB,
124                                         double FEA,
125                                         double FEBB,
126                                         double FEA,
127                                         double FEB,
128                                         double *startconc),
129                                         "Use vrna_pf_dimer_concentrations() instead");
130
131 #endif
132
133 #endif

```

## 18.28 ViennaRNA/constraints.h File Reference

Use [ViennaRNA/constraints/basic.h](#) instead.

Include dependency graph for constraints.h:

### 18.28.1 Detailed Description

Use [ViennaRNA/constraints/basic.h](#) instead.

**Deprecated** Use [ViennaRNA/constraints/basic.h](#) instead

## 18.29 constraints.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 #ifdef VRNA_WARN_DEPRECATED
12 #warning "Including deprecated header file <ViennaRNA/constraints.h>! Use <ViennaRNA/constraints/basic.h>
   instead!"
13 #endif
14 #include <ViennaRNA/constraints/basic.h>
15 #include <ViennaRNA/constraints/hard.h>
16 #include <ViennaRNA/constraints/soft.h>
17 #include <ViennaRNA/constraints/SHAPE.h>
18 #include <ViennaRNA/constraints/ligand.h>
19 #endif
20
21 #endif

```

## 18.30 ViennaRNA/constraints/hard.h File Reference

Functions and data structures for handling of secondary structure hard constraints.

Include dependency graph for hard.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct `vrna_hc_s`  
*The hard constraints data structure. [More...](#)*
- struct `vrna_hc_up_s`  
*A single hard constraint for a single nucleotide. [More...](#)*

### Macros

- `#define VRNA_CONSTRAINT_NO_HEADER 0`  
*do not print the header information line*
- `#define VRNA_CONSTRAINT_DB 16384U`  
*Flag for `vrna_constraints_add()` to indicate that constraint is passed in pseudo dot-bracket notation.*
- `#define VRNA_CONSTRAINT_DB_ENFORCE_BP 32768U`  
*Switch for dot-bracket structure constraint to enforce base pairs.*
- `#define VRNA_CONSTRAINT_DB_PIPE 65536U`  
*Flag that is used to indicate the pipe '|` sign in pseudo dot-bracket notation of hard constraints.*
- `#define VRNA_CONSTRAINT_DB_DOT 131072U`  
*dot '.' switch for structure constraints (no constraint at all)*
- `#define VRNA_CONSTRAINT_DB_X 262144U`  
*'x' switch for structure constraint (base must not pair)*
- `#define VRNA_CONSTRAINT_DB_ANG_BRACK 524288U`  
*angle brackets '<','>' switch for structure constraint (paired downstream/upstream)*
- `#define VRNA_CONSTRAINT_DB_RND_BRACK 1048576U`  
*round brackets '(',')' switch for structure constraint (base i pairs base j)*
- `#define VRNA_CONSTRAINT_DB_INTRAMOL 2097152U`  
*Flag that is used to indicate the character 'l' in pseudo dot-bracket notation of hard constraints.*
- `#define VRNA_CONSTRAINT_DB_INTERMOL 4194304U`  
*Flag that is used to indicate the character 'e' in pseudo dot-bracket notation of hard constraints.*
- `#define VRNA_CONSTRAINT_DB_GQUAD 8388608U`  
 *'+' switch for structure constraint (base is involved in a gquad)*
- `#define VRNA_CONSTRAINT_DB_WUSS 33554432U`  
*Flag to indicate Washington University Secondary Structure (WUSS) notation of the hard constraint string.*
- `#define VRNA_CONSTRAINT_DB_DEFAULT`  
*Switch for dot-bracket structure constraint with default symbols.*
- `#define VRNA_CONSTRAINT_CONTEXT_EXT_LOOP (unsigned char)0x01`  
*Hard constraints flag, base pair in the exterior loop.*
- `#define VRNA_CONSTRAINT_CONTEXT_HP_LOOP (unsigned char)0x02`  
*Hard constraints flag, base pair encloses hairpin loop.*
- `#define VRNA_CONSTRAINT_CONTEXT_INT_LOOP (unsigned char)0x04`  
*Hard constraints flag, base pair encloses an interior loop.*
- `#define VRNA_CONSTRAINT_CONTEXT_INT_LOOP_ENC (unsigned char)0x08`  
*Hard constraints flag, base pair encloses a multi branch loop.*
- `#define VRNA_CONSTRAINT_CONTEXT_MB_LOOP (unsigned char)0x10`  
*Hard constraints flag, base pair is enclosed in an interior loop.*
- `#define VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC (unsigned char)0x20`  
*Hard constraints flag, base pair is enclosed in a multi branch loop.*

- #define **VRNA\_CONSTRAINT\_CONTEXT\_ENFORCE** (unsigned char)0x40  
*Hard constraint flag to indicate enforcement of constraints.*
- #define **VRNA\_CONSTRAINT\_CONTEXT\_NO\_REMOVE** (unsigned char)0x80  
*Hard constraint flag to indicate not to remove base pairs that conflict with a given constraint.*
- #define **VRNA\_CONSTRAINT\_CONTEXT\_NONE** (unsigned char)0  
*Constraint context flag that forbids any loop.*
- #define **VRNA\_CONSTRAINT\_CONTEXT\_CLOSING\_LOOPS**  
*Constraint context flag indicating base pairs that close any loop.*
- #define **VRNA\_CONSTRAINT\_CONTEXT\_ENCLOSED\_LOOPS**  
*Constraint context flag indicating base pairs enclosed by any loop.*
- #define **VRNA\_CONSTRAINT\_CONTEXT\_ALL\_LOOPS**  
*Constraint context flag indicating any loop context.*

## Typedefs

- typedef struct [vrna\\_hc\\_s](#) **vrna\_hc\_t**  
*Typename for the hard constraints data structure [vrna\\_hc\\_s](#).*
- typedef struct [vrna\\_hc\\_up\\_s](#) **vrna\_hc\_up\_t**  
*Typename for the single nucleotide hard constraint data structure [vrna\\_hc\\_up\\_s](#).*
- typedef unsigned char() **vrna\_callback\_hc\_evaluate**(int i, int j, int k, int l, unsigned char d, void \*data)  
*Callback to evaluate whether or not a particular decomposition step is contributing to the solution space.*

## Enumerations

- enum [vrna\\_hc\\_type\\_e](#) { **VRNA\_HC\_DEFAULT** , **VRNA\_HC\_WINDOW** }  
*The hard constraints type.*

## Functions

- void **vrna\_message\_constraint\_options** (unsigned int option)  
*Print a help message for pseudo dot-bracket structure constraint characters to stdout. (constraint support is specified by option parameter)*
- void **vrna\_message\_constraint\_options\_all** (void)  
*Print structure constraint characters to stdout (full constraint support)*
- void **vrna\_hc\_init** ([vrna\\_fold\\_compound\\_t](#) \*vc)  
*Initialize/Reset hard constraints to default values.*
- void **vrna\_hc\_add\_up** ([vrna\\_fold\\_compound\\_t](#) \*vc, int i, unsigned char option)  
*Make a certain nucleotide unpaired.*
- int **vrna\_hc\_add\_up\_batch** ([vrna\\_fold\\_compound\\_t](#) \*vc, [vrna\\_hc\\_up\\_t](#) \*constraints)  
*Apply a list of hard constraints for single nucleotides.*
- int **vrna\_hc\_add\_bp** ([vrna\\_fold\\_compound\\_t](#) \*vc, int i, int j, unsigned char option)  
*Favorize/Enforce a certain base pair (i,j)*
- void **vrna\_hc\_add\_bp\_nonspecific** ([vrna\\_fold\\_compound\\_t](#) \*vc, int i, int d, unsigned char option)  
*Enforce a nucleotide to be paired (upstream/downstream)*
- void **vrna\_hc\_free** ([vrna\\_hc\\_t](#) \*hc)  
*Free the memory allocated by a [vrna\\_hc\\_t](#) data structure.*
- void **vrna\_hc\_add\_f** ([vrna\\_fold\\_compound\\_t](#) \*vc, [vrna\\_callback\\_hc\\_evaluate](#) \*f)  
*Add a function pointer pointer for the generic hard constraint feature.*
- void **vrna\_hc\_add\_data** ([vrna\\_fold\\_compound\\_t](#) \*vc, void \*data, [vrna\\_callback\\_free\\_auxdata](#) \*f)  
*Add an auxiliary data structure for the generic hard constraints callback function.*
- int **vrna\_hc\_add\_from\_db** ([vrna\\_fold\\_compound\\_t](#) \*vc, const char \*constraint, unsigned int options)  
*Add hard constraints from pseudo dot-bracket notation.*

- void [print\\_tty\\_constraint](#) (unsigned int option)  
*Print structure constraint characters to stdout. (constraint support is specified by option parameter)*
- void [print\\_tty\\_constraint\\_full](#) (void)  
*Print structure constraint characters to stdout (full constraint support)*
- void [constrain\\_ptypes](#) (const char \*constraint, unsigned int length, char \*ptype, int \*BP, int min\_loop\_size, unsigned int idx\_type)  
*Insert constraining pair types according to constraint structure string.*

### 18.30.1 Detailed Description

Functions and data structures for handling secondary structure hard constraints.

### 18.30.2 Macro Definition Documentation

#### 18.30.2.1 VRNA\_CONSTRAINT\_NO\_HEADER

```
#define VRNA_CONSTRAINT_NO_HEADER 0
do not print the header information line
```

**Deprecated** This mode is not supported anymore!

#### 18.30.2.2 VRNA\_CONSTRAINT\_DB\_ANG\_BRACK

```
#define VRNA_CONSTRAINT_DB_ANG_BRACK 524288U
angle brackets '<', '>' switch for structure constraint (paired downstream/upstream)
```

See also

[vrna\\_hc\\_add\\_from\\_db\(\)](#), [vrna\\_constraints\\_add\(\)](#), [vrna\\_message\\_constraint\\_options\(\)](#), [vrna\\_message\\_constraint\\_options\\_all\(\)](#)

### 18.30.3 Enumeration Type Documentation

#### 18.30.3.1 vrna\_hc\_type\_e

enum [vrna\\_hc\\_type\\_e](#)

The hard constraints type.

Global and local structure prediction methods use a slightly different way to handle hard constraints internally. This enum is used to distinguish both types.

Enumerator

|                 |  |
|-----------------|--|
| VRNA_HC_DEFAULT | Default Hard Constraints.  |
| VRNA_HC_WINDOW  | Hard Constraints suitable for local structure prediction using window approach.<br><br>See also<br><br><a href="#">vrna_mfe_window()</a> , <a href="#">vrna_mfe_window_zscore()</a> , <a href="#">pfl_fold()</a> |

### 18.30.4 Function Documentation

#### 18.30.4.1 vrna\_hc\_add\_data()

```
void vrna_hc_add_data (
    vrna_fold_compound_t * vc,
    void * data,
    vrna_callback_free_auxdata * f )
```

Add an auxiliary data structure for the generic hard constraints callback function.

See also

[vrna\\_hc\\_add\\_f\(\)](#)

#### Parameters

|             |   |
|-------------|---|
| <i>vc</i>   | The fold compound the generic hard constraint function should be bound to           |
| <i>data</i> | A pointer to the data structure that holds required data for function 'f'           |
| <i>f</i>    | A pointer to a function that free's the memory occupied by <i>data</i> (Maybe NULL) |

#### 18.30.4.2 print\_tty\_constraint()

```
void print_tty_constraint (
    unsigned int option )
```

Print structure constraint characters to stdout. (constraint support is specified by option parameter)

**Deprecated** Use [vrna\\_message\\_constraints\(\)](#) instead!

#### Parameters

|               |  |
|---------------|--|
| <i>option</i> | Option switch that tells which constraint help will be printed |
|---------------|--|

#### 18.30.4.3 print\_tty\_constraint\_full()

```
void print_tty_constraint_full (
    void )
```

Print structure constraint characters to stdout (full constraint support)

**Deprecated** Use [vrna\\_message\\_constraint\\_options\\_all\(\)](#) instead!

#### 18.30.4.4 constrain\_ptypes()

```
void constrain_ptypes (
    const char * constraint,
    unsigned int length,
    char * ptype,
    int * BP,
    int min_loop_size,
    unsigned int idx_type )
```

Insert constraining pair types according to constraint structure string.

**Deprecated** Do not use this function anymore! Structure constraints are now handled through [vrna\\_hc\\_t](#) and related functions.

**Parameters**

|                      |   |
|----------------------|---|
| <i>constraint</i>    | The structure constraint string                                       |
| <i>length</i>        | The actual length of the sequence (constraint may be shorter)         |
| <i>ptype</i>         | A pointer to the basepair type array                                  |
| <i>BP</i>            | (not used anymore)  |
| <i>min_loop_size</i> | The minimal loop size (usually TURN )                                 |
| <i>idx_type</i>      | Define the access type for base pair type array (0 = indx, 1 = iindx) |

**18.31 hard.h**[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_HARD_H
2 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_HARD_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 # define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 # define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 # define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
16
34 typedef struct vrna_hc_s vrna_hc_t;
35
40 typedef struct vrna_hc_up_s vrna_hc_up_t;
41
42 typedef struct vrna_hc_depot_s vrna_hc_depot_t;
43
44 #include <ViennaRNA/fold_compound.h>
45 #include <ViennaRNA/constraints/basic.h>
46
78 typedef unsigned char (vrna_callback_hc_evaluate)(int          i,
79                           int          j,
80                           int          k,
81                           int          l,
82                           unsigned char d,
83                           void        *data);
84
90 #define VRNA_CONSTRAINT_NO_HEADER      0
91
100 #define VRNA_CONSTRAINT_DB           16384U
101
113 #define VRNA_CONSTRAINT_DB_ENFORCE_BP    32768U
114
126 #define VRNA_CONSTRAINT_DB_PIPE       65536U
127
136 #define VRNA_CONSTRAINT_DB_DOT        131072U
145 #define VRNA_CONSTRAINT_DB_X         262144U
152 #define VRNA_CONSTRAINT_DB_ANG_BRACK 524288U
161 #define VRNA_CONSTRAINT_DB_RND_BRACK 1048576U
162
174 #define VRNA_CONSTRAINT_DB_INTRAMOL   2097152U
175
187 #define VRNA_CONSTRAINT_DB_INTERMOL   4194304U
188
199 #define VRNA_CONSTRAINT_DB_GQUAD     8388608U
200
201 #define VRNA_CONSTRAINT_DB_CANONICAL_BP 16777216U
202
211 #define VRNA_CONSTRAINT_DB_WUSS      33554432U
212
213
225 #define VRNA_CONSTRAINT_DB_DEFAULT \
226   (VRNA_CONSTRAINT_DB \
227   | VRNA_CONSTRAINT_DB_PIPE \
228   | VRNA_CONSTRAINT_DB_DOT \
229   | VRNA_CONSTRAINT_DB_X \
230   | VRNA_CONSTRAINT_DB_ANG_BRACK \
231   | VRNA_CONSTRAINT_DB_RND_BRACK \
232   | VRNA_CONSTRAINT_DB_INTRAMOL \
233   | VRNA_CONSTRAINT_DB_INTERMOL \

```

```

234     | VRNA_CONSTRAINT_DB_GQUAD \
235   )
236
243 #define VRNA_CONSTRAINT_CONTEXT_EXT_LOOP      (unsigned char)0x01
244
251 #define VRNA_CONSTRAINT_CONTEXT_HP_LOOP       (unsigned char)0x02
252
259 #define VRNA_CONSTRAINT_CONTEXT_INT_LOOP      (unsigned char)0x04
260
267 #define VRNA_CONSTRAINT_CONTEXT_INT_LOOP_ENC   (unsigned char)0x08
268
275 #define VRNA_CONSTRAINT_CONTEXT_MB_LOOP       (unsigned char)0x10
276
283 #define VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC   (unsigned char)0x20
284
288 #define VRNA_CONSTRAINT_CONTEXT_ENFORCE      (unsigned char)0x40
289
293 #define VRNA_CONSTRAINT_CONTEXT_NO_REMOVE    (unsigned char)0x80
294
295
299 #define VRNA_CONSTRAINT_CONTEXT_NONE        (unsigned char)0
300
304 #define VRNA_CONSTRAINT_CONTEXT_CLOSING_LOOPS (unsigned char)(VRNA_CONSTRAINT_CONTEXT_EXT_LOOP | \
305                                         VRNA_CONSTRAINT_CONTEXT_HP_LOOP | \
306                                         VRNA_CONSTRAINT_CONTEXT_INT_LOOP | \
307                                         VRNA_CONSTRAINT_CONTEXT_MB_LOOP)
308
312 #define VRNA_CONSTRAINT_CONTEXT_ENCLOSED_LOOPS (unsigned char)(VRNA_CONSTRAINT_CONTEXT_INT_LOOP_ENC | \
313                                         VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC)
314
321 #define VRNA_CONSTRAINT_CONTEXT_ALL_LOOPS     (unsigned char)(VRNA_CONSTRAINT_CONTEXT_CLOSING_LOOPS | \
322                                         VRNA_CONSTRAINT_CONTEXT_ENCLOSED_LOOPS)
323
324
325 #define VRNA_CONSTRAINT_WINDOW_UPDATE_5      1U
326
327 #define VRNA_CONSTRAINT_WINDOW_UPDATE_3      2U
328
335 typedef enum {
336   VRNA_HC_DEFAULT,
337   VRNA_HC_WINDOW
341 } vrna_hc_type_e;
342
343
368 struct vrna_hc_s {
369   vrna_hc_type_e type;
370   unsigned int n;
371
372   unsigned char state;
373
374 #ifndef VRNA_DISABLE_C11_FEATURES
375   /* C11 support for unnamed unions/structs */
376   union {
377     struct {
378 #endif
379     unsigned char *mx;
380 #ifndef VRNA_DISABLE_C11_FEATURES
381   };
382   struct {
383 #endif
384     unsigned char **matrix_local;
385 #ifndef VRNA_DISABLE_C11_FEATURES
386   };
387   };
388 #endif
389
390   int           *up_ext;
393   int           *up_hp;
396   int           *up_int;
399   int           *up_ml;
403   vrna_callback_hc_evaluate *f;
407   void          *data;
412   vrna_callback_free_auxdata *free_data;
423   vrna_hc_depot_t *depot;
424 };
425
431 struct vrna_hc_up_s {
432   int position;
433   int strand;
434   unsigned char options;
435 };
436
459 void vrna_message_constraint_options(unsigned int option);
460
461
472 void vrna_message_constraint_options_all(void);
473

```

```
474
489 void vrna_hc_init(vrna_fold_compound_t *vc);
490
491
492 void vrna_hc_init_window(vrna_fold_compound_t *vc);
493
494
495 int
496 vrna_hc_prepare(vrna_fold_compound_t *fc,
497                  unsigned int          options);
498
499 void
500 vrna_hc_update(vrna_fold_compound_t *fc,
501                  unsigned int          i,
502                  unsigned int          options);
503
504
519 void vrna_hc_add_up(vrna_fold_compound_t *vc,
520                      int                 i,
521                      unsigned char        option);
522
523
524 int
525 vrna_hc_add_up_strand(vrna_fold_compound_t *fc,
526                         unsigned int          i,
527                         unsigned int          strand,
528                         unsigned char        option);
529
539 int
540 vrna_hc_add_up_batch(vrna_fold_compound_t *vc,
541                        vrna_hc_up_t        *constraints);
542
543 int
544 vrna_hc_add_up_strand_batch(vrna_fold_compound_t *fc,
545                               vrna_hc_up_t        *constraints);
546
563 int vrna_hc_add_bp(vrna_fold_compound_t *vc,
564                      int                 i,
565                      int                 j,
566                      unsigned char        option);
567
568
569 int
570 vrna_hc_add_bp_strand(vrna_fold_compound_t *fc,
571                         unsigned int          i,
572                         unsigned int          strand_i,
573                         unsigned int          j,
574                         unsigned int          strand_j,
575                         unsigned char        option);
576
594 void vrna_hc_add_bp_nonspecific(vrna_fold_compound_t *vc,
595                                   int                 i,
596                                   int                 d,
597                                   unsigned char        option);
598
599
611 void vrna_hc_free(vrna_hc_t *hc);
612
613
618 void vrna_hc_add_f(vrna_fold_compound_t      *vc,
619                      vrna_callback_hc_evaluate *f);
620
621
633 void vrna_hc_add_data(vrna_fold_compound_t      *vc,
634                         void                *data,
635                         vrna_callback_free_auxdata *f);
636
637
656 int
657 vrna_hc_add_from_db(vrna_fold_compound_t    *vc,
658                      const char           *constraint,
659                      unsigned int          options);
660
661
662 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
663
671 DEPRECATED(void print_tty_constraint(unsigned int option),
672             "Use vrna_message_constraint_options() instead");
673
680 DEPRECATED(void print_tty_constraint_full(void),
681             "Use vrna_message_constraint_options_all() instead");
682
695 DEPRECATED(void constrain_ptypes(const char   *constraint,
696                                  unsigned int length,
697                                  char          *ptype,
698                                  int           *BP,
699                                  int           min_loop_size,
```

```

700         unsigned int idx_type),
701         "Use the new API and the hard constraint framework instead");
702
703 #endif
704
705 #endif

```

## 18.32 ViennaRNA/constraints/ligand.h File Reference

Functions for incorporation of ligands binding to hairpin and interior loop motifs using the soft constraints framework.  
Include dependency graph for ligand.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [vrna\\_sc\\_motif\\_s](#)

### Typedefs

- typedef struct [vrna\\_sc\\_motif\\_s](#) [vrna\\_sc\\_motif\\_t](#)  
*Type definition for soft constraint motif.*

### Functions

- int [vrna\\_sc\\_add\\_hi\\_motif](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, const char \*seq, const char \*structure, [FLT\\_OR\\_DBL](#) energy, unsigned int options)  
*Add soft constraints for hairpin or interior loop binding motif.*

#### 18.32.1 Detailed Description

Functions for incorporation of ligands binding to hairpin and interior loop motifs using the soft constraints framework.

## 18.33 ligand.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_LIGAND_H
2 #define VIENNA_RNA_PACKAGE_LIGAND_H
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19 typedef struct vrna_sc_motif_s vrna_sc_motif_t;
20
21 #include <ViennaRNA/datastructures/basic.h>
22 #include <ViennaRNA/fold_compound.h>
23
24 struct vrna_sc_motif_s {
25     int i;
26     int j;
27     int k;
28     int l;
29     int number;
30 };
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49 int
50 vrna_sc_add_hi_motif(vrna_fold_compound_t *fc,
51                     const char           *seq,
52                     const char           *structure,
53                     FLT\_OR\_DBL          energy,
54                     unsigned int          options);
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79 #endif

```

## 18.34 ViennaRNA/constraints/SHAPE.h File Reference

This module provides function to incorporate SHAPE reactivity data into the folding recursions by means of soft constraints.

Include dependency graph for SHAPE.h: This graph shows which files directly or indirectly include this file:

### Functions

- int [vrna\\_sc\\_add\\_SHAPE\\_deigan](#) (*vrna\_fold\_compound\_t* \*vc, const double \*reactivities, double m, double b, unsigned int options)
   
*Add SHAPE reactivity data as soft constraints (Deigan et al. method)*
- int [vrna\\_sc\\_add\\_SHAPE\\_deigan\\_ali](#) (*vrna\_fold\_compound\_t* \*vc, const char \*\*shape\_files, const int \*shape\_file\_association, double m, double b, unsigned int options)
   
*Add SHAPE reactivity data from files as soft constraints for consensus structure prediction (Deigan et al. method)*
- int [vrna\\_sc\\_add\\_SHAPE\\_zarringhalam](#) (*vrna\_fold\_compound\_t* \*vc, const double \*reactivities, double m, double default\_value, const char \*shape\_conversion, unsigned int options)
   
*Add SHAPE reactivity data as soft constraints (Zarringhalam et al. method)*
- int [vrna\\_sc\\_SHAPE\\_parse\\_method](#) (const char \*method\_string, char \*method, float \*param\_1, float \*param\_2)
   
*Parse a character string and extract the encoded SHAPE reactivity conversion method and possibly the parameters for conversion into pseudo free energies.*
- int [vrna\\_sc\\_SHAPE\\_to\\_pr](#) (const char \*shape\_conversion, double \*values, int length, double default\_value)
   
*Convert SHAPE reactivity values to probabilities for being unpaired.*

### 18.34.1 Detailed Description

This module provides function to incorporate SHAPE reactivity data into the folding recursions by means of soft constraints.

### 18.34.2 Function Documentation

#### 18.34.2.1 [vrna\\_sc\\_SHAPE\\_parse\\_method\(\)](#)

```
int vrna_sc_SHAPE_parse_method (
    const char * method_string,
    char * method,
    float * param_1,
    float * param_2 )
```

Parse a character string and extract the encoded SHAPE reactivity conversion method and possibly the parameters for conversion into pseudo free energies.

#### Parameters

|                      |  |
|----------------------|--|
| <i>method_string</i> | The string that contains the encoded SHAPE reactivity conversion method                                |
| <i>method</i>        | A pointer to the memory location where the method character will be stored                             |
| <i>param_1</i>       | A pointer to the memory location where the first parameter of the corresponding method will be stored  |
| <i>param_2</i>       | A pointer to the memory location where the second parameter of the corresponding method will be stored |

**Returns**

1 on successful extraction of the method, 0 on errors

## 18.35 SHAPE.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_SHAPE_H
2 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_SHAPE_H
3
4 #include <ViennaRNA/fold_compound.h>
5
23 void vrna_constraints_add_SHAPE(vrna_fold_compound_t *vc,
24                                 const char *shape_file,
25                                 const char *shape_method,
26                                 const char *shape_conversion,
27                                 int verbose,
28                                 unsigned int constraint_type);
29
30 void vrna_constraints_add_SHAPE_ali(vrna_fold_compound_t *vc,
31                                     const char *shape_method,
32                                     const char **shape_files,
33                                     const int *shape_file_association,
34                                     int verbose,
35                                     unsigned int constraint_type);
61 int vrna_sc_add_SHAPE_deigan( vrna_fold_compound_t *vc,
62                               const double *reactivities,
63                               double m,
64                               double b,
65                               unsigned int options);
66
79 int vrna_sc_add_SHAPE_deigan_ali( vrna_fold_compound_t *vc,
80                                   const char **shape_files,
81                                   const int *shape_file_association,
82                                   double m,
83                                   double b,
84                                   unsigned int options);
85
108 int vrna_sc_add_SHAPE_zarringhalam( vrna_fold_compound_t *vc,
109                                       const double *reactivities,
110                                       double b,
111                                       double default_value,
112                                       const char *shape_conversion,
113                                       unsigned int options);
114
127 int vrna_sc_SHAPE_parse_method( const char *method_string,
128                                 char *method,
129                                 float *param_1,
130                                 float *param_2);
131
146 int vrna_sc_SHAPE_to_pr(const char *shape_conversion,
147                           double *values,
148                           int length,
149                           double default_value);
150
151 #endif

```

## 18.36 ViennaRNA/constraints/soft.h File Reference

Functions and data structures for secondary structure soft constraints.

Include dependency graph for soft.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct `vrna_sc_bp_storage_t`  
*A base pair constraint.*
- struct `vrna_sc_s`  
*The soft constraints data structure.* [More...](#)

### Typedefs

- typedef struct `vrna_sc_s` `vrna_sc_t`  
*Typename for the soft constraints data structure* `vrna_sc_s`.

- `typedef int() vrna_callback_sc_energy(int i, int j, int k, int l, unsigned char d, void *data)`  
*Callback to retrieve pseudo energy contribution for soft constraint feature.*
- `typedef FLT_OR_DBL() vrna_callback_sc_exp_energy(int i, int j, int k, int l, unsigned char d, void *data)`  
*Callback to retrieve pseudo energy contribution as Boltzmann Factors for soft constraint feature.*
- `typedef vrna_basepair_t *() vrna_callback_sc_backtrack(int i, int j, int k, int l, unsigned char d, void *data)`  
*Callback to retrieve auxiliary base pairs for soft constraint feature.*

## Enumerations

- enum `vrna_sc_type_e { VRNA_SC_DEFAULT , VRNA_SC_WINDOW }`  
*The type of a soft constraint.*

## Functions

- `void vrna_sc_init (vrna_fold_compound_t *vc)`  
*Initialize an empty soft constraints data structure within a `vrna_fold_compound_t`.*
- `int vrna_sc_set_bp (vrna_fold_compound_t *vc, const FLT_OR_DBL **constraints, unsigned int options)`  
*Set soft constraints for paired nucleotides.*
- `int vrna_sc_add_bp (vrna_fold_compound_t *vc, int i, int j, FLT_OR_DBL energy, unsigned int options)`  
*Add soft constraints for paired nucleotides.*
- `int vrna_sc_set_up (vrna_fold_compound_t *vc, const FLT_OR_DBL *constraints, unsigned int options)`  
*Set soft constraints for unpaired nucleotides.*
- `int vrna_sc_add_up (vrna_fold_compound_t *vc, int i, FLT_OR_DBL energy, unsigned int options)`  
*Add soft constraints for unpaired nucleotides.*
- `void vrna_sc_remove (vrna_fold_compound_t *vc)`  
*Remove soft constraints from `vrna_fold_compound_t`.*
- `void vrna_sc_free (vrna_sc_t *sc)`  
*Free memory occupied by a `vrna_sc_t` data structure.*
- `int vrna_sc_add_data (vrna_fold_compound_t *vc, void *data, vrna_callback_free_auxdata *free_data)`  
*Add an auxiliary data structure for the generic soft constraints callback function.*
- `int vrna_sc_add_f (vrna_fold_compound_t *vc, vrna_callback_sc_energy *f)`  
*Bind a function pointer for generic soft constraint feature (MFE version)*
- `int vrna_sc_add_bt (vrna_fold_compound_t *vc, vrna_callback_sc_backtrack *f)`  
*Bind a backtracking function pointer for generic soft constraint feature.*
- `int vrna_sc_add_exp_f (vrna_fold_compound_t *vc, vrna_callback_sc_exp_energy *exp_f)`  
*Bind a function pointer for generic soft constraint feature (PF version)*

### 18.36.1 Detailed Description

Functions and data structures for secondary structure soft constraints.

### 18.36.2 Enumeration Type Documentation

#### 18.36.2.1 vrna\_sc\_type\_e

enum `vrna_sc_type_e`

The type of a soft constraint.

Enumerator

|                              |                           |
|------------------------------|---------------------------|
| <code>VRNA_SC_DEFAULT</code> | Default Soft Constraints. |
|------------------------------|---------------------------|

## Enumerator

|                |  |
|----------------|--|
| VRNA_SC_WINDOW | <p>Soft Constraints suitable for local structure prediction using window approach.</p> <p><b>See also</b></p> <p><a href="#">vrna_mfe_window()</a>, <a href="#">vrna_mfe_window_zscore()</a>, <a href="#">pfl_fold()</a></p> |
|----------------|--|

## 18.37 soft.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_SOFT_H
2 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_SOFT_H
3
4 typedef struct vrna_sc_s vrna_sc_t;
5
6 #include <ViennaRNA/datastructures/basic.h>
7 #include <ViennaRNA/fold_compound.h>
8 #include <ViennaRNA/constraints/basic.h>
9
10 typedef int (vrna_callback_sc_energy)(int i,
11                            int j,
12                            int k,
13                            int l,
14                            unsigned char d,
15                            void *data);
16
17
18 typedef FLT_OR_DBL (vrna_callback_sc_exp_energy)(int i,
19                            int j,
20                            int k,
21                            int l,
22                            unsigned char d,
23                            void *data);
24
25
26 typedef vrna_basepair_t *(vrna_callback_sc_backtrack)(int i,
27                            int j,
28                            int k,
29                            int l,
30                            unsigned char d,
31                            void *data);
32
33
34 typedef enum {
35   VRNA_SC_DEFAULT,
36   VRNA_SC_WINDOW
37 } vrna_sc_type_e;
38
39
40 typedef struct {
41   unsigned int interval_start;
42   unsigned int interval_end;
43   int e;
44 } vrna_sc_bp_storage_t;
45
46
47 struct vrna_sc_s {
48   const vrna_sc_type_e type;
49   unsigned int n;
50
51   unsigned char state;
52
53   int **energy_up;
54   FLT_OR_DBL **exp_energy_up;
55   int *up_storage;
56   vrna_sc_bp_storage_t **bp_storage;
57 #ifndef VRNA_DISABLE_C11_FEATURES
58   /* C11 support for unnamed unions/structs */
59   union {
60     struct {
61       int *energy_bp;
62       FLT_OR_DBL *exp_energy_bp;
63     };
64     struct {
65       int **energy_bp_local;
66       FLT_OR_DBL **exp_energy_bp_local;
67     };
68   };
69 #endif
70 #ifndef VRNA_DISABLE_C11_FEATURES
71   /* C11 support for unnamed unions/structs */
72   union {
73     struct {
74       int *energy_bp;
75       FLT_OR_DBL *exp_energy_bp;
76     };
77     struct {
78       int **energy_bp_local;
79       FLT_OR_DBL **exp_energy_bp_local;
80     };
81   };
82 #endif
83 #endif
84
85
86
87
88
89
90
91
92
93
94
95
96
97

```

```
198     /* C11 support for unnamed unions/structs */
199     };
200 }
201 #endif
202
203 int           *energy_stack;
204 FLT_OR_DBL    *exp_energy_stack;
205 /* generic soft constraints below */
206 vrna_callback_sc_energy   *f;
207 vrna_callback_sc_backtrack *bt;
208 vrna_callback_sc_exp_energy *exp_f;
209 void           *data;
210 vrna_callback_free_auxdata *free_data;
211 };
212
213 void vrna_sc_init(vrna_fold_compound_t *vc);
214
215 void
216 vrna_sc_prepare(vrna_fold_compound_t *vc,
217                   unsigned int      options);
218
219
220
221 int
222 vrna_sc_update(vrna_fold_compound_t *vc,
223                  unsigned int      i,
224                  unsigned int      options);
225
226
227 int
228 vrna_sc_set_bp(vrna_fold_compound_t *vc,
229                   const FLT_OR_DBL *constraints,
230                   unsigned int      options);
231
232
233
234 int
235 vrna_sc_add_bp(vrna_fold_compound_t *vc,
236                   int             i,
237                   int             j,
238                   FLT_OR_DBL     energy,
239                   unsigned int    options);
240
241
242
243 int
244 vrna_sc_add_up(vrna_fold_compound_t *vc,
245                   int             i,
246                   FLT_OR_DBL     energy,
247                   unsigned int    options);
248
249
250
251 int
252 vrna_sc_set_up(vrna_fold_compound_t *vc,
253                   const FLT_OR_DBL *constraints,
254                   unsigned int      options);
255
256
257
258 int
259 vrna_sc_set_stack(vrna_fold_compound_t *vc,
260                     const FLT_OR_DBL *constraints,
261                     unsigned int      options);
262
263
264
265 int
266 vrna_sc_set_stack_comparative(vrna_fold_compound_t *fc,
267                                 const FLT_OR_DBL *constraints,
268                                 unsigned int      options);
269
270
271
272 int
273 vrna_sc_add_stack(vrna_fold_compound_t *vc,
274                     int             i,
275                     FLT_OR_DBL     energy,
276                     unsigned int    options);
277
278
279
280 int
281 vrna_sc_add_stack_comparative(vrna_fold_compound_t *fc,
282                                 int             i,
283                                 const FLT_OR_DBL *energies,
284                                 unsigned int    options);
285
286
287
288 void
289 vrna_sc_remove(vrna_fold_compound_t *vc);
290
291 void
```

```

392 vrna_sc_free(vrna_sc_t *sc);
393
394
407 int
408 vrna_sc_add_data(vrna_fold_compound_t      *vc,
409                     void                  *data,
410                     vrna_callback_free_auxdata *free_data);
411
412
413 int
414 vrna_sc_add_data_comparative(vrna_fold_compound_t      *vc,
415                                 void                  **data,
416                                 vrna_callback_free_auxdata **free_data);
417
418
419
420 int
421 vrna_sc_add_f(vrna_fold_compound_t      *vc,
422                 vrna_callback_sc_energy *f);
423
424
425 int
426 vrna_sc_add_f_comparative(vrna_fold_compound_t      *vc,
427                             vrna_callback_sc_energy **f);
428
429
430 int
431 vrna_sc_add_bt(vrna_fold_compound_t      *vc,
432                  vrna_callback_sc_backtrack *f);
433
434
435 int
436 vrna_sc_add_exp_f(vrna_fold_compound_t      *vc,
437                      vrna_callback_sc_exp_energy *exp_f);
438
439
440 int
441 vrna_sc_add_exp_f_comparative(vrna_fold_compound_t      *vc,
442                                     vrna_callback_sc_exp_energy **exp_f);
443
444
445 #endif

```

## 18.38 ViennaRNA/constraints\_hard.h File Reference

Use [ViennaRNA/constraints/hard.h](#) instead.

Include dependency graph for constraints\_hard.h:

### 18.38.1 Detailed Description

Use [ViennaRNA/constraints/hard.h](#) instead.

**Deprecated** Use [ViennaRNA/constraints/hard.h](#) instead

## 18.39 constraints\_hard.h

Go to the documentation of this file.

```

1 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_HARD_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_HARD_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 # ifdef VRNA_WARN_DEPRECATED
6 # warning "Including deprecated header file <ViennaRNA/constraints_hard.h>! Use
7     <ViennaRNA/constraints/hard.h> instead!"
8 # endif
9 #include <ViennaRNA/constraints/hard.h>
10#endif
11#endif
12#endif
13#endif
14#endif
15#endif
16#endif
17#endif

```

## 18.40 ViennaRNA/constraints\_ligand.h File Reference

Use [ViennaRNA/constraints/ligand.h](#) instead.

Include dependency graph for constraints\_ligand.h:

### 18.40.1 Detailed Description

Use [ViennaRNA/constraints/ligand.h](#) instead.

**Deprecated** Use [ViennaRNA/constraints/ligand.h](#) instead

## 18.41 constraints\_ligand.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_LIGAND_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_LIGAND_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 # ifdef VRNA_WARN_DEPRECATED
6 # warning "Including deprecated header file <ViennaRNA/constraints_ligand.h>! Use
7     <ViennaRNA/constraints/ligand.h> instead!"
8 # endif
9 #include <ViennaRNA/constraints/ligand.h>
10#endif
11
12#endif
```

## 18.42 ViennaRNA/constraints\_SHAPE.h File Reference

Use [ViennaRNA/constraints/SHAPE.h](#) instead.

Include dependency graph for constraints\_SHAPE.h:

### 18.42.1 Detailed Description

Use [ViennaRNA/constraints/SHAPE.h](#) instead.

**Deprecated** Use [ViennaRNA/constraints/SHAPE.h](#) instead

## 18.43 constraints\_SHAPE.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_SHAPE_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_SHAPE_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 # ifdef VRNA_WARN_DEPRECATED
6 # warning "Including deprecated header file <ViennaRNA/constraints_SHAPE.h>! Use
7     <ViennaRNA/constraints/SHAPE.h> instead!"
8 # endif
9 #include <ViennaRNA/constraints/SHAPE.h>
10#endif
11
12#endif
```

## 18.44 ViennaRNA/constraints\_soft.h File Reference

Use [ViennaRNA/constraints/soft.h](#) instead.

Include dependency graph for constraints\_soft.h:

### 18.44.1 Detailed Description

Use [ViennaRNA/constraints/soft.h](#) instead.

**Deprecated** Use [ViennaRNA/constraints/soft.h](#) instead

## 18.45 constraints\_soft.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_SOFT_DEPRECATED_H
```

```

2 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_SOFT_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 #warning "Including deprecated header file <ViennaRNA/constraints_soft.h>! Use
           <ViennaRNA/constraints/soft.h> instead!"
13 # endif
14 #include <ViennaRNA/constraints/soft.h>
15 #endif
16
17 #endif

```

## 18.46 ViennaRNA/convert\_epars.h File Reference

Use [ViennaRNA/params/convert.h](#) instead.

Include dependency graph for convert\_epars.h:

### 18.46.1 Detailed Description

Use [ViennaRNA/params/convert.h](#) instead.

**Deprecated** Use [ViennaRNA/params/convert.h](#) instead

## 18.47 convert\_epars.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PARAMS_CONVERT_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_PARAMS_CONVERT_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 #warning "Including deprecated header file <ViennaRNA/convert_epars.h>! Use <ViennaRNA/params/convert.h>
           instead!"
13 # endif
14 #include <ViennaRNA/params/convert.h>
15 #endif
16
17 #endif

```

## 18.48 ViennaRNA/data\_structures.h File Reference

Use [ViennaRNA/datastructures/basic.h](#) instead.

Include dependency graph for data\_structures.h:

### 18.48.1 Detailed Description

Use [ViennaRNA/datastructures/basic.h](#) instead.

**Deprecated** Use [ViennaRNA/datastructures/basic.h](#) instead

## 18.49 data\_structures.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_DATA_STRUCTURES_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_DATA_STRUCTURES_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 #warning "Including deprecated header file <ViennaRNA/data_structures.h>! Use
           <ViennaRNA/datastructures/basic.h> instead!"
13 # endif
14 #include <ViennaRNA/datastructures/basic.h>
15 #endif
16
17 #endif

```

## 18.50 ViennaRNA/datastructures/hash\_tables.h File Reference

Implementations of hash table functions.

### Data Structures

- struct [vrna\\_ht\\_entry\\_db\\_t](#)  
*Default hash table entry. [More...](#)*

### Functions

#### Dot-Bracket / Free Energy entries

- int [vrna\\_ht\\_db\\_comp](#) (void \*x, void \*y)  
*Default hash table entry comparison.*
- unsigned int [vrna\\_ht\\_db\\_hash\\_func](#) (void \*x, unsigned long hashtable\_size)  
*Default hash function.*
- int [vrna\\_ht\\_db\\_free\\_entry](#) (void \*hash\_entry)  
*Default function to free memory occupied by a hash entry.*

### Abstract interface

- [typedef struct vrna\\_hash\\_table\\_s \\* vrna\\_hash\\_table\\_t](#)  
*A hash table object.*
- [typedef int\(\) vrna\\_callback\\_ht\\_compare\\_entries](#)(void \*x, void \*y)  
*Callback function to compare two hash table entries.*
- [typedef unsigned int\(\) vrna\\_callback\\_ht\\_hash\\_function](#)(void \*x, unsigned long hashtable\_size)  
*Callback function to generate a hash key, i.e. hash function.*
- [typedef int\(\) vrna\\_callback\\_ht\\_free\\_entry](#)(void \*x)  
*Callback function to free a hash table entry.*
- [vrna\\_hash\\_table\\_t vrna\\_ht\\_init](#) (unsigned int b, [vrna\\_callback\\_ht\\_compare\\_entries](#) \*compare\_function, [vrna\\_callback\\_ht\\_hash\\_function](#) \*hash\_function, [vrna\\_callback\\_ht\\_free\\_entry](#) \*free\_hash\_entry)  
*Get an initialized hash table.*
- [unsigned long vrna\\_ht\\_size](#) ([vrna\\_hash\\_table\\_t](#) ht)  
*Get the size of the hash table.*
- [unsigned long vrna\\_ht\\_collisions](#) ([struct vrna\\_hash\\_table\\_s](#) \*ht)  
*Get the number of collisions in the hash table.*
- [void \\* vrna\\_ht\\_get](#) ([vrna\\_hash\\_table\\_t](#) ht, void \*x)  
*Get an element from the hash table.*
- [int vrna\\_ht\\_insert](#) ([vrna\\_hash\\_table\\_t](#) ht, void \*x)  
*Insert an object into a hash table.*
- [void vrna\\_ht\\_remove](#) ([vrna\\_hash\\_table\\_t](#) ht, void \*x)  
*Remove an object from the hash table.*
- [void vrna\\_ht\\_clear](#) ([vrna\\_hash\\_table\\_t](#) ht)  
*Clear the hash table.*
- [void vrna\\_ht\\_free](#) ([vrna\\_hash\\_table\\_t](#) ht)  
*Free all memory occupied by the hash table.*

#### 18.50.1 Detailed Description

Implementations of hash table functions.

## 18.51 hash\_tables.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_HASH_UTIL_H
2 #define VIENNA_RNA_PACKAGE_HASH_UTIL_H
3
4 /* Taken from the barriers tool and modified by GE. */
5
65 typedef struct vrna_hash_table_s *vrna_hash_table_t;
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94 vrna_hash_table_t
95 vrna_ht_init(unsigned int b,
96                 vrna_callback_ht_compare_entries *compare_function,
97                 vrna_callback_ht_hash_function *hash_function,
98                 vrna_callback_ht_free_entry *free_hash_entry);
99
100
101 unsigned long
102 vrna_ht_size(vrna_hash_table_t ht);
103
104
105 unsigned long
106 vrna_ht_collisions(struct vrna_hash_table_s *ht);
107
108
109
110
111
112
113 void *
114 vrna_ht_get(vrna_hash_table_t ht,
115               void *x);
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135 void *
136 vrna_ht_insert(vrna_hash_table_t ht,
137                  void *x);
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155 void
156 vrna_ht_remove(vrna_hash_table_t ht,
157                   void *x);
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187 void
188 vrna_ht_clear(vrna_hash_table_t ht);
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205 /* End of abstract interface */
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274 /* End of dot-bracket interface */
275
276
277
278
279
280
281 #endif
```

## 18.52 ViennaRNA/datastructures/heap.h File Reference

Implementation of an abstract heap data structure.

### Typedefs

- `typedef struct vrna_heap_s *vrna_heap_t`  
*An abstract heap data structure.*
- `typedef int() vrna_callback_heap_cmp(const void *a, const void *b, void *data)`  
*Heap compare function prototype.*
- `typedef size_t() vrna_callback_heap_get_pos(const void *a, void *data)`  
*Retrieve the position of a particular heap entry within the heap.*
- `typedef void() vrna_callback_heap_set_pos(const void *a, size_t pos, void *data)`  
*Store the position of a particular heap entry within the heap.*

### Functions

- `vrna_heap_t vrna_heap_init(size_t n, vrna_callback_heap_cmp *cmp, vrna_callback_heap_get_pos *get_entry_pos, vrna_callback_heap_set_pos *set_entry_pos, void *data)`  
*Initialize a heap data structure.*
- `void vrna_heap_free(vrna_heap_t h)`  
*Free memory occupied by a heap data structure.*
- `size_t vrna_heap_size(struct vrna_heap_s *h)`  
*Get the size of a heap data structure, i.e. the number of stored elements.*
- `void vrna_heap_insert(vrna_heap_t h, void *v)`  
*Insert an element into the heap.*
- `void *vrna_heap_pop(vrna_heap_t h)`  
*Pop (remove and return) the object at the root of the heap.*
- `const void *vrna_heap_top(vrna_heap_t h)`  
*Get the object at the root of the heap.*
- `void *vrna_heap_remove(vrna_heap_t h, const void *v)`  
*Remove an arbitrary element within the heap.*
- `void *vrna_heap_update(vrna_heap_t h, void *v)`  
*Update an arbitrary element within the heap.*

### 18.52.1 Detailed Description

Implementation of an abstract heap data structure.

## 18.53 heap.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_HEAP_H
2 #define VIENNA_RNA_PACKAGE_HEAP_H
3
44 typedef int (vrna_callback_heap_cmp)(const void *a,
45                                     const void *b,
46                                     void      *data);
47
48
56 typedef size_t (vrna_callback_heap_get_pos)(const void *a,
57                                              void      *data);
58
59
67 typedef void (vrna_callback_heap_set_pos)(const void *a,
68                                         size_t    pos,
69                                         void      *data);

```

```

70
71
105 vrna_heap_t
106 vrna_heap_init(size_t n,
107                 vrna_callback_heap_cmp *cmp,
108                 vrna_callback_heap_get_pos *get_entry_pos,
109                 vrna_callback_heap_set_pos *set_entry_pos,
110                 void *data);
111
112
120 void
121 vrna_heap_free(vrna_heap_t h);
122
123
130 size_t
131 vrna_heap_size(struct vrna_heap_s *h);
132
133
143 void
144 vrna_heap_insert(vrna_heap_t h,
145                     void *v);
146
147
159 void *
160 vrna_heap_pop(vrna_heap_t h);
161
162
172 const void *
173 vrna_heap_top(vrna_heap_t h);
174
175
189 void *
190 vrna_heap_remove(vrna_heap_t h,
191                     const void *v);
192
193
210 void *
211 vrna_heap_update(vrna_heap_t h,
212                     void *v);
213
214
219 #endif

```

## 18.54 lists.h

```

1 /*
2  $Log: lists.h,v $
3  Revision 1.2  2000/10/10 08:50:01  ivo
4  some annotation for lclint
5
6  Revision 1.1  1997/08/04 21:05:32  walter
7  Initial revision
8
9 */
10
11 #ifndef __LIST_H
12 #define __LIST_H
13
14 /*----- Macros and type definitions -----*/
15
16 typedef struct LST_BUCKET {
17     struct LST_BUCKET *next;
18 }
19 LST_BUCKET;
20
21 typedef struct {
22     int count;          /* Number of elements currently in list */
23     LST_BUCKET *head;   /* Pointer to head element of list */
24     LST_BUCKET *z;      /* Pointer to last node of list */
25     LST_BUCKET hz[2];   /* Space for head and z nodes */
26 }
27 LIST;
28
29 /* Return a pointer to the user space given the address of the header of
30 * a node.
31 */
32
33 #define LST_USERSPACE(h) ((void*)((LST_BUCKET*)(h) + 1))
34
35 /* Return a pointer to the header of a node, given the address of the
36 * user space.
37 */
38
39 #define LST_HEADER(n) ((LST_BUCKET*)(n) - 1)
40

```

```

41 /* Return a pointer to the user space of the list's head node. This user
42 * space does not actually exist, but it is useful to be able to address
43 * it to enable insertion at the start of the list.
44 */
45
46 #define LST_HEAD(l) LST_USERSPACE((l)->head)
47
48 /* Determine if a list is empty
49 */
50
51 #define LST_EMPTY(l) ((l)->count == 0)
52
53 /***** Function Prototypes *****/
54
55 /*@only@//@out@*/ void *lst_newnode (int size);
56 void lst_freenode /*@only@*/ void *node);
57 /*@only@//@out@*/ LIST *lst_init (void);
58 void lst_kill (LIST * l, void (*freeNode) ());
59 void lst_insertafter (LIST * l, /*@keep@*/ void *node, void *after);
60 void *lst_deletenext /*@only@*/ LIST * l, void *node);
61 /*@dependent@*/ void *lst_first (LIST * l);
62 /*@dependent@*/ void *lst_next (void *prev);
63 void lst_mergesort (LIST * l, int (*cmp_func) ());
64
65 #endif

```

## 18.55 ViennaRNA/dist\_vars.h File Reference

Global variables for Distance-Package.

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [Postorder\\_list](#)  
*Postorder data structure.*
- struct [Tree](#)  
*Tree data structure.*
- struct [swString](#)  
*Some other data structure.*

### Variables

- int [edit\\_backtrack](#)  
*Produce an alignment of the two structures being compared by tracing the editing path giving the minimum distance.*
- char \* [aligned\\_line](#) [4]  
*Contains the two aligned structures after a call to one of the distance functions with [edit\\_backtrack](#) set to 1.*
- int [cost\\_matrix](#)  
*Specify the cost matrix to be used for distance calculations.*

#### 18.55.1 Detailed Description

Global variables for Distance-Package.

#### 18.55.2 Variable Documentation

##### 18.55.2.1 [edit\\_backtrack](#)

```
int edit_backtrack [extern]
```

Produce an alignment of the two structures being compared by tracing the editing path giving the minimum distance.  
set to 1 if you want backtracking

### 18.55.2.2 cost\_matrix

```
int cost_matrix [extern]
```

Specify the cost matrix to be used for distance calculations.

if 0, use the default cost matrix (upper matrix in example), otherwise use Shapiro's costs (lower matrix).

## 18.56 dist\_vars.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_DIST_VARS_H
2 #define VIENNA_RNA_PACKAGE_DIST_VARS_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5
6 extern int edit_backtrack;
7
8 extern char *aligned_line[4];
9
10 extern int cost_matrix;
11
12 /* Global type defs for Distance-Package */
13
14 typedef struct {
15     int type;
16     int weight;
17     int father;
18     int sons;
19     int leftmostleaf;
20 } Postorder_list;
21
22 typedef struct {
23     Postorder_list *postorder_list;
24     int *keyroots;
25 } Tree;
26
27 typedef struct {
28     int type;
29     int sign;
30     float weight;
31 } swString;
32
33#endif
34
35#endif
```

## 18.57 ViennaRNA/dp\_matrices.h File Reference

Functions to deal with standard dynamic programming (DP) matrices.

Include dependency graph for dp\_matrices.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [vrna\\_mx\\_mfe\\_s](#)

*Minimum Free Energy (MFE) Dynamic Programming (DP) matrices data structure required within the vrna\_fold\_compound\_t. More...*

- struct [vrna\\_mx\\_pf\\_s](#)

*Partition function (PF) Dynamic Programming (DP) matrices data structure required within the vrna\_fold\_compound\_t. More...*

### Typedefs

- typedef struct [vrna\\_mx\\_mfe\\_s](#) [vrna\\_mx\\_mfe\\_t](#)

*Typename for the Minimum Free Energy (MFE) DP matrices data structure vrna\_mx\_mfe\_s.*

- typedef struct [vrna\\_mx\\_pf\\_s](#) [vrna\\_mx\\_pf\\_t](#)

*Typename for the Partition Function (PF) DP matrices data structure vrna\_mx\_pf\_s.*

## Enumerations

- enum `vrna_mx_type_e` { `VRNA_MX_DEFAULT` , `VRNA_MX_WINDOW` , `VRNA_MX_2DFOLD` }

*An enumerator that is used to specify the type of a polymorphic Dynamic Programming (DP) matrix data structure.*

## Functions

- int `vrna_mx_add` (`vrna_fold_compound_t` \*`vc`, `vrna_mx_type_e` `type`, unsigned int `options`)  
*Add Dynamic Programming (DP) matrices (allocate memory)*
- void `vrna_mx_mfe_free` (`vrna_fold_compound_t` \*`vc`)  
*Free memory occupied by the Minimum Free Energy (MFE) Dynamic Programming (DP) matrices.*
- void `vrna_mx_pf_free` (`vrna_fold_compound_t` \*`vc`)  
*Free memory occupied by the Partition Function (PF) Dynamic Programming (DP) matrices.*

### 18.57.1 Detailed Description

Functions to deal with standard dynamic programming (DP) matrices.

## 18.58 dp\_matrices.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_DP_MATRICES_H
2 #define VIENNA_RNA_PACKAGE_DP_MATRICES_H
3
40 typedef struct vrna_mx_mfe_s vrna_mx_mfe_t;
41 typedef struct vrna_mx_pf_s vrna_mx_pf_t;
42
43 #include <ViennaRNA/datastructures/basic.h>
44 #include <ViennaRNA/fold_compound.h>
45
46 typedef enum {
47     VRNA_MX_DEFAULT,
48     VRNA_MX_WINDOW,
49     VRNA_MX_2DFOLD
50 } vrna_mx_type_e;
51
52 struct vrna_mx_mfe_s {
53     const vrna_mx_type_e type;
54     unsigned int length;
55     unsigned int strands;
56 #ifndef VRNA_DISABLE_C11_FEATURES
57     /* C11 support for unnamed unions/structs */
58     union {
59         struct {
60             int *c;
61             int *f5;
62             int *f3;
63             int **fms5;
64             int **fms3;
65             int *FML;
66             int *FM1;
67             int *FM2;
68             int *ggg;
69             int Fc;
70             int FcH;
71             int FcI;
72             int FcM;
73 #ifndef VRNA_DISABLE_C11_FEATURES
74             /* C11 support for unnamed unions/structs */
75             int **c_local;
76             int *f3_local;
77             int **fML_local;
78             int **ggg_local;
79 #endif
80         };
81     };
82 #endif
83     int ***E_F5;
84     int **l_min_F5;
85 }
86
87 struct {
88     int ***E_F5;
89     int **l_min_F5;
90 }
91
92 struct {
93     int ***E_F5;
94     int **l_min_F5;
95 }
96
97 struct {
98     int ***E_F5;
99     int **l_min_F5;
100 }
101
102 struct {
103     int ***E_F5;
104     int **l_min_F5;
105 }
106
107 struct {
108     int ***E_F5;
109     int **l_min_F5;
110 }
111
112 struct {
113     int ***E_F5;
114     int **l_min_F5;
115 }
116
117 struct {
118     int ***E_F5;
119     int **l_min_F5;
120 }
121
122 struct {
123     int ***E_F5;
124     int **l_min_F5;
125 }
126
127 struct {
128     int ***E_F5;
129     int **l_min_F5;
130 }
131
132 struct {
133     int ***E_F5;
134     int **l_min_F5;
135 }
136
137 struct {
138     int ***E_F5;
139     int **l_min_F5;
140 }
141
142 struct {
143     int ***E_F5;
144     int **l_min_F5;
145 }
146
147 struct {
148     int ***E_F5;
149     int **l_min_F5;
150 }
151
152 struct {
153     int ***E_F5;
154     int **l_min_F5;
155 }
156
157 struct {
158     int ***E_F5;
159     int **l_min_F5;
160 }
161
162 struct {
163     int ***E_F5;
164     int **l_min_F5;
165 }
166
167 struct {
168     int ***E_F5;
169     int **l_min_F5;
170 }
171
172 struct {
173     int ***E_F5;
174     int **l_min_F5;
175 }
176
177 struct {
178     int ***E_F5;
179     int **l_min_F5;
180 }
181
182 struct {
183     int ***E_F5;
184     int **l_min_F5;
185 }
186
187 struct {
188     int ***E_F5;
189     int **l_min_F5;
190 }
191
192 struct {
193     int ***E_F5;
194     int **l_min_F5;
195 }
196
197 struct {
198     int ***E_F5;
199     int **l_min_F5;
200 }
201
202 struct {
203     int ***E_F5;
204     int **l_min_F5;
205 }
206
207 struct {
208     int ***E_F5;
209     int **l_min_F5;
210 }
211
212 struct {
213     int ***E_F5;
214     int **l_min_F5;
215 }
216
217 struct {
218     int ***E_F5;
219     int **l_min_F5;
220 }
221
222 struct {
223     int ***E_F5;
224     int **l_min_F5;
225 }
226
227 struct {
228     int ***E_F5;
229     int **l_min_F5;
230 }
231
232 struct {
233     int ***E_F5;
234     int **l_min_F5;
235 }
236
237 struct {
238     int ***E_F5;
239     int **l_min_F5;
240 }
241
242 struct {
243     int ***E_F5;
244     int **l_min_F5;
245 }
246
247 struct {
248     int ***E_F5;
249     int **l_min_F5;
250 }
251
252 struct {
253     int ***E_F5;
254     int **l_min_F5;
255 }
256
257 struct {
258     int ***E_F5;
259     int **l_min_F5;
260 }
261
262 struct {
263     int ***E_F5;
264     int **l_min_F5;
265 }
266
267 struct {
268     int ***E_F5;
269     int **l_min_F5;
270 }
271
272 struct {
273     int ***E_F5;
274     int **l_min_F5;
275 }
276
277 struct {
278     int ***E_F5;
279     int **l_min_F5;
280 }
281
282 struct {
283     int ***E_F5;
284     int **l_min_F5;
285 }
286
287 struct {
288     int ***E_F5;
289     int **l_min_F5;
290 }
291
292 struct {
293     int ***E_F5;
294     int **l_min_F5;
295 }
296
297 struct {
298     int ***E_F5;
299     int **l_min_F5;
300 }
301
302 struct {
303     int ***E_F5;
304     int **l_min_F5;
305 }
306
307 struct {
308     int ***E_F5;
309     int **l_min_F5;
310 }
311
312 struct {
313     int ***E_F5;
314     int **l_min_F5;
315 }
316
317 struct {
318     int ***E_F5;
319     int **l_min_F5;
320 }
321
322 struct {
323     int ***E_F5;
324     int **l_min_F5;
325 }
326
327 struct {
328     int ***E_F5;
329     int **l_min_F5;
330 }
331
332 struct {
333     int ***E_F5;
334     int **l_min_F5;
335 }
336
337 struct {
338     int ***E_F5;
339     int **l_min_F5;
340 }
341
342 struct {
343     int ***E_F5;
344     int **l_min_F5;
345 }
346
347 struct {
348     int ***E_F5;
349     int **l_min_F5;
350 }
351
352 struct {
353     int ***E_F5;
354     int **l_min_F5;
355 }
356
357 struct {
358     int ***E_F5;
359     int **l_min_F5;
360 }
361
362 struct {
363     int ***E_F5;
364     int **l_min_F5;
365 }
366
367 struct {
368     int ***E_F5;
369     int **l_min_F5;
370 }
371
372 struct {
373     int ***E_F5;
374     int **l_min_F5;
375 }
376
377 struct {
378     int ***E_F5;
379     int **l_min_F5;
380 }
381
382 struct {
383     int ***E_F5;
384     int **l_min_F5;
385 }
386
387 struct {
388     int ***E_F5;
389     int **l_min_F5;
390 }
391
392 struct {
393     int ***E_F5;
394     int **l_min_F5;
395 }
396
397 struct {
398     int ***E_F5;
399     int **l_min_F5;
400 }
401
402 struct {
403     int ***E_F5;
404     int **l_min_F5;
405 }
406
407 struct {
408     int ***E_F5;
409     int **l_min_F5;
410 }
411
412 struct {
413     int ***E_F5;
414     int **l_min_F5;
415 }
416
417 struct {
418     int ***E_F5;
419     int **l_min_F5;
420 }
421
422 struct {
423     int ***E_F5;
424     int **l_min_F5;
425 }
426
427 struct {
428     int ***E_F5;
429     int **l_min_F5;
430 }
431
432 struct {
433     int ***E_F5;
434     int **l_min_F5;
435 }
436
437 struct {
438     int ***E_F5;
439     int **l_min_F5;
440 }
441
442 struct {
443     int ***E_F5;
444     int **l_min_F5;
445 }
446
447 struct {
448     int ***E_F5;
449     int **l_min_F5;
450 }
451
452 struct {
453     int ***E_F5;
454     int **l_min_F5;
455 }
456
457 struct {
458     int ***E_F5;
459     int **l_min_F5;
460 }
461
462 struct {
463     int ***E_F5;
464     int **l_min_F5;
465 }
466
467 struct {
468     int ***E_F5;
469     int **l_min_F5;
470 }
471
472 struct {
473     int ***E_F5;
474     int **l_min_F5;
475 }
476
477 struct {
478     int ***E_F5;
479     int **l_min_F5;
480 }
481
482 struct {
483     int ***E_F5;
484     int **l_min_F5;
485 }
486
487 struct {
488     int ***E_F5;
489     int **l_min_F5;
490 }
491
492 struct {
493     int ***E_F5;
494     int **l_min_F5;
495 }
496
497 struct {
498     int ***E_F5;
499     int **l_min_F5;
500 }
501
502 struct {
503     int ***E_F5;
504     int **l_min_F5;
505 }
506
507 struct {
508     int ***E_F5;
509     int **l_min_F5;
510 }
511
512 struct {
513     int ***E_F5;
514     int **l_min_F5;
515 }
516
517 struct {
518     int ***E_F5;
519     int **l_min_F5;
520 }
521
522 struct {
523     int ***E_F5;
524     int **l_min_F5;
525 }
526
527 struct {
528     int ***E_F5;
529     int **l_min_F5;
530 }
531
532 struct {
533     int ***E_F5;
534     int **l_min_F5;
535 }
536
537 struct {
538     int ***E_F5;
539     int **l_min_F5;
540 }
541
542 struct {
543     int ***E_F5;
544     int **l_min_F5;
545 }
546
547 struct {
548     int ***E_F5;
549     int **l_min_F5;
550 }
551
552 struct {
553     int ***E_F5;
554     int **l_min_F5;
555 }
556
557 struct {
558     int ***E_F5;
559     int **l_min_F5;
560 }
561
562 struct {
563     int ***E_F5;
564     int **l_min_F5;
565 }
566
567 struct {
568     int ***E_F5;
569     int **l_min_F5;
570 }
571
572 struct {
573     int ***E_F5;
574     int **l_min_F5;
575 }
576
577 struct {
578     int ***E_F5;
579     int **l_min_F5;
580 }
581
582 struct {
583     int ***E_F5;
584     int **l_min_F5;
585 }
586
587 struct {
588     int ***E_F5;
589     int **l_min_F5;
590 }
591
592 struct {
593     int ***E_F5;
594     int **l_min_F5;
595 }
596
597 struct {
598     int ***E_F5;
599     int **l_min_F5;
600 }
601
602 struct {
603     int ***E_F5;
604     int **l_min_F5;
605 }
606
607 struct {
608     int ***E_F5;
609     int **l_min_F5;
610 }
611
612 struct {
613     int ***E_F5;
614     int **l_min_F5;
615 }
616
617 struct {
618     int ***E_F5;
619     int **l_min_F5;
620 }
621
622 struct {
623     int ***E_F5;
624     int **l_min_F5;
625 }
626
627 struct {
628     int ***E_F5;
629     int **l_min_F5;
630 }
631
632 struct {
633     int ***E_F5;
634     int **l_min_F5;
635 }
636
637 struct {
638     int ***E_F5;
639     int **l_min_F5;
640 }
641
642 struct {
643     int ***E_F5;
644     int **l_min_F5;
645 }
646
647 struct {
648     int ***E_F5;
649     int **l_min_F5;
650 }
651
652 struct {
653     int ***E_F5;
654     int **l_min_F5;
655 }
656
657 struct {
658     int ***E_F5;
659     int **l_min_F5;
660 }
661
662 struct {
663     int ***E_F5;
664     int **l_min_F5;
665 }
666
667 struct {
668     int ***E_F5;
669     int **l_min_F5;
670 }
671
672 struct {
673     int ***E_F5;
674     int **l_min_F5;
675 }
676
677 struct {
678     int ***E_F5;
679     int **l_min_F5;
680 }
681
682 struct {
683     int ***E_F5;
684     int **l_min_F5;
685 }
686
687 struct {
688     int ***E_F5;
689     int **l_min_F5;
690 }
691
692 struct {
693     int ***E_F5;
694     int **l_min_F5;
695 }
696
697 struct {
698     int ***E_F5;
699     int **l_min_F5;
700 }
701
702 struct {
703     int ***E_F5;
704     int **l_min_F5;
705 }
706
707 struct {
708     int ***E_F5;
709     int **l_min_F5;
710 }
711
712 struct {
713     int ***E_F5;
714     int **l_min_F5;
715 }
716
717 struct {
718     int ***E_F5;
719     int **l_min_F5;
720 }
721
722 struct {
723     int ***E_F5;
724     int **l_min_F5;
725 }
726
727 struct {
728     int ***E_F5;
729     int **l_min_F5;
730 }
731
732 struct {
733     int ***E_F5;
734     int **l_min_F5;
735 }
736
737 struct {
738     int ***E_F5;
739     int **l_min_F5;
740 }
741
742 struct {
743     int ***E_F5;
744     int **l_min_F5;
745 }
746
747 struct {
748     int ***E_F5;
749     int **l_min_F5;
750 }
751
752 struct {
753     int ***E_F5;
754     int **l_min_F5;
755 }
756
757 struct {
758     int ***E_F5;
759     int **l_min_F5;
760 }
761
762 struct {
763     int ***E_F5;
764     int **l_min_F5;
765 }
766
767 struct {
768     int ***E_F5;
769     int **l_min_F5;
770 }
771
772 struct {
773     int ***E_F5;
774     int **l_min_F5;
775 }
776
777 struct {
778     int ***E_F5;
779     int **l_min_F5;
780 }
781
782 struct {
783     int ***E_F5;
784     int **l_min_F5;
785 }
786
787 struct {
788     int ***E_F5;
789     int **l_min_F5;
790 }
791
792 struct {
793     int ***E_F5;
794     int **l_min_F5;
795 }
796
797 struct {
798     int ***E_F5;
799     int **l_min_F5;
800 }
801
802 struct {
803     int ***E_F5;
804     int **l_min_F5;
805 }
806
807 struct {
808     int ***E_F5;
809     int **l_min_F5;
810 }
811
812 struct {
813     int ***E_F5;
814     int **l_min_F5;
815 }
816
817 struct {
818     int ***E_F5;
819     int **l_min_F5;
820 }
821
822 struct {
823     int ***E_F5;
824     int **l_min_F5;
825 }
826
827 struct {
828     int ***E_F5;
829     int **l_min_F5;
830 }
831
832 struct {
833     int ***E_F5;
834     int **l_min_F5;
835 }
836
837 struct {
838     int ***E_F5;
839     int **l_min_F5;
840 }
841
842 struct {
843     int ***E_F5;
844     int **l_min_F5;
845 }
846
847 struct {
848     int ***E_F5;
849     int **l_min_F5;
850 }
851
852 struct {
853     int ***E_F5;
854     int **l_min_F5;
855 }
856
857 struct {
858     int ***E_F5;
859     int **l_min_F5;
860 }
861
862 struct {
863     int ***E_F5;
864     int **l_min_F5;
865 }
866
867 struct {
868     int ***E_F5;
869     int **l_min_F5;
870 }
871
872 struct {
873     int ***E_F5;
874     int **l_min_F5;
875 }
876
877 struct {
878     int ***E_F5;
879     int **l_min_F5;
880 }
881
882 struct {
883     int ***E_F5;
884     int **l_min_F5;
885 }
886
887 struct {
888     int ***E_F5;
889     int **l_min_F5;
890 }
891
892 struct {
893     int ***E_F5;
894     int **l_min_F5;
895 }
896
897 struct {
898     int ***E_F5;
899     int **l_min_F5;
900 }
901
902 struct {
903     int ***E_F5;
904     int **l_min_F5;
905 }
906
907 struct {
908     int ***E_F5;
909     int **l_min_F5;
910 }
911
912 struct {
913     int ***E_F5;
914     int **l_min_F5;
915 }
916
917 struct {
918     int ***E_F5;
919     int **l_min_F5;
920 }
921
922 struct {
923     int ***E_F5;
924     int **l_min_F5;
925 }
926
927 struct {
928     int ***E_F5;
929     int **l_min_F5;
930 }
931
932 struct {
933     int ***E_F5;
934     int **l_min_F5;
935 }
936
937 struct {
938     int ***E_F5;
939     int **l_min_F5;
940 }
941
942 struct {
943     int ***E_F5;
944     int **l_min_F5;
945 }
946
947 struct {
948     int ***E_F5;
949     int **l_min_F5;
950 }
951
952 struct {
953     int ***E_F5;
954     int **l_min_F5;
955 }
956
957 struct {
958     int ***E_F5;
959     int **l_min_F5;
960 }
961
962 struct {
963     int ***E_F5;
964     int **l_min_F5;
965 }
966
967 struct {
968     int ***E_F5;
969     int **l_min_F5;
970 }
971
972 struct {
973     int ***E_F5;
974     int **l_min_F5;
975 }
976
977 struct {
978     int ***E_F5;
979     int **l_min_F5;
980 }
981
982 struct {
983     int ***E_F5;
984     int **l_min_F5;
985 }
986
987 struct {
988     int ***E_F5;
989     int **l_min_F5;
990 }
991
992 struct {
993     int ***E_F5;
994     int **l_min_F5;
995 }
996
997 struct {
998     int ***E_F5;
999     int **l_min_F5;
1000 }
1001
1002 struct {
1003     int ***E_F5;
1004     int **l_min_F5;
1005 }
1006
1007 struct {
1008     int ***E_F5;
1009     int **l_min_F5;
1010 }
1011
1012 struct {
1013     int ***E_F5;
1014     int **l_min_F5;
1015 }
1016
1017 struct {
1018     int ***E_F5;
1019     int **l_min_F5;
1020 }
1021
1022 struct {
1023     int ***E_F5;
1024     int **l_min_F5;
1025 }
1026
1027 struct {
1028     int ***E_F5;
1029     int **l_min_F5;
1030 }
1031
1032 struct {
1033     int ***E_F5;
1034     int **l_min_F5;
1035 }
1036
1037 struct {
1038     int ***E_F5;
1039     int **l_min_F5;
1040 }
1041
1042 struct {
1043     int ***E_F5;
1044     int **l_min_F5;
1045 }
1046
1047 struct {
1048     int ***E_F5;
1049     int **l_min_F5;
1050 }
1051
1052 struct {
1053     int ***E_F5;
1054     int **l_min_F5;
1055 }
1056
1057 struct {
1058     int ***E_F5;
1059     int **l_min_F5;
1060 }
1061
1062 struct {
1063     int ***E_F5;
1064     int **l_min_F5;
1065 }
1066
1067 struct {
1068     int ***E_F5;
1069     int **l_min_F5;
1070 }
1071
1072 struct {
1073     int ***E_F5;
1074     int **l_min_F5;
1075 }
1076
1077 struct {
1078     int ***E_F5;
1079     int **l_min_F5;
1080 }
1081
1082 struct {
1083     int ***E_F5;
1084     int **l_min_F5;
1085 }
1086
1087 struct {
1088     int ***E_F5;
1089     int **l_min_F5;
1090 }
1091
1092 struct {
1093     int ***E_F5;
1094     int **l_min_F5;
1095 }
1096
1097 struct {
1098     int ***E_F5;
1099     int **l_min_F5;
1100 }
1101
1102 struct {
1103     int ***E_F5;
1104     int **l_min_F5;
1105 }
1106
1107 struct {
1108     int ***E_F5;
1109     int **l_min_F5;
1110 }
1111
1112 struct {
1113     int ***E_F5;
1114     int **l_min_F5;
1115 }
1116
1117 struct {
1118     int ***E_F5;
1119     int **l_min_F5;
1120 }
1121
1122 struct {
1123     int ***E_F5;
1124     int **l_min_F5;
1125 }
1126
1127 struct {
1128     int ***E_F5;
1129     int **l_min_F5;
1130 }
1131
1132 struct {
1133     int ***E_F5;
1134     int **l_min_F5;
1135 }
1136
1137 struct {
1138     int ***E_F5;
1139     int **l_min_F5;
1140 }
1141
1142 struct {
1143     int ***E_F5;
1144     int **l_min_F5;
1145 }
1146
1147 struct {
1148     int ***E_F5;
1149     int **l_min_F5;
1150 }
1151
1152 struct {
1153     int ***E_F5;
1154     int **l_min_F5;
1155 }
1156
1157 struct {
1158     int ***E_F5;
1159     int **l_min_F5;
1160 }
1161
1162 struct {
1163     int ***E_F5;
1164     int **l_min_F5;
1165 }
1166
1167 struct {
1168     int ***E_F5;
1169     int **l_min_F5;
1170 }
1171
1172 struct {
1173     int ***E_F5;
1174     int **l_min_F5;
1175 }
1176
1177 struct {
1178     int ***E_F5;
1179     int **l_min_F5;
1180 }
1181
1182 struct {
1183     int ***E_F5;
1184     int **l_min_F5;
1185 }
1186
1187 struct {
1188     int ***E_F5;
1189     int **l_min_F5;
1190 }
1191
1192 struct {
1193     int ***E_F5;
1194     int **l_min_F5;
1195 }
1196
1197 struct {
1198     int ***E_F5;
1199     int **l_min_F5;
1200 }
1201
1202 struct {
1203     int ***E_F5;
1204     int **l_min_F5;
1205 }
1206
1207 struct {
1208     int ***E_F5;
1209     int **l_min_F5;
1210 }
1211
1212 struct {
1213     int ***E_F5;
1214     int **l_min_F5;
1215 }
1216
1217 struct {
1218     int ***E_F5;
1219     int **l_min_F5;
1220 }
1221
1222 struct {
1223     int ***E_F5;
1224     int **l_min_F5;
1225 }
1226
1227 struct {
1228     int ***E_F5;
1229     int **l_min_F5;
1230 }
1231
1232 struct {
1233     int ***E_F5;
1234     int **l_min_F5;
1235 }
1236
1237 struct {
1238     int ***E_F5;
1239     int **l_min_F5;
1240 }
1241
1242 struct {
1243     int ***E_F5;
1244     int **l_min_F5;
1245 }
1246
1247 struct {
1248     int ***E_F5;
1249     int **l_min_F5;
1250 }
1251
1252 struct {
1253     int ***E_F5;
1254     int **l_min_F5;
1255 }
1256
1257 struct {
1258     int ***E_F5;
1259     int **l_min_F5;
1260 }
1261
1262 struct {
1263     int ***E_F5;
1264     int **l_min_F5;
1265 }
1266
1267 struct {
1268     int ***E_F5;
1269     int **l_min_F5;
1270 }
1271
1272 struct {
1273     int ***E_F5;
1274     int **l_min_F5;
1275 }
1276
1277 struct {
1278     int ***E_F5;
1279     int **l_min_F5;
1280 }
1281
1282 struct {
1283     int ***E_F5;
1284     int **l_min_F5;
1285 }
1286
1287 struct {
1288     int ***E_F5;
1289     int **l_min_F5;
1290 }
1291
1292 struct {
1293     int ***E_F5;
1294     int **l_min_F5;
1295 }
1296
1297 struct {
1298     int ***E_F5;
1299     int **l_min_F5;
1300 }
1301
1302 struct {
1303     int ***E_F5;
1304     int **l_min_F5;
1305 }
1306
1307 struct {
1308     int ***E_F5;
1309     int **l_min_F5;
1310 }
1311
1312 struct {
1313     int ***E_F5;
1314     int **l_min_F5;
1315 }
1316
1317 struct {
1318     int ***E_F5;
1319     int **l_min_F5;
1320 }
1321
1322 struct {
1323     int ***E_F5;
1324     int **l_min_F5;
1325 }
1326
1327 struct {
1328     int ***E_F5;
1329     int **l_min_F5;
1330 }
1331
1332 struct {
1333     int ***E_F5;
1334     int **l_min_F5;
1335 }
1336
1337 struct {
1338     int ***E_F5;
1339     int **l_min_F5;
1340 }
1341
1342 struct {
1343     int ***E_F5;
1344     int **l_min_F5;
1345 }
1346
1347 struct {
1348     int ***E_F5;
1349     int **l_min_F5;
1350 }
1351
1352 struct {
1353     int ***E_F5;
1354     int **l_min_F5;
1355 }
1356
1357 struct {
1358     int ***E_F5;
1359     int **l_min_F5;
1360 }
1361
1362 struct {
1363     int ***E_F5;
1364     int **l_min_F5;
1365 }
1366
1367 struct {
1368     int ***E_F5;
1369     int **l_min_F5;
1370 }
1371
1372 struct {
1373     int ***E_F5;
1374     int **l_min_F5;
1375 }
1376
1377 struct {
1378     int ***E_F5;
1379     int **l_min_F5;
1380 }
1381
1382 struct {
1383     int ***E_F5;
1384     int **l_min_F5;
1385 }
1386
1387 struct {
1388     int ***E_F5;
1389     int **l_min_F5;
1390 }
1391
1392 struct {
1393     int ***E_F5;
1394     int **l_min_F5;
1395 }
1396
1397 struct {
1398     int ***E_F5;
1399     int **l_min_F5;
1400 }
1401
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
```

```

114 int      ***l_max_F5;
115 int      *k_min_F5;
116 int      *k_max_F5;
117
118 int      ***E_F3;
119 int      ***l_min_F3;
120 int      ***l_max_F3;
121 int      *k_min_F3;
122 int      *k_max_F3;
123
124 int      ***E_C;
125 int      ***l_min_C;
126 int      ***l_max_C;
127 int      *k_min_C;
128 int      *k_max_C;
129
130 int      ***E_M;
131 int      ***l_min_M;
132 int      ***l_max_M;
133 int      *k_min_M;
134 int      *k_max_M;
135
136 int      ***E_M1;
137 int      ***l_min_M1;
138 int      ***l_max_M1;
139 int      *k_min_M1;
140 int      *k_max_M1;
141
142 int      ***E_M2;
143 int      ***l_min_M2;
144 int      ***l_max_M2;
145 int      *k_min_M2;
146 int      *k_max_M2;
147
148 int      **E_Fc;
149 int      *l_min_Fc;
150 int      *l_max_Fc;
151 int      k_min_Fc;
152 int      k_max_Fc;
153
154 int      **E_FcH;
155 int      *l_min_FcH;
156 int      *l_max_FcH;
157 int      k_min_FcH;
158 int      k_max_FcH;
159
160 int      **E_FcI;
161 int      *l_min_FcI;
162 int      *l_max_FcI;
163 int      k_min_FcI;
164 int      k_max_FcI;
165
166 int      **E_FcM;
167 int      *l_min_FcM;
168 int      *l_max_FcM;
169 int      k_min_FcM;
170 int      k_max_FcM;
171
172 /* auxiliary arrays for remaining set of coarse graining (k,l) > (k_max, l_max) */
173 int      *E_F5_rem;
174 int      *E_F3_rem;
175 int      *E_C_rem;
176 int      *E_M_rem;
177 int      *E_M1_rem;
178 int      *E_M2_rem;
179
180 int      E_Fc_rem;
181 int      E_FcH_rem;
182 int      E_FcI_rem;
183 int      E_FcM_rem;
184
185 #ifdef COUNT_STATES
186 unsigned long ***N_F5;
187 unsigned long ***N_C;
188 unsigned long ***N_M;
189 unsigned long ***N_M1;
190#endif
191
192 #ifndef VRNA_DISABLE_C11_FEATURES
193 /* C11 support for unnamed unions/structs */
194 };
195 };
196#endif
197 };
198 };
199 };
200#endif
201 };
202
203 struct vrna_mx_pf_s {
204     const vrna_mx_type_e type;

```

```
211     unsigned int          length;
212     FLT_OR_DBL           *scale;
213     FLT_OR_DBL           *expMLbase;
214 #ifndef VRNA_DISABLE_C11_FEATURES
215     /* C11 support for unnamed unions/structs */
216     union {
217         struct {
218             #endif
219             FLT_OR_DBL *q;
220             FLT_OR_DBL *qb;
221             FLT_OR_DBL *qm;
222             FLT_OR_DBL *qm1;
223             FLT_OR_DBL *probs;
224             FLT_OR_DBL *qlk;
225             FLT_OR_DBL *qln;
226             FLT_OR_DBL *G;
227             #endif
228             FLT_OR_DBL qo;
229             FLT_OR_DBL *qm2;
230             FLT_OR_DBL qho;
231             FLT_OR_DBL qio;
232             FLT_OR_DBL qmo;
233             #endif
234 #ifndef VRNA_DISABLE_C11_FEATURES
235             /* C11 support for unnamed unions/structs */
236         };
237         #endif
238         struct {
239             #endif
240             FLT_OR_DBL **q_local;
241             FLT_OR_DBL **qb_local;
242             FLT_OR_DBL **qm_local;
243             FLT_OR_DBL **pR;
244             FLT_OR_DBL **qm2_local;
245             FLT_OR_DBL **Q15;
246             FLT_OR_DBL **q21;
247             FLT_OR_DBL **qmb;
248             FLT_OR_DBL **G_local;
249 #ifndef VRNA_DISABLE_C11_FEATURES
250             /* C11 support for unnamed unions/structs */
251         };
252         struct {
253             #endif
254             #endif
255             FLT_OR_DBL ***Q;
256             int **l_min_Q;
257             int **l_max_Q;
258             int *k_min_Q;
259             int *k_max_Q;
260             #endif
261             #endif
262             #endif
263             #endif
264             #endif
265             #endif
266             #endif
267             #endif
268             #endif
269             #endif
270             #endif
271             #endif
272             #endif
273             #endif
274             #endif
275             #endif
276             #endif
277             #endif
278             #endif
279             #endif
280             #endif
281             #endif
282             #endif
283             #endif
284             #endif
285             #endif
286             #endif
287             #endif
288             #endif
289             #endif
290             #endif
291             #endif
292             #endif
293             #endif
294             #endif
295             #endif
296             #endif
297             #endif
298             #endif
299             #endif
300             #endif
301             #endif
302             #endif
303             #endif
304             #endif
305             #endif
306             #endif
307             #endif
308             #endif
309             #endif
310             #endif
311             #endif
312             #endif
313             #endif
314             #endif
315             #endif
316             #endif
317             #endif
318             #endif
319             #endif
320             #endif
321             #endif
322             #endif
323             #endif
324             #endif
325             #endif
```

```

326
327     FLT_OR_DBL **Q_cI;
328     int *l_min_Q_cI;
329     int *l_max_Q_cI;
330     int k_min_Q_cI;
331     int k_max_Q_cI;
332
333     FLT_OR_DBL **Q_cM;
334     int *l_min_Q_cM;
335     int *l_max_Q_cM;
336     int k_min_Q_cM;
337     int k_max_Q_cM;
338
339     /* auxiliary arrays for remaining set of coarse graining (k,l) > (k_max, l_max) */
340     FLT_OR_DBL *Q_rem;
341     FLT_OR_DBL *Q_B_rem;
342     FLT_OR_DBL *Q_M_rem;
343     FLT_OR_DBL *Q_M1_rem;
344     FLT_OR_DBL *Q_M2_rem;
345
346     FLT_OR_DBL Q_c_rem;
347     FLT_OR_DBL Q_ch_rem;
348     FLT_OR_DBL Q_ci_rem;
349     FLT_OR_DBL Q_cM_rem;
350 #ifndef VRNA_DISABLE_C11_FEATURES
351     /* C11 support for unnamed unions/structs */
352 };
353 };
354#endif
355 };
356 };
357 };
358#endif
359 };
360
361 int
362 vrna_mx_add(vrna_fold_compound_t    *vc,
363               vrna_mx_type_e      type,
364               unsigned int         options);
365
366 int
367 vrna_mx_mfe_add(vrna_fold_compound_t *vc,
368                   vrna_mx_type_e      mx_type,
369                   unsigned int         options);
370
371 int
372 vrna_mx_pf_add(vrna_fold_compound_t *vc,
373                   vrna_mx_type_e      mx_type,
374                   unsigned int         options);
375
376 int
377 vrna_mx_prepare(vrna_fold_compound_t *vc,
378                   unsigned int         options);
379
380 void
381 vrna_mx_mfe_free(vrna_fold_compound_t *vc);
382
383 void
384 vrna_mx_pf_free(vrna_fold_compound_t *vc);
385
386#endif

```

## 18.59 ViennaRNA/duplex.h File Reference

Functions for simple RNA-RNA duplex interactions.

Include dependency graph for duplex.h:

### 18.59.1 Detailed Description

Functions for simple RNA-RNA duplex interactions.

## 18.60 duplex.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_DUPLEX_H
2 #define VIENNA_RNA_PACKAGE_DUPLEX_H

```

```

3
4 #include <ViennaRNA/datastructures/basic.h>
5
6 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
7
15 duplexT duplexfold(const char *s1,
16                     const char *s2);
17
18
19 duplexT *duplex_subopt(const char *s1,
20                         const char *s2,
21                         int      delta,
22                         int      w);
23
24
25 duplexT aliduplexfold(const char *s1[],
26                         const char *s2[]);
27
28
29 duplexT *aliduplex_subopt(const char *s1[],
30                            const char *s2[],
31                            int      delta,
32                            int      w);
33
34
35 #endif
36
37 #endif

```

## 18.61 ViennaRNA/edit\_cost.h File Reference

global variables for Edit Costs included by treelist.c and stringdist.c

### 18.61.1 Detailed Description

global variables for Edit Costs included by treelist.c and stringdist.c

## 18.62 edit\_cost.h

[Go to the documentation of this file.](#)

```

1
6 #define PRIVATE static
7
8 PRIVATE char sep = ':';
9 PRIVATE char *coding = "Null:U:P:H:B:I:M:S:E:R";
10
11 #define DIST_INF 10000 /* infinity */
12
13 typedef int CostMatrix[10][10];
14
15 PRIVATE CostMatrix *EditCost; /* will point to UsualCost or ShapiroCost */
16
17 PRIVATE CostMatrix UsualCost =
18 {
19
20 /* Null, U, P, H, B, I, M, S, E, R
21 */
22 { 0, 1, 2, 2, 2, 2, 2, 1, 1, DIST_INF},
23 { 1, 0, 1, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF},
24 { 2, 1, 0, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF},
25 { 2, DIST_INF, DIST_INF, 0, 2, 2, 2, DIST_INF, DIST_INF, DIST_INF},
26 { 2, DIST_INF, DIST_INF, 2, 0, 1, 2, DIST_INF, DIST_INF, DIST_INF},
27 { 2, DIST_INF, DIST_INF, 2, 1, 0, 2, DIST_INF, DIST_INF, DIST_INF},
28 { 2, DIST_INF, DIST_INF, 2, 2, 2, 0, DIST_INF, DIST_INF, DIST_INF},
29 { 1, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, 0, DIST_INF, DIST_INF},
30 { 1, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, 0, DIST_INF},
31 { DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, 0},
32 /* R replaced */

```

```

33 };
34
35
36 PRIVATE CostMatrix ShapiroCost =
37 {
38
39 /* Null,      U,      P,      H,      B,      I,      M,      S,      E,      R
40   */
41 {      0,      1,      2,     100,      5,      5,     75,      5,      5, DIST_INF},
42 {      1,      0,      1, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF},
43 {      2,      1,      0, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF},
44 {     100, DIST_INF, DIST_INF,      0,      8,      8,     8, DIST_INF, DIST_INF, DIST_INF},
45 {      5, DIST_INF, DIST_INF,      8,      0,      3,     8, DIST_INF, DIST_INF, DIST_INF},
46 {      5, DIST_INF, DIST_INF,      8,      3,      0,     8, DIST_INF, DIST_INF, DIST_INF},
47 {     75, DIST_INF, DIST_INF,      8,      8,      8,     0, DIST_INF, DIST_INF, DIST_INF},
48 {      5, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF,      0, DIST_INF, DIST_INF},
49 {      5, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF,      0, DIST_INF},
50 { DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, 0},
51 {      /* R replaced */
52 };
53

```

## 18.63 ViennaRNA/energy\_const.h File Reference

Use [ViennaRNA/params/constants.h](#) instead.

Include dependency graph for energy\_const.h:

### 18.63.1 Detailed Description

Use [ViennaRNA/params/constants.h](#) instead.

**Deprecated** Use [ViennaRNA/params/constants.h](#) instead

## 18.64 energy\_const.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PARAMS_CONSTANTS_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_PARAMS_CONSTANTS_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 # ifdef VRNA_WARN_DEPRECATED
6 # warning "Including deprecated header file <ViennaRNA/energy_const.h>! Use <ViennaRNA/params/constants.h>
7 instead!"
8 # endif
9 #include <ViennaRNA/params/constants.h>
10 #endif
11
12
13
14
15
16
17

```

## 18.65 ViennaRNA/energy\_par.h File Reference

Use [ViennaRNA/params/default.h](#) instead.

Include dependency graph for energy\_par.h:

### 18.65.1 Detailed Description

Use [ViennaRNA/params/default.h](#) instead.

**Deprecated** Use [ViennaRNA/params/default.h](#) instead

## 18.66 energy\_par.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PARAMS_DEFAULT_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_PARAMS_DEFAULT_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 # ifdef VRNA_WARN_DEPRECATED
6 # warning "Including deprecated header file <ViennaRNA/energy_par.h>! Use <ViennaRNA/params/default.h>
7 instead!"
8 # endif
9 #include <ViennaRNA/params/default.h>
10#endif
11
12#endif
13
14#endif
15
16
17#endif

```

## 18.67 ViennaRNA/equilibrium\_probs.h File Reference

Equilibrium Probability implementations.

Include dependency graph for equilibrium\_probs.h: This graph shows which files directly or indirectly include this file:

### Functions

#### Base pair probabilities and derived computations

- int [vrna\\_pairing\\_probs](#) (vrna\_fold\_compound\_t \*vc, char \*structure)
- double [vrna\\_mean\\_bp\\_distance\\_pr](#) (int length, FLT\_OR\_DBL \*pr)
 

*Get the mean base pair distance in the thermodynamic ensemble from a probability matrix.*
- double [vrna\\_mean\\_bp\\_distance](#) (vrna\_fold\_compound\_t \*vc)
 

*Get the mean base pair distance in the thermodynamic ensemble.*
- double [vrna\\_ensemble\\_defect\\_pt](#) (vrna\_fold\_compound\_t \*fc, const short \*pt)
 

*Compute the Ensemble Defect for a given target structure provided as a vrna\_ptable.*
- double [vrna\\_ensemble\\_defect](#) (vrna\_fold\_compound\_t \*fc, const char \*structure)
 

*Compute the Ensemble Defect for a given target structure.*
- double \* [vrna\\_positional\\_entropy](#) (vrna\_fold\_compound\_t \*fc)
 

*Compute a vector of positional entropies.*
- [vrna\\_ep\\_t \\* vrna\\_stack\\_prob](#) (vrna\_fold\_compound\_t \*vc, double cutoff)
 

*Compute stacking probabilities.*

#### Multimer probabilities computations

- void [vrna\\_pf\\_dimer\\_probs](#) (double FAB, double FA, double FB, [vrna\\_ep\\_t](#) \*prAB, const [vrna\\_ep\\_t](#) \*prA, const [vrna\\_ep\\_t](#) \*prB, int Alength, const [vrna\\_exp\\_param\\_t](#) \*exp\_params)
 

*Compute Boltzmann probabilities of dimerization without homodimers.*

#### Structure probability computations

- double [vrna\\_pr\\_structure](#) (vrna\_fold\_compound\_t \*fc, const char \*structure)
 

*Compute the equilibrium probability of a particular secondary structure.*
- double [vrna\\_pr\\_energy](#) (vrna\_fold\_compound\_t \*vc, double e)

### 18.67.1 Detailed Description

Equilibrium Probability implementations.

This file includes various implementations for equilibrium probability computations based on the partition function of an RNA sequence, two concatenated sequences, or a sequence alignment.

## 18.68 equilibrium\_probs.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_EQUILIBRIUM_PROBS_H
2 #define VIENNA_RNA_PACKAGE_EQUILIBRIUM_PROBS_H
3
4 #include <ViennaRNA/datastructures/basic.h>
5 #include <ViennaRNA/fold_compound.h>
6 #include <ViennaRNA/utils/structures.h>
7 #include <ViennaRNA/params/basic.h>
8
9 /* 
10 ##### BASE PAIR PROBABILITY RELATED FUNCTIONS #####
11 ##### #####
12 */
13
14
15 int
16 vrna_pairing_probs(vrna_fold_compound_t *vc,
17                      char                  *structure);
18
19
20 double
21 vrna_mean_bp_distance_pr(int      length,
22                           FLT_OR_DBL *pr);
23
24
25
26
27 double
28 vrna_mean_bp_distance(vrna_fold_compound_t *vc);
29
30
31
32 double
33 vrna_ensemble_defect_pt(vrna_fold_compound_t *fc,
34                          const short       *pt);
35
36
37 double
38 vrna_ensemble_defect(vrna_fold_compound_t *fc,
39                       const char        *structure);
40
41
42 double *
43 vrna_positional_entropy(vrna_fold_compound_t *fc);
44
45
46 vrna_ep_t *
47 vrna_stack_prob(vrna_fold_compound_t *vc,
48                  double           cutoff);
49
50
51 /* End base pair related functions */
52 void
53 vrna_pf_dimer_probs(double          FAB,
54                      double          FA,
55                      double          FB,
56                      vrna_ep_t       *prAB,
57                      const vrna_ep_t *prA,
58                      const vrna_ep_t *prB,
59                      int             Alength,
60                      const vrna_exp_param_t *exp_params);
61
62
63 /* End multimer probability related functions */
64 double
65 vrna_pr_structure(vrna_fold_compound_t *fc,
66                     const char        *structure);
67
68
69 double
70 vrna_pr_energy(vrna_fold_compound_t *vc,
71                 double           e);
72
73
74 /* End structure probability related functions */
75 /* End thermodynamics */
76
77
78 #endif

```

## 18.69 ViennaRNA/eval.h File Reference

Functions and variables related to energy evaluation of sequence/structure pairs.

Include dependency graph for eval.h: This graph shows which files directly or indirectly include this file:

## Macros

- `#define VRNA_VERBOSITY QUIET -1`  
*Quiet level verbosity setting.*
- `#define VRNA_VERBOSITY_DEFAULT 1`  
*Default level verbosity setting.*

## Functions

- `int vrna_eval_loop_pt (vrna_fold_compound_t *fc, int i, const short *pt)`  
*Calculate energy of a loop.*
- `int vrna_eval_loop_pt_v (vrna_fold_compound_t *fc, int i, const short *pt, int verbosity_level)`  
*Calculate energy of a loop.*
- `float vrna_eval_move (vrna_fold_compound_t *fc, const char *structure, int m1, int m2)`  
*Calculate energy of a move (closing or opening of a base pair)*
- `int vrna_eval_move_pt (vrna_fold_compound_t *fc, short *pt, int m1, int m2)`  
*Calculate energy of a move (closing or opening of a base pair)*
- `float energy_of_structure (const char *string, const char *structure, int verbosity_level)`  
*Calculate the free energy of an already folded RNA using global model detail settings.*
- `float energy_of_struct_par (const char *string, const char *structure, vrna_param_t *parameters, int verbosity_level)`  
*Calculate the free energy of an already folded RNA.*
- `float energy_of_circ_structure (const char *string, const char *structure, int verbosity_level)`  
*Calculate the free energy of an already folded circular RNA.*
- `float energy_of_circ_struct_par (const char *string, const char *structure, vrna_param_t *parameters, int verbosity_level)`  
*Calculate the free energy of an already folded circular RNA.*
- `int energy_of_structure_pt (const char *string, short *ptable, short *s, short *s1, int verbosity_level)`  
*Calculate the free energy of an already folded RNA.*
- `int energy_of_struct_pt_par (const char *string, short *ptable, short *s, short *s1, vrna_param_t *parameters, int verbosity_level)`  
*Calculate the free energy of an already folded RNA.*
- `float energy_of_move (const char *string, const char *structure, int m1, int m2)`  
*Calculate energy of a move (closing or opening of a base pair)*
- `int energy_of_move_pt (short *pt, short *s, short *s1, int m1, int m2)`  
*Calculate energy of a move (closing or opening of a base pair)*
- `int loop_energy (short *ptable, short *s, short *s1, int i)`  
*Calculate energy of a loop.*
- `float energy_of_struct (const char *string, const char *structure)`
- `int energy_of_struct_pt (const char *string, short *ptable, short *s, short *s1)`
- `float energy_of_circ_struct (const char *string, const char *structure)`

### Basic Energy Evaluation Interface with Dot-Bracket Structure String

- `float vrna_eval_structure (vrna_fold_compound_t *fc, const char *structure)`  
*Calculate the free energy of an already folded RNA.*
- `float vrna_eval_covar_structure (vrna_fold_compound_t *fc, const char *structure)`  
*Calculate the pseudo energy derived by the covariance scores of a set of aligned sequences.*
- `float vrna_eval_structure_verbose (vrna_fold_compound_t *fc, const char *structure, FILE *file)`  
*Calculate the free energy of an already folded RNA and print contributions on a per-loop base.*
- `float vrna_eval_structure_v (vrna_fold_compound_t *fc, const char *structure, int verbosity_level, FILE *file)`  
*Calculate the free energy of an already folded RNA and print contributions on a per-loop base.*
- `float vrna_eval_structure_cstr (vrna_fold_compound_t *fc, const char *structure, int verbosity_level, vrna_cstr_t output_stream)`

### Basic Energy Evaluation Interface with Structure Pair Table

- int **vrna\_eval\_structure\_pt** (*vrna\_fold\_compound\_t* \*fc, const short \*pt)  
*Calculate the free energy of an already folded RNA.*
- int **vrna\_eval\_structure\_pt\_verbose** (*vrna\_fold\_compound\_t* \*fc, const short \*pt, FILE \*file)  
*Calculate the free energy of an already folded RNA.*
- int **vrna\_eval\_structure\_pt\_v** (*vrna\_fold\_compound\_t* \*fc, const short \*pt, int verbosity\_level, FILE \*file)  
*Calculate the free energy of an already folded RNA.*

### Simplified Energy Evaluation with Sequence and Dot-Bracket Strings

- float **vrna\_eval\_structure\_simple** (const char \*string, const char \*structure)  
*Calculate the free energy of an already folded RNA.*
- float **vrna\_eval\_circ\_structure** (const char \*string, const char \*structure)  
*Evaluate the free energy of a sequence/structure pair where the sequence is circular.*
- float **vrna\_eval\_gquad\_structure** (const char \*string, const char \*structure)  
*Evaluate the free energy of a sequence/structure pair where the structure may contain G-Quadruplexes.*
- float **vrna\_eval\_circ\_gquad\_structure** (const char \*string, const char \*structure)  
*Evaluate the free energy of a sequence/structure pair where the sequence is circular and the structure may contain G-Quadruplexes.*
- float **vrna\_eval\_structure\_simple\_verbose** (const char \*string, const char \*structure, FILE \*file)  
*Calculate the free energy of an already folded RNA and print contributions per loop.*
- float **vrna\_eval\_structure\_simple\_v** (const char \*string, const char \*structure, int verbosity\_level, FILE \*file)  
*Calculate the free energy of an already folded RNA and print contributions per loop.*
- float **vrna\_eval\_circ\_structure\_v** (const char \*string, const char \*structure, int verbosity\_level, FILE \*file)  
*Evaluate free energy of a sequence/structure pair, assume sequence to be circular and print contributions per loop.*
- float **vrna\_eval\_gquad\_structure\_v** (const char \*string, const char \*structure, int verbosity\_level, FILE \*file)  
*Evaluate free energy of a sequence/structure pair, allow for G-Quadruplexes in the structure and print contributions per loop.*
- float **vrna\_eval\_circ\_gquad\_structure\_v** (const char \*string, const char \*structure, int verbosity\_level, FILE \*file)  
*Evaluate free energy of a sequence/structure pair, assume sequence to be circular, allow for G-Quadruplexes in the structure, and print contributions per loop.*

### Simplified Energy Evaluation with Sequence Alignments and Consensus Structure Dot-Bracket String

- float **vrna\_eval\_consensus\_structure\_simple** (const char \*\*alignment, const char \*structure)  
*Calculate the free energy of an already folded RNA sequence alignment.*
- float **vrna\_eval\_circ\_consensus\_structure** (const char \*\*alignment, const char \*structure)  
*Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the sequences are circular.*
- float **vrna\_eval\_gquad\_consensus\_structure** (const char \*\*alignment, const char \*structure)  
*Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the structure may contain G-Quadruplexes.*
- float **vrna\_eval\_circ\_gquad\_consensus\_structure** (const char \*\*alignment, const char \*structure)  
*Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the sequence is circular and the structure may contain G-Quadruplexes.*
- float **vrna\_eval\_consensus\_structure\_simple\_verbose** (const char \*\*alignment, const char \*structure, FILE \*file)  
*Evaluate the free energy of a consensus structure for an RNA sequence alignment and print contributions per loop.*
- float **vrna\_eval\_consensus\_structure\_simple\_v** (const char \*\*alignment, const char \*structure, int verbosity\_level, FILE \*file)  
*Evaluate the free energy of a consensus structure for an RNA sequence alignment and print contributions per loop.*
- float **vrna\_eval\_circ\_consensus\_structure\_v** (const char \*\*alignment, const char \*structure, int verbosity\_level, FILE \*file)  
*Evaluate the free energy of a consensus structure for an RNA sequence alignment and print contributions per loop.*

*Evaluate the free energy of a consensus structure for an alignment of circular RNA sequences and print contributions per loop.*

- float [vrna\\_eval\\_gquad\\_consensus\\_structure\\_v](#) (const char \*\*alignment, const char \*structure, int verbosity\_level, FILE \*file)

*Evaluate the free energy of a consensus structure for an RNA sequence alignment, allow for annotated G-Quadruplexes in the structure and print contributions per loop.*

- float [vrna\\_eval\\_circ\\_gquad\\_consensus\\_structure\\_v](#) (const char \*\*alignment, const char \*structure, int verbosity\_level, FILE \*file)

*Evaluate the free energy of a consensus structure for an alignment of circular RNA sequences, allow for annotated G-Quadruplexes in the structure and print contributions per loop.*

### Simplified Energy Evaluation with Sequence String and Structure Pair Table

- int [vrna\\_eval\\_structure\\_pt\\_simple](#) (const char \*string, const short \*pt)

*Calculate the free energy of an already folded RNA.*

- int [vrna\\_eval\\_structure\\_pt\\_simple\\_verbose](#) (const char \*string, const short \*pt, FILE \*file)

*Calculate the free energy of an already folded RNA.*

- int [vrna\\_eval\\_structure\\_pt\\_simple\\_v](#) (const char \*string, const short \*pt, int verbosity\_level, FILE \*file)

*Calculate the free energy of an already folded RNA.*

### Simplified Energy Evaluation with Sequence Alignment and Consensus Structure Pair Table

- int [vrna\\_eval\\_consensus\\_structure\\_pt\\_simple](#) (const char \*\*alignment, const short \*pt)

*Evaluate the Free Energy of a Consensus Secondary Structure given a Sequence Alignment.*

- int [vrna\\_eval\\_consensus\\_structure\\_pt\\_simple\\_verbose](#) (const char \*\*alignment, const short \*pt, FILE \*file)

- int [vrna\\_eval\\_consensus\\_structure\\_pt\\_simple\\_v](#) (const char \*\*alignment, const short \*pt, int verbosity\_level, FILE \*file)

## Variables

- int **cut\_point**

*first pos of second seq for cofolding*

- int **eos\_debug**

*verbose info from energy\_of\_struct*

### 18.69.1 Detailed Description

Functions and variables related to energy evaluation of sequence/structure pairs.

## 18.70 eval.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_EVAL_H
2 #define VIENNA_RNA_PACKAGE_EVAL_H
3
4 #include <stdio.h>
5 #include <ViennaRNA/datastructures/basic.h>
6 #include <ViennaRNA/fold_compound.h>
7 #include <ViennaRNA/datastructures/char_stream.h>
8 #include <ViennaRNA/landscape/move.h>
9 #include <ViennaRNA/params/basic.h> /* for deprecated functions */
10
11 #ifdef VRNA_WARN_DEPRECATED
12 # if defined(__clang__)
13 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
14 # elif defined(__GNUC__)
15 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
16 # else
17 #  define DEPRECATED(func, msg) func
18 # endif
19 #else
20 # define DEPRECATED(func, msg) func
21 #endif
22
23

```

```
58 #define VRNA_VERBOSITY QUIET      -1
59
60
64 #define VRNA_VERBOSITY_DEFAULT     1
65
66
93 float
94 vrna_eval_structure(vrna_fold_compound_t *fc,
95                      const char           *structure);
96
97
118 float
119 vrna_eval_covar_structure(vrna_fold_compound_t *fc,
120                            const char           *structure);
121
122
136 float
137 vrna_eval_structure_verbose(vrna_fold_compound_t *fc,
138                           const char           *structure,
139                           FILE                 *file);
140
141
167 float
168 vrna_eval_structure_v(vrna_fold_compound_t *fc,
169                        const char           *structure,
170                        int                  verbosity_level,
171                        FILE                 *file);
172
173
174 float
175 vrna_eval_structure_cstr(vrna_fold_compound_t *fc,
176                           const char           *structure,
177                           int                  verbosity_level,
178                           vrna_cstr_t          output_stream);
179
180
181 /* End basic eval interface */
208 int
209 vrna_eval_structure_pt(vrna_fold_compound_t *fc,
210                         const short          *pt);
211
212
226 int
227 vrna_eval_structure_pt_verbose(vrna_fold_compound_t *fc,
228                                 const short          *pt,
229                                 FILE                 *file);
230
231
257 int
258 vrna_eval_structure_pt_v(vrna_fold_compound_t *fc,
259                           const short          *pt,
260                           int                  verbosity_level,
261                           FILE                 *file);
262
263
264 /* End basic eval interface with pair table */
288 float
289 vrna_eval_structure_simple(const char *string,
290                            const char *structure);
291
292
303 float
304 vrna_eval_circ_structure(const char *string,
305                           const char *structure);
306
307
325 float
326 vrna_eval_gquad_structure(const char *string,
327                            const char *structure);
328
329
348 float
349 vrna_eval_circ_gquad_structure(const char *string,
350                                 const char *structure);
351
352
367 float
368 vrna_eval_structure_simple_verbose(const char *string,
369                                     const char *structure,
370                                     FILE       *file);
371
372
396 float
397 vrna_eval_structure_simple_v(const char *string,
398                               const char *structure,
399                               int        verbosity_level,
400                               FILE       *file);
```

```
401
402
418 float
419 vrna_eval_circ_structure_v(const char *string,
420                     const char *structure,
421                     int      verbosity_level,
422                     FILE    *file);
423
424
447 float
448 vrna_eval_gquad_structure_v(const char *string,
449                     const char *structure,
450                     int      verbosity_level,
451                     FILE    *file);
452
453
474 float
475 vrna_eval_circ_gquad_structure_v(const char *string,
476                     const char *structure,
477                     int      verbosity_level,
478                     FILE    *file);
479
480
481 /* End simplified eval interface */
511 float
512 vrna_eval_consensus_structure_simple(const char **alignment,
513                     const char *structure);
514
515
531 float
532 vrna_eval_circ_consensus_structure(const char **alignment,
533                     const char *structure);
534
535
558 float
559 vrna_eval_gquad_consensus_structure(const char **alignment,
560                     const char *structure);
561
562
585 float
586 vrna_eval_circ_gquad_consensus_structure(const char **alignment,
587                     const char *structure);
588
589
609 float
610 vrna_eval_consensus_structure_simple_verbose(const char **alignment,
611                     const char *structure,
612                     FILE    *file);
613
614
639 float
640 vrna_eval_consensus_structure_simple_v(const char **alignment,
641                     const char *structure,
642                     int      verbosity_level,
643                     FILE    *file);
644
645
665 float
666 vrna_eval_circ_consensus_structure_v(const char **alignment,
667                     const char *structure,
668                     int      verbosity_level,
669                     FILE    *file);
670
671
698 float
699 vrna_eval_gquad_consensus_structure_v(const char **alignment,
700                     const char *structure,
701                     int      verbosity_level,
702                     FILE    *file);
703
704
731 float
732 vrna_eval_circ_gquad_consensus_structure_v(const char **alignment,
733                     const char *structure,
734                     int      verbosity_level,
735                     FILE    *file);
736
737
738 /* End simplified comparative eval interface */
761 int
762 vrna_eval_structure_pt_simple(const char *string,
763                     const short *pt);
764
765
779 int
780 vrna_eval_structure_pt_simple_verbose(const char *string,
781                     const short *pt,
```

```
782             FILE           *file);
783
784
785 int
786 vrna_eval_structure_pt_simple_v(const char *string,
787                                 const short *pt,
788                                 int          verbosity_level,
789                                 FILE         *file);
790
791
792
793 int
794 vrna_eval_consensus_structure_pt_simple(const char **alignment,
795                                         const short *pt);
796
797
798
799 int
800 vrna_eval_consensus_structure_pt_simple_verbose(const char **alignment,
801                                                 const short *pt,
802                                                 FILE        *file);
803
804
805
806 int
807 vrna_eval_consensus_structure_pt_simple_v(const char **alignment,
808                                         const short *pt,
809                                         int          verbosity_level,
810                                         FILE        *file);
811
812
813
814
815 /* End simplified eval interface with pair table */
816
817 int
818 vrna_eval_loop_pt(vrna_fold_compound_t *fc,
819                     int            i,
820                     const short   *pt);
821
822
823 int
824 vrna_eval_loop_pt_v(vrna_fold_compound_t *fc,
825                      int            i,
826                      const short   *pt,
827                      int            verbosity_level);
828
829
830 float
831 vrna_eval_move(vrna_fold_compound_t *fc,
832                  const char     *structure,
833                  int            m1,
834                  int            m2);
835
836
837 int
838 vrna_eval_move_pt(vrna_fold_compound_t *fc,
839                     short          *pt,
840                     int            m1,
841                     int            m2);
842
843
844 int
845 vrna_eval_move_pt_simple(const char *string,
846                           short        *pt,
847                           int          m1,
848                           int          m2);
849
850
851
852 int
853 vrna_eval_move_pt_simple_v(const char *string,
854                             short       *pt,
855                             int        m1,
856                             int        m2);
857
858
859
860 int
861 vrna_eval_move_shift_pt(vrna_fold_compound_t *fc,
862                          vrna_move_t    *m,
863                          short         *structure);
864
865
866
867 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
868
869
870 extern int cut_point;
871
872 extern int eos_debug;
873
874
875 DEPRECATED(float energy_of_structure(const char *string,
876                                       const char *structure,
877                                       int        verbosity_level),
878             "Use vrna_eval_structure_simple() and vrna_eval_structure() instead");
879
880
881 DEPRECATED(float energy_of_struct_par(const char      *string,
882                                       const char      *structure,
883                                       vrna_param_t   *parameters,
884                                       int            verbosity_level),
885             "Use vrna_eval_structure() instead");
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
```

```

1072 DEPRECATED(float energy_of_circ_structure(const char *string,
1073                           const char *structure,
1074                           int verbosity_level),
1075                           "Use vrna_eval_circ_structure_simple() and vrna_eval_structure() instead");
1076
1092 DEPRECATED(float energy_of_circ_struct_par(const char *string,
1093                           const char *structure,
1094                           vrna_param_t *parameters,
1095                           int verbosity_level),
1096                           "Use vrna_eval_structure() instead");
1097
1098
1099 DEPRECATED(float energy_of_gquad_structure(const char *string,
1100                           const char *structure,
1101                           int verbosity_level),
1102                           "Use vrna_eval_structure_simple() instead");
1103
1104 DEPRECATED(float energy_of_gquad_struct_par(const char *string,
1105                           const char *structure,
1106                           vrna_param_t *parameters,
1107                           int verbosity_level),
1108                           "Use vrna_eval_structure() instead");
1109
1110
1131 DEPRECATED(int energy_of_structure_pt(const char *string,
1132                           short *ptable,
1133                           short *s,
1134                           short *s1,
1135                           int verbosity_level),
1136                           "Use vrna_eval_structure_pt_simple() and vrna_eval_structure_pt() instead");
1137
1155 DEPRECATED(int energy_of_struct_pt_par(const char *string,
1156                           short *ptable,
1157                           short *s,
1158                           short *s1,
1159                           vrna_param_t *parameters,
1160                           int verbosity_level),
1161                           "Use vrna_eval_structure_pt() instead");
1162
1163
1180 DEPRECATED(float energy_of_move(const char *string,
1181                           const char *structure,
1182                           int m1,
1183                           int m2),
1184                           "Use vrna_eval_move() instead");
1185
1186
1205 DEPRECATED(int energy_of_move_pt(short *pt,
1206                           short *s,
1207                           short *s1,
1208                           int m1,
1209                           int m2),
1210                           "Use vrna_eval_move_pt_simple() and vrna_eval_move_pt() instead");
1211
1225 DEPRECATED(int loop_energy(short *ptable,
1226                           short *s,
1227                           short *s1,
1228                           int i),
1229                           "Use vrna_eval_loop_pt() instead");
1230
1245 DEPRECATED(float energy_of_struct(const char *string,
1246                           const char *structure),
1247                           "Use vrna_eval_structure_simple() instead");
1248
1265 DEPRECATED(int energy_of_struct_pt(const char *string,
1266                           short *ptable,
1267                           short *s,
1268                           short *s1),
1269                           "Use vrna_eval_structure_pt_simple() instead");
1270
1285 DEPRECATED(float energy_of_circ_struct(const char *string,
1286                           const char *structure),
1287                           "Use vrna_eval_circ_structure_simple() and vrna_eval_structure() instead");
1288
1289 #endif
1290
1295 #endif

```

## 18.71 ViennaRNA/exterior\_loops.h File Reference

Use [ViennaRNA/loops/external.h](#) instead.

Include dependency graph for exterior\_loops.h:

### 18.71.1 Detailed Description

Use [ViennaRNA/loops/external.h](#) instead.

**Deprecated** Use [ViennaRNA/loops/external.h](#) instead

## 18.72 exterior\_loops.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_LOOPS_EXTERNAL_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_LOOPS_EXTERNAL_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 #warning "Including deprecated header file <ViennaRNA/exterior_loops.h>! Use <ViennaRNA/loops/external.h>
           instead!"
13 # endif
14 #include <ViennaRNA/loops/external.h>
15 #endif
16
17 #endif

```

## 18.73 ViennaRNA/file\_formats.h File Reference

Use [ViennaRNA/io/file\\_formats.h](#) instead.

Include dependency graph for file\_formats.h:

### 18.73.1 Detailed Description

Use [ViennaRNA/io/file\\_formats.h](#) instead.

**Deprecated** Use [ViennaRNA/io/file\\_formats.h](#) instead

## 18.74 file\_formats.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_FILE_FORMATS_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_FILE_FORMATS_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 #warning "Including deprecated header file <ViennaRNA/file_formats.h>! Use <ViennaRNA/io/file_formats.h>
           instead!"
13 # endif
14 #include <ViennaRNA/io/file_formats.h>
15 #include <ViennaRNA/io/file_formats_msa.h>
16 #endif
17
18 #endif

```

## 18.75 ViennaRNA/io/file\_formats.h File Reference

Read and write different file formats for RNA sequences, structures.

Include dependency graph for file\_formats.h: This graph shows which files directly or indirectly include this file:

### Macros

- `#define VRNA_OPTION_MULTILINE 32U`
  - Tell a function that an input is assumed to span several lines.*
- `#define VRNA_CONSTRAINT_MULTILINE 32U`
  - parse multiline constraint*

## Functions

- void `vrna_file_helixlist` (const char \*seq, const char \*db, float energy, FILE \*file)  
*Print a secondary structure as helix list.*
- void `vrna_file_connect` (const char \*seq, const char \*db, float energy, const char \*identifier, FILE \*file)  
*Print a secondary structure as connect table.*
- void `vrna_file_bpseq` (const char \*seq, const char \*db, FILE \*file)  
*Print a secondary structure in bpseq format.*
- void `vrna_file_json` (const char \*seq, const char \*db, double energy, const char \*identifier, FILE \*file)  
*Print a secondary structure in jsonformat.*
- unsigned int `vrna_file_fasta_read_record` (char \*\*header, char \*\*sequence, char \*\*\*rest, FILE \*file, unsigned int options)
- char \* `vrna_extract_record_rest_structure` (const char \*\*lines, unsigned int length, unsigned int option)  
*Extract a dot-bracket structure string from (multiline)character array.*
- int `vrna_file_SHAPE_read` (const char \*file\_name, int length, double default\_value, char \*sequence, double \*values)  
*Read data from a given SHAPE reactivity input file.*
- void `vrna_extract_record_rest_constraint` (char \*\*cstruc, const char \*\*lines, unsigned int option)  
*Extract a hard constraint encoded as pseudo dot-bracket string.*
- unsigned int `read_record` (char \*\*header, char \*\*sequence, char \*\*\*rest, unsigned int options)  
*Get a data record from stdin.*

### 18.75.1 Detailed Description

Read and write different file formats for RNA sequences, structures.

## 18.76 file\_formats.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_FILE_FORMATS_H
2 #define VIENNA_RNA_PACKAGE_FILE_FORMATS_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 #  define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
16 #include <stdio.h>
17
18 #include <ViennaRNA/datastructures/basic.h>
19
20 void
21 vrna_file_helixlist(const char *seq,
22                      const char *db,
23                      float energy,
24                      FILE *file);
25
26 void
27 vrna_file_connect(const char *seq,
28                      const char *db,
29                      float energy,
30                      const char *identifier,
31                      FILE *file);
32
33 void
34 vrna_file_bpseq(const char *seq,
35                      const char *db,
36                      FILE *file);

```

```

89
90
91 #if VRNA_WITH_JSON_SUPPORT
92
102 void
103 vrna_file_json(const char *seq,
104                  const char *db,
105                  double energy,
106                  const char *identifier,
107                  FILE      *file);
108
109
110 #endif
111
121 #define VRNA_OPTION_MULTILINE      32U
126 #define VRNA_CONSTRAINT_MULTILINE 32U
127
193 unsigned int
194 vrna_file_fasta_read_record(char          **header,
195                               char          **sequence,
196                               char          ***rest,
197                               FILE         *file,
198                               unsigned int options);
199
200
217 char *
218 vrna_extract_record_rest_structure(const char **lines,
219                                     unsigned int length,
220                                     unsigned int option);
221
222
235 int
236 vrna_file_SHAPE_read(const char *file_name,
237                        int      length,
238                        double   default_value,
239                        char     *sequence,
240                        double   *values);
241
242 #define VRNA_INPUT_VERBOSE 16384U
243
244
245 int
246 vrna_file_connect_read_record(FILE      *fp,
247                                 char      **id,
248                                 char      **sequence,
249                                 char      **structure,
250                                 char      **remainder,
251                                 unsigned int options);
252
253 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
254
271 DEPRECATED(void vrna_extract_record_rest_constraint(char          **cstruc,
272                           const char    **lines,
273                           unsigned int  option),
274                           "This function is obsolete");
275
280 DEPRECATED(char *extract_record_rest_structure(const char **lines,
281                                     unsigned int length,
282                                     unsigned int option),
283                           "Use vrna_extract_record_rest_structure() instead");
284
291 DEPRECATED(unsigned int read_record(char          **header,
292                           char          **sequence,
293                           char          ***rest,
294                           unsigned int options),
295                           "Use vrna_file_fasta_read_record() instead");
296
297
298 DEPRECATED(unsigned int get_multi_input_line(char          **string,
299                           unsigned int options),
300                           "This function is obsolete");
301
302 #endif
303
308 #endif

```

## 18.77 ViennaRNA/file\_formats\_msa.h File Reference

Use [ViennaRNA/io/file\\_formats\\_msa.h](#) instead.

Include dependency graph for file\_formats\_msa.h:

### 18.77.1 Detailed Description

Use [ViennaRNA/io/file\\_formats\\_msa.h](#) instead.

**Deprecated** Use [ViennaRNA/io/file\\_formats\\_msa.h](#) instead

## 18.78 file\_formats\_msa.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_FILE_FORMATS_MSA_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_FILE_FORMATS_MSA_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 # warning "Including deprecated header file <ViennaRNA/file_formats_msa.h>! Use
13 # <ViennaRNA/io/file_formats_msa.h> instead!"
14 # endif
15 #include <ViennaRNA/io/file_formats_msa.h>
16 #endif
17 #endif

```

## 18.79 ViennaRNA/io/file\_formats\_msa.h File Reference

Functions dealing with file formats for Multiple Sequence Alignments (MSA)

Include dependency graph for file\_formats\_msa.h: This graph shows which files directly or indirectly include this file:

### Macros

- `#define VRNA_FILE_FORMAT_MSA_CLUSTAL 1U`  
*Option flag indicating ClustalW formatted files.*
- `#define VRNA_FILE_FORMAT_MSA_STOCKHOLM 2U`  
*Option flag indicating Stockholm 1.0 formatted files.*
- `#define VRNA_FILE_FORMAT_MSA_FASTA 4U`  
*Option flag indicating FASTA (Pearson) formatted files.*
- `#define VRNA_FILE_FORMAT_MSA_MAF 8U`  
*Option flag indicating MAF formatted files.*
- `#define VRNA_FILE_FORMAT_MSA_MIS 16U`  
*Option flag indicating most informative sequence (MIS) output.*
- `#define VRNA_FILE_FORMAT_MSA_DEFAULT`  
*Option flag indicating the set of default file formats.*
- `#define VRNA_FILE_FORMAT_MSA_NOCHECK 4096U`  
*Option flag to disable validation of the alignment.*
- `#define VRNA_FILE_FORMAT_MSA_UNKNOWN 8192U`  
*Return flag of `vrna_file_msa_detect_format()` to indicate unknown or malformatted alignment.*
- `#define VRNA_FILE_FORMAT_MSA_APPEND 16384U`  
*Option flag indicating to append data to a multiple sequence alignment file rather than overwriting it.*
- `#define VRNA_FILE_FORMAT_MSA QUIET 32768U`  
*Option flag to suppress unnecessary spam messages on `stderr`*
- `#define VRNA_FILE_FORMAT_MSA_SILENT 65536U`  
*Option flag to completely silence any warnings on `stderr`*

# Functions

- int **vrna\_file\_msa\_read** (const char \*filename, char \*\*\*names, char \*\*\*aln, char \*\*id, char \*\*structure, unsigned int options)  
*Read a multiple sequence alignment from file.*
  - int **vrna\_file\_msa\_read\_record** (FILE \*fp, char \*\*\*names, char \*\*\*aln, char \*\*id, char \*\*structure, unsigned int options)  
*Read a multiple sequence alignment from file handle.*
  - unsigned int **vrna\_file\_msa\_detect\_format** (const char \*filename, unsigned int options)  
*Detect the format of a multiple sequence alignment file.*
  - int **vrna\_file\_msa\_write** (const char \*filename, const char \*\*names, const char \*\*aln, const char \*id, const char \*structure, const char \*source, unsigned int options)  
*Write multiple sequence alignment file.*

### **18.79.1 Detailed Description**

## Functions dealing with file formats for Multiple Sequence Alignments (MSA)

3

## 18.80 file\_formats\_msa.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_FILE_FORMATS_MSA_H
2 #define VIENNA_RNA_PACKAGE_FILE_FORMATS_MSA_H
3
4 #include <stdio.h>
5
6 #define VRNA_FILE_FORMAT_MSA_CLUSTAL 1U
7
8 #define VRNA_FILE_FORMAT_MSA_STOCKHOLM 2U
9
10 #define VRNA_FILE_FORMAT_MSA_FASTA 4U
11
12 #define VRNA_FILE_FORMAT_MSA_MAF 8U
13
14 #define VRNA_FILE_FORMAT_MSA_MIS 16U
15
16 #define VRNA_FILE_FORMAT_MSA_DEFAULT ( \
17     VRNA_FILE_FORMAT_MSA_CLUSTAL \
18     | VRNA_FILE_FORMAT_MSA_STOCKHOLM \
19     | VRNA_FILE_FORMAT_MSA_FASTA \
20     | VRNA_FILE_FORMAT_MSA_MAF \
21 )
22
23
24 #define VRNA_FILE_FORMAT_MSA_NOCHECK 4096
25
26 #define VRNA_FILE_FORMAT_MSA_UNKNOWN 8192
27
28 #define VRNA_FILE_FORMAT_MSA_APPEND 16384
29
30 #define VRNA_FILE_FORMAT_MSA QUIET 32768
31
32 #define VRNA_FILE_FORMAT_MSA_SILENT 65536
33
34 int
35 vrna_file_msa_read(const char *filename,
36                     char ***names,
37                     char ***aln,
38                     char **id,
39                     char **structure,
40                     unsigned int options);
41
42
43 int
44 vrna_file_msa_read_record(FILE *fp,
45                            char ***name,
46                            char ***aln,
47                            char **id,
48                            char **str,
49                            unsigned int option);
50
51
52 unsigned int
53 vrna_file_msa_detect_format(const char *file)
54 {
55     if (strcmp(file, "clustal") == 0)
56         return VRNA_FILE_FORMAT_MSA_CLUSTAL;
57     else if (strcmp(file, "stockholm") == 0)
58         return VRNA_FILE_FORMAT_MSA_STOCKHOLM;
59     else if (strcmp(file, "fasta") == 0)
60         return VRNA_FILE_FORMAT_MSA_FASTA;
61     else if (strcmp(file, "maf") == 0)
62         return VRNA_FILE_FORMAT_MSA_MAF;
63     else
64         return VRNA_FILE_FORMAT_MSA_DEFAULT;
65 }

```

```

246             unsigned int  options);
247
248
249 int
250 vrna_file_msa_write(const char      *filename,
251                      const char    **names,
252                      const char    **aln,
253                      const char    *id,
254                      const char    *structure,
255                      const char    *source,
256                      unsigned int   options);
257
258
259 #endif

```

## 18.81 ViennaRNA/file\_utils.h File Reference

Use [ViennaRNA/io/utils.h](#) instead.

Include dependency graph for file\_utils.h:

### 18.81.1 Detailed Description

Use [ViennaRNA/io/utils.h](#) instead.

**Deprecated** Use [ViennaRNA/io/utils.h](#) instead

## 18.82 file\_utils.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_FILE_UTILS_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_FILE_UTILS_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 # ifdef VRNA_WARN_DEPRECATED
6 # warning "Including deprecated header file <ViennaRNA/file_utils.h>! Use <ViennaRNA/io/utils.h> instead!"
7 # endif
8 #include <ViennaRNA/io/utils.h>
9 #endif
10
11#endif

```

## 18.83 ViennaRNA/findpath.h File Reference

Use [ViennaRNA/landscape/findpath.h](#) instead.

Include dependency graph for findpath.h:

### 18.83.1 Detailed Description

Use [ViennaRNA/landscape/findpath.h](#) instead.

**Deprecated** Use [ViennaRNA/landscape/findpath.h](#) instead

## 18.84 findpath.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_FINDPATH_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_FINDPATH_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 # ifdef VRNA_WARN_DEPRECATED
6 # warning "Including deprecated header file <ViennaRNA/findpath.h>! Use <ViennaRNA/landscape/findpath.h>
7           instead!"
8 # endif
9 #include <ViennaRNA/landscape/findpath.h>
10
11#endif

```

## 18.85 ViennaRNA/landscape/findpath.h File Reference

A breadth-first search heuristic for optimal direct folding paths.

Include dependency graph for findpath.h: This graph shows which files directly or indirectly include this file:

### Functions

- int `vrna_path_findpath_saddle (vrna_fold_compound_t *fc, const char *s1, const char *s2, int width)`  
*Find energy of a saddle point between 2 structures (search only direct path)*
- int `vrna_path_findpath_saddle_ub (vrna_fold_compound_t *fc, const char *s1, const char *s2, int width, int maxE)`  
*Find energy of a saddle point between 2 structures (search only direct path)*
- `vrna_path_t * vrna_path_findpath (vrna_fold_compound_t *fc, const char *s1, const char *s2, int width)`  
*Find refolding path between 2 structures (search only direct path)*
- `vrna_path_t * vrna_path_findpath_ub (vrna_fold_compound_t *fc, const char *s1, const char *s2, int width, int maxE)`  
*Find refolding path between 2 structures (search only direct path)*
- int `find_saddle (const char *seq, const char *s1, const char *s2, int width)`  
*Find energy of a saddle point between 2 structures (search only direct path)*
- void `free_path (vrna_path_t *path)`  
*Free memory allocated by `get_path()` function.*
- `vrna_path_t * get_path (const char *seq, const char *s1, const char *s2, int width)`  
*Find refolding path between 2 structures (search only direct path)*

### 18.85.1 Detailed Description

A breadth-first search heuristic for optimal direct folding paths.

## 18.86 findpath.h

Go to the documentation of this file.

```

1 #ifndef VIENNA_RNA_PACKAGE_FIND_PATH_H
2 #define VIENNA_RNA_PACKAGE_FIND_PATH_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 #  define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
30 #include <ViennaRNA/fold_compound.h>
31 #include <ViennaRNA/landscape/paths.h>
32
54 int
55 vrna_path_findpath_saddle(vrna_fold_compound_t *fc,
56                           const char *s1,
57                           const char *s2,
58                           int width);
59
60
87 int
88 vrna_path_findpath_saddle_ub(vrna_fold_compound_t *fc,
89                               const char *s1,
90                               const char *s2,
91                               int width,
92                               int maxE);
93
94
116 vrna_path_t *
117 vrna_path_findpath(vrna_fold_compound_t *fc,
118                      const char *s1,
119                      const char *s2,
```

```

120           int           width);
121
122
150 vrna_path_t *
151 vrna_path_findpath_ub(vrna_fold_compound_t *fc,
152                         const char   *s1,
153                         const char   *s2,
154                         int          width,
155                         int          maxE);
156
157
158 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
159
176 DEPRECATED(int
177             find_saddle(const char *seq,
178                         const char *s1,
179                         const char *s2,
180                         int          width),
181             "Use vrna_path_findpath_saddle() instead!");
182
183
193 DEPRECATED(void
194             free_path(vrna_path_t *path),
195             "Use vrna_path_free() instead!");
196
197
214 DEPRECATED(vrna_path_t *
215             get_path(const char *seq,
216                         const char *s1,
217                         const char *s2,
218                         int          width),
219             "Use vrna_path_findpath() instead!");
220
221
222 #endif
223
228 #endif

```

## 18.87 ViennaRNA/fold.h File Reference

MFE calculations for single RNA sequences.

Include dependency graph for fold.h:

### Functions

- float **fold\_par** (const char \*sequence, char \*structure, **vrna\_param\_t** \*parameters, int is\_constrained, int is←\_circular)  
*Compute minimum free energy and an appropriate secondary structure of an RNA sequence.*
- float **fold** (const char \*sequence, char \*structure)  
*Compute minimum free energy and an appropriate secondary structure of an RNA sequence.*
- float **circfold** (const char \*sequence, char \*structure)  
*Compute minimum free energy and an appropriate secondary structure of a circular RNA sequence.*
- void **free\_arrays** (void)  
*Free arrays for mfe folding.*
- void **update\_fold\_params** (void)  
*Recalculate energy parameters.*
- void **update\_fold\_params\_par** (**vrna\_param\_t** \*parameters)  
*Recalculate energy parameters.*
- void **export\_fold\_arrays** (int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*indx\_p, char \*\*ptype\_p)
- void **export\_fold\_arrays\_par** (int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*indx\_p, char \*\*ptype\_p, **vrna\_param\_t** \*\*P\_p)
- void **export\_circfold\_arrays** (int \*Fc\_p, int \*FcH\_p, int \*FcI\_p, int \*FcM\_p, int \*\*fM2\_p, int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*indx\_p, char \*\*ptype\_p)
- void **export\_circfold\_arrays\_par** (int \*Fc\_p, int \*FcH\_p, int \*FcI\_p, int \*FcM\_p, int \*\*fM2\_p, int \*\*f5\_p, int \*\*c\_p, int \*\*fML\_p, int \*\*fM1\_p, int \*\*indx\_p, char \*\*ptype\_p, **vrna\_param\_t** \*\*P\_p)
- int **LoopEnergy** (int n1, int n2, int type, int type\_2, int si1, int sj1, int sp1, int sq1)
- int **HairpinE** (int size, int type, int si1, int sj1, const char \*string)
- void **initialize\_fold** (int length)

### **18.87.1 Detailed Description**

## MFE calculations for single RNA sequences.

**18.88 fold.h**

[Go to the documentation of this file.](#)

```

197             int **indx_p,
198             char **ptype_p),
199 "Use the new API with vrna_fold_compound_t datastructure instead");
200
206 DEPRECATED(void
207 export_circfold_arrays_par( int *Fc_p,
208                             int *FcH_p,
209                             int *FcI_p,
210                             int *FcM_p,
211                             int **fM2_p,
212                             int **f5_p,
213                             int **c_p,
214                             int **fML_p,
215                             int **fM1_p,
216                             int **indx_p,
217                             char **ptype_p,
218                             vrna_param_t ***P_p),
219 "Use the new API with vrna_fold_compound_t datastructure instead");
220
221
222
223 /* finally moved the loop energy function declarations to this header... */
224 /* BUT: The functions only exist for backward compatibility reasons! */
225 /* You better include "loop_energies.h" and call the functions: */
226 /* E_Hairpin() and E_IntLoop() which are (almost) threadsafe as they get */
227 /* a pointer to the energy parameter data structure as additional argument */
228
233 DEPRECATED(int LoopEnergy(int n1,
234                            int n2,
235                            int type,
236                            int type_2,
237                            int sil,
238                            int sjl,
239                            int spl,
240                            int sq1),
241 "This function is obsolete");
242
247 DEPRECATED(int HairpinE(int size,
248                           int type,
249                           int sil,
250                           int sjl,
251                           const char *string),
252 "Use E_Hairpin() instead");
253
259 DEPRECATED(void initialize_fold(int length),
260 "This function is obsolete");
261
265 DEPRECATED(char *backtrack_fold_from_pair(char *sequence,
266                                              int i,
267                                              int j),
268 "This function is obsolete. Consider using vrna_backtrack_from_intervals() instead");
269
270
271 #endif
272
277 #endif

```

## 18.89 ViennaRNA/fold\_compound.h File Reference

The Basic Fold Compound API.

Include dependency graph for fold\_compound.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct `vrna_fc_s`

*The most basic data structure required by many functions throughout the RNAlib.* [More...](#)

### Macros

- `#define VRNA_STATUS_MFE_PRE` (unsigned char)1

*Status message indicating that MFE computations are about to begin.*

- `#define VRNA_STATUS_MFE_POST` (unsigned char)2

*Status message indicating that MFE computations are finished.*

- `#define VRNA_STATUS_PF_PRE` (unsigned char)3

*Status message indicating that Partition function computations are about to begin.*

- `#define VRNA_STATUS_PF_POST` (unsigned char)4  
*Status message indicating that Partition function computations are finished.*
- `#define VRNA_OPTION_DEFAULT` 0U  
*Option flag to specify default settings/requirements.*
- `#define VRNA_OPTION_MFE` 1U  
*Option flag to specify requirement of Minimum Free Energy (MFE) DP matrices and corresponding set of energy parameters.*
- `#define VRNA_OPTION_PF` 2U  
*Option flag to specify requirement of Partition Function (PF) DP matrices and corresponding set of Boltzmann factors.*
- `#define VRNA_OPTION_HYBRID` 4U  
*Option flag to specify requirement of dimer DP matrices.*
- `#define VRNA_OPTION_EVAL_ONLY` 8U  
*Option flag to specify that neither MFE, nor PF DP matrices are required.*
- `#define VRNA_OPTION_WINDOW` 16U  
*Option flag to specify requirement of DP matrices for local folding approaches.*

## Typedefs

- `typedef struct vrna_fc_s vrna_fold_compound_t`  
*Typename for the fold\_compound data structure `vrna_fc_s`.*
- `typedef void() vrna_callback_free_auxdata(void *data)`  
*Callback to free memory allocated for auxiliary user-provided data.*
- `typedef void() vrna_callback_recursion_status(unsigned char status, void *data)`  
*Callback to perform specific user-defined actions before, or after recursive computations.*

## Enumerations

- `enum vrna_fc_type_e { VRNA_FC_TYPE_SINGLE , VRNA_FC_TYPE_COMPARATIVE }`  
*An enumerator that is used to specify the type of a `vrna_fold_compound_t`.*

## Functions

- `vrna_fold_compound_t * vrna_fold_compound (const char *sequence, const vrna_md_t *md_p, unsigned int options)`  
*Retrieve a `vrna_fold_compound_t` data structure for single sequences and hybridizing sequences.*
- `vrna_fold_compound_t * vrna_fold_compound_comparative (const char **sequences, vrna_md_t *md_p, unsigned int options)`  
*Retrieve a `vrna_fold_compound_t` data structure for sequence alignments.*
- `void vrna_fold_compound_free (vrna_fold_compound_t *fc)`  
*Free memory occupied by a `vrna_fold_compound_t`.*
- `void vrna_fold_compound_add_auxdata (vrna_fold_compound_t *fc, void *data, vrna_callback_free_auxdata *f)`  
*Add auxiliary data to the `vrna_fold_compound_t`.*
- `void vrna_fold_compound_add_callback (vrna_fold_compound_t *fc, vrna_callback_recursion_status *f)`  
*Add a recursion status callback to the `vrna_fold_compound_t`.*

### 18.89.1 Detailed Description

The Basic Fold Compound API.

## 18.90 fold\_compound.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_FOLD_COMPOUND_H
2 #define VIENNA_RNA_PACKAGE_FOLD_COMPOUND_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 #  define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38 typedef struct vrna_fc_s vrna_fold_compound_t;
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58 typedef void (vrna_callback_free_auxdata)(void *data);
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78 typedef void (vrna_callback_recursion_status)(unsigned char status,
79                                     void           *data);
80
81
82
83
84
85
86
87 #define VRNA_STATUS_MFE_PRE      (unsigned char)1
88
89
90
91
92
93
94
95 #define VRNA_STATUS_MFE_POST    (unsigned char)2
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112 #include <ViennaRNA/model.h>
113 #include <ViennaRNA/params/basic.h>
114 #include <ViennaRNA/sequence.h>
115 #include <ViennaRNA/dp_matrices.h>
116 #include <ViennaRNA/constraints/hard.h>
117 #include <ViennaRNA/constraints/soft.h>
118 #include <ViennaRNA/grammar.h>
119 #include <ViennaRNA/structured_domains.h>
120 #include <ViennaRNA/unstructured_domains.h>
121
122 #ifdef VRNA_WITH_SVM
123 #include <ViennaRNA/zscore.h>
124 #endif
125
126
127
128
129
130 typedef enum {
131     VRNA_FC_TYPE_SINGLE,
132     VRNA_FC_TYPE_COMPARATIVE
133 } vrna_fc_type_e;
134
135
136
137
138 struct vrna_fc_s {
139     const vrna_fc_type_e type;
140     unsigned int          length;
141 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
142     DEPRECATED(int cutpoint,
143                "Use strand_* members instead");
144 #endif
145     unsigned int          *strand_number;
146     unsigned int          *strand_order;
147     unsigned int          *strand_order_uniq;
148     unsigned int          *strand_start;
149     unsigned int          *strand_end;
150     unsigned int          strands;
151     vrna_seq_t            *nucleotides;
152     vrna_msa_t            *alignment;
153     vrna_hc_t              *hc;
154     vrna_mx_mfe_t          *matrices;
155     vrna_mx_pf_t            *exp_matrices;
156     vrna_param_t            *params;
157     vrna_exp_param_t        *exp_params;
158     int                     *iindx;
159     int                     *jindx;
160     vrna_callback_recursion_status *stat_cb;
161     void                   *auxdata;
162     vrna_callback_free_auxdata   *free_auxdata;
163 /* data structure to adjust additional structural domains, such as G-quadruplexes */
164     vrna_sd_t               *domains_struct;
165 /* data structure to adjust additional contributions to unpaired stretches, e.g. due to protein
166     binding */
167     vrna_ud_t               *domains_up;
168 /* auxiliary (user-defined) extension to the folding grammar */
169 
```

```
222     vrna_gr_aux_t *aux_grammar;
223 #ifndef VRNA_DISABLE_C11_FEATURES
224     /* C11 support for unnamed unions/structs */
225     union {
226         struct {
227             char *sequence;
228             short *sequence_encoding;
229             short *encoding5;
230             short *encoding3;
231             short *sequence_encoding2;
232         };
233     };
234 #endif
235     char *ptype;
236     char *ptype_pf_compat;
237     vrna_sc_t *sc;
238 #ifndef VRNA_DISABLE_C11_FEATURES
239     /* C11 support for unnamed unions/structs */
240     };
241     struct {
242         char **sequences;
243         unsigned int n_seq;
244         char *cons_seq;
245         short *S_cons;
246         short **S;
247         short **S5;
248         short **S3;
249         char **Ss;
250         unsigned int **a2s;
251         int *pscore;
252         int **pscore_local;
253         short *pscore_pf_compat;
254         vrna_sc_t **scs;
255         int oldAliEn;
256     };
257 #ifndef VRNA_DISABLE_C11_FEATURES
258     };
259     };
260     };
261     };
262     };
263 #endif
264     unsigned int maxD1;
265     unsigned int maxD2;
266     short *reference_pt1;
267     short *reference_pt2;
268     unsigned int *referenceBPs1;
269     unsigned int *referenceBPs2;
270     unsigned int *bpdist;
271     unsigned int *mml;
272     unsigned int *mm2;
273     int window_size;
274     char **ptype_local;
275 #ifdef VRNA_WITH_SVM
276     vrna_zsc_dat_t zscore_data;
277 #endif
278     };
279     };
280     };
281     };
282     };
283     };
284     };
285     };
286     };
287     };
288     };
289     };
290     };
291     };
292     };
293     };
294     };
295     };
296     };
297     };
298     };
299     };
300     };
301     };
302     };
303     };
304     };
305     };
306     };
307     };
308     };
309     };
310     };
311     };
312     };
313     };
314     };
315     };
316     };
317     };
318     };
319     };
320     };
321     };
322     };
323     };
324     };
325     };
326     };
327     };
328     };
329     };
330     };
331     };
332     };
333     };
334     };
335     };
336     };
337     };
338     };
339     };
340     };
341     };
342     };
343     };
344     };
345     };
346     };
347     };
348     };
349     };
350     };
351     };
352     };
353     };
354     };
355     };
356     };
357     };
358     };
359     };
360     };
361     };
362     };
363     };
364     };
365     };
366     };
367     };
368     };
369     };
370     /* the definitions below should be used for functions that return/receive/destroy fold compound data
371     structures */
372     };
373     #define VRNA_OPTION_DEFAULT          0U
374     #define VRNA_OPTION_MFE              1U
375     #define VRNA_OPTION_PF               2U
376     #define VRNA_OPTION_HYBRID           4U
377     #define VRNA_OPTION_EVAL_ONLY        8U
378     #define VRNA_OPTION_WINDOW           16U
379     };
380     vrna_fold_compound_t *
381     vrna_fold_compound(const char      *sequence,
382                         const vrna_md_t *md_p,
383                         unsigned int    options);
384     };
385     vrna_fold_compound_t *
386     vrna_fold_compound_comparative(const char   **sequences,
387                                     vrna_md_t   *md_p,
388                                     unsigned int options);
389     
```

```

498
499
500 vrna_fold_compound_t *
501 vrna_fold_compound_comparative2(const char          **sequences,
502                                     const char          **names,
503                                     const unsigned char *orientation,
504                                     const unsigned long long *start,
505                                     const unsigned long long *genome_size,
506                                     vrna_md_t           *md_p,
507                                     unsigned int         options);
508
509
510 vrna_fold_compound_t *
511 vrna_fold_compound_TwoD(const char      *sequence,
512                           const char      *s1,
513                           const char      *s2,
514                           vrna_md_t       *md_p,
515                           unsigned int     options);
516
517
518 int
519 vrna_fold_compound_prepare(vrna_fold_compound_t *fc,
520                             unsigned int     options);
521
522
523 void
524 vrna_fold_compound_free(vrna_fold_compound_t *fc);
525
526
527 void
528 vrna_fold_compound_add_auxdata(vrna_fold_compound_t *fc,
529                                 void            *data,
530                                 vrna_callback_free_auxdata *f);
531
532
533
534 void
535 vrna_fold_compound_add_callback(vrna_fold_compound_t *fc,
536                                  vrna_callback_recursion_status *f);
537
538
539
540 #endif

```

## 18.91 ViennaRNA/fold\_vars.h File Reference

Here all declarations of the global variables used throughout RNAlib.

Include dependency graph for fold\_vars.h: This graph shows which files directly or indirectly include this file:

### Variables

- int **fold\_constrained**  
*Global switch to activate/deactivate folding with structure constraints.*
- int **csv**  
*generate comma seperated output*
- char \* **RibosumFile**
- int **james\_rule**
- int **logML**
- int **cut\_point**  
*Marks the position (starting from 1) of the first nucleotide of the second molecule within the concatenated sequence.*
- **bondT** \* **base\_pair**  
*Contains a list of base pairs after a call to [fold\(\)](#).*
- **FLT\_OR\_DBL** \* **pr**  
*A pointer to the base pair probability matrix.*
- int \* **iindx**  
*index array to move through pr.*

### 18.91.1 Detailed Description

Here all declarations of the global variables used throughout RNAlib.

## 18.91.2 Variable Documentation

### 18.91.2.1 RibosumFile

```
char* RibosumFile [extern]
warning this variable will vanish in the future ribosums will be compiled in instead
```

### 18.91.2.2 james\_rule

```
int james_rule [extern]
interior loops of size 2 get energy 0.8Kcal and no mismatches, default 1
```

### 18.91.2.3 logML

```
int logML [extern]
use logarithmic multiloop energy function
```

### 18.91.2.4 cut\_point

```
int cut_point [extern]
Marks the position (starting from 1) of the first nucleotide of the second molecule within the concatenated sequence. To evaluate the energy of a duplex structure (a structure formed by two strands), concatenate the two sequences and set it to the first base of the second strand in the concatenated sequence. The default value of -1 stands for single molecule folding. The cut_point variable is also used by vrna\_file\_PS\_rnplot\(\) and PS\_dot\_plot\(\) to mark the chain break in postscript plots.
```

### 18.91.2.5 base\_pair

```
bondT* base_pair [extern]
Contains a list of base pairs after a call to fold\(\).
base_pair[0].i contains the total number of pairs.
```

**Deprecated** Do not use this variable anymore!

### 18.91.2.6 pr

```
FLOAT_OR_DOUBLE* pr [extern]
A pointer to the base pair probability matrix.
```

**Deprecated** Do not use this variable anymore!

### 18.91.2.7 iindx

```
int* iindx [extern]
index array to move through pr.
The probability for base i and j to form a pair is in pr[iindx[i]-j].
```

**Deprecated** Do not use this variable anymore!

## 18.92 fold\_vars.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_FOLD_VARS_H
2 #define VIENNA_RNA_PACKAGE_FOLD_VARS_H
3
4 #include <ViennaRNA/datastructures/basic.h>
```

```

5 /* For now, we include model.h by default to provide backwards compatibility
6 However, this will most likely change, since fold_vars.h is scheduled to
7 vanish from the sources at latest in ViennaRNA Package v3
8 */
9 #include <ViennaRNA/model.h>
10
11
12 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
13
14 extern int    fold_constrained;
15
16 extern int    csv;
17
18 extern char *RibosumFile;
19
20 extern int    james_rule;
21
22 extern int    logML;
23
24 extern int    cut_point;
25
26 extern bondT *base_pair;
27
28 extern FLT_OR_DBL *pr;
29
30 extern int    *iindx;
31
32
33 #endif
34
35
36 #endif
37
38
39 #endif
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86

```

## 18.93 ViennaRNA/gquad.h File Reference

G-quadruplexes.

Include dependency graph for gquad.h:

### Functions

- int \* [get\\_gquad\\_matrix](#) (short \*S, [vrna\\_param\\_t](#) \*P)  
*Get a triangular matrix prefilled with minimum free energy contributions of G-quadruplexes.*
- int [parse\\_gquad](#) (const char \*struc, int \*L, int l[3])
- PRIVATE int [backtrack\\_GQuad\\_IntLoop](#) (int c, int i, int j, int type, short \*S, int \*ggg, int \*index, int \*p, int \*q, [vrna\\_param\\_t](#) \*P)
- PRIVATE int [backtrack\\_GQuad\\_IntLoop\\_L](#) (int c, int i, int j, int type, short \*S, int \*\*ggg, int maxdist, int \*p, int \*q, [vrna\\_param\\_t](#) \*P)

### 18.93.1 Detailed Description

G-quadruplexes.

## 18.94 gquad.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_GQUAD_H
2 #define VIENNA_RNA_PACKAGE_GQUAD_H
3
4 #include <ViennaRNA/datastructures/basic.h>
5 #include <ViennaRNA/fold_compound.h>
6 #include <ViennaRNA/params/basic.h>
7
8 #ifndef INLINE
9 #ifdef __GNUC__
10 # define INLINE inline
11 #else
12 # define INLINE
13 #endif
14 #endif
15
16 int      E_gquad(int           L,
17                  int           l[3],
18                  vrna\_param\_t *P);
19
20
21
22
23
24
25
26
27
28
29
30
31
32

```

```

33
34
35 FLT_OR_DBL exp_E_gquad(int i, L,
36           int l[3],
37           vrna_exp_param_t *pf);
38
39
40 void E_gquad_ali_en(int i,
41           int L,
42           int l[3],
43           const short **S,
44           unsigned int **a2s,
45           unsigned int n_seq,
46           vrna_param_t *P,
47           int en[2]);
48
49
66 int *get_gquad_matrix(short *S,
67           vrna_param_t *P);
68
69
70 int *get_gquad_ali_matrix(unsigned int n,
71           short *S_cons,
72           short **S,
73           unsigned int **a2s,
74           int n_seq,
75           vrna_param_t *P);
76
77
78 FLT_OR_DBL *get_gquad_pf_matrix(short *S,
79           FLT_OR_DBL *scale,
80           vrna_exp_param_t *pf);
81
82
83 FLT_OR_DBL *get_gquad_pf_matrix_comparative(unsigned int n,
84           short *S_cons,
85           short **S,
86           unsigned int **a2s,
87           FLT_OR_DBL *scale,
88           unsigned int n_seq,
89           vrna_exp_param_t *pf);
90
91
92 int **get_gquad_L_matrix(short *S,
93           int start,
94           int maxdist,
95           int n,
96           int **g,
97           vrna_param_t *P);
98
99
100 void vrna_gquad_mx_local_update(vrna_fold_compound_t *vc,
101           int start);
102
103
104 void get_gquad_pattern_mfe(short *S,
105           int i,
106           int j,
107           vrna_param_t *P,
108           int *L,
109           int l[3]);
110
111
112 void
113 get_gquad_pattern_exhaustive(short *S,
114           int i,
115           int j,
116           vrna_param_t *P,
117           int *L,
118           int *l,
119           int threshold);
120
121
122 void get_gquad_pattern_pf(short *S,
123           int i,
124           int j,
125           vrna_exp_param_t *pf,
126           int *L,
127           int l[3]);
128
129
130 plist *get_plist_gquad_from_pr(short *S,
131           int gi,
132           int gj,
133           FLT_OR_DBL *G,
134           FLT_OR_DBL *probs,
135           FLT_OR_DBL *scale,

```

```

136                         vrna_exp_param_t *pf);
137
138
139 plist *get_plist_gquad_from_pr_max(short           *S,
140                                     int                gi,
141                                     int                gj,
142                                     FLT_OR_DBL        *G,
143                                     FLT_OR_DBL        *probs,
144                                     FLT_OR_DBL        *scale,
145                                     int                *L,
146                                     int                l[3],
147                                     vrna_exp_param_t *pf);
148
149
150 plist *get_plist_gquad_from_db(const char *structure,
151                                 float      pr);
152
153
154 plist *
155 vrna_get_plist_gquad_from_pr(vrna_fold_compound_t *fc,
156                               int                gi,
157                               int                gj);
158
159
160 plist *
161 vrna_get_plist_gquad_from_pr_max(vrna_fold_compound_t *fc,
162                                   int                gi,
163                                   int                gj,
164                                   int                *lmax,
165                                   int                lmax[3]);
166
167
168 int      get_gquad_count(short *S,
169                          int   i,
170                          int   j);
171
172
173 int      get_gquad_layer_count(short *S,
174                                int   i,
175                                int   j);
176
177
178 void get_gquad_pattern_mfe_ali(short      **S,
179                                  unsigned int **a2s,
180                                  short       *S_cons,
181                                  int        n_seq,
182                                  int        i,
183                                  int        j,
184                                  vrna_param_t *P,
185                                  int        *L,
186                                  int        l[3]);
187
188
189 int parse_gquad(const char *struc,
200                  int   *L,
201                  int   l[3]);
202
203
204 INLINE PRIVATE int backtrack_GQuad_IntLoop(int      c,
205                                              int      i,
206                                              int      j,
207                                              int      type,
208                                              short   *S,
209                                              int      *ggg,
210                                              int      *index,
211                                              int      *p,
212                                              int      *q,
213                                              vrna_param_t *P);
214
215
216 INLINE PRIVATE int backtrack_GQuad_IntLoop_comparative(int      c,
217                                                       int      i,
218                                                       int      j,
219                                                       unsigned int *type,
220                                                       short   *S_cons,
221                                                       short   **S5,
222                                                       short   **S3,
223                                                       unsigned int **a2s,
224                                                       int      *ggg,
225                                                       int      *index,
226                                                       int      *p,
227                                                       int      *q,
228                                                       int      n_seq,
229                                                       vrna_param_t *P);
230
231
232 INLINE PRIVATE int backtrack_GQuad_IntLoop_L(int      c,

```

```

233                               int      i,
234                               int      j,
235                               int      type,
236                               short   *S,
237                               int      **ggg,
238                               int      maxdist,
239                               int      *p,
240                               int      *q,
241                               vrna_param_t *P);
242
243
244 PRIVATE INLINE int
245 vrna_BT_gquad_int(vrna_fold_compound_t *vc,
246                     int      i,
247                     int      j,
248                     int      en,
249                     vrna_bp_stack_t *bp_stack,
250                     int      *stack_count);
251
252
253 PRIVATE INLINE int
254 vrna_BT_gquad_mfe(vrna_fold_compound_t *vc,
255                      int      i,
256                      int      j,
257                      vrna_bp_stack_t *bp_stack,
258                      int      *stack_count)
259 {
260     /*
261      * here we do some fancy stuff to backtrace the stacksize and linker lengths
262      * of the g-quadruplex that should reside within position i,j
263      */
264     short      *S;
265     int       l[3], L, a, n_seq;
266     vrna_param_t *P;
267
268     if (vc) {
269         P = vc->params;
270         switch (vc->type) {
271             case VRNA_FC_TYPE_SINGLE:
272                 S = vc->sequence_encoding2;
273                 L = -1;
274
275                 get_gquad_pattern_mfe(S, i, j, P, &L, l);
276                 break;
277
278             case VRNA_FC_TYPE_COMPARATIVE:
279                 n_seq = vc->n_seq;
280                 L     = -1;
281                 get_gquad_pattern_mfe_ali(vc->S, vc->a2s, vc->S_cons, n_seq, i, j, P, &L, l);
282                 break;
283         }
284
285         if (L != -1) {
286             /* fill the G's of the quadruplex into base_pair2 */
287             for (a = 0; a < L; a++) {
288                 bp_stack[(*stack_count)].i = i + a;
289                 bp_stack[(*stack_count)].j = i + a;
290                 bp_stack[(*stack_count)].i = i + L + l[0] + a;
291                 bp_stack[(*stack_count)].j = i + L + l[0] + a;
292                 bp_stack[(*stack_count)].i = i + L + l[0] + L + l[1] + a;
293                 bp_stack[(*stack_count)].j = i + L + l[0] + L + l[1] + a;
294                 bp_stack[(*stack_count)].i = i + L + l[0] + L + l[1] + L + l[2] + a;
295                 bp_stack[(*stack_count)].j = i + L + l[0] + L + l[1] + L + l[2] + a;
296             }
297             return 1;
298         } else {
299             return 0;
300         }
301     }
302
303     return 0;
304 }
305
306
307 PRIVATE INLINE int
308 vrna_BT_gquad_int(vrna_fold_compound_t *vc,
309                     int      i,
310                     int      j,
311                     int      en,
312                     vrna_bp_stack_t *bp_stack,
313                     int      *stack_count)
314 {
315     int      energy, dangles, *idx, ij, p, q, maxl, minl, c0, l1, *ggg;
316     unsigned char type;
317     char    *ptype;
318     short   si, sj, *S, *S1;
319

```

```

320     vrna_param_t *P;
321     vrna_md_t     *md;
322
323     idx    = vc->jindx;
324     ij     = idx[j] + i;
325     P      = vc->params;
326     md    = &(P->model_details);
327     ptype = vc->pctype;
328     type   = (unsigned char)pctype[ij];
329     S1    = vc->sequence_encoding;
330     S     = vc->sequence_encoding2;
331     dangles = md->dangles;
332     si     = S1[i + 1];
333     sj     = S1[j - 1];
334     ggg    = vc->matrices->ggg;
335     energy = 0;
336
337     if (dangles == 2)
338         energy += P->mismatchI[type][si][sj];
339
340     if (type > 2)
341         energy += P->TerminalAU;
342
343     p = i + 1;
344     if (S1[p] == 3) {
345         if (p < j - VRNA_GQUAD_MIN_BOX_SIZE) {
346             minl = j - i + p - MAXLOOP - 2;
347             c0   = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
348             minl = MAX2(c0, minl);
349             c0   = j - 3;
350             maxl = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
351             maxl = MIN2(c0, maxl);
352             for (q = minl; q < maxl; q++) {
353                 if (S[q] != 3)
354                     continue;
355
356                 if (en == energy + ggg[idx[q] + p] + P->internal_loop[j - q - 1])
357                     return vrna_BT_gquad_mfe(vc, p, q, bp_stack, stack_count);
358             }
359         }
360     }
361
362     for (p = i + 2;
363          p < j - VRNA_GQUAD_MIN_BOX_SIZE;
364          p++) {
365         l1 = p - i - 1;
366         if (l1 > MAXLOOP)
367             break;
368
369         if (S1[p] != 3)
370             continue;
371
372         minl = j - i + p - MAXLOOP - 2;
373         c0   = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
374         minl = MAX2(c0, minl);
375         c0   = j - 1;
376         maxl = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
377         maxl = MIN2(c0, maxl);
378         for (q = minl; q < maxl; q++) {
379             if (S1[q] != 3)
380                 continue;
381
382             if (en == energy + ggg[idx[q] + p] + P->internal_loop[l1 + j - q - 1])
383                 return vrna_BT_gquad_mfe(vc, p, q, bp_stack, stack_count);
384         }
385     }
386
387     q = j - 1;
388     if (S1[q] == 3)
389         for (p = i + 4;
390              p < j - VRNA_GQUAD_MIN_BOX_SIZE;
391              p++) {
392             l1 = p - i - 1;
393             if (l1 > MAXLOOP)
394                 break;
395
396             if (S1[p] != 3)
397                 continue;
398
399             if (en == energy + ggg[idx[q] + p] + P->internal_loop[l1])
400                 return vrna_BT_gquad_mfe(vc, p, q, bp_stack, stack_count);
401     }
402
403     return 0;
404 }
405

```

```

424 INLINE PRIVATE int
425 backtrack_GQuad_IntLoop(int          c,
426                           int          i,
427                           int          j,
428                           int          type,
429                           short        *S,
430                           int          *ggg,
431                           int          *index,
432                           int          *p,
433                           int          *q,
434                           vrna_param_t *P)
435 {
436     int   energy, dangles, k, l, maxl, minl, c0, ll;
437     short si, sj;
438
439     dangles = P->model_details.dangles;
440     si      = S[i + 1];
441     sj      = S[j - 1];
442     energy  = 0;
443
444     if (dangles == 2)
445         energy += P->mismatchI[type][si][sj];
446
447     if (type > 2)
448         energy += P->TerminalAU;
449
450     k = i + 1;
451     if (S[k] == 3) {
452         if (k < j - VRNA_GQUAD_MIN_BOX_SIZE) {
453             minl = j - i + k - MAXLOOP - 2;
454             c0   = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
455             minl = MAX2(c0, minl);
456             c0   = j - 3;
457             maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
458             maxl = MIN2(c0, maxl);
459             for (l = minl; l < maxl; l++) {
460                 if (S[l] != 3)
461                     continue;
462
463                 if (c == energy + ggg[index[l] + k] + P->internal_loop[j - l - 1]) {
464                     *p   = k;
465                     *q   = l;
466                     return 1;
467                 }
468             }
469         }
470     }
471
472     for (k = i + 2;
473          k < j - VRNA_GQUAD_MIN_BOX_SIZE;
474          k++) {
475         ll = k - i - 1;
476         if (ll > MAXLOOP)
477             break;
478
479         if (S[k] != 3)
480             continue;
481
482         minl = j - i + k - MAXLOOP - 2;
483         c0   = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
484         minl = MAX2(c0, minl);
485         c0   = j - 1;
486         maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
487         maxl = MIN2(c0, maxl);
488         for (l = minl; l < maxl; l++) {
489             if (S[l] != 3)
490                 continue;
491
492             if (c == energy + ggg[index[l] + k] + P->internal_loop[ll + j - l - 1]) {
493                 *p   = k;
494                 *q   = l;
495                 return 1;
496             }
497         }
498     }
499
500     l = j - 1;
501     if (S[l] == 3)
502         for (k = i + 4;
503              k < j - VRNA_GQUAD_MIN_BOX_SIZE;
504              k++) {
505             ll = k - i - 1;
506             if (ll > MAXLOOP)
507                 break;
508
509             if (S[k] != 3)
510                 continue;

```

```

511     if (c == energy + ggg[index[l] + k] + P->internal_loop[ll]) {
512         *p = k;
513         *q = l;
514         return 1;
515     }
516 }
517 }
518
519 return 0;
520 }
521
522
523 #define MAXLOOP 1000
524 #define MIN2(a, b) ((a) < (b) ? (a) : (b))
525 #define MAX2(a, b) ((a) > (b) ? (a) : (b))
526
527 #include "RNA.h"
528
529 #include "GQuad.h"
530
531 #include "mismatch.h"
532
533 #include "TerminalAU.h"
534
535 #include "RNAParam.h"
536
537
538 {
539     int energy, dangles, k, l, maxl, minl, c0, ll, ss, tt, u1, u2, eee;
540
541     dangles = P->model_details.dangles;
542     energy = 0;
543
544     for (ss = 0; ss < n_seq; ss++) {
545         tt = type[ss];
546         if (tt == 0)
547             tt = 7;
548
549         if (dangles == 2)
550             energy += P->mismatchI[tt][S3[ss][i]][S5[ss][j]];
551
552         if (tt > 2)
553             energy += P->TerminalAU;
554     }
555
556     k = i + 1;
557     if (S_cons[k] == 3) {
558         if (k < j - VRNA_GQUAD_MIN_BOX_SIZE) {
559             minl = j - i + k - MAXLOOP - 2;
560             c0 = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
561             minl = MAX2(c0, minl);
562             c0 = j - 3;
563             maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
564             maxl = MIN2(c0, maxl);
565             for (l = minl; l < maxl; l++) {
566                 if (S_cons[l] != 3)
567                     continue;
568
569                 eee = 0;
570
571                 for (ss = 0; ss < n_seq; ss++) {
572                     u1 = a2s[ss][j - 1] - a2s[ss][l];
573                     eee += P->internal_loop[u1];
574                 }
575
576                 if (c == energy + ggg[index[l] + k] + eee) {
577                     *p = k;
578                     *q = l;
579                     return 1;
580                 }
581             }
582         }
583     }
584
585     for (k = i + 2;
586          k < j - VRNA_GQUAD_MIN_BOX_SIZE;
587          k++) {
588         ll = k - i - 1;
589         if (ll > MAXLOOP)
590             break;
591
592         if (S_cons[k] != 3)
593             continue;
594
595         minl = j - i + k - MAXLOOP - 2;
596         c0 = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
597         minl = MAX2(c0, minl);

```

```

598     c0      = j - 1;
599     maxl   = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
600     maxl   = MIN2(c0, maxl);
601     for (l = minl; l < maxl; l++) {
602       if (S_cons[l] != 3)
603         continue;
604       eee = 0;
605
606       for (ss = 0; ss < n_seq; ss++) {
607         u1   = a2s[ss][k - 1] - a2s[ss][i];
608         u2   = a2s[ss][j - 1] - a2s[ss][l];
609         eee += P->internal_loop[u1 + u2];
610       }
611
612       if (c == energy + ggg[index[l] + k] + eee) {
613         *p   = k;
614         *q   = l;
615         return 1;
616       }
617     }
618   }
619 }
620
621 l = j - 1;
622 if (S_cons[l] == 3)
623   for (k = i + 4;
624        k < j - VRNA_GQUAD_MIN_BOX_SIZE;
625        k++) {
626     l1 = k - i - 1;
627     if (l1 > MAXLOOP)
628       break;
629
630     if (S_cons[k] != 3)
631       continue;
632
633     eee = 0;
634
635     for (ss = 0; ss < n_seq; ss++) {
636       u1   = a2s[ss][k - 1] - a2s[ss][i];
637       eee += P->internal_loop[u1];
638     }
639
640     if (c == energy + ggg[index[l] + k] + eee) {
641       *p   = k;
642       *q   = l;
643       return 1;
644     }
645   }
646
647 return 0;
648 }
649
650
651 INLINE PRIVATE int
652 backtrack_GQuad_IntLoop_L(int          c,
653                            int          i,
654                            int          j,
655                            int          type,
656                            short        *S,
657                            int          **ggg,
658                            int          maxdist,
659                            int          *p,
660                            int          *q,
661                            vrna_param_t *P)
662 {
663   int   energy, dangles, k, l, maxl, minl, c0, l1;
664   short si, sj;
665
666   dangles = P->model_details.dangles;
667   si      = S[i + 1];
668   sj      = S[j - 1];
669   energy  = 0;
670
671   if (dangles == 2)
672     energy += P->mismatchI[type][si][sj];
673
674   if (type > 2)
675     energy += P->TerminalAU;
676
677   k = i + 1;
678   if (S[k] == 3) {
679     if (k < j - VRNA_GQUAD_MIN_BOX_SIZE) {
680       minl = j - i + k - MAXLOOP - 2;
681       c0   = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
682       minl = MAX2(c0, minl);
683       c0   = j - 3;
684       maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
685     }
686   }

```

```

701     maxl = MIN2(c0, maxl);
702     for (l = minl; l < maxl; l++) {
703         if (S[l] != 3)
704             continue;
705
706         if (c == energy + ggg[k][l - k] + P->internal_loop[j - l - 1]) {
707             *p = k;
708             *q = l;
709             return 1;
710         }
711     }
712 }
713 }
714
715 for (k = i + 2;
716      k < j - VRNA_GQUAD_MIN_BOX_SIZE;
717      k++) {
718     l1 = k - i - 1;
719     if (l1 > MAXLOOP)
720         break;
721
722     if (S[k] != 3)
723         continue;
724
725     minl = j - i + k - MAXLOOP - 2;
726     c0 = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
727     minl = MAX2(c0, minl);
728     c0 = j - 1;
729     maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
730     maxl = MIN2(c0, maxl);
731     for (l = minl; l < maxl; l++) {
732         if (S[l] != 3)
733             continue;
734
735         if (c == energy + ggg[k][l - k] + P->internal_loop[l1 + j - l - 1]) {
736             *p = k;
737             *q = l;
738             return 1;
739         }
740     }
741 }
742
743 l = j - 1;
744 if (S[l] == 3)
745     for (k = i + 4;
746          k < j - VRNA_GQUAD_MIN_BOX_SIZE;
747          k++) {
748     l1 = k - i - 1;
749     if (l1 > MAXLOOP)
750         break;
751
752     if (S[k] != 3)
753         continue;
754
755     if (c == energy + ggg[k][l - k] + P->internal_loop[l1]) {
756         *p = k;
757         *q = l;
758         return 1;
759     }
760 }
761
762 return 0;
763 }
764
765
766 INLINE PRIVATE int
767 backtrack_GQuad_IntLoop_L_comparative(int           c,
768                                         int           i,
769                                         int           j,
770                                         unsigned int  *type,
771                                         short        *S_cons,
772                                         short        **S5,
773                                         short        **S3,
774                                         unsigned int  **a2s,
775                                         int           **ggg,
776                                         int           *p,
777                                         int           *q,
778                                         int           n_seq,
779                                         vrna_param_t *P)
780 {
781     /*
782     * The case that is handled here actually resembles something like
783     * an interior loop where the enclosing base pair is of regular
784     * kind and the enclosed pair is not a canonical one but a g-quadruplex
785     * that should then be decomposed further...
786     */
787     int mm, dangle_model, k, l, maxl, minl, c0, l1, ss, tt, eee, u1, u2;

```

```

788     dangle_model = P->model_details.dangles;
789
790     mm = 0;
791     for (ss = 0; ss < n_seq; ss++) {
792         tt = type[ss];
793
794         if (dangle_model == 2)
795             mm += P->mismatchI[tt][S3[ss][i]][S5[ss][j]];
796
797         if (tt > 2)
798             mm += P->TerminalAU;
799     }
800
801     for (k = i + 2;
802          k < j - VRNA_GQUAD_MIN_BOX_SIZE;
803          k++) {
804         if (S_cons[k] != 3)
805             continue;
806
807         l1 = k - i - 1;
808         if (l1 > MAXLOOP)
809             break;
810
811         minl = j - i + k - MAXLOOP - 2;
812         c0 = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
813         minl = MAX2(c0, minl);
814         c0 = j - 1;
815         maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
816         maxl = MIN2(c0, maxl);
817         for (l = minl; l < maxl; l++) {
818             if (S_cons[l] != 3)
819                 continue;
820
821             eee = 0;
822
823             for (ss = 0; ss < n_seq; ss++) {
824                 u1 = a2s[ss][k - 1] - a2s[ss][i];
825                 u2 = a2s[ss][j - 1] - a2s[ss][l];
826                 eee += P->internal_loop[u1 + u2];
827             }
828
829             c0 = mm +
830                 ggg[k][l - k] +
831                 eee;
832
833             if (c == c0) {
834                 *p = k;
835                 *q = l;
836                 return 1;
837             }
838         }
839     }
840
841     k = i + 1;
842     if (S_cons[k] == 3) {
843         if (k < j - VRNA_GQUAD_MIN_BOX_SIZE) {
844             minl = j - i + k - MAXLOOP - 2;
845             c0 = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
846             minl = MAX2(c0, minl);
847             c0 = j - 3;
848             maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
849             maxl = MIN2(c0, maxl);
850             for (l = minl; l < maxl; l++) {
851                 if (S_cons[l] != 3)
852                     continue;
853
854                 eee = 0;
855
856                 for (ss = 0; ss < n_seq; ss++) {
857                     u1 = a2s[ss][j - 1] - a2s[ss][l];
858                     eee += P->internal_loop[u1];
859                 }
860
861                 if (c == mm + ggg[k][l - k] + eee) {
862                     *p = k;
863                     *q = l;
864                     return 1;
865                 }
866             }
867         }
868     }
869
870     l = j - 1;
871     if (S_cons[l] == 3) {
872         for (k = i + 4; k < j - VRNA_GQUAD_MIN_BOX_SIZE; k++) {
873             l1 = k - i - 1;
874             if (l1 > MAXLOOP)

```

```

875     break;
876
877     if (S_cons[k] != 3)
878         continue;
879
880     eee = 0;
881
882     for (ss = 0; ss < n_seq; ss++) {
883         u1 = a2s[ss][k - 1] - a2s[ss][i];
884         eee += P->internal_loop[u1];
885     }
886
887     if (c == mm + ggg[k][l - k] + eee) {
888         *p = k;
889         *q = l;
890         return 1;
891     }
892 }
893 }
894
895 return 0;
896 }
897
898
899 PRIVATE INLINE
900 int
901 E_GQuad_IntLoop(int i,
902                   int j,
903                   type,
904                   short *S,
905                   int *ggg,
906                   int *index,
907                   vrna_param_t *P)
908 {
909     int energy, ge, dangles, p, q, ll, minq, maxq, c0;
910     short si, sj;
911
912     dangles = P->model_details.dangles;
913     si = S[i + 1];
914     sj = S[j - 1];
915     energy = 0;
916
917     if (dangles == 2)
918         energy += P->mismatchI[type][si][sj];
919
920     if (type > 2)
921         energy += P->TerminalAU;
922
923     ge = INF;
924
925     p = i + 1;
926     if (S[p] == 3) {
927         if (p < j - VRNA_GQUAD_MIN_BOX_SIZE) {
928             minq = j - i + p - MAXLOOP - 2;
929             c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
930             minq = MAX2(c0, minq);
931             c0 = j - 3;
932             maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
933             maxq = MIN2(c0, maxq);
934             for (q = minq; q < maxq; q++) {
935                 if (S[q] != 3)
936                     continue;
937
938                 c0 = energy + ggg[index[q] + p] + P->internal_loop[j - q - 1];
939                 ge = MIN2(ge, c0);
940             }
941         }
942     }
943
944     for (p = i + 2;
945          p < j - VRNA_GQUAD_MIN_BOX_SIZE;
946          p++) {
947         ll = p - i - 1;
948         if (ll > MAXLOOP)
949             break;
950
951         if (S[p] != 3)
952             continue;
953
954         minq = j - i + p - MAXLOOP - 2;
955         c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
956         minq = MAX2(c0, minq);
957         c0 = j - 1;
958         maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
959         maxq = MIN2(c0, maxq);
960         for (q = minq; q < maxq; q++) {
961             if (S[q] != 3)

```

```

962         continue;
963
964     c0 = energy + ggg[index[q] + p] + P->internal_loop[l1 + j - q - 1];
965     ge = MIN2(ge, c0);
966 }
967 }
968
969 q = j - 1;
970 if (S[q] == 3)
971   for (p = i + 4;
972       p < j - VRNA_GQUAD_MIN_BOX_SIZE;
973       p++) {
974     l1 = p - i - 1;
975     if (l1 > MAXLOOP)
976       break;
977
978     if (S[p] != 3)
979       continue;
980
981     c0 = energy + ggg[index[q] + p] + P->internal_loop[l1];
982     ge = MIN2(ge, c0);
983   }
984
985 #if 0
986 /* here comes the additional stuff for the odd dangle models */
987 if (dangles % 1) {
988   en1 = energy + P->dangle5[type][si];
989   en2 = energy + P->dangle5[type][sj];
990   en3 = energy + P->mismatchI[type][si][sj];
991
992 /* first case with 5' dangle (i.e. j-1) onto enclosing pair */
993 p = i + 1;
994 if (S[p] == 3) {
995   if (p < j - VRNA_GQUAD_MIN_BOX_SIZE) {
996     minq = j - i + p - MAXLOOP - 2;
997     c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
998     minq = MAX2(c0, minq);
999     c0 = j - 4;
1000    maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
1001    maxq = MIN2(c0, maxq);
1002    for (q = minq; q < maxq; q++) {
1003      if (S[q] != 3)
1004        continue;
1005
1006      c0 = en1 + ggg[index[q] + p] + P->internal_loop[j - q - 1];
1007      ge = MIN2(ge, c0);
1008    }
1009  }
1010 }
1011
1012 for (p = i + 2; p < j - VRNA_GQUAD_MIN_BOX_SIZE; p++) {
1013   l1 = p - i - 1;
1014   if (l1 > MAXLOOP)
1015     break;
1016
1017   if (S[p] != 3)
1018     continue;
1019
1020   minq = j - i + p - MAXLOOP - 2;
1021   c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
1022   minq = MAX2(c0, minq);
1023   c0 = j - 2;
1024   maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
1025   maxq = MIN2(c0, maxq);
1026   for (q = minq; q < maxq; q++) {
1027     if (S[q] != 3)
1028       continue;
1029
1030     c0 = en1 + ggg[index[q] + p] + P->internal_loop[l1 + j - q - 1];
1031     ge = MIN2(ge, c0);
1032   }
1033 }
1034
1035 q = j - 2;
1036 if (S[q] == 3)
1037   for (p = i + 4; p < j - VRNA_GQUAD_MIN_BOX_SIZE; p++) {
1038     l1 = p - i - 1;
1039     if (l1 > MAXLOOP)
1040       break;
1041
1042     if (S[p] != 3)
1043       continue;
1044
1045     c0 = en1 + ggg[index[q] + p] + P->internal_loop[l1 + 1];
1046     ge = MIN2(ge, c0);
1047   }
1048 }
```

```

1049     /* second case with 3' dangle (i.e. i+1) onto enclosing pair */
1050 }
1051
1052 #endif
1053     return ge;
1054 }
1055
1056
1057 PRIVATE INLINE
1058 int
1059 E_GQuad_IntLoop_comparative(int          i,
1060                               int          j,
1061                               unsigned int *tt,
1062                               short        *S_cons,
1063                               short        **S5,
1064                               short        **S3,
1065                               unsigned int **a2s,
1066                               int          *ggg,
1067                               int          *index,
1068                               int          n_seq,
1069                               vrna_param_t *P)
1070 {
1071     unsigned int type;
1072     int         eee, energy, ge, p, q, ll, u1, u2, minq, maxq, c0, s;
1073     vrna_md_t  *md;
1074
1075     md      = &(P->model_details);
1076     energy = 0;
1077
1078     for (s = 0; s < n_seq; s++) {
1079         type = tt[s];
1080         if (md->dangles == 2)
1081             energy += P->mismatchI[type][S3[s][i]][S5[s][j]];
1082
1083         if (type > 2)
1084             energy += P->TerminalAU;
1085     }
1086
1087     ge = INF;
1088
1089     p = i + 1;
1090     if (S_cons[p] == 3) {
1091         if (p < j - VRNA_GQUAD_MIN_BOX_SIZE) {
1092             minq = j - i + p - MAXLOOP - 2;
1093             c0   = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
1094             minq = MAX2(c0, minq);
1095             c0   = j - 3;
1096             maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
1097             maxq = MIN2(c0, maxq);
1098             for (q = minq; q < maxq; q++) {
1099                 if (S_cons[q] != 3)
1100                     continue;
1101
1102                 eee = 0;
1103
1104                 for (s = 0; s < n_seq; s++) {
1105                     u1 = a2s[s][j - 1] - a2s[s][q];
1106                     eee += P->internal_loop[u1];
1107                 }
1108
1109                 c0 = energy +
1110                     ggg[index[q] + p] +
1111                     eee;
1112                 ge = MIN2(ge, c0);
1113             }
1114         }
1115     }
1116
1117     for (p = i + 2;
1118          p < j - VRNA_GQUAD_MIN_BOX_SIZE;
1119          p++) {
1120         ll = p - i - 1;
1121         if (ll > MAXLOOP)
1122             break;
1123
1124         if (S_cons[p] != 3)
1125             continue;
1126
1127         minq = j - i + p - MAXLOOP - 2;
1128         c0   = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
1129         minq = MAX2(c0, minq);
1130         c0   = j - 1;
1131         maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
1132         maxq = MIN2(c0, maxq);
1133         for (q = minq; q < maxq; q++) {
1134             if (S_cons[q] != 3)
1135                 continue;

```

```

1136     eee = 0;
1137
1138     for (s = 0; s < n_seq; s++) {
1139         u1 = a2s[s][p - 1] - a2s[s][i];
1140         u2 = a2s[s][j - 1] - a2s[s][q];
1141         eee += P->internal_loop[u1 + u2];
1142     }
1143
1144     c0 = energy +
1145         ggg[index[q] + p] +
1146         eee;
1147     ge = MIN2(ge, c0);
1148 }
1149
1150 }
1151
1152 q = j - 1;
1153 if (S_cons[q] == 3)
1154     for (p = i + 4;
1155         p < j - VRNA_GQUAD_MIN_BOX_SIZE;
1156         p++) {
1157     l1 = p - i - 1;
1158     if (l1 > MAXLOOP)
1159         break;
1160
1161     if (S_cons[p] != 3)
1162         continue;
1163
1164     eee = 0;
1165
1166     for (s = 0; s < n_seq; s++) {
1167         u1 = a2s[s][p - 1] - a2s[s][i];
1168         eee += P->internal_loop[u1];
1169     }
1170
1171     c0 = energy +
1172         ggg[index[q] + p] +
1173         eee;
1174     ge = MIN2(ge, c0);
1175 }
1176
1177 return ge;
1178 }
1179
1180
1181 PRIVATE INLINE
1182 int
1183 E_GQuad_IntLoop_L_comparative(int i,
1184                                 int j,
1185                                 unsigned int *tt,
1186                                 short *S_cons,
1187                                 short **S5,
1188                                 short **S3,
1189                                 unsigned int **a2s,
1190                                 int **ggg,
1191                                 int n_seq,
1192                                 vrna_param_t *P)
1193 {
1194     unsigned int type;
1195     int eee, energy, ge, p, q, l1, ul, u2, minq, maxq, c0, s;
1196     vrna_md_t *md;
1197
1198     md = &(P->model_details);
1199     energy = 0;
1200
1201     for (s = 0; s < n_seq; s++) {
1202         type = tt[s];
1203         if (md->dangles == 2)
1204             energy += P->mismatchI[type][S3[s][i]][S5[s][j]];
1205
1206         if (type > 2)
1207             energy += P->TerminalAU;
1208     }
1209
1210     ge = INF;
1211
1212     p = i + 1;
1213     if (S_cons[p] == 3) {
1214         if (p < j - VRNA_GQUAD_MIN_BOX_SIZE) {
1215             minq = j - i + p - MAXLOOP - 2;
1216             c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
1217             minq = MAX2(c0, minq);
1218             c0 = j - 3;
1219             maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
1220             maxq = MIN2(c0, maxq);
1221             for (q = minq; q < maxq; q++) {
1222                 if (S_cons[q] != 3)

```

```

1223         continue;
1224     eee = 0;
1225
1226     for (s = 0; s < n_seq; s++) {
1227         u1 = a2s[s][j - 1] - a2s[s][q];
1228         eee += P->internal_loop[u1];
1229     }
1230
1231     c0 = energy +
1232         ggg[p][q - p] +
1233         eee;
1234     ge = MIN2(ge, c0);
1235 }
1236 }
1237 }
1238 }
1239
1240 for (p = i + 2;
1241     p < j - VRNA_GQUAD_MIN_BOX_SIZE;
1242     p++) {
1243     l1 = p - i - 1;
1244     if (l1 > MAXLOOP)
1245         break;
1246
1247     if (S_cons[p] != 3)
1248         continue;
1249
1250     minq = j - i + p - MAXLOOP - 2;
1251     c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
1252     minq = MAX2(c0, minq);
1253     c0 = j - 1;
1254     maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
1255     maxq = MIN2(c0, maxq);
1256     for (q = minq; q < maxq; q++) {
1257         if (S_cons[q] != 3)
1258             continue;
1259
1260         eee = 0;
1261
1262         for (s = 0; s < n_seq; s++) {
1263             u1 = a2s[s][p - 1] - a2s[s][i];
1264             u2 = a2s[s][j - 1] - a2s[s][q];
1265             eee += P->internal_loop[u1 + u2];
1266         }
1267
1268         c0 = energy +
1269             ggg[p][q - p] +
1270             eee;
1271         ge = MIN2(ge, c0);
1272     }
1273 }
1274
1275 q = j - 1;
1276 if (S_cons[q] == 3)
1277     for (p = i + 4;
1278         p < j - VRNA_GQUAD_MIN_BOX_SIZE;
1279         p++) {
1280     l1 = p - i - 1;
1281     if (l1 > MAXLOOP)
1282         break;
1283
1284     if (S_cons[p] != 3)
1285         continue;
1286
1287     eee = 0;
1288
1289     for (s = 0; s < n_seq; s++) {
1290         u1 = a2s[s][p - 1] - a2s[s][i];
1291         eee += P->internal_loop[u1];
1292     }
1293
1294     c0 = energy +
1295         ggg[p][q - p] +
1296         eee;
1297     ge = MIN2(ge, c0);
1298 }
1299
1300 return ge;
1301 }
1302
1303
1304 PRIVATE INLINE
1305 int *
1306 E_GQuad_IntLoop_exhaustive(int          i,
1307                             int          j,
1308                             int          **p_p,
1309                             int          **q_p,

```

```

1310             int      type,
1311             short   *S,
1312             int      *ggg,
1313             int      threshold,
1314             int      *index,
1315             vrna_param_t *P)
1316 {
1317     int    energy, *ge, dangles, p, q, ll, minq, maxq, c0;
1318     short si, sj;
1319     int    cnt = 0;
1320
1321     dangles = P->model_details.dangles;
1322     si      = S[i + 1];
1323     sj      = S[j - 1];
1324     energy  = 0;
1325
1326     if (dangles == 2)
1327         energy += P->mismatchI[type][si][sj];
1328
1329     if (type > 2)
1330         energy += P->TerminalAU;
1331
1332     /* guess how many gquads are possible in interval [i+1,j-1] */
1333     *p_p  = (int *)vrna_alloc(sizeof(int) * 256);
1334     *q_p  = (int *)vrna_alloc(sizeof(int) * 256);
1335     ge    = (int *)vrna_alloc(sizeof(int) * 256);
1336
1337     p = i + 1;
1338     if (S[p] == 3) {
1339         if (p < j - VRNA_GQUAD_MIN_BOX_SIZE) {
1340             minq = j - i + p - MAXLOOP - 2;
1341             c0   = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
1342             minq = MAX2(c0, minq);
1343             c0   = j - 3;
1344             maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
1345             maxq = MIN2(c0, maxq);
1346             for (q = minq; q < maxq; q++) {
1347                 if (S[q] != 3)
1348                     continue;
1349
1350                 c0 = energy + ggg[index[q] + p] + P->internal_loop[j - q - 1];
1351                 if (c0 <= threshold) {
1352                     ge[cnt]      = energy + P->internal_loop[j - q - 1];
1353                     (*p_p)[cnt]  = p;
1354                     (*q_p)[cnt++] = q;
1355                 }
1356             }
1357         }
1358     }
1359
1360     for (p = i + 2;
1361          p < j - VRNA_GQUAD_MIN_BOX_SIZE;
1362          p++) {
1363         ll = p - i - 1;
1364         if (ll > MAXLOOP)
1365             break;
1366
1367         if (S[p] != 3)
1368             continue;
1369
1370         minq = j - i + p - MAXLOOP - 2;
1371         c0   = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
1372         minq = MAX2(c0, minq);
1373         c0   = j - 1;
1374         maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
1375         maxq = MIN2(c0, maxq);
1376         for (q = minq; q < maxq; q++) {
1377             if (S[q] != 3)
1378                 continue;
1379
1380             c0 = energy + ggg[index[q] + p] + P->internal_loop[ll + j - q - 1];
1381             if (c0 <= threshold) {
1382                 ge[cnt]      = energy + P->internal_loop[ll + j - q - 1];
1383                 (*p_p)[cnt]  = p;
1384                 (*q_p)[cnt++] = q;
1385             }
1386         }
1387     }
1388
1389     q = j - 1;
1390     if (S[q] == 3)
1391         for (p = i + 4;
1392              p < j - VRNA_GQUAD_MIN_BOX_SIZE;
1393              p++) {
1394             ll = p - i - 1;
1395             if (ll > MAXLOOP)
1396                 break;

```

```

1397     if (S[p] != 3)
1398         continue;
1400
1401     c0 = energy + ggg[index[q] + p] + P->internal_loop[l1];
1402     if (c0 <= threshold) {
1403         ge[cnt] = energy + P->internal_loop[l1];
1404         (*p_p)[cnt] = p;
1405         (*q_p)[cnt++] = q;
1406     }
1407 }
1408
1409 (*p_p)[cnt] = -1;
1410
1411 return ge;
1412 }
1413
1414
1415 PRIVATE INLINE
1416 int
1417 E_GQuad_IntLoop_L(int i,
1418                     int j,
1419                     int type,
1420                     short *S,
1421                     int **ggg,
1422                     int maxdist,
1423                     vrna_param_t *P)
1424 {
1425     int energy, ge, dangles, p, q, l1, minq, maxq, c0;
1426     short si, sj;
1427
1428     dangles = P->model_details.dangles;
1429     si = S[i + 1];
1430     sj = S[j - 1];
1431     energy = 0;
1432
1433     if (dangles == 2)
1434         energy += P->mismatchI[type][si][sj];
1435
1436     if (type > 2)
1437         energy += P->TerminalAU;
1438
1439     ge = INF;
1440
1441     p = i + 1;
1442     if (S[p] == 3) {
1443         if (p < j - VRNA_GQUAD_MIN_BOX_SIZE) {
1444             minq = j - i + p - MAXLOOP - 2;
1445             c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
1446             minq = MAX2(c0, minq);
1447             c0 = j - 3;
1448             maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
1449             maxq = MIN2(c0, maxq);
1450             for (q = minq; q < maxq; q++) {
1451                 if (S[q] != 3)
1452                     continue;
1453
1454                 c0 = energy + ggg[p][q - p] + P->internal_loop[j - q - 1];
1455                 ge = MIN2(ge, c0);
1456             }
1457         }
1458     }
1459
1460     for (p = i + 2;
1461          p < j - VRNA_GQUAD_MIN_BOX_SIZE;
1462          p++) {
1463         l1 = p - i - 1;
1464         if (l1 > MAXLOOP)
1465             break;
1466
1467         if (S[p] != 3)
1468             continue;
1469
1470         minq = j - i + p - MAXLOOP - 2;
1471         c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
1472         minq = MAX2(c0, minq);
1473         c0 = j - 1;
1474         maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
1475         maxq = MIN2(c0, maxq);
1476         for (q = minq; q < maxq; q++) {
1477             if (S[q] != 3)
1478                 continue;
1479
1480             c0 = energy + ggg[p][q - p] + P->internal_loop[l1 + j - q - 1];
1481             ge = MIN2(ge, c0);
1482         }
1483     }

```

```

1484     q = j - 1;
1485     if (S[q] == 3)
1486         for (p = i + 4;
1487              p < j - VRNA_GQUAD_MIN_BOX_SIZE;
1488              p++) {
1489             l1 = p - i - 1;
1490             if (l1 > MAXLOOP)
1491                 break;
1492             if (S[p] != 3)
1493                 continue;
1494             c0 = energy + ggg[p][q - p] + P->internal_loop[l1];
1495             ge = MIN2(ge, c0);
1496         }
1497     return ge;
1498 }
1499
1500
1501
1502
1503
1504
1505 PRIVATE INLINE
1506 FLT_OR_DBL
1507 exp_E_GQuad_IntLoop(int i,
1508                      int j,
1509                      int type,
1510                      short *S,
1511                      FLT_OR_DBL *G,
1512                      FLT_OR_DBL *scale,
1513                      int *index,
1514                      vrna_exp_param_t *pf)
1515 {
1516     int k, l, minl, maxl, u, r;
1517     FLT_OR_DBL q, qe;
1518     double *expintern;
1519     short si, sj;
1520
1521     q = 0;
1522     si = S[i + 1];
1523     sj = S[j - 1];
1524     qe = (FLT_OR_DBL) pf->exp mismatchI[type][si][sj];
1525     expintern = &(pf->expinternal[0]);
1526
1527     if (type > 2)
1528         qe *= (FLT_OR_DBL) pf->expTermAU;
1529
1530     k = i + 1;
1531     if (S[k] == 3) {
1532         if (k < j - VRNA_GQUAD_MIN_BOX_SIZE) {
1533             minl = j - MAXLOOP - 1;
1534             u = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
1535             minl = MAX2(u, minl);
1536             u = j - 3;
1537             maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
1538             maxl = MIN2(u, maxl);
1539             for (l = minl; l < maxl; l++) {
1540                 if (S[l] != 3)
1541                     continue;
1542
1543                 if (G[index[k] - 1] == 0.)
1544                     continue;
1545
1546                 q += qe
1547                     * G[index[k] - 1]
1548                     * (FLT_OR_DBL) expintern[j - l - 1]
1549                     * scale[j - l + 1];
1550             }
1551         }
1552     }
1553
1554     for (k = i + 2;
1555          k <= j - VRNA_GQUAD_MIN_BOX_SIZE;
1556          k++) {
1557         u = k - i - 1;
1558         if (u > MAXLOOP)
1559             break;
1560
1561         if (S[k] != 3)
1562             continue;
1563
1564         minl = j - i + k - MAXLOOP - 2;
1565         r = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
1566         minl = MAX2(r, minl);
1567         maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
1568         r = j - 1;
1569         maxl = MIN2(r, maxl);
1570         for (l = minl; l < maxl; l++) {

```

```

1571     if (S[1] != 3)
1572         continue;
1573
1574     if (G[index[k] - 1] == 0.)
1575         continue;
1576
1577     q += qe
1578         * G[index[k] - 1]
1579         * (FLT_OR_DBL)expintern[u + j - 1 - 1]
1580         * scale[u + j - 1 + 1];
1581     }
1582 }
1583
1584 l = j - 1;
1585 if (S[l] == 3)
1586     for (k = i + 4; k <= j - VRNA_GQUAD_MIN_BOX_SIZE; k++) {
1587         u = k - i - 1;
1588         if (u > MAXLOOP)
1589             break;
1590
1591         if (S[k] != 3)
1592             continue;
1593
1594         if (G[index[k] - 1] == 0.)
1595             continue;
1596
1597         q += qe
1598             * G[index[k] - 1]
1599             * (FLT_OR_DBL)expintern[u]
1600             * scale[u + 2];
1601     }
1602
1603 return q;
1604 }
1605
1606
1607 PRIVATE INLINE
1608 FLT_OR_DBL
1609 exp_E_GQuad_IntLoop_comparative(int
1610                                     int
1611                                     unsigned int
1612                                     short
1613                                     short
1614                                     short
1615                                     unsigned int
1616                                     FLT_OR_DBL
1617                                     FLT_OR_DBL
1618                                     int
1619                                     int
1620                                     vrna_exp_param_t
1621                                     i,
1622                                     j,
1623                                     *tt,
1624                                     *S_cons,
1625                                     **S5,
1626                                     **S3,
1627                                     **a2s,
1628                                     *G,
1629                                     *scale,
1630                                     *index,
1631                                     n_seq,
1632                                     *pf)
1633 {
1634     unsigned int type;
1635     int k, l, minl, maxl, u, ul, u2, r, s;
1636     FLT_OR_DBL q, qe, qqq;
1637     double *expintern;
1638     vrna_md_t *md;
1639
1640     q = 0;
1641     qe = 1.;
1642     md = &(pf->model_details);
1643     expintern = &(pf->expinternal[0]);
1644
1645     for (s = 0; s < n_seq; s++) {
1646         type = tt[s];
1647         if (md->dangles == 2)
1648             qe *= (FLT_OR_DBL)pf->exp mismatchI[type][S3[s][i]][S5[s][j]];
1649
1650         if (type > 2)
1651             qe *= (FLT_OR_DBL)pf->expTermAU;
1652     }
1653
1654     k = i + 1;
1655     if (S_cons[k] == 3) {
1656         if (k < j - VRNA_GQUAD_MIN_BOX_SIZE) {
1657             minl = j - MAXLOOP - 1;
1658             u = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
1659             minl = MAX2(u, minl);
1660             u = j - 3;
1661             maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
1662             maxl = MIN2(u, maxl);
1663             for (l = minl; l < maxl; l++) {
1664                 if (S_cons[l] != 3)
1665                     continue;
1666
1667                 if (G[index[k] - 1] == 0.)
1668                     continue;
1669             }
1670         }
1671     }

```

```

1658     qqq = 1.;
1659
1660     for (s = 0; s < n_seq; s++) {
1661         u1 = a2s[s][j - 1] - a2s[s][1];
1662         qqq *= expintern[u1];
1663     }
1664
1665     q += qe *
1666         G[index[k] - 1] *
1667         qqq *
1668         scale[j - 1 + 1];
1669 }
1670 }
1671 }
1672
1673 for (k = i + 2;
1674     k <= j - VRNA_GQUAD_MIN_BOX_SIZE;
1675     k++) {
1676     u = k - i - 1;
1677     if (u > MAXLOOP)
1678         break;
1679
1680     if (S_cons[k] != 3)
1681         continue;
1682
1683     minl = j - i + k - MAXLOOP - 2;
1684     r = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
1685     minl = MAX2(r, minl);
1686     maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
1687     r = j - 1;
1688     maxl = MIN2(r, maxl);
1689     for (l = minl; l < maxl; l++) {
1690         if (S_cons[l] != 3)
1691             continue;
1692
1693         if (G[index[k] - 1] == 0.)
1694             continue;
1695
1696     qqq = 1.;
1697
1698     for (s = 0; s < n_seq; s++) {
1699         u1 = a2s[s][k - 1] - a2s[s][i];
1700         u2 = a2s[s][j - 1] - a2s[s][1];
1701         qqq *= expintern[u1 + u2];
1702     }
1703
1704     q += qe *
1705         G[index[k] - 1] *
1706         qqq *
1707         scale[u + j - 1 + 1];
1708 }
1709 }
1710
1711 l = j - 1;
1712 if (S_cons[l] == 3)
1713     for (k = i + 4; k <= j - VRNA_GQUAD_MIN_BOX_SIZE; k++) {
1714         u = k - i - 1;
1715         if (u > MAXLOOP)
1716             break;
1717
1718         if (S_cons[k] != 3)
1719             continue;
1720
1721         if (G[index[k] - 1] == 0.)
1722             continue;
1723
1724     qqq = 1.;
1725
1726     for (s = 0; s < n_seq; s++) {
1727         u1 = a2s[s][k - 1] - a2s[s][i];
1728         qqq *= expintern[u1];
1729     }
1730
1731     q += qe *
1732         G[index[k] - 1] *
1733         qqq *
1734         scale[u + 2];
1735 }
1736
1737 return q;
1738 }
1739
1740
1741 #endif

```

## 18.95 ViennaRNA/grammar.h File Reference

Implementations for the RNA folding grammar.

Include dependency graph for grammar.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct `vrna_gr_aux_s`

#### 18.95.1 Detailed Description

Implementations for the RNA folding grammar.

## 18.96 grammar.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_GRAMMAR_H
2 #define VIENNA_RNA_PACKAGE_GRAMMAR_H
3
16 #include <ViennaRNA/fold_compound.h>
17
18 typedef int (vrna_callback_gr_rule) (vrna_fold_compound_t *vc,
19                         int i,
20                         int j,
21                         void *data);
22
23
24 typedef void (vrna_callback_gr_rule_aux) (vrna_fold_compound_t *vc,
25                         int i,
26                         int j,
27                         void *data);
28
29
30 typedef FLT_OR_DBL (vrna_callback_gr_rule_exp) (vrna_fold_compound_t *vc,
31                         int i,
32                         int j,
33                         void *data);
34
35
36 typedef void (vrna_callback_gr_rule_aux_exp) (vrna_fold_compound_t *vc,
37                         int i,
38                         int j,
39                         void *data);
40
41
42 typedef void (vrna_callback_gr_cond) (vrna_fold_compound_t *fc,
43                         unsigned char stage,
44                         void *data);
45
46
47 typedef void (vrna_callback_gr_free_data) (void *data);
48
49
50 typedef struct vrna_gr_aux_s vrna_gr_aux_t;
51
52
53 struct vrna_gr_aux_s {
54     vrna_callback_gr_cond     *cb_proc;
55     vrna_callback_gr_rule     *cb_aux_f;
56     vrna_callback_gr_rule     *cb_aux_c;
57     vrna_callback_gr_rule     *cb_aux_m;
58     vrna_callback_gr_rule     *cb_aux_ml;
59     vrna_callback_gr_rule_aux *cb_aux;
60
61     vrna_callback_gr_rule_exp *cb_aux_exp_f;
62     vrna_callback_gr_rule_exp *cb_aux_exp_c;
63     vrna_callback_gr_rule_exp *cb_aux_exp_m;
64     vrna_callback_gr_rule_exp *cb_aux_exp_ml;
65     vrna_callback_gr_rule_exp *cb_aux_exp;
66
67     void *data;
68     vrna_callback_gr_free_data *free_data;
69
70 };
71
72
73 int
74 vrna_gr_set_aux_f (vrna_fold_compound_t *fc,
75                     vrna_callback_gr_rule *cb);
76

```

```

77
78 int
79 vrna_gr_set_aux_exp_f(vrna_fold_compound_t *fc,
80                         vrna_callback_gr_rule_exp *cb);
81
82
83 int
84 vrna_gr_set_aux_c(vrna_fold_compound_t *fc,
85                     vrna_callback_gr_rule *cb);
86
87
88 int
89 vrna_gr_set_aux_exp_c(vrna_fold_compound_t *fc,
90                         vrna_callback_gr_rule_exp *cb);
91
92
93 int
94 vrna_gr_set_aux_m(vrna_fold_compound_t *fc,
95                     vrna_callback_gr_rule *cb);
96
97
98 int
99 vrna_gr_set_aux_exp_m(vrna_fold_compound_t *fc,
100                        vrna_callback_gr_rule_exp *cb);
101
102
103 int
104 vrna_gr_set_aux_m1(vrna_fold_compound_t *fc,
105                      vrna_callback_gr_rule *cb);
106
107
108 int
109 vrna_gr_set_aux_exp_m1(vrna_fold_compound_t *fc,
110                          vrna_callback_gr_rule_exp *cb);
111
112
113 int
114 vrna_gr_set_aux(vrna_fold_compound_t *fc,
115                     vrna_callback_gr_rule_aux *cb);
116
117
118 int
119 vrna_gr_set_aux_exp(vrna_fold_compound_t *fc,
120                      vrna_callback_gr_rule_aux_exp *cb);
121
122
123 int
124 vrna_gr_set_data(vrna_fold_compound_t *fc,
125                     void *data,
126                     vrna_callback_gr_free_data *free_data);
127
128
129 int
130 vrna_gr_set_cond(vrna_fold_compound_t *fc,
131                     vrna_callback_gr_cond *cb);
132
133
134 int
135 vrna_gr_reset(vrna_fold_compound_t *fc);
136
137
147 #endif

```

## 18.97 ViennaRNA/hairpin\_loops.h File Reference

Use [ViennaRNA/loops/hairpin.h](#) instead.

Include dependency graph for hairpin\_loops.h:

### 18.97.1 Detailed Description

Use [ViennaRNA/loops/hairpin.h](#) instead.

**Deprecated** Use [ViennaRNA/loops/hairpin.h](#) instead

## 18.98 hairpin\_loops.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_LOOPS_HAIRPIN_DEPRECATED_H
```

```

2 #define VIENNA_RNA_PACKAGE_LOOPSS_HAIRPIN_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 #warning "Including deprecated header file <ViennaRNA/hairpin_loops.h>! Use <ViennaRNA/loops/hairpin.h>
           instead!"
13 # endif
14 #include <ViennaRNA/loops/hairpin.h>
15 #endif
16
17 #endif

```

## 18.99 ViennaRNA/heat\_capacity.h File Reference

Compute heat capacity for an RNA.

Include dependency graph for heat\_capacity.h:

### Data Structures

- struct [vrna\\_heat\\_capacity\\_s](#)  
*A single result from heat capacity computations.* [More...](#)

### Typedefs

- typedef void() [vrna\\_heat\\_capacity\\_callback](#)(float temp, float heat\_capacity, void \*data)  
*The callback for heat capacity predictions.*
- typedef struct [vrna\\_heat\\_capacity\\_s](#) [vrna\\_heat\\_capacity\\_t](#)  
*A single result from heat capacity computations.*

### Functions

#### Basic heat capacity function interface

- [vrna\\_heat\\_capacity\\_t](#) \* [vrna\\_heat\\_capacity](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, float T\_min, float T\_max, float T\_increment, unsigned int mpoints)  
*Compute the specific heat for an RNA.*
- int [vrna\\_heat\\_capacity\\_cb](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, float T\_min, float T\_max, float T\_increment, unsigned int mpoints, [vrna\\_heat\\_capacity\\_callback](#) \*cb, void \*data)  
*Compute the specific heat for an RNA (callback variant)*

#### Simplified heat capacity computation

- [vrna\\_heat\\_capacity\\_t](#) \* [vrna\\_heat\\_capacity\\_simple](#) (const char \*sequence, float T\_min, float T\_max, float T\_increment, unsigned int mpoints)  
*Compute the specific heat for an RNA (simplified variant)*

### 18.99.1 Detailed Description

Compute heat capacity for an RNA.

This file includes the interface to all functions related to predicting the heat capacity for an RNA.

## 18.100 heat\_capacity.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_MELTING_H
2 #define VIENNA_RNA_PACKAGE_MELTING_H
3
4 #include <stdio.h>
5
6 #include <ViennaRNA/datastructures/basic.h>
7
38 typedef void (vrna\_heat\_capacity\_callback) (float temp,
39                                     float heat_capacity,

```

```

40                               void    *data);
41
42
48 typedef struct vrna_heat_capacity_s vrna_heat_capacity_t;
49
50
56 struct vrna_heat_capacity_s {
57     float temperature;
58     float heat_capacity;
59 };
60
61
91 vrna_heat_capacity_t *
92 vrna_heat_capacity(vrna_fold_compound_t *fc,
93             float T_min,
94             float T_max,
95             float T_increment,
96             unsigned int mpoints);
97
98
128 int
129 vrna_heat_capacity_cb(vrna_fold_compound_t *fc,
130             float T_min,
131             float T_max,
132             float T_increment,
133             unsigned int mpoints,
134             vrna_heat_capacity_callback *cb,
135             void *data);
136
137
138 /* End basic interface */
172 vrna_heat_capacity_t *
173 vrna_heat_capacity_simple(const char *sequence,
174             float T_min,
175             float T_max,
176             float T_increment,
177             unsigned int mpoints);
178
179 /* End basic interface */
182 /* End thermodynamics */
185 #endif

```

## 18.101 ViennaRNA/interior\_loops.h File Reference

Use [ViennaRNA/loops/internal.h](#) instead.

Include dependency graph for interior\_loops.h:

### 18.101.1 Detailed Description

Use [ViennaRNA/loops/internal.h](#) instead.

**Deprecated** Use [ViennaRNA/loops/internal.h](#) instead

## 18.102 interior\_loops.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_LOOPS_INTERNAL_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_LOOPS_INTERNAL_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 # warning "Including deprecated header file <ViennaRNA/interior_loops.h>! Use <ViennaRNA/loops/internal.h>
instead!"
13 # endif
14 #include <ViennaRNA/loops/internal.h>
15 #endif
16
17 #endif

```

## 18.103 ViennaRNA/inverse.h File Reference

Inverse folding routines.

## Functions

- float `inverse_fold` (char \*start, const char \*target)  
*Find sequences with predefined structure.*
- float `inverse_pf_fold` (char \*start, const char \*target)  
*Find sequence that maximizes probability of a predefined structure.*

## Variables

- char \* `symbolset`  
*This global variable points to the allowed bases, initially "AUGC". It can be used to design sequences from reduced alphabets.*
- float `final_cost`
- int `give_up`
- int `inv_verbose`

### 18.103.1 Detailed Description

Inverse folding routines.

## 18.104 inverse.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_INVERSE_H
2 #define VIENNA_RNA_PACKAGE_INVERSE_H
3
21 extern char *symbolset;
23 extern float final_cost;
25 extern int give_up;
27 extern int inv_verbose;
28
45 float inverse_fold( char *start,
46                      const char *target);
47
61 float inverse_pf_fold(char *start,
62                      const char *target);
63
67 #endif

```

## 18.105 ViennaRNA/landscape/move.h File Reference

Methods to operate with structural neighbors of RNA secondary structures.

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct `vrna_move_s`  
*An atomic representation of the transition / move from one structure to its neighbor.* [More...](#)

## Macros

- #define `VRNA_MOVESET_INSERTION` 4  
*Option flag indicating insertion move.*
- #define `VRNA_MOVESET_DELETION` 8  
*Option flag indicating deletion move.*
- #define `VRNA_MOVESET_SHIFT` 16  
*Option flag indicating shift move.*
- #define `VRNA_MOVESET_NO_LP` 32  
*Option flag indicating moves without lonely base pairs.*
- #define `VRNA_MOVESET_DEFAULT` (`VRNA_MOVESET_INSERTION` | `VRNA_MOVESET_DELETION`)  
*Option flag indicating default move set, i.e. insertions/deletion of a base pair.*

## Typedefs

- `typedef struct vrna_move_s vrna_move_t`  
*A single move that transforms a secondary structure into one of its neighbors.*

## Functions

- `vrna_move_t vrna_move_init (int pos_5, int pos_3)`  
*Create an atomic move.*
- `void vrna_move_list_free (vrna_move_t *moves)`
- `void vrna_move_apply (short *pt, const vrna_move_t *m)`  
*Apply a particular move / transition to a secondary structure, i.e. transform a structure.*
- `int vrna_move_is_removal (const vrna_move_t *m)`  
*Test whether a move is a base pair removal.*
- `int vrna_move_is_insertion (const vrna_move_t *m)`  
*Test whether a move is a base pair insertion.*
- `int vrna_move_is_shift (const vrna_move_t *m)`  
*Test whether a move is a base pair shift.*
- `int vrna_move_compare (const vrna_move_t *a, const vrna_move_t *b, const short *pt)`  
*Compare two moves.*

### 18.105.1 Detailed Description

Methods to operate with structural neighbors of RNA secondary structures.

## 18.106 move.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_MOVE_H
2 #define VIENNA_RNA_PACKAGE_MOVE_H
3
4
20 typedef struct vrna_move_s vrna_move_t;
21
26 #define VRNA_MOVESET_INSERTION 4
27
32 #define VRNA_MOVESET_DELETION 8
33
38 #define VRNA_MOVESET_SHIFT 16
43 #define VRNA_MOVESET_NO_LP 32
44
49 #define VRNA_MOVESET_DEFAULT ((VRNA_MOVESET_INSERTION | VRNA_MOVESET_DELETION))
50
51
73 struct vrna_move_s {
74     int         pos_5;
75     int         pos_3;
76     vrna_move_t *next;
79 };
80
81
91 vrna_move_t
92 vrna_move_init(int pos_5,
93                 int pos_3);
94
95
99 void
100 vrna_move_list_free(vrna_move_t *moves);
101
102
109 void
110 vrna_move_apply(short           *pt,
111                   const vrna_move_t *m);
112
113
114 void
115 vrna_move_apply_db(char            *structure,
116                      const short      *pt,
117                      const vrna_move_t *m);
118

```

```

119
126 int
127 vrna_move_is_removal(const vrna_move_t *m);
128
129
136 int
137 vrna_move_is_insertion(const vrna_move_t *m);
138
139
146 int
147 vrna_move_is_shift(const vrna_move_t *m);
148
149
170 int
171 vrna_move_compare(const vrna_move_t *a,
172                     const vrna_move_t *b,
173                     const short      *pt);
174
175
180 #endif

```

## 18.107 ViennaRNA/landscape/paths.h File Reference

API for computing (optimal) (re-)folding paths between secondary structures.

Include dependency graph for paths.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [vrna\\_path\\_s](#)  
*An element of a refolding path list. [More...](#)*

### Macros

- `#define VRNA_PATH_TYPE_DOT_BRACKET 1U`  
*Flag to indicate producing a (re-)folding path as list of dot-bracket structures.*
- `#define VRNA_PATH_TYPE_MOVES 2U`  
*Flag to indicate producing a (re-)folding path as list of transition moves.*

### Typedefs

- `typedef struct vrna_path_s vrna_path_t`  
*Typename for the refolding path data structure [vrna\\_path\\_s](#).*
- `typedef struct vrna_path_options_s *vrna_path_options_t`  
*Options data structure for (re-)folding path implementations.*
- `typedef struct vrna_path_s path_t`  
*Old typename of [vrna\\_path\\_s](#).*

### Functions

- `void vrna_path_free(vrna_path_t *path)`  
*Release (free) memory occupied by a (re-)folding path.*
- `void vrna_path_options_free(vrna_path_options_t options)`  
*Release (free) memory occupied by an options data structure for (re-)folding path implementations.*
- `vrna_path_options_t vrna_path_options_findpath(int width, unsigned int type)`  
*Create options data structure for findpath direct (re-)folding path heuristic.*
- `vrna_path_t * vrna_path_direct(vrna_fold_compound_t *fc, const char *s1, const char *s2, vrna_path_options_t options)`  
*Determine an optimal direct (re-)folding path between two secondary structures.*
- `vrna_path_t * vrna_path_direct_ub(vrna_fold_compound_t *fc, const char *s1, const char *s2, int maxE, vrna_path_options_t options)`  
*Determine an optimal direct (re-)folding path between two secondary structures.*

### 18.107.1 Detailed Description

API for computing (optimal) (re-)folding paths between secondary structures.

## 18.108 paths.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_PATHS_H
2 #define VIENNA_RNA_PACKAGE_PATHS_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 #  define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
16
17 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
18
19 #define vrna_path_s vrna_path_t;
20
21
22 #define vrna_path_options_s *vrna_path_options_t;
23
24
25 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
26
27 #define vrna_path_t vrna_path_s path_t,
28     "Use vrna_path_t instead!";
29
30 #endif
31
32 #include <ViennaRNA/fold_compound.h>
33 #include <ViennaRNA/landscape/move.h>
34
35
36 #define VRNA_PATH_TYPE_DOT_BRACKET    1U
37
38 #define VRNA_PATH_TYPE_MOVES          2U
39
40 struct vrna_path_s {
41     unsigned int type;
42     double      en;
43     char        *s;
44     vrna_move_t move;
45 };
46
47
48 void
49 vrna_path_free(vrna_path_t *path);
50
51
52 void
53 vrna_path_options_free(vrna_path_options_t options);
54
55
56 vrna_path_options_t
57 vrna_path_options_findpath(int           width,
58                             unsigned int type);
59
60
61 vrna_path_t *
62 vrna_path_direct(vrna_fold_compound_t *fc,
63                  const char        *s1,
64                  const char        *s2,
65                  vrna_path_options_t options);
66
67
68 vrna_path_t *
69 vrna_path_direct_ub(vrna_fold_compound_t *fc,
70                      const char        *s1,
71                      const char        *s2,
72                      int               maxE,
73                      vrna_path_options_t options);
74
75
76 #endif // VRNA_DISABLE_BACKWARD_COMPATIBILITY
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242 #endif
```

## 18.109 ViennaRNA/Lfold.h File Reference

Functions for locally optimal MFE structure prediction.

Include dependency graph for Lfold.h:

### Functions

- float [Lfold](#) (const char \*string, const char \*structure, int maxdist)  
*The local analog to [fold\(\)](#).*
- float [Lfoldz](#) (const char \*string, const char \*structure, int maxdist, int zsc, double min\_z)

### 18.109.1 Detailed Description

Functions for locally optimal MFE structure prediction.

## 18.110 Lfold.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_LFOLD_H
2 #define VIENNA_RNA_PACKAGE_LFOLD_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5
12 #ifdef VRNA_WARN_DEPRECATED
13 # if defined(__clang__)
14 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
15 # elif defined(__GNUC__)
16 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
17 # else
18 #  define DEPRECATED(func, msg) func
19 # endif
20 #else
21 # define DEPRECATED(func, msg) func
22 #endif
23
24 #include <ViennaRNA/mfe_window.h>
25
36 DEPRECATED(float Lfold(const char *string,
37                  const char *structure,
38                  int maxdist),
39                  "Use vrna_Lfold() or vrna_Lfold_cb() instead");
40
41 #ifdef VRNA_WITH_SVM
42 DEPRECATED(float Lfoldz(const char *string,
43                  const char *structure,
44                  int maxdist,
45                  int zsc,
46                  double min_z),
47                  "Use vrna_Lfoldz() or vrna_Lfoldz_cb() instead");
48 #endif
49
52 DEPRECATED(float aliLfold(const char **AS,
53                  const char *structure,
54                  int maxdist),
55                  "Use vrna_aliLfold() or vrna_aliLfold_cb() instead");
56
58 DEPRECATED(float aliLfold\_cb(const char **AS,
59                  int maxdist,
60                  vrna_mfe_window_callback *cb,
61                  void *data),
62                  "Use vrna_aliLfold() or vrna_aliLfold_cb() instead");
63
64 #endif
65 #endif
66
67
68 #endif
69
70
71 #endif
72
73 #endif
74
75
76
77
78
79
80
81 #endif
82
83 #endif

```

## 18.111 ViennaRNA/loop\_energies.h File Reference

Use [ViennaRNA/loops/all.h](#) instead.

Include dependency graph for loop\_energies.h:

### 18.111.1 Detailed Description

Use [ViennaRNA/loops/all.h](#) instead.

**Deprecated** Use [ViennaRNA/loops/all.h](#) instead

## 18.112 loop\_energies.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_LOOPS_ALL_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_LOOPS_ALL_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 # ifdef VRNA_WARN_DEPRECATED
6 # warning "Including deprecated header file <ViennaRNA/loop_energies.h>! Use <ViennaRNA/loops/all.h>
7 instead!"
8 # endif
9 #include <ViennaRNA/loops/all.h>
10#endif
11
12#endif
13
14#endif
15
16
17#endif

```

## 18.113 ViennaRNA/loops/all.h File Reference

Energy evaluation for MFE and partition function calculations.

Include dependency graph for all.h: This graph shows which files directly or indirectly include this file:

### 18.113.1 Detailed Description

Energy evaluation for MFE and partition function calculations.

This file contains functions for the calculation of the free energy  $\Delta G$  of a hairpin- [ [E\\_Hairpin\(\)](#) ] or interior-loop [ [E\\_IntLoop\(\)](#) ].

The unit of the free energy returned is  $10^{-2} * \text{kcal/mol}$

In case of computing the partition function, this file also supplies functions which return the Boltzmann weights  $e^{-\Delta G/kT}$  for a hairpin- [ [exp\\_E\\_Hairpin\(\)](#) ] or interior-loop [ [exp\\_E\\_IntLoop\(\)](#) ].

## 18.114 all.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_LOOPS_ALL_H
2 #define VIENNA_RNA_PACKAGE_LOOPS_ALL_H
3
4 /* below we include the loop type specific energy evaluation functions */
5
6 #include <ViennaRNA/loops/external.h>
7
8 #include <ViennaRNA/loops/hairpin.h>
9
10 #include <ViennaRNA/loops/internal.h>
11
12 #include <ViennaRNA/loops/multibranch.h>
13
14#endif
15
16#endif
17
18#endif
19
20#endif
21
22#endif
23
24#endif
25
26#endif
27
28#endif
29
30#endif
31
32#endif
33
34#endif
35
36#endif
37
38#endif
39
40#endif
41#endif

```

## 18.115 ViennaRNA/loops/external.h File Reference

Energy evaluation of exterior loops for MFE and partition function calculations.

Include dependency graph for external.h: This graph shows which files directly or indirectly include this file:

### Functions

- int [E\\_Stem](#) (int type, int si1, int sj1, int extLoop, [vrna\\_param\\_t](#) \*P)  
*Compute the energy contribution of a stem branching off a loop-region.*

- **FLT\_OR\_DBL exp\_E\_ExtLoop** (int type, int si1, int sj1, **vrna\_exp\_param\_t** \*P)
- **FLT\_OR\_DBL exp\_E\_Stem** (int type, int si1, int sj1, int extLoop, **vrna\_exp\_param\_t** \*P)

### Basic free energy interface

- int **vrna\_E\_ext\_stem** (unsigned int type, int n5d, int n3d, **vrna\_param\_t** \*p)  
*Evaluate a stem branching off the exterior loop.*
- int **vrna\_eval\_ext\_stem** (**vrna\_fold\_compound\_t** \*fc, int i, int j)  
*Evaluate the free energy of a base pair in the exterior loop.*
- int **vrna\_E\_ext\_loop\_5** (**vrna\_fold\_compound\_t** \*fc)
- int **vrna\_E\_ext\_loop\_3** (**vrna\_fold\_compound\_t** \*fc, int i)

### Boltzmann weight (partition function) interface

- **typedef struct vrna\_mx\_pf\_aux\_el\_s \* vrna\_mx\_pf\_aux\_el\_t**  
*Auxiliary helper arrays for fast exterior loop computations.*
- **FLT\_OR\_DBL vrna\_exp\_E\_ext\_stem** (unsigned int type, int n5d, int n3d, **vrna\_exp\_param\_t** \*p)  
*Evaluate a stem branching off the exterior loop (Boltzmann factor version)*
- **vrna\_mx\_pf\_aux\_el\_t vrna\_exp\_E\_ext\_fast\_init** (**vrna\_fold\_compound\_t** \*fc)
- void **vrna\_exp\_E\_ext\_fast\_rotate** (**vrna\_mx\_pf\_aux\_el\_t** aux\_mx)
- void **vrna\_exp\_E\_ext\_fast\_free** (**vrna\_mx\_pf\_aux\_el\_t** aux\_mx)
- **FLT\_OR\_DBL vrna\_exp\_E\_ext\_fast** (**vrna\_fold\_compound\_t** \*fc, int i, int j, **vrna\_mx\_pf\_aux\_el\_t** aux\_mx)
- void **vrna\_exp\_E\_ext\_fast\_update** (**vrna\_fold\_compound\_t** \*fc, int j, **vrna\_mx\_pf\_aux\_el\_t** aux\_mx)

#### 18.115.1 Detailed Description

Energy evaluation of exterior loops for MFE and partition function calculations.

, ,

## 18.116 external.h

Go to the documentation of this file.

```

1 #ifndef VIENNA_RNA_PACKAGE_LOOPS_EXTERNAL_H
2 #define VIENNA_RNA_PACKAGE_LOOPS_EXTERNAL_H
3
4 #include <ViennaRNA/datastructures/basic.h>
5 #include <ViennaRNA/fold_compound.h>
6 #include <ViennaRNA/params/basic.h>
7
8 #ifdef VRNA_WARN_DEPRECATED
9 # if defined(DEPRECATED)
10 #   undef DEPRECATED
11 # endif
12 # if defined(__clang__)
13 #   define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
14 # elif defined(__GNUC__)
15 #   define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
16 # else
17 #   define DEPRECATED(func, msg) func
18 # endif
19 #else
20 # define DEPRECATED(func, msg) func
21 #endif
22
23 int
24 vrna_E_ext_stem(unsigned int type,
25                 int          n5d,
26                 int          n3d,
27                 vrna_param_t *p);
28
29
30 int
31 vrna_eval_ext_stem(vrna_fold_compound_t *fc,
32                     int          i,
33                     int          j);
34
35
36 int
37 vrna_E_ext_loop_5(vrna_fold_compound_t *fc);
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93

```

```

94
95 int
96 vrna_E_ext_loop_3(vrna_fold_compound_t *fc,
97                     int i);
98
99
100 /* End basic interface */
116 typedef struct vrna_mx_pf_aux_el_s *vrna_mx_pf_aux_el_t;
117
118
136 FLT_OR_DBL
137 vrna_exp_E_ext_stem(unsigned int type,
138                      int n5d,
139                      int n3d,
140                      vrna_exp_param_t *p);
141
142
143 vrna_mx_pf_aux_el_t
144 vrna_exp_E_ext_fast_init(vrna_fold_compound_t *fc);
145
146
147 void
148 vrna_exp_E_ext_fast_rotate(vrna_mx_pf_aux_el_t aux_mx);
149
150
151 void
152 vrna_exp_E_ext_fast_free(vrna_mx_pf_aux_el_t aux_mx);
153
154
155 FLT_OR_DBL
156 vrna_exp_E_ext_fast(vrna_fold_compound_t *fc,
157                      int i,
158                      int j,
159                      vrna_mx_pf_aux_el_t aux_mx);
160
161
162 void
163 vrna_exp_E_ext_fast_update(vrna_fold_compound_t *fc,
164                             int j,
165                             vrna_mx_pf_aux_el_t aux_mx);
166
167
168 /* End partition function interface */
182 int
183 vrna_BT_ext_loop_f5(vrna_fold_compound_t *fc,
184                      int *k,
185                      int *i,
186                      int *j,
187                      vrna_bp_stack_t *bp_stack,
188                      int *stack_count);
189
190
191 int
192 vrna_BT_ext_loop_f3(vrna_fold_compound_t *fc,
193                      int *k,
194                      int maxdist,
195                      int *i,
196                      int *j,
197                      vrna_bp_stack_t *bp_stack,
198                      int *stack_count);
199
200
201 int
202 vrna_BT_ext_loop_f3_pp(vrna_fold_compound_t *fc,
203                          int *i,
204                          int maxdist);
205
206
211 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
212
268 DEPRECATED(int E_Stem(int type,
269                      int sil,
270                      int sjl,
271                      int extLoop,
272                      vrna_param_t *P),
273             "This function is obsolete. Use vrna_E_ext_stem() or E_MLstem() instead");
274
275
276 DEPRECATED(int E_ExtLoop(int type,
277                           int sil,
278                           int sjl,
279                           vrna_param_t *P),
280             "Use vrna_E_ext_stem() instead");
281
282
290 DEPRECATED(FLT_OR_DBL exp_E_ExtLoop(int type,
291                                         int sil,

```

```

292                     int          sj1,
293                     vrna_exp_param_t *P),
294             "Use vrna_exp_E_ext_stem() instead");
295
296
305 DEPRECATED(FLT_OR_DBL exp_E_Stack(int           type,
306                                     int           sil,
307                                     int           sj1,
308                                     int           extLoop,
309                                     vrna_exp_param_t *P),
310             "This function is obsolete");
311
312 #endif
313 #endif
314
320 #endif

```

## 18.117 ViennaRNA/loops/hairpin.h File Reference

Energy evaluation of hairpin loops for MFE and partition function calculations.

Include dependency graph for hairpin.h: This graph shows which files directly or indirectly include this file:

### Functions

- int [vrna\\_BT\\_hp\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j, int en, [vrna\\_bp\\_stack\\_t](#) \*bp\_stack, int \*stack\_← count)  
*Backtrack a hairpin loop closed by (i, j).*

#### Basic free energy interface

- int [vrna\\_E\\_hp\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j)  
*Evaluate the free energy of a hairpin loop and consider hard constraints if they apply.*
- int [vrna\\_E\\_ext\\_hp\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j)  
*Evaluate the free energy of an exterior hairpin loop and consider possible hard constraints.*
- int [vrna\\_eval\\_ext\\_hp\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j)  
*Evaluate free energy of an exterior hairpin loop.*
- int [vrna\\_eval\\_hp\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j)  
*Evaluate free energy of a hairpin loop.*
- PRIVATE int [E\\_Hairpin](#) (int size, int type, int si1, int sj1, const char \*string, [vrna\\_param\\_t](#) \*P)  
*Compute the Energy of a hairpin-loop.*

#### Boltzmann weight (partition function) interface

- PRIVATE [FLT\\_OR\\_DBL exp\\_E\\_Hairpin](#) (int u, int type, short si1, short sj1, const char \*string, [vrna\\_exp\\_param\\_t](#) \*P)  
*Compute Boltzmann weight  $e^{-\Delta G/kT}$  of a hairpin loop.*
- [FLT\\_OR\\_DBL vrna\\_exp\\_E\\_hp\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j)  
*High-Level function for hairpin loop energy evaluation (partition function variant)*

### 18.117.1 Detailed Description

Energy evaluation of hairpin loops for MFE and partition function calculations.

, ,

## 18.118 hairpin.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_LOOPS_HAIRPIN_H
2 #define VIENNA_RNA_PACKAGE_LOOPS_HAIRPIN_H
3
4 #include <math.h>
5 #include <string.h>
6 #include <ViennaRNA/utils/basic.h>
7 #include <ViennaRNA/datastructures/basic.h>
8 #include <ViennaRNA/fold_compound.h>

```

```
9 #include <ViennaRNA/params/basic.h>
10
11 #ifndef VRNA_WARN_DEPRECATED
12 # if defined(DEPRECATED)
13 #   undef DEPRECATED
14 # endif
15 # if defined(__clang__)
16 #   define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
17 # elif defined(__GNUC__)
18 #   define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
19 # else
20 #   define DEPRECATED(func, msg) func
21 # endif
22 #else
23 # define DEPRECATED(func, msg) func
24 #endif
25
26 #ifdef __GNUC__
27 # define INLINE inline
28 #else
29 # define INLINE
30 #endif
31
32 int
33 vrna_E_hp_loop(vrna_fold_compound_t *fc,
34                  int           i,
35                  int           j);
36
37
38 int
39 vrna_E_ext_hp_loop(vrna_fold_compound_t *fc,
40                      int           i,
41                      int           j);
42
43
44 int
45 vrna_eval_ext_hp_loop(vrna_fold_compound_t *fc,
46                       int           i,
47                       int           j);
48
49
50 int
51 vrna_eval_hp_loop(vrna_fold_compound_t *fc,
52                    int           i,
53                    int           j);
54
55
56 PRIVATE INLINE int
57 E_Hairpin(int      size,
58            int      type,
59            int      sil,
60            int      sjl,
61            const char *string,
62            vrna_param_t *P)
63 {
64     int energy;
65
66     if (size <= 30)
67         energy = P->hairpin[size];
68     else
69         energy = P->hairpin[30] + (int)(P->lxc * log((size) / 30.));
70
71     if (size < 3)
72         return energy; /* should only be the case when folding alignments */
73
74     if ((string) && (P->model_details.special_hp)) {
75         if (size == 4) {
76             /* check for tetraloop bonus */
77             char tl[7] = {
78                 0
79             }, *ts;
80             memcpy(tl, string, sizeof(char) * 6);
81             tl[6] = '\0';
82             if ((ts = strstr(P->Tetraloops, tl)))
83                 return P->Tetraloop_E[(ts - P->Tetraloops) / 7];
84         } else if (size == 6) {
85             char tl[9] = {
86                 0
87             }, *ts;
88             memcpy(tl, string, sizeof(char) * 8);
89             tl[8] = '\0';
90             if ((ts = strstr(P->Hexaloops, tl)))
91                 return P->Hexaloop_E[(ts - P->Hexaloops) / 9];
92         } else if (size == 3) {
93             char tl[6] = {
94                 0
95             }, *ts;
```

```

188     memcpy(tl, string, sizeof(char) * 5);
189     tl[5] = '\0';
190     if ((ts = strstr(P->Triloops, tl)))
191         return P->Triloop_E[(ts - P->Triloops) / 6];
192
193     return energy + (type > 2 ? P->TerminalAU : 0);
194 }
195 }
196
197 energy += P->mismatchH[type][sil][sj1];
198
199 return energy;
200 }
201
202
203 /* End basic interface */
231 PRIVATE INLINE FLT_OR_DBL
232 exp_E_Hairpin(int
233                 int          u,
234                 short        type,
235                 short        sil,
236                 const char   *string,
237                 vrna_exp_param_t *P)
238 {
239     double q, kT;
240
241     kT = P->kT; /* kT in cal/mol */
242
243     if (u <= 30)
244         q = P->exphairpin[u];
245     else
246         q = P->exphairpin[30] * exp(-(P->lxc * log(u / 30.)) * 10. / kT);
247
248     if (u < 3)
249         return (FLT_OR_DBL)q; /* should only be the case when folding alignments */
250
251     if ((string) && (P->model_details.special_hp)) {
252         if (u == 4) {
253             char tl[7] = {
254                 0
255             }, *ts;
256             memcpy(tl, string, sizeof(char) * 6);
257             tl[6] = '\0';
258             if ((ts = strstr(P->Tetraloops, tl))) {
259                 if (type != 7)
260                     return (FLT_OR_DBL)(P->exptetra[(ts - P->Tetraloops) / 7]);
261                 else
262                     q *= P->exptetra[(ts - P->Tetraloops) / 7];
263             }
264         } else if (u == 6) {
265             char tl[9] = {
266                 0
267             }, *ts;
268             memcpy(tl, string, sizeof(char) * 8);
269             tl[8] = '\0';
270             if ((ts = strstr(P->Hexaloops, tl)))
271                 return (FLT_OR_DBL)(P->exphex[(ts - P->Hexaloops) / 9]);
272         } else if (u == 3) {
273             char tl[6] = {
274                 0
275             }, *ts;
276             memcpy(tl, string, sizeof(char) * 5);
277             tl[5] = '\0';
278             if ((ts = strstr(P->Triloops, tl)))
279                 return (FLT_OR_DBL)(P->exptri[(ts - P->Triloops) / 6]);
280
281             if (type > 2)
282                 return (FLT_OR_DBL)(q * P->expTermAU);
283             else
284                 return (FLT_OR_DBL)q;
285         }
286     }
287
288     q *= P->expmismatchH[type][sil][sj1];
289
290     return (FLT_OR_DBL)q;
291 }
292
293
303 FLT_OR_DBL
304 vrna_exp_E_hp_loop(vrna_fold_compound_t *fc,
305                      int          i,
306                      int          j);
307
308
309 /* End partition function interface */
330 int

```

```

331 vrna_BT_hp_loop(vrna_fold_compound_t *fc,
332     int i,
333     int j,
334     int en,
335     vrna_bp_stack_t *bp_stack,
336     int *stack_count);
337
338
343 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
344
354 #endif
355
356 #endif

```

## 18.119 ViennaRNA/loops/internal.h File Reference

Energy evaluation of interior loops for MFE and partition function calculations.

Include dependency graph for internal.h: This graph shows which files directly or indirectly include this file:

### Functions

- int **vrna\_BT\_stack** (**vrna\_fold\_compound\_t** \*fc, int \*i, int \*j, int \*en, **vrna\_bp\_stack\_t** \*bp\_stack, int \*stack\_count)  
*Backtrack a stacked pair closed by (i, j).*
- int **vrna\_BT\_int\_loop** (**vrna\_fold\_compound\_t** \*fc, int \*i, int \*j, int en, **vrna\_bp\_stack\_t** \*bp\_stack, int \*stack\_count)  
*Backtrack an interior loop closed by (i, j).*
- PRIVATE int **E\_IntLoop** (int n1, int n2, int type, int type\_2, int si1, int sj1, int sp1, int sq1, **vrna\_param\_t** \*P)
- PRIVATE **FLT\_OR\_DBL exp\_E\_IntLoop** (int u1, int u2, int type, int type2, short si1, short sj1, short sp1, short sq1, **vrna\_exp\_param\_t** \*P)

### Basic free energy interface

- int **vrna\_E\_int\_loop** (**vrna\_fold\_compound\_t** \*fc, int i, int j)
- int **vrna\_eval\_int\_loop** (**vrna\_fold\_compound\_t** \*fc, int i, int j, int k, int l)  
*Evaluate the free energy contribution of an interior loop with delimiting base pairs (i, j) and (k, l).*
- int **vrna\_E\_ext\_int\_loop** (**vrna\_fold\_compound\_t** \*fc, int i, int j, int \*ip, int \*iq)
- int **vrna\_E\_stack** (**vrna\_fold\_compound\_t** \*fc, int i, int j)

### Boltzmann weight (partition function) interface

- **FLT\_OR\_DBL vrna\_exp\_E\_int\_loop** (**vrna\_fold\_compound\_t** \*fc, int i, int j)
- **FLT\_OR\_DBL vrna\_exp\_E\_interior\_loop** (**vrna\_fold\_compound\_t** \*fc, int i, int j, int k, int l)

### 18.119.1 Detailed Description

Energy evaluation of interior loops for MFE and partition function calculations.

, ,

## 18.120 internal.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_LOOPS_INTERNAL_H
2 #define VIENNA_RNA_PACKAGE_LOOPS_INTERNAL_H
3
4 #include <math.h>
5
6 #include <ViennaRNA/utils/basic.h>
7 #include <ViennaRNA/params/default.h>
8 #include <ViennaRNA/datastructures/basic.h>
9 #include <ViennaRNA/fold_compound.h>
10 #include <ViennaRNA/params/basic.h>
11 #include <ViennaRNA/constraints/hard.h>
12 #include <ViennaRNA/constraints/soft.h>
13
14 #ifdef VRNA_WARN_DEPRECATED

```

```

15 # if defined(DEPRECATED)
16 #   undef DEPRECATED
17 # endif
18 # if defined(__clang__)
19 #   define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
20 # elif defined(__GNUC__)
21 #   define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
22 # else
23 #   define DEPRECATED(func, msg) func
24 # endif
25 #else
26 #   define DEPRECATED(func, msg) func
27 #endif
28
29 #ifdef __GNUC__
30 # define INLINE inline
31 #else
32 # define INLINE
33 #endif
34
35 int
36 vrna_E_int_loop(vrna_fold_compound_t *fc,
37                  int i,
38                  int j);
39
40 int
41 vrna_eval_int_loop(vrna_fold_compound_t *fc,
42                     int i,
43                     int j,
44                     int k,
45                     int l);
46
47 int
48 vrna_E_ext_int_loop(vrna_fold_compound_t *fc,
49                      int i,
50                      int j,
51                      int *ip,
52                      int *iq);
53
54
55 int
56 vrna_E_stack(vrna_fold_compound_t *fc,
57               int i,
58               int j);
59
60
61 /* End basic interface */
62 /* j < i indicates circular folding, i.e. collect contributions for exterior int loops */
63 #define FLT_OR_DBL
64 vrna_exp_E_int_loop(vrna_fold_compound_t *fc,
65                      int i,
66                      int j);
67
68
69 #define FLT_OR_DBL
70 vrna_exp_E_interior_loop(vrna_fold_compound_t *fc,
71                           int i,
72                           int j,
73                           int k,
74                           int l);
75
76
77 /* End partition function interface */
78 int
79 vrna_BT_stack(vrna_fold_compound_t *fc,
80                int *i,
81                int *j,
82                int *en,
83                vrna_bp_stack_t *bp_stack,
84                int *stack_count);
85
86
87 int
88 vrna_BT_int_loop(vrna_fold_compound_t *fc,
89                   int *i,
90                   int *j,
91                   int en,
92                   vrna_bp_stack_t *bp_stack,
93                   int *stack_count);
94
95
96 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
97 #if defined(ON_SAME_STRAND)
98 #undef ON_SAME_STRAND
99 #endif

```

```

167
168 #define ON_SAME_STRAND(I, J, C) ((I) >= (C)) || ((J) < (C))
169
214 PRIVATE INLINE int E_IntLoop(int n1,
215                               int n2,
216                               int type,
217                               int type_2,
218                               int sil,
219                               int sj1,
220                               int spl,
221                               int sq1,
222                               vrna_param_t *P);
223
224
244 PRIVATE INLINE FLT_OR_DBL exp_E_IntLoop(int u1,
245                                            int u2,
246                                            int type,
247                                            int type2,
248                                            short sil,
249                                            short sj1,
250                                            short spl,
251                                            short sq1,
252                                            vrna_exp_param_t *P);
253
254
255 PRIVATE INLINE int E_IntLoop_Co(int type,
256                                  int type_2,
257                                  int i,
258                                  int j,
259                                  int p,
260                                  int q,
261                                  int cutpoint,
262                                  short sil,
263                                  short sj1,
264                                  short spl,
265                                  short sq1,
266                                  int dangles,
267                                  vrna_param_t *P);
268
269
270 /*
271  * ugly but fast interior loop evaluation
272  *
273  * Avoid including this function in your own code. It only serves
274  * as a fast inline block internally re-used throughout the RNAlib. It
275  * evaluates the free energy of interior loops in single sequences or sequence
276  * hybrids. Soft constraints are also applied if available.
277  *
278  * NOTE: do not include into doxygen reference manual!
279 */
280 PRIVATE INLINE int
281 ubf_eval_int_loop(int i,
282                    int j,
283                    int p,
284                    int q,
285                    int il,
286                    int jl,
287                    int pl,
288                    int ql,
289                    short si,
290                    short sj,
291                    short sp,
292                    short sq,
293                    unsigned char type,
294                    unsigned char type_2,
295                    int *rtype,
296                    int ij,
297                    int cp,
298                    vrna_param_t *P,
299                    vrna_sc_t *sc)
300 {
301     int energy, u1, u2;
302
303     u1 = pl - i;
304     u2 = jl - q;
305
306     if ((cp < 0) || (ON_SAME_STRAND(i, p, cp) && ON_SAME_STRAND(q, j, cp))) {
307         /* regular interior loop */
308         energy = E_IntLoop(u1, u2, type, type_2, si, sj, sp, sq, P);
309     } else {
310         /* interior loop like cofold structure */
311         short Si, Sj;
312         Si = ON_SAME_STRAND(i, il, cp) ? si : -1;
313         Sj = ON_SAME_STRAND(jl, j, cp) ? sj : -1;
314         energy = E_IntLoop_Co(rtype[type], rtype[type_2],
315                               i, j, p, q,
316                               cp,

```

```

317             Si, Sj,
318             sp, sq,
319             P->model_details.dangles,
320             P);
321     }
322
323 /* add soft constraints */
324 if (sc) {
325     if (sc->energy_up)
326         energy += sc->energy_up[i1][u1]
327             + sc->energy_up[q1][u2];
328
329     if (sc->energy_bp)
330         energy += sc->energy_bp[ij];
331
332     if (sc->energy_stack)
333         if (u1 + u2 == 0) {
334             int a = sc->energy_stack[i]
335                 + sc->energy_stack[p]
336                 + sc->energy_stack[q]
337                 + sc->energy_stack[j];
338             energy += a;
339         }
340
341     if (sc->f)
342         energy += sc->f(i, j, p, q, VRNA_DECOMP_PAIR_IL, sc->data);
343 }
344
345 return energy;
346 }
347
348
349 PRIVATE INLINE int
350 ubf_eval_int_loop2(int i,
351                     int j,
352                     int p,
353                     int q,
354                     int il,
355                     int j1,
356                     int p1,
357                     int q1,
358                     short si,
359                     short sj,
360                     short sp,
361                     short sq,
362                     unsigned char type,
363                     unsigned char type_2,
364                     int *rtype,
365                     int ij,
366                     unsigned int *sn,
367                     unsigned int *ss,
368                     vrna_param_t *p,
369                     vrna_sc_t *sc)
370 {
371     int energy, u1, u2;
372
373     u1 = p1 - i;
374     u2 = j1 - q;
375
376     if ((sn[i] == sn[p]) && (sn[q] == sn[j])) {
377         /* regular interior loop */
378         energy = E_IntLoop(u1, u2, type, type_2, si, sj, sp, sq, P);
379     } else {
380         /* interior loop like cofold structure */
381         short Si, Sj;
382         Si = (sn[i1] == sn[i]) ? si : -1;
383         Sj = (sn[j] == sn[j1]) ? sj : -1;
384         energy = E_IntLoop_Co(rtype[type], rtype[type_2],
385                               i, j, p, q,
386                               ss[1],
387                               Si, Sj,
388                               sp, sq,
389                               P->model_details.dangles,
390                               P);
391     }
392
393 /* add soft constraints */
394 if (sc) {
395     if (sc->energy_up)
396         energy += sc->energy_up[i1][u1]
397             + sc->energy_up[q1][u2];
398
399     if (sc->energy_bp)
400         energy += sc->energy_bp[ij];
401
402     if (sc->energy_stack)
403         if (u1 + u2 == 0) {

```

```

404     int a = sc->energy_stack[i]
405     + sc->energy_stack[p]
406     + sc->energy_stack[q]
407     + sc->energy_stack[j];
408     energy += a;
409 }
410
411 if (sc->f)
412     energy += sc->f(i, j, p, q, VRNA_DECOMP_PAIR_IL, sc->data);
413 }
414
415 return energy;
416 }
417
418
419 /*
420 * ugly but fast exterior interior loop evaluation
421 *
422 * Avoid including this function in your own code. It only serves
423 * as a fast inline block internally re-used throughout the RNALib. It
424 * evaluates the free energy of interior loops in single sequences or sequence
425 * hybrids. Soft constraints are also applied if available.
426 *
427 * NOTE: do not include into doxygen reference manual!
428 */
429 PRIVATE INLINE int
430 ubf_eval_ext_int_loop(int          i,
431                      int          j,
432                      int          p,
433                      int          q,
434                      int          il,
435                      int          jl,
436                      int          pl,
437                      int          ql,
438                      short        si,
439                      short        sj,
440                      short        sp,
441                      short        sq,
442                      unsigned char type,
443                      unsigned char type_2,
444                      int          length,
445                      vrna_param_t *P,
446                      vrna_sc_t    *sc)
447 {
448     int energy, u1, u2, u3;
449
450     u1 = il;
451     u2 = pl - j;
452     u3 = length - q;
453
454     energy = E_IntLoop(u2, u1 + u3, type, type_2, si, sj, sp, sq, P);
455
456     /* add soft constraints */
457     if (sc) {
458         if (sc->energy_up) {
459             energy += sc->energy_up[jl][u2]
460                     + ((u3 > 0) ? sc->energy_up[ql][u3] : 0)
461                     + ((u1 > 0) ? sc->energy_up[il][u1] : 0);
462         }
463
464         if (sc->energy_stack)
465             if (u1 + u2 + u3 == 0)
466                 energy += sc->energy_stack[i]
467                         + sc->energy_stack[p]
468                         + sc->energy_stack[q]
469                         + sc->energy_stack[j];
470
471         if (sc->f)
472             energy += sc->f(i, j, p, q, VRNA_DECOMP_PAIR_IL, sc->data);
473     }
474
475     return energy;
476 }
477
478
479 PRIVATE INLINE int
480 E_IntLoop(int          n1,
481            int          n2,
482            int          type,
483            int          type_2,
484            int          sii,
485            int          sjl,
486            int          spl,
487            int          sql,
488            vrna_param_t *P)
489 {
490     /* compute energy of degree 2 loop (stack bulge or interior) */

```

```

491     int nl, ns, u, energy;
492
493     if (nl > n2) {
494         nl = nl;
495         ns = n2;
496     } else {
497         nl = n2;
498         ns = nl;
499     }
500
501     if (nl == 0)
502         return P->stack[type][type_2]; /* stack */
503
504     if (ns == 0) {
505         /* bulge */
506         energy = (nl <= MAXLOOP) ? P->bulge[nl] :
507             (P->bulge[30] + (int)(P->lxc * log(nl / 30.)));
508         if (nl == 1) {
509             energy += P->stack[type][type_2];
510         } else {
511             if (type > 2)
512                 energy += P->TerminalAU;
513
514             if (type_2 > 2)
515                 energy += P->TerminalAU;
516         }
517
518         return energy;
519     } else {
520         /* interior loop */
521         if (ns == 1) {
522             if (nl == 1) /* 1x1 loop */
523                 return P->int11[type][type_2][sil][sj1];
524
525             if (nl == 2) {
526                 /* 2x1 loop */
527                 if (nl == 1)
528                     energy = P->int21[type][type_2][sil][sq1][sj1];
529                 else
530                     energy = P->int21[type_2][type][sq1][sil][sp1];
531
532             return energy;
533         } else {
534             /* 1xn loop */
535             energy =
536             (nl + 1 <=
537                 MAXLOOP) ? (P->internal_loop[nl + 1]) : (P->internal_loop[30] +
538                     (int)(P->lxc * log((nl + 1) / 30.)));
539             energy += MIN2(MAX_NINIO, (nl - ns) * P->ninio[2]);
540             energy += P->mismatch1nI[type][sil][sj1] + P->mismatch1nI[type_2][sq1][sp1];
541             return energy;
542         }
543     } else if (ns == 2) {
544         if (nl == 2) {
545             /* 2x2 loop */
546             return P->int22[type][type_2][sil][sp1][sq1][sj1];
547         } else if (nl == 3) {
548             /* 2x3 loop */
549             energy = P->internal_loop[5] + P->ninio[2];
550             energy += P->mismatch23I[type][sil][sj1] + P->mismatch23I[type_2][sq1][sp1];
551             return energy;
552         }
553     }
554
555     /* generic interior loop (no else here!) */
556     u = nl + ns;
557     energy =
558     (u <=
559         MAXLOOP) ? (P->internal_loop[u]) : (P->internal_loop[30] + (int)(P->lxc * log((u) / 30.)));
560
561     energy += MIN2(MAX_NINIO, (nl - ns) * P->ninio[2]);
562
563     energy += P->mismatchI[type][sil][sj1] + P->mismatchI[type_2][sq1][sp1];
564
565 }
566 }
567
568 return energy;
569 }
570
571
572 PRIVATE INLINE FLT_OR_DBL
573 exp_E_IntLoop(int          u1,
574                 int          u2,
575                 int          type,
576                 int          type2,
577                 short        sil,

```

```

578         short      sj1,
579         short      spl,
580         short      sq1,
581         vrna_exp_param_t *P)
582 {
583     int      ul, us, no_close = 0;
584     double   z           = 0.;
585     int      noGUclosure = P->model_details.noGUclosure;
586
587     if ((noGUclosure) && ((type2 == 3) || (type2 == 4) || (type == 3) || (type == 4)))
588     no_close = 1;
589
590     if (ul > u2) {
591         ul = u1;
592         us = u2;
593     } else {
594         ul = u2;
595         us = u1;
596     }
597
598     if (ul == 0) {
599         /* stack */
600         z = P->expstack[type][type2];
601     } else if (!no_close) {
602         if (us == 0) {
603             /* bulge */
604             z = P->expbulge[ul];
605             if (ul == 1) {
606                 z *= P->expstack[type][type2];
607             } else {
608                 if (type > 2)
609                     z *= P->expTermAU;
610
611                 if (type2 > 2)
612                     z *= P->expTermAU;
613             }
614
615             return (FLT_OR_DBL)z;
616         } else if (us == 1) {
617             if (ul == 1)          /* 1x1 loop */
618                 return (FLT_OR_DBL)(P->expint11[type][type2][sil][sj1]);
619
620             if (ul == 2) {
621                 /* 2x1 loop */
622                 if (ul == 1)
623                     return (FLT_OR_DBL)(P->expint21[type][type2][sil][sq1][sj1]);
624                 else
625                     return (FLT_OR_DBL)(P->expint21[type2][type][sq1][sil][sp1]);
626             } else {
627                 /* 1x1 loop */
628                 z = P->expinternal[ul + us] * P->expmismatch1nI[type][sil][sj1] *
629                     P->expmismatch1nI[type2][sq1][sp1];
630                 return (FLT_OR_DBL)(z * P->expninio[2][ul - us]);
631             }
632         } else if (us == 2) {
633             if (ul == 2) {
634                 /* 2x2 loop */
635                 return (FLT_OR_DBL)(P->expint22[type][type2][sil][sp1][sq1][sj1]);
636             } else if (ul == 3) {
637                 /* 2x3 loop */
638                 z = P->expinternal[5] * P->expmismatch23I[type][sil][sj1] *
639                     P->expmismatch23I[type2][sq1][sp1];
640                 return (FLT_OR_DBL)(z * P->expninio[2][1]);
641             }
642         }
643
644         /* generic interior loop (no else here!)*/
645         z = P->expinternal[ul + us] * P->expmismatchI[type][sil][sj1] *
646             P->expmismatchI[type2][sq1][sp1];
647         return (FLT_OR_DBL)(z * P->expninio[2][ul - us]);
648     }
649
650     return (FLT_OR_DBL)z;
651 }
652
653
654 PRIVATE INLINE int
655 E_IntLoop_Co(int      type,
656               int      type_2,
657               int      i,
658               int      j,
659               int      p,
660               int      q,
661               int      cutpoint,
662               short   sil,
663               short   sj1,
664               short   spl,

```

```

665         short      sql,
666         int       dangles,
667         vrna_param_t *P)
668 {
669     int e, energy, ci, cj, cp, cq, d3, d5, d5_2, d3_2, tmm, tmm_2;
670
671     energy = 0;
672     if (type > 2)
673         energy += P->TerminalAU;
674
675     if (type_2 > 2)
676         energy += P->TerminalAU;
677
678     if (!dangles)
679         return energy;
680
681     ci = ON_SAME_STRAND(i, i + 1, cutpoint);
682     cj = ON_SAME_STRAND(j - 1, j, cutpoint);
683     cp = ON_SAME_STRAND(p - 1, p, cutpoint);
684     cq = ON_SAME_STRAND(q, q + 1, cutpoint);
685
686     d3 = ci ? P->dangle3[type][sil] : 0;
687     d5 = cj ? P->dangle5[type][sjl] : 0;
688     d5_2 = cp ? P->dangle5[type_2][sp1] : 0;
689     d3_2 = cq ? P->dangle3[type_2][sq1] : 0;
690
691     tmm = (cj && ci) ? P->mismatchExt[type][sjl][sil] : d5 + d3;
692     tmm_2 = (cp && cq) ? P->mismatchExt[type_2][sp1][sq1] : d5_2 + d3_2;
693
694     if (dangles == 2)
695         return energy + tmm + tmm_2;
696
697     /* now we may have non-double dangles only */
698     if (p - i > 2) {
699         if (j - q > 2) {
700             /* all degrees of freedom */
701             e = MIN2(tmm, d5);
702             e = MIN2(e, d3);
703             energy += e;
704             e = MIN2(tmm_2, d5_2);
705             e = MIN2(e, d3_2);
706             energy += e;
707         } else if (j - q == 2) {
708             /* all degrees of freedom in 5' part between i and p */
709             e = MIN2(tmm + d5_2, d3 + d5_2);
710             e = MIN2(e, d5 + d5_2);
711             e = MIN2(e, d3 + tmm_2);
712             e = MIN2(e, d3 + d3_2);
713             e = MIN2(e, tmm_2); /* no dangles on enclosing pair */
714             e = MIN2(e, d5_2); /* no dangles on enclosing pair */
715             e = MIN2(e, d3_2); /* no dangles on enclosing pair */
716             energy += e;
717         } else {
718             /* no unpaired base between q and j */
719             energy += d3 + d5_2;
720         }
721     } else if (p - i == 2) {
722         if (j - q > 2) {
723             /* all degrees of freedom in 3' part between q and j */
724             e = MIN2(tmm + d3_2, d5 + d3_2);
725             e = MIN2(e, d5 + d3_2);
726             e = MIN2(e, d3 + d3_2);
727             e = MIN2(e, d5 + tmm_2);
728             e = MIN2(e, tmm_2);
729             e = MIN2(e, d5_2);
730             e = MIN2(e, d3_2);
731             energy += e;
732         } else if (j - q == 2) {
733             /* one possible dangling base between either side */
734             e = MIN2(tmm, tmm_2);
735             e = MIN2(e, d3);
736             e = MIN2(e, d5);
737             e = MIN2(e, d5_2);
738             e = MIN2(e, d3_2);
739             e = MIN2(e, d3 + d3_2);
740             e = MIN2(e, d5 + d5_2);
741             energy += e;
742         } else {
743             /* one unpaired base between i and p */
744             energy += MIN2(d3, d5_2);
745         }
746     } else {
747         /* no unpaired base between i and p */
748         if (j - q > 2) {
749             /* all degrees of freedom in 3' part between q and j */
750             energy += d5 + d3_2;
751         } else if (j - q == 2) {

```

```

752     /* one unpaired base between q and j */
753     energy += MIN2(d5, d3_2);
754 }
755 }
756
757 return energy;
758 }
759
760
765 #endif
766
767 #endif

```

## 18.121 ViennaRNA/loops/multibranch.h File Reference

Energy evaluation of multibranch loops for MFE and partition function calculations.

Include dependency graph for multibranch.h: This graph shows which files directly or indirectly include this file:

### Functions

- int [vrna\\_BT\\_mb\\_loop](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int \*i, int \*j, int \*k, int en, int \*component1, int \*component2)

*Backtrack the decomposition of a multi branch loop closed by (i, j).*

#### Basic free energy interface

- int [vrna\\_E\\_mb\\_loop\\_stack](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j)  
*Evaluate energy of a multi branch helices stacking onto closing pair (i,j)*
- int [vrna\\_E\\_mb\\_loop\\_fast](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j, int \*dml1, int \*dml2)
- int [E\\_ml\\_rightmost\\_stem](#) (int i, int j, [vrna\\_fold\\_compound\\_t](#) \*fc)
- int [vrna\\_E\\_ml\\_stems\\_fast](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j, int \*fmi, int \*dml)

#### Boltzmann weight (partition function) interface

- [typedef struct vrna\\_mx\\_pf\\_aux\\_ml\\_s \\* vrna\\_mx\\_pf\\_aux\\_ml\\_t](#)  
*Auxiliary helper arrays for fast exterior loop computations.*
- [FLT\\_OR\\_DBL vrna\\_exp\\_E\\_mb\\_loop\\_fast](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j, [vrna\\_mx\\_pf\\_aux\\_ml\\_t](#) aux\_mx)
- [vrna\\_mx\\_pf\\_aux\\_ml\\_t vrna\\_exp\\_E\\_ml\\_fast\\_init](#) ([vrna\\_fold\\_compound\\_t](#) \*fc)
- void [vrna\\_exp\\_E\\_ml\\_fast\\_rotate](#) ([vrna\\_mx\\_pf\\_aux\\_ml\\_t](#) aux\_mx)
- void [vrna\\_exp\\_E\\_ml\\_fast\\_free](#) ([vrna\\_mx\\_pf\\_aux\\_ml\\_t](#) aux\_mx)
- const [FLT\\_OR\\_DBL \\* vrna\\_exp\\_E\\_ml\\_fast\\_qqm](#) ([vrna\\_mx\\_pf\\_aux\\_ml\\_t](#) aux\_mx)
- const [FLT\\_OR\\_DBL \\* vrna\\_exp\\_E\\_ml\\_fast\\_qqm1](#) ([vrna\\_mx\\_pf\\_aux\\_ml\\_t](#) aux\_mx)
- [FLT\\_OR\\_DBL vrna\\_exp\\_E\\_ml\\_fast](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int i, int j, [vrna\\_mx\\_pf\\_aux\\_ml\\_t](#) aux\_mx)

### 18.121.1 Detailed Description

Energy evaluation of multibranch loops for MFE and partition function calculations.

, ,

## 18.122 multibranch.h

#### Go to the documentation of this file.

```

1 #ifndef VIENNA_RNA_PACKAGE_LOOPS_MULTIBRANCH_H
2 #define VIENNA_RNA_PACKAGE_LOOPS_MULTIBRANCH_H
3
4 #include <ViennaRNA/utils/basic.h>
5 #include <ViennaRNA/datastructures/basic.h>
6 #include <ViennaRNA/fold_compound.h>
7 #include <ViennaRNA/params/basic.h>
8
9 #ifdef VRNA_WARN_DEPRECATED
10 # if defined(DEPRECATED)
11 #   undef DEPRECATED

```

```

12 # endif
13 # if defined(__clang__)
14 # define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
15 # elif defined(__GNUC__)
16 # define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
17 # else
18 # define DEPRECATED(func, msg) func
19 # endif
20 #else
21 # define DEPRECATED(func, msg) func
22#endif
23
24 #ifdef __GNUC__
25 # define INLINE inline
26 #else
27 # define INLINE
28#endif
29
30 int
31 vrna_E_mb_loop_stack(vrna_fold_compound_t *fc,
32                      int i,
33                      int j);
34
35
36 int
37 vrna_E_mb_loop_fast(vrna_fold_compound_t *fc,
38                      int i,
39                      int j,
40                      int *dmlil,
41                      int *dmli2);
42
43
44 int
45 E_ml_rightmost_stem(int i,
46                      int j,
47                      vrna_fold_compound_t *fc);
48
49
50 int
51 vrna_E_ml_stems_fast(vrna_fold_compound_t *fc,
52                      int i,
53                      int j,
54                      int *fmi,
55                      int *dmli);
56
57
58 /* End basic interface */
59 typedef struct vrna_mx_pf_aux_ml_s *vrna_mx_pf_aux_ml_t;
60
61
62 FLT_OR_DBL
63 vrna_exp_E_mb_loop_fast(vrna_fold_compound_t *fc,
64                         int i,
65                         int j,
66                         vrna_mx_pf_aux_ml_t aux_mx);
67
68
69 vrna_mx_pf_aux_ml_t
70 vrna_exp_E_ml_fast_init(vrna_fold_compound_t *fc);
71
72
73 void
74 vrna_exp_E_ml_fast_rotate(vrna_mx_pf_aux_ml_t aux_mx);
75
76
77 void
78 vrna_exp_E_ml_fast_free(vrna_mx_pf_aux_ml_t aux_mx);
79
80
81 const FLT_OR_DBL *
82 vrna_exp_E_ml_fast_qqm(vrna_mx_pf_aux_ml_t aux_mx);
83
84
85 const FLT_OR_DBL *
86 vrna_exp_E_ml_fast_qqm1(vrna_mx_pf_aux_ml_t aux_mx);
87
88
89 FLT_OR_DBL
90 vrna_exp_E_ml_fast(vrna_fold_compound_t *fc,
91                     int i,
92                     int j,
93                     vrna_mx_pf_aux_ml_t aux_mx);
94
95
96 /* End partition function interface */
97 int
98 vrna_BT_mb_loop(vrna_fold_compound_t *fc,
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164

```

```

165             int          *i,
166             int          *j,
167             int          *k,
168             int          en,
169             int          *component1,
170             int          *component2);
171
172 int
173 vrna_BT_mb_loop_split(vrna_fold_compound_t *fc,
174                         int          *i,
175                         int          *j,
176                         int          *k,
177                         int          *l,
178                         int          *component1,
179                         int          *component2,
180                         vrna_bp_stack_t *bp_stack,
181                         int          *stack_count);
182
183
184
190 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
191
216 PRIVATE INLINE int
217 E_MLstem(int      type,
218           int      sil,
219           int      sj1,
220           vrna_param_t *P)
221 {
222     int energy = 0;
223
224     if (sil >= 0 && sj1 >= 0)
225         energy += P->mismatchM[type][sil][sj1];
226     else if (sil >= 0)
227         energy += P->dangle5[type][sil];
228     else if (sj1 >= 0)
229         energy += P->dangle3[type][sj1];
230
231     if (type > 2)
232         energy += P->TerminalAU;
233
234     energy += P->MLintern[type];
235
236     return energy;
237 }
238
239
246 PRIVATE INLINE FLT_OR_DBL
247 exp_E_MLstem(int      type,
248               int      sil,
249               int      sj1,
250               vrna_exp_param_t *P)
251 {
252     double energy = 1.0;
253
254     if (sil >= 0 && sj1 >= 0)
255         energy = P->expMismatchM[type][sil][sj1];
256     else if (sil >= 0)
257         energy = P->expDangle5[type][sil];
258     else if (sj1 >= 0)
259         energy = P->expDangle3[type][sj1];
260
261     if (type > 2)
262         energy *= P->expTermAU;
263
264     energy *= P->expMLintern[type];
265     return (FLT_OR_DBL)energy;
266 }
267
268
269 #endif
270
275 #endif

```

## 18.123 ViennaRNA/LPfold.h File Reference

Partition function and equilibrium probability implementation for the sliding window algorithm.  
Include dependency graph for LPfold.h:

### Functions

- void [update\\_pf\\_paramsLP](#) (int length)

- `vrna_ep_t * pfl_fold` (char \*sequence, int winSize, int pairSize, float cutoffb, double \*\*pU, `vrna_ep_t` \*\*dpp2, FILE \*pUfp, FILE \*spup)
   
*Compute partition functions for locally stable secondary structures.*
- `vrna_ep_t * pfl_fold_par` (char \*sequence, int winSize, int pairSize, float cutoffb, double \*\*pU, `vrna_ep_t` \*\*dpp2, FILE \*pUfp, FILE \*spup, `vrna_exp_param_t` \*parameters)
   
*Compute partition functions for locally stable secondary structures.*
- void `putoutpU_prob` (double \*\*pU, int length, int ulength, FILE \*fp, int energies)
   
*Writes the unpaired probabilities (pU) or opening energies into a file.*
- void `putoutpU_prob_bin` (double \*\*pU, int length, int ulength, FILE \*fp, int energies)
   
*Writes the unpaired probabilities (pU) or opening energies into a binary file.*
- void `init_pf_foldLP` (int length)

### 18.123.1 Detailed Description

Partition function and equilibrium probability implementation for the sliding window algorithm.

This file contains the implementation for sliding window partition function and equilibrium probabilities. It also provides the unpaired probability implementation from Bernhart et al. 2011 [4]

### 18.123.2 Function Documentation

#### 18.123.2.1 init\_pf\_foldLP()

```
void init_pf_foldLP (
    int length )
```

Dunno if this function was ever used by external programs linking to RNAlib, but it was declared PUBLIC before. Anyway, never use this function as it will be removed soon and does nothing at all

## 18.124 LPfold.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_LPFOLD_H
2 #define VIENNA_RNA_PACKAGE_LPFOLD_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5
16 #include <stdio.h>
17
18 #include <ViennaRNA/datastructures/basic.h>
19 #include <ViennaRNA/params/basic.h>
20 #include <ViennaRNA/part_func_window.h>
21
22 #ifdef VRNA_WARN_DEPRECATED
23 # if defined(__clang__)
24 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
25 # elif defined(__GNUC__)
26 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
27 # else
28 #  define DEPRECATED(func, msg) func
29 # endif
30 #else
31 # define DEPRECATED(func, msg) func
32 #endif
33
41 DEPRECATED(void update_pf_paramsLP(int length),
42 "This function is obsolete");
43
44
51 DEPRECATED(void update_pf_paramsLP_par(int           length,
52                                         vrna_exp_param_t *parameters),
53 "Use the new API with vrna_folc_compound_t instead");
54
55
93 DEPRECATED(vrna_ep_t *pfl_fold(char           *sequence,
94                                 int             winSize,
95                                 int             pairSize,
96                                 float            cutoffb,
97                                 double           **pU,
```

```

98           vrna_ep_t      **dpp2,
99           FILE          *pUfp,
100          FILE          *spup),
101 "Use vrna_pfl_fold(), vrna_pfl_fold_cb(), vrna_pfl_fold_up(), or vrna_pfl_fold_up_cb() instead");
102
103
110 DEPRECATED(vrna_ep_t *pfl_fold_par(char           *sequence,
111              int            winsize,
112              int            pairSize,
113              float           cutoffb,
114              double          **pU,
115              vrna_ep_t      **dpp2,
116              FILE          *pUfp,
117              FILE          *spup,
118              vrna_exp_param_t *parameters),
119 "Use the new API and vrna_probs_window() instead");
120
121
122 DEPRECATED(void putoutpU_prob_par(double           **pU,
123              int            length,
124              int            ulength,
125              FILE          *fp,
126              int            energies,
127              vrna_exp_param_t *parameters),
128 "");
129
130
145 DEPRECATED(void    putoutpU_prob(double  **pU,
146              int            length,
147              int            ulength,
148              FILE          *fp,
149              int            energies),
150 "");
151
152
153 DEPRECATED(void    putoutpU_prob_bin_par(double           **pU,
154              int            length,
155              int            ulength,
156              FILE          *fp,
157              int            energies,
158              vrna_exp_param_t *parameters),
159 "");
160
161
176 DEPRECATED(void    putoutpU_prob_bin(double  **pU,
177              int            length,
178              int            ulength,
179              FILE          *fp,
180              int            energies),
181 "");
182
183
189 DEPRECATED(void init_pf_foldLP(int length),
190 "This function is obsolete");
191
192 #endif
193
194 #endif

```

## 18.125 ViennaRNA/MEA.h File Reference

Computes a MEA (maximum expected accuracy) structure.

Include dependency graph for MEA.h:

### Functions

- `char * vrna_MEAE (vrna_fold_compound_t *fc, double gamma, float *mea)`  
*Compute a MEA (maximum expected accuracy) structure.*
- `char * vrna_MEAE_from_plist (vrna_ep_t *plist, const char *sequence, double gamma, vrna_md_t *md, float *mea)`  
*Compute a MEA (maximum expected accuracy) structure from a list of probabilities.*
- `float MEAE (plist *p, char *structure, double gamma)`  
*Computes a MEA (maximum expected accuracy) structure.*

### 18.125.1 Detailed Description

Computes a MEA (maximum expected accuracy) structure.

## 18.126 MEA.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_MEA_H
2 #define VIENNA_RNA_PACKAGE_MEA_H
3
4 #include <ViennaRNA/datastructures/basic.h>
5 #include <ViennaRNA/params/basic.h>
6
35 char *
36 vrna_MEAE(vrna_fold_compound_t *fc,
37             double gamma,
38             float *mea);
39
40
70 char *
71 vrna_MEAE_from plist(vrna_ep_t *plist,
72                       const char *sequence,
73                       double gamma,
74                       vrna_md_t *md,
75                       float *mea);
76
77
78 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
79
80 #ifdef VRNA_WARN_DEPRECATED
81 # if defined(__clang__)
82 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
83 # elif defined(__GNUC__)
84 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
85 # else
86 #  define DEPRECATED(func, msg) func
87 # endif
88 #else
89 # define DEPRECATED(func, msg) func
90 #endif
91
92
108 DEPRECATED(float
109              MEA(plist *p,
110                  char *structure,
111                  double gamma),
112                  "Use vrna_MEAE() or vrna_MEAE_from plist() instead!");
113
114
115 DEPRECATED(float
116              MEA_seq(plist *p,
117                      const char *sequence,
118                      char *structure,
119                      double gamma,
120                      vrna_exp_param_t *pf),
121                  "Use vrna_MEAE() or vrna_MEAE_from plist() instead!");
122
123
124 #endif
125
126 #endif

```

## 18.127 ViennaRNA/mfe.h File Reference

Compute Minimum Free energy (MFE) and backtrace corresponding secondary structures from RNA sequence data.

Include dependency graph for mfe.h: This graph shows which files directly or indirectly include this file:

### Functions

- float `vrna_backtrack5 (vrna_fold_compound_t *fc, unsigned int length, char *structure)`  
*Backtrack an MFE (sub)structure.*

#### Basic global MFE prediction interface

- float `vrna_mfe (vrna_fold_compound_t *vc, char *structure)`

*Compute minimum free energy and an appropriate secondary structure of an RNA sequence, or RNA sequence alignment.*

- float **vrna\_mfe\_dimer** (*vrna\_fold\_compound\_t* \*vc, *char* \*structure)

*Compute the minimum free energy of two interacting RNA molecules.*

### Simplified global MFE prediction using sequence(s) or multiple sequence alignment(s)

- float **vrna\_fold** (*const char* \*sequence, *char* \*structure)

*Compute Minimum Free Energy (MFE), and a corresponding secondary structure for an RNA sequence.*

- float **vrna\_circfold** (*const char* \*sequence, *char* \*structure)

*Compute Minimum Free Energy (MFE), and a corresponding secondary structure for a circular RNA sequence.*

- float **vrna\_alifold** (*const char* \*\*sequences, *char* \*structure)

*Compute Minimum Free Energy (MFE), and a corresponding consensus secondary structure for an RNA sequence alignment using a comparative method.*

- float **vrna\_circalifold** (*const char* \*\*sequences, *char* \*structure)

*Compute Minimum Free Energy (MFE), and a corresponding consensus secondary structure for a sequence alignment of circular RNAs using a comparative method.*

- float **vrna\_cofold** (*const char* \*sequence, *char* \*structure)

*Compute Minimum Free Energy (MFE), and a corresponding secondary structure for two dimerized RNA sequences.*

### 18.127.1 Detailed Description

Compute Minimum Free energy (MFE) and backtrace corresponding secondary structures from RNA sequence data.

,

This file includes (almost) all function declarations within the RNAlib that are related to MFE folding...

## 18.128 mfe.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_MFE_H
2 #define VIENNA_RNA_PACKAGE_MFE_H
3
4 #include <stdio.h>
5 #include <ViennaRNA/datastructures/basic.h>
6 #include <ViennaRNA/fold_compound.h>
7
8 #ifdef VRNA_WARN_DEPRECATED
9 # if defined(__clang__)
10 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
11 # elif defined(__GNUC__)
12 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
13 # else
14 #  define DEPRECATED(func, msg) func
15 # endif
16 #else
17 # define DEPRECATED(func, msg) func
18 #endif
19
20
97 float
98 vrna_mfe(vrna_fold_compound_t *vc,
99           char                  *structure);
100
101
117 DEPRECATED(float
118           vrna_mfe_dimer(vrna_fold_compound_t *vc,
119                         char                  *structure),
120           "Use vrna_mfe() instead");
121
122
153 float
154 vrna_fold(const char *sequence,
155             char                  *structure);
156
157
180 float
181 vrna_circfold(const char *sequence,
182                 char                  *structure);
183
184

```

```

206 float
207 vrna_alifold(const char **sequences,
208                 char      *structure);
209
210
211 float
212 vrna_circalifold(const char **sequences,
213                     char      *structure);
214
215
216 DEPRECATED(float
217             vrna_cofold(const char *sequence,
218                         char      *structure),
219             "USE vrna_fold() instead");
220
221
222 int
223 vrna_backtrack_from_intervals(vrna_fold_compound_t   *vc,
224                                vrna_bp_stack_t     *bp_stack,
225                                sect                bt_stack[],
226                                int                  s);
227
228
229
230 float
231 vrna_backtrack5(vrna_fold_compound_t   *fc,
232                   unsigned int        length,
233                   char               *structure);
234
235
236 int
237 vrna_backtrack_window(vrna_fold_compound_t   *fc,
238                        const char          *Lfold_filename,
239                        long                file_pos,
240                        char               **structure,
241                        double              mfe);
242
243
244
245 #endif

```

## 18.129 ViennaRNA/mfe\_window.h File Reference

Compute local Minimum Free Energy (MFE) using a sliding window approach and backtrace corresponding secondary structures.

Include dependency graph for mfe\_window.h: This graph shows which files directly or indirectly include this file:

### Typedefs

- **typedef void() vrna\_mfe\_window\_callback**(int start, int end, const char \*structure, float en, void \*data)  
*The default callback for sliding window MFE structure predictions.*

### Functions

#### Basic local (sliding window) MFE prediction interface

- **float vrna\_mfe\_window** (*vrna\_fold\_compound\_t* \*vc, *FILE* \*file)  
*Local MFE prediction using a sliding window approach.*
- **float vrna\_mfe\_window\_cb** (*vrna\_fold\_compound\_t* \*vc, *vrna\_mfe\_window\_callback* \*cb, void \*data)
- **float vrna\_mfe\_window\_zscore** (*vrna\_fold\_compound\_t* \*vc, double min\_z, *FILE* \*file)  
*Local MFE prediction using a sliding window approach (with z-score cut-off)*
- **float vrna\_mfe\_window\_zscore\_cb** (*vrna\_fold\_compound\_t* \*vc, double min\_z, *vrna\_mfe\_window\_zscore\_callback* \*cb, void \*data)

#### Simplified local MFE prediction using sequence(s) or multiple sequence alignment(s)

- **float vrna\_Lfold** (*const char* \*string, int *window\_size*, *FILE* \*file)  
*Local MFE prediction using a sliding window approach (simplified interface)*
- **float vrna\_Lfold\_cb** (*const char* \*string, int *window\_size*, *vrna\_mfe\_window\_callback* \*cb, void \*data)
- **float vrna\_Lfoldz** (*const char* \*string, int *window\_size*, double *min\_z*, *FILE* \*file)  
*Local MFE prediction using a sliding window approach with z-score cut-off (simplified interface)*
- **float vrna\_Lfoldz\_cb** (*const char* \*string, int *window\_size*, double *min\_z*, *vrna\_mfe\_window\_zscore\_callback* \*cb, void \*data)
- **float vrna\_aliLfold** (*const char* \*\**alignment*, int *maxdist*, *FILE* \*fp)
- **float vrna\_aliLfold\_cb** (*const char* \*\**alignment*, int *maxdist*, *vrna\_mfe\_window\_callback* \*cb, void \*data)

## **18.129.1 Detailed Description**

Compute local Minimum Free Energy (MFE) using a sliding window approach and backtrace corresponding secondary structures.

This file includes the interface to all functions related to predicting locally stable secondary structures.

## 18.130 mfe\_window.h

[Go to the documentation of this file.](#)

```

232     FILE      *file);
233
234
235 float
236 vrna_Lfoldz_cb(const char
237             int           *string,
238             double        window_size,
239             vrna_mfe_window_zscore_callback *cb,
240             void           min_z,
241             *data);
242
243 #endif
244
245 float vrna_aliLfold(const char   **alignment,
246                      int            maxdist,
247                      FILE           *fp);
248
249
250 float vrna_aliLfold_cb(const char      **alignment,
251                         int           maxdist,
252                         vrna_mfe_window_callback *cb,
253                         void           *data);
254
255
256 /* End simplified local MFE interface */
257 /* End group mfe_fold_window */
263 #endif

```

## 18.131 ViennaRNA/mm.h File Reference

Several Maximum Matching implementations.

Include dependency graph for mm.h:

### Functions

- int [vrna\\_maximum\\_matching \(vrna\\_fold\\_compound\\_t \\*fc\)](#)
- int [vrna\\_maximum\\_matching\\_simple \(const char \\*sequence\)](#)

#### 18.131.1 Detailed Description

Several Maximum Matching implementations.

This file contains the declarations for several maximum matching implementations

#### 18.131.2 Function Documentation

##### 18.131.2.1 vrna\_maximum\_matching()

```
int vrna_maximum_matching (
    vrna_fold_compound_t * fc )
```

**SWIG Wrapper Notes** This function is attached as method **maximum\_matching()** to objects of type **fold\_<compound** (i.e. [vrna\\_fold\\_compound\\_t](#)).

##### 18.131.2.2 vrna\_maximum\_matching\_simple()

```
int vrna_maximum_matching_simple (
    const char * sequence )
```

**SWIG Wrapper Notes** This function is available as global function **maximum\_matching()**.

## 18.132 mm.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_MM_H
2 #define VIENNA_RNA_PACKAGE_MM_H
3
13 #include <ViennaRNA/fold_compound.h>
14
15 int
16 vrna_maximum_matching(vrna_fold_compound_t *fc);
17
18
19 int
20 vrna_maximum_matching_simple(const char *sequence);
21
22
23 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
24
25 unsigned int
26 maximumMatching(const char *string);
27
28
29 unsigned int *
30 maximumMatchingConstraint(const char *string,
31                           short      *ptable);
32
33
34 unsigned int *
35 maximumMatching2Constraint(const char *string,
36                           short      *ptable,
37                           short      *ptable2);
38
39
40 #endif
41
42 #endif

```

## 18.133 ViennaRNA/model.h File Reference

The model details data structure and its corresponding modifiers.

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct `vrna_md_s`

*The data structure that contains the complete model details used throughout the calculations.* [More...](#)

### Macros

- `#define VRNA_MODEL_DEFAULT_TEMPERATURE 37.0`  
*Default temperature for structure prediction and free energy evaluation in °C*
- `#define VRNA_MODEL_DEFAULT_PF_SCALE -1`  
*Default scaling factor for partition function computations.*
- `#define VRNA_MODEL_DEFAULT_BETA_SCALE 1.`  
*Default scaling factor for absolute thermodynamic temperature in Boltzmann factors.*
- `#define VRNA_MODEL_DEFAULT_DANGLES 2`  
*Default dangling end model.*
- `#define VRNA_MODEL_DEFAULT_SPECIAL_HP 1`  
*Default model behavior for lookup of special tri-, tetra-, and hexa-loops.*
- `#define VRNA_MODEL_DEFAULT_NO_LP 0`  
*Default model behavior for so-called 'lonely pairs'.*
- `#define VRNA_MODEL_DEFAULT_NO_GU 0`  
*Default model behavior for G-U base pairs.*
- `#define VRNA_MODEL_DEFAULT_NO_GU_CLOSURE 0`  
*Default model behavior for G-U base pairs closing a loop.*
- `#define VRNA_MODEL_DEFAULT_CIRC 0`

- `#define VRNA_MODEL_DEFAULT_GQUAD 0`

*Default model behavior to treat a molecule as a circular RNA (DNA)*
- `#define VRNA_MODEL_DEFAULT_UNIQ_ML 0`

*Default model behavior regarding the treatment of G-Quadruplexes.*
- `#define VRNA_MODEL_DEFAULT_ENERGY_SET 0`

*Default behavior of the model regarding unique multi-branch loop decomposition.*
- `#define VRNA_MODEL_DEFAULT_BACKTRACK 1`

*Default model behavior with regards to backtracking of structures.*
- `#define VRNA_MODEL_DEFAULT_BACKTRACK_TYPE 'F'`

*Default model behavior on what type of backtracking to perform.*
- `#define VRNA_MODEL_DEFAULT_COMPUTE_BPP 1`

*Default model behavior with regards to computing base pair probabilities.*
- `#define VRNA_MODEL_DEFAULT_MAX_BP_SPAN -1`

*Default model behavior for the allowed maximum base pair span.*
- `#define VRNA_MODEL_DEFAULT_WINDOW_SIZE -1`

*Default model behavior for the sliding window approach.*
- `#define VRNA_MODEL_DEFAULT_LOG_ML 0`

*Default model behavior on how to evaluate the energy contribution of multi-branch loops.*
- `#define VRNA_MODEL_DEFAULT_ALI_OLD_EN 0`

*Default model behavior for consensus structure energy evaluation.*
- `#define VRNA_MODEL_DEFAULT_ALI_RIBO 0`

*Default model behavior for consensus structure co-variance contribution assessment.*
- `#define VRNA_MODEL_DEFAULT_ALI_CV_FACT 1.`

*Default model behavior for weighting the co-variance score in consensus structure prediction.*
- `#define VRNA_MODEL_DEFAULT_ALI_NC_FACT 1.`

*Default model behavior for weighting the nucleotide conservation? in consensus structure prediction.*
- `#define MAXALPHA 20`

*Maximal length of alphabet.*

## Typedefs

- `typedef struct vrna_md_s vrna_md_t`

*Typename for the model details data structure `vrna_md_s`.*

## Functions

- `void vrna_md_set_default (vrna_md_t *md)`

*Apply default model details to a provided `vrna_md_t` data structure.*
- `void vrna_md_update (vrna_md_t *md)`

*Update the model details data structure.*
- `vrna_md_t * vrna_md_copy (vrna_md_t *md_to, const vrna_md_t *md_from)`

*Copy/Clone a `vrna_md_t` model.*
- `char * vrna_md_option_string (vrna_md_t *md)`

*Get a corresponding commandline parameter string of the options in a `vrna_md_t`.*
- `void vrna_md_defaults_reset (vrna_md_t *md_p)`

*Reset the global default model details to a specific set of parameters, or their initial values.*
- `void vrna_md_defaults_temperature (double T)`

*Set default temperature for energy evaluation of loops.*
- `double vrna_md_defaults_temperature_get (void)`

*Get default temperature for energy evaluation of loops.*
- `void vrna_md_defaults_betaScale (double b)`

- `double vrna_md_defaults_betaScale_get (void)`  
    *Set default scaling factor of thermodynamic temperature in Boltzmann factors.*
- `void vrna_md_defaults_dangles (int d)`  
    *Get default scaling factor of thermodynamic temperature in Boltzmann factors.*
- `int vrna_md_defaults_dangles_get (void)`  
    *Set default dangle model for structure prediction.*
- `void vrna_md_defaults_special_hp (int flag)`  
    *Get default dangle model for structure prediction.*
- `int vrna_md_defaults_special_hp_get (void)`  
    *Set default behavior for lookup of tabulated free energies for special hairpin loops, such as Tri-, Tetra-, or Hexa-loops.*
- `void vrna_md_defaults_noLP (int flag)`  
    *Get default behavior for lookup of tabulated free energies for special hairpin loops, such as Tri-, Tetra-, or Hexa-loops.*
- `int vrna_md_defaults_noLP_get (void)`  
    *Set default behavior for prediction of canonical secondary structures.*
- `void vrna_md_defaults_noGU (int flag)`  
    *Get default behavior for prediction of canonical secondary structures.*
- `int vrna_md_defaults_noGU_get (void)`  
    *Set default behavior for treatment of G-U wobble pairs.*
- `void vrna_md_defaults_noGUclosure (int flag)`  
    *Get default behavior for treatment of G-U wobble pairs.*
- `int vrna_md_defaults_noGUclosure_get (void)`  
    *Set default behavior for G-U pairs as closing pair for loops.*
- `void vrna_md_defaults_logML (int flag)`  
    *Get default behavior for G-U pairs as closing pair for loops.*
- `int vrna_md_defaults_logML_get (void)`  
    *Set default behavior recomputing free energies of multi-branch loops using a logarithmic model.*
- `void vrna_md_defaults_circ (int flag)`  
    *Get default behavior recomputing free energies of multi-branch loops using a logarithmic model.*
- `int vrna_md_defaults_circ_get (void)`  
    *Set default behavior whether input sequences are circularized.*
- `void vrna_md_defaults_gquad (int flag)`  
    *Get default behavior whether input sequences are circularized.*
- `int vrna_md_defaults_gquad_get (void)`  
    *Set default behavior for treatment of G-Quadruplexes.*
- `void vrna_md_defaults_uniq_ML (int flag)`  
    *Get default behavior for treatment of G-Quadruplexes.*
- `int vrna_md_defaults_uniq_ML_get (void)`  
    *Set default behavior for creating additional matrix for unique multi-branch loop prediction.*
- `void vrna_md_defaults_energy_set (int e)`  
    *Get default behavior for creating additional matrix for unique multi-branch loop prediction.*
- `int vrna_md_defaults_energy_set_get (void)`  
    *Set default energy set.*
- `void vrna_md_defaults_backtrack (int flag)`  
    *Get default energy set.*
- `int vrna_md_defaults_backtrack_get (void)`  
    *Set default behavior for whether to backtrack secondary structures.*
- `void vrna_md_defaults_backtrack_type (char t)`  
    *Get default behavior for whether to backtrack secondary structures.*
- `char vrna_md_defaults_backtrack_type_get (void)`  
    *Set default backtrack type, i.e. which DP matrix is used.*
- `int vrna_md_defaults_backtrack_type_get (void)`  
    *Get default backtrack type, i.e. which DP matrix is used.*

- void `vrna_md_defaults_compute_bpp` (int flag)
 

*Set the default behavior for whether to compute base pair probabilities after partition function computation.*
- int `vrna_md_defaults_compute_bpp_get` (void)
 

*Get the default behavior for whether to compute base pair probabilities after partition function computation.*
- void `vrna_md_defaults_max_bp_span` (int span)
 

*Set default maximal base pair span.*
- int `vrna_md_defaults_max_bp_span_get` (void)
 

*Get default maximal base pair span.*
- void `vrna_md_defaults_min_loop_size` (int size)
 

*Set default minimal loop size.*
- int `vrna_md_defaults_min_loop_size_get` (void)
 

*Get default minimal loop size.*
- void `vrna_md_defaults_window_size` (int size)
 

*Set default window size for sliding window structure prediction approaches.*
- int `vrna_md_defaults_window_size_get` (void)
 

*Get default window size for sliding window structure prediction approaches.*
- void `vrna_md_defaults_oldAliEn` (int flag)
 

*Set default behavior for whether to use old energy model for comparative structure prediction.*
- int `vrna_md_defaults_oldAliEn_get` (void)
 

*Get default behavior for whether to use old energy model for comparative structure prediction.*
- void `vrna_md_defaults_ribo` (int flag)
 

*Set default behavior for whether to use Ribosum Scoring in comparative structure prediction.*
- int `vrna_md_defaults_ribo_get` (void)
 

*Get default behavior for whether to use Ribosum Scoring in comparative structure prediction.*
- void `vrna_md_defaults_cv_fact` (double factor)
 

*Set the default co-variance scaling factor used in comparative structure prediction.*
- double `vrna_md_defaults_cv_fact_get` (void)
 

*Get the default co-variance scaling factor used in comparative structure prediction.*
- void `vrna_md_defaults_nc_fact` (double factor)
 

*Set the default scaling factor used to avoid under-/overflows in partition function computation.*
- double `vrna_md_defaults_nc_fact_get` (void)
 

*Get the default scaling factor used to avoid under-/overflows in partition function computation.*
- void `set_model_details` (`vrna_md_t` \*md)
 

*Set default model details.*

## Variables

- double `temperature`

*Rescale energy parameters to a temperature in degC.*
- double `pf_scale`

*A scaling factor used by `pf_fold()` to avoid overflows.*
- int `dangles`

*Switch the energy model for dangling end contributions (0, 1, 2, 3)*
- int `tetra_loop`

*Include special stabilizing energies for some tri-, tetra- and hexa-loops;*
- int `noLonelyPairs`

*Global switch to avoid/allow helices of length 1.*
- int `noGU`

*Global switch to forbid/allow GU base pairs at all.*

- int **no\_closingGU**  
*GU allowed only inside stacks if set to 1.*
- int **circ**  
*backward compatibility variable.. this does not effect anything*
- int **gquad**  
*Allow G-quadruplex formation.*
- int **uniq\_ML**  
*do ML decomposition uniquely (for subopt)*
- int **energy\_set**  
*0 = BP; 1=any with GC; 2=any with AU-parameter*
- int **do\_backtrack**  
*do backtracking, i.e. compute secondary structures or base pair probabilities*
- char **backtrack\_type**  
*A backtrack array marker for `inverse_fold()`*
- char \* **nonstandards**  
*contains allowed non standard base pairs*
- int **max\_bp\_span**  
*Maximum allowed base pair span.*
- int **oldAliEn**  
*use old alifold energies (with gaps)*
- int **ribo**  
*use ribosum matrices*
- int **logML**  
*if nonzero use logarithmic ML energy in `energy_of_struct`*

### 18.133.1 Detailed Description

The model details data structure and its corresponding modifiers.

## 18.134 model.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_MODEL_H
2 #define VIENNA_RNA_PACKAGE_MODEL_H
3
17 #ifndef NBASES
18 #define NBASES 8
19 #endif
20
22 typedef struct vrna_md_s vrna_md_t;
23
30 #define VRNA_MODEL_DEFAULT_TEMPERATURE 37.0
31
36 #define VRNA_MODEL_DEFAULT_PF_SCALE -1
37
42 #define VRNA_MODEL_DEFAULT_BETA_SCALE 1.
43
47 #define VRNA_MODEL_DEFAULT_DANGLES 2
48
53 #define VRNA_MODEL_DEFAULT_SPECIAL_HP 1
54
59 #define VRNA_MODEL_DEFAULT_NO_LP 0
60
65 #define VRNA_MODEL_DEFAULT_NO_GU 0
66
71 #define VRNA_MODEL_DEFAULT_NO_GU_CLOSURE 0
72
77 #define VRNA_MODEL_DEFAULT_CIRC 0
78
83 #define VRNA_MODEL_DEFAULT_GQUAD 0
84
89 #define VRNA_MODEL_DEFAULT_UNIQ_ML 0
90
95 #define VRNA_MODEL_DEFAULT_ENERGY_SET 0

```

```
96
101 #define VRNA_MODEL_DEFAULT_BACKTRACK      1
102
107 #define VRNA_MODEL_DEFAULT_BACKTRACK_TYPE 'F'
108
113 #define VRNA_MODEL_DEFAULT_COMPUTE_BPP      1
114
119 #define VRNA_MODEL_DEFAULT_MAX_BP_SPAN     -1
120
125 #define VRNA_MODEL_DEFAULT_WINDOW_SIZE     -1
126
131 #define VRNA_MODEL_DEFAULT_LOG_ML          0
132
137 #define VRNA_MODEL_DEFAULT_ALI_OLD_EN       0
138
143 #define VRNA_MODEL_DEFAULT_ALI_RIBO        0
144
149 #define VRNA_MODEL_DEFAULT_ALI_CV_FACT     1.
150
154 #define VRNA_MODEL_DEFAULT_ALI_NC_FACT     1.
155
156
157 #define VRNA_MODEL_DEFAULT_PF_SMOOTH       1
158
159
160 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
161
162 #ifndef MAXALPHA
166 #define MAXALPHA                         20
167 #endif
168
169 #endif
170
180 struct vrna_md_s {
181     double   temperature;
182     double   betaScale;
183     int      pf_smooth;
184     int      dangles;
208     int      special_hp;
209     int      noLP;
210     int      noGU;
211     int      noGUclosure;
212     int      logML;
213     int      circ;
214     int      gquad;
215     int      uniq_ML;
216     int      energy_set;
217     int      backtrack;
218     char    backtrack_type;
219     int      compute_bpp;
220     char    nonstandards[64];
221     int      max_bp_span;
223     int      min_loop_size;
227     int      window_size;
228     int      oldAliEn;
229     int      ribo;
230     double   cv_fact;
231     double   nc_fact;
232     double   sfact;
233     int      rtype[8];
234     short   alias[MAXALPHA + 1];
235     int      pair[MAXALPHA + 1][MAXALPHA + 1];
236     float   pair_dist[7][7];
237 };
238
239
248 void
249 vrna_md_set_default(vrna_md_t *md);
250
251
264 void
265 vrna_md_update(vrna_md_t *md);
266
267
278 vrna_md_t *
279 vrna_md_copy(vrna_md_t      *md_to,
280                 const vrna_md_t *md_from);
281
282
288 char *
289 vrna_md_option_string(vrna_md_t *md);
290
291
292 void
293 vrna_md_set_nonstandards(vrna_md_t *md,
294                           const char *ns_bases);
295
```

```
296
314 void
315 vrna_md_defaults_reset(vrna_md_t *md_p);
316
317
323 void
324 vrna_md_defaults_temperature(double T);
325
326
332 double
333 vrna_md_defaults_temperature_get(void);
334
335
343 void
344 vrna_md_defaults_betaScale(double b);
345
346
353 double
354 vrna_md_defaults_betaScale_get(void);
355
356
357 void
358 vrna_md_defaults_pf_smooth(int s);
359
360
361 int
362 vrna_md_defaults_pf_smooth_get(void);
363
364
370 void
371 vrna_md_defaults_dangles(int d);
372
373
379 int
380 vrna_md_defaults_dangles_get(void);
381
382
388 void
389 vrna_md_defaults_special_hp(int flag);
390
391
397 int
398 vrna_md_defaults_special_hp_get(void);
399
400
406 void
407 vrna_md_defaults_noLP(int flag);
408
409
415 int
416 vrna_md_defaults_noLP_get(void);
417
418
424 void
425 vrna_md_defaults_noGU(int flag);
426
427
433 int
434 vrna_md_defaults_noGU_get(void);
435
436
442 void
443 vrna_md_defaults_noGUclosure(int flag);
444
445
451 int
452 vrna_md_defaults_noGUclosure_get(void);
453
454
460 void
461 vrna_md_defaults_logML(int flag);
462
463
469 int
470 vrna_md_defaults_logML_get(void);
471
472
478 void
479 vrna_md_defaults_circ(int flag);
480
481
487 int
488 vrna_md_defaults_circ_get(void);
489
490
496 void
497 vrna_md_defaults_gquad(int flag);
```

```
498
499
505 int
506 vrna_md_defaults_gquad_get(void);
507
508
515 void
516 vrna_md_defaults_uniq_ML(int flag);
517
518
524 int
525 vrna_md_defaults_uniq_ML_get(void);
526
527
533 void
534 vrna_md_defaults_energy_set(int e);
535
536
542 int
543 vrna_md_defaults_energy_set_get(void);
544
545
551 void
552 vrna_md_defaults_backtrack(int flag);
553
554
560 int
561 vrna_md_defaults_backtrack_get(void);
562
563
569 void
570 vrna_md_defaults_backtrack_type(char t);
571
572
578 char
579 vrna_md_defaults_backtrack_type_get(void);
580
581
587 void
588 vrna_md_defaults_compute_bpp(int flag);
589
590
596 int
597 vrna_md_defaults_compute_bpp_get(void);
598
599
605 void
606 vrna_md_defaults_max_bp_span(int span);
607
608
614 int
615 vrna_md_defaults_max_bp_span_get(void);
616
617
623 void
624 vrna_md_defaults_min_loop_size(int size);
625
626
632 int
633 vrna_md_defaults_min_loop_size_get(void);
634
635
641 void
642 vrna_md_defaults_window_size(int size);
643
644
650 int
651 vrna_md_defaults_window_size_get(void);
652
653
661 void
662 vrna_md_defaults_oldAliEn(int flag);
663
664
670 int
671 vrna_md_defaults_oldAliEn_get(void);
672
673
679 void
680 vrna_md_defaults_ribo(int flag);
681
682
688 int
689 vrna_md_defaults_ribo_get(void);
690
691
697 void
```

```
698 vrna_md_defaults_cv_fact(double factor);
699
700
706 double
707 vrna_md_defaults_cv_fact_get(void);
708
709
715 void
716 vrna_md_defaults_nc_fact(double factor);
717
718
724 double
725 vrna_md_defaults_nc_fact_get(void);
726
727
733 void
734 vrna_md_defaults_sfact(double factor);
735
736
742 double
743 vrna_md_defaults_sfact_get(void);
744
745
746 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
747
748 #define model_detailsT           vrna_md_t          /* restore compatibility of struct rename */
749
750 /* BEGIN deprecated global variables: */
751
761 extern double temperature;
762
774 extern double pf_scale;
775
797 extern int dangles;
798
804 extern int tetra_loop;
805
813 extern int noLonelyPairs;
814
818 extern int noGU;
819
823 extern int no_closingGU;
824
828 extern int circ;
829
833 extern int gquad;
834
838 extern int uniq_ML;
839
847 extern int energy_set;
848
855 extern int do_backtrack;
856
864 extern char backtrack_type;
865
873 extern char *nonstandards;
874
880 extern int max_bp_span;
881
885 extern int oldAliEn;
886
890 extern int ribo;
891
892 extern double cv_fact;
893
894 extern double nc_fact;
895
897 extern int logML;
898
899 /* END deprecated global variables: */
900
914 void
915 set_model_details(vrna_md_t *md);
916
917
918 char *
919 option_string(void);
920
921
922 #endif
927 #endif
```

## 18.135 move\_set.h

```
1 #ifndef __MOVE_SET_H
```

```

2 #define __MOVE_SET_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5
6 typedef struct _struct_en{
7     int energy;           /* energy in 10kcal/mol*/
8     short *structure;    /* structure in energy_of_move format*/
9 } struct_en;
10
11 /* prints structure*/
12 void print_stren(FILE *out, struct_en *str);
13 void print_str(FILE *out, short *str);
14
15 /* copying functions*/
16 void copy_arr(short *dest, short *src); /*just copy*/
17 short *allocopy(short *src);           /*copy and make space*/
18
19 enum MOVE_TYPE {GRADIENT, FIRST, ADAPTIVE};
20
21 /* walking methods (verbose_lvl 0-2, shifts = use shift moves? noLP = no lone pairs? (not compatible with
shifts))
22
23     input:   seq - sequence
24             ptable - structure encoded with make_pair_table() from pair_mat.h
25             s, sl - sequence encoded with encode_sequence from pair_mat.h
26
27     methods: deepest - lowest energy structure is used
28             first - first found lower energy structure is used
29             rand - random lower energy structure is used
30
31     returns local minima structure in ptable and its energy in 10kcal/mol as output */
32
33 int move_gradient( char *seq,
34                     short *ptable,
35                     short *s,
36                     short *sl,
37                     int verbosity_level,
38                     int shifts,
39                     int noLP);
34
35 int move_first( char *seq,
36                  short *ptable,
37                  short *s,
38                  short *sl,
39                  int verbosity_level,
40                  int shifts,
41                  int noLP);
42
43 int move_adaptive( char *seq,
44                     short *ptable,
45                     short *s,
46                     short *sl,
47                     int verbosity_level);
48
49 /* standardized method that encapsulates above "_pt" methods
50
51     input:   seq - sequence
52             struc - structure in dot-bracket notation
53             type - type of move selection according to MOVE_TYPE enum
54
55     return: energy of LM
56             structure of LM in struc in bracket-dot notation
57 */
58
59 int move_standard(char *seq,
60                     char *struc,
61                     enum MOVE_TYPE type,
62                     int verbosity_level,
63                     int shifts,
64                     int noLP);
65
66
67 /* browse_neighbours and perform funct function on each of them (used mainly for user specified flooding)
68
69     input:   seq - sequence
70             ptable - structure encoded with make_pair_table() from pair_mat.h
71             s, sl - sequence encoded with encode_sequence from pair_mat.h
72             funct - function (structure from neighbourhood, structure from input) to perform on every
73             structure in neighbourhood (if the function returns non-zero, the iteration through neighbourhood
74             stops.)
75
76     returns energy of the structure funct sets as second argument*/
77
78 int browse_neighs_pt( char *seq,
79                     short *ptable,
80                     short *s,
81                     short *sl,
82                     int verbosity_level,
83                     int shifts,
84                     int noLP,
85                     int (*funct) (struct_en*, struct_en*));
86
87
88 int browse_neighs( char *seq,
89                     char *struc,
90                     int verbosity_level,
91                     int shifts,
92                     int noLP,
93                     int (*funct) (struct_en*, struct_en*));

```

```

89
90 #endif
91
92 #endif
93
94

```

## 18.136 ViennaRNA/multibranch\_loops.h File Reference

Use [ViennaRNA/loops/multibranch.h](#) instead.

Include dependency graph for multibranch\_loops.h:

### 18.136.1 Detailed Description

Use [ViennaRNA/loops/multibranch.h](#) instead.

**Deprecated** Use [ViennaRNA/loops/multibranch.h](#) instead

## 18.137 multibranch\_loops.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_LOOPS_MULTIBRANCH_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_LOOPS_MULTIBRANCH_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 # ifdef VRNA_WARN_DEPRECATED
6 # warning "Including deprecated header file <ViennaRNA/multibranch_loops.h>! Use
7     <ViennaRNA/loops/multibranch.h> instead!"
8 # endif
9 #include <ViennaRNA/loops/multibranch.h>
10 #endif
11
12
13
14
15
16
17 #endif

```

## 18.138 ViennaRNA/naview.h File Reference

Use ViennaRNA/plotting/naview/naview.h instead.

### 18.138.1 Detailed Description

Use ViennaRNA/plotting/naview/naview.h instead.

**Deprecated** Use ViennaRNA/plotting/naview/naview.h instead

## 18.139 naview.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PLOT_NAVIEW_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_PLOT_NAVIEW_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 # ifdef VRNA_WARN_DEPRECATED
6 # warning "Including deprecated header file <ViennaRNA/naview.h>! Use <ViennaRNA/plotting/naview.h>
7     instead!"
8 # endif
9 # ifdef VRNA_WITH_NAVIEW_LAYOUT
10 # include <ViennaRNA/plotting/naview/naview.h>
11 # else
12 #   warning "Naview Layout algorithm is not available in this version!"
13 # endif
14 #endif
15
16
17
18
19 #endif
20
21 #endif

```

## 18.140 ViennaRNA/landscape/neighbor.h File Reference

Methods to compute the neighbors of an RNA secondary structure.

Include dependency graph for neighbor.h: This graph shows which files directly or indirectly include this file:

### Macros

- `#define VRNA_NEIGHBOR_CHANGE 1`  
*State indicator for a neighbor that has been changed.*
- `#define VRNA_NEIGHBOR_INVALID 2`  
*State indicator for a neighbor that has been invalidated.*
- `#define VRNA_NEIGHBOR_NEW 3`  
*State indicator for a neighbor that has become newly available.*

### Typedefs

- `typedef void() vrna_callback_move_update(vrna_fold_compound_t *fc, vrna_move_t neighbor, unsigned int state, void *data)`  
*Prototype of the neighborhood update callback.*

### Functions

- `void vrna_loopidx_update (int *loopidx, const short *pt, int length, const vrna_move_t *m)`  
*Alters the loopIndices array that was constructed with `vrna_loopidx_from_ptable()`.*
- `vrna_move_t * vrna_neighbors (vrna_fold_compound_t *vc, const short *pt, unsigned int options)`  
*Generate neighbors of a secondary structure.*
- `vrna_move_t * vrna_neighbors_successive (const vrna_fold_compound_t *vc, const vrna_move_t *curr←_move, const short *prev_pt, const vrna_move_t *prev_neighbors, int size_prev_neighbors, int *size←_neighbors, unsigned int options)`  
*Generate neighbors of a secondary structure (the fast way)*
- `int vrna_move_neighbor_diff_cb (vrna_fold_compound_t *fc, short *ptable, vrna_move_t move, vrna_callback_move_update *cb, void *data, unsigned int options)`  
*Apply a move to a secondary structure and indicate which neighbors have changed consequentially.*
- `vrna_move_t * vrna_move_neighbor_diff (vrna_fold_compound_t *fc, short *ptable, vrna_move_t move, vrna_move_t **invalid_moves, unsigned int options)`  
*Apply a move to a secondary structure and indicate which neighbors have changed consequentially.*

### 18.140.1 Detailed Description

Methods to compute the neighbors of an RNA secondary structure.

## 18.141 neighbor.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_NEIGHBOR_H
2 #define VIENNA_RNA_PACKAGE_NEIGHBOR_H
3
111 #include <ViennaRNA/fold_compound.h>
112 #include <ViennaRNA/landscape/move.h>
113
124 typedef void (vrna_callback_move_update)(vrna_fold_compound_t *fc,
125                                     vrna_move_t             neighbor,
126                                     unsigned int           state,
127                                     void                  *data);
128
129
135 #define VRNA_NEIGHBOR_CHANGE    1
136
137
143 #define VRNA_NEIGHBOR_INVALID   2
144

```

```

145
151 #define VRNA_NEIGHBOR_NEW      3
152
153
154 void
155 vrna_loopidx_update(int          *loopidx,
156                      const short    *pt,
157                      int             length,
158                      const vrna_move_t *m);
159
160
161
162 vrna_move_t *
163 vrna_neighbors(vrna_fold_compound_t *vc,
164                 const short     *pt,
165                 unsigned int    options);
166
167
168 vrna_move_t *
169 vrna_neighbors_successive(const vrna_fold_compound_t   *vc,
170                           const vrna_move_t      *curr_move,
171                           const short            *prev_pt,
172                           const vrna_move_t      *prev_neighbors,
173                           int                     size_prev_neighbors,
174                           int                     size_neighbors,
175                           unsigned int           options);
176
177
178
179 vrna_move_t *
180 vrna_move_neighbor_diff_cb(vrna_fold_compound_t      *fc,
181                            short                *ptable,
182                            vrna_move_t          move,
183                            vrna_callback_move_update *cb,
184                            void                 *data,
185                            unsigned int         options);
186
187
188 vrna_move_t *
189 vrna_move_neighbor_diff(vrna_fold_compound_t      *fc,
190                         short                *ptable,
191                         vrna_move_t          move,
192                         vrna_move_t          **invalid_moves,
193                         unsigned int         options);
194
195
196
197 #endif /* VIENNA_RNA_PACKAGE_NEIGHBOR_H */

```

## 18.142 ViennaRNA/neighbor.h File Reference

Use [ViennaRNA/landscape/neighbor.h](#) instead.

Include dependency graph for neighbor.h:

### 18.142.1 Detailed Description

Use [ViennaRNA/landscape/neighbor.h](#) instead.

**Deprecated** Use [ViennaRNA/landscape/neighbor.h](#) instead

## 18.143 neighbor.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_NEIGHBOR_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_NEIGHBOR_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 #warning "Including deprecated header file <ViennaRNA/neighbor.h>! Use <ViennaRNA/landscape/neighbor.h>
6 instead!"
7 #endif
8 #include <ViennaRNA/landscape/neighbor.h>
9 #include <ViennaRNA/landscape/move.h>
10#endif
11
12#endif

```

## 18.144 pair\_mat.h

```

1 #ifndef VIENNA_RNA_PACKAGE_PAIR_MAT_H
2 #define VIENNA_RNA_PACKAGE_PAIR_MAT_H
3
4 #include <ctype.h>
5 #include <ViennaRNA/utils/basic.h>
6 #include <ViennaRNA/fold_vars.h>
7
8 #define NBASES 8
9 /*@notnull@*/
10
11 #ifndef INLINE
12 # ifdef __GNUC__
13 #   define INLINE inline
14 # else
15 #   define INLINE
16 # endif
17 #endif
18
19 static const char Law_and_Order[] = "_ACGUTXKI";
20 static int BP_pair[NBASES][NBASES] =
21 /* _ A C G U X K I */
22 { { 0, 0, 0, 0, 0, 0, 0, 0 },
23 { 0, 0, 0, 0, 5, 0, 0, 5 },
24 { 0, 0, 0, 1, 0, 0, 0, 0 },
25 { 0, 0, 2, 0, 3, 0, 0, 0 },
26 { 0, 6, 0, 4, 0, 0, 0, 6 },
27 { 0, 0, 0, 0, 0, 2, 0, 0 },
28 { 0, 0, 0, 0, 0, 1, 0, 0 },
29 { 0, 6, 0, 0, 5, 0, 0, 0 } };
30
31 #define MAXALPHA 20 /* maximal length of alphabet */
32
33 static short alias[MAXALPHA + 1];
34 static int pair[MAXALPHA + 1][MAXALPHA + 1];
35 /* rtype[pair[i][j]]:=pair[j][i] */
36 static int rtype[8] = {
37 0, 2, 1, 4, 3, 6, 5, 7
38 };
39
40 #ifdef _OPENMP
41 #pragma omp threadprivate(Law_and_Order, BP_pair, alias, pair, rtype)
42 #endif
43
44 /* for backward compatibility */
45 #define ENCODE(c) encode_char(c)
46
47 static INLINE int
48 encode_char(char c)
49 {
50 /* return numerical representation of base used e.g. in pair[][] */
51 int code;
52
53 c = toupper(c);
54
55 if (energy_set > 0) {
56 code = (int)(c - 'A') + 1;
57 } else {
58 const char *pos;
59 pos = strchr(Law_and_Order, c);
60 if (pos == NULL)
61 code = 0;
62 else
63 code = (int)(pos - Law_and_Order);
64
65 if (code > 5)
66 code = 0;
67
68 if (code > 4)
69 code--; /* make T and U equivalent */
70 }
71
72 return code;
73 }
74
75
76 /*@+boolint +charint@*/
77 /*@null@*/
78 extern char *nonstandards;
79
80 static INLINE void
81 make_pair_matrix(void)
82 {
83 int i, j;
84
85 if (energy_set == 0) {

```

```

86     for (i = 0; i < 5; i++)
87         alias[i] = (short)i;
88     alias[5] = 3; /* X <-> G */
89     alias[6] = 2; /* K <-> C */
90     alias[7] = 0; /* I <-> default base '@' */
91     for (i = 0; i < NBASES; i++)
92         for (j = 0; j < NBASES; j++)
93             pair[i][j] = BP_pair[i][j];
94     if (nogu)
95         pair[3][4] = pair[4][3] = 0;
96
97     if (nonstandards != NULL) {
98         /* allow nonstandard bp's */
99         for (i = 0; i < (int)strlen(nonstandards); i += 2)
100             pair[encode_char(nonstandards[i])]
101                 [encode_char(nonstandards[i + 1])] = 7;
102     }
103
104     for (i = 0; i < NBASES; i++)
105         for (j = 0; j < NBASES; j++)
106             rtype[pair[i][j]] = pair[j][i];
107 } else {
108     for (i = 0; i <= MAXALPHA; i++)
109         for (j = 0; j <= MAXALPHA; j++)
110             pair[i][j] = 0;
111     if (energy_set == 1) {
112         for (i = 1; i < MAXALPHA; ) {
113             alias[i++] = 3; /* A <-> G */
114             alias[i++] = 2; /* B <-> C */
115         }
116         for (i = 1; i < MAXALPHA; i++) {
117             pair[i][i + 1] = 2; /* AB <-> GC */
118             i++;
119             pair[i][i - 1] = 1; /* BA <-> CG */
120         }
121     } else if (energy_set == 2) {
122         for (i = 1; i < MAXALPHA; ) {
123             alias[i++] = 1; /* A <-> A */
124             alias[i++] = 4; /* B <-> U */
125         }
126         for (i = 1; i < MAXALPHA; i++) {
127             pair[i][i + 1] = 5; /* AB <-> AU */
128             i++;
129             pair[i][i - 1] = 6; /* BA <-> UA */
130         }
131     } else if (energy_set == 3) {
132         for (i = 1; i < MAXALPHA - 2; ) {
133             alias[i++] = 3; /* A <-> G */
134             alias[i++] = 2; /* B <-> C */
135             alias[i++] = 1; /* C <-> A */
136             alias[i++] = 4; /* D <-> U */
137         }
138         for (i = 1; i < MAXALPHA - 2; i++) {
139             pair[i][i + 1] = 2; /* AB <-> GC */
140             i++;
141             pair[i][i - 1] = 1; /* BA <-> CG */
142             i++;
143             pair[i][i + 1] = 5; /* CD <-> AU */
144             i++;
145             pair[i][i - 1] = 6; /* DC <-> UA */
146         }
147     } else {
148         vrna_message_error("What energy_set are YOU using??");
149     }
150
151     for (i = 0; i <= MAXALPHA; i++)
152         for (j = 0; j <= MAXALPHA; j++)
153             rtype[pair[i][j]] = pair[j][i];
154 }
155 }
156
157
158 static INLINE short *
159 encode_sequence(const char *sequence,
160                  short how)
161 {
162     unsigned int i, l = (unsigned int)strlen(sequence);
163     short *S = (short *)vrna_alloc(sizeof(short) * (l + 2));
164
165     switch (how) {
166         /* standard encoding as always used for S */
167         case 0:
168             for (i = 1; i <= l; i++) /* make numerical encoding of sequence */
169                 S[i] = (short)encode_char(sequence[i - 1]);
170             S[l + 1] = S[l];
171             S[0] = (short)l;
172             break;

```

```

173     /* encoding for mismatches of nonstandard bases (normally used for S1) */
174     case 1:
175         for (i = 1; i <= l; i++)
176             S[i] = alias[(short)encode_char(sequence[i - 1])];
177         S[l + 1] = S[1];
178         S[0] = S[1];
179         break;
180     }
181
182     return S;
183 }
184
185
186 #endif /* VIENNA_RNA_PACKAGE_PAIR_MAT_H */

```

## 18.145 ViennaRNA/params.h File Reference

Use [ViennaRNA/params/basic.h](#) instead.

Include dependency graph for params.h:

### 18.145.1 Detailed Description

Use [ViennaRNA/params/basic.h](#) instead.

**Deprecated** Use [ViennaRNA/params/basic.h](#) instead

## 18.146 params.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PARAMS_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_PARAMS_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 #warning "Including deprecated header file <ViennaRNA/params.h>! Use <ViennaRNA/params/basic.h> instead!"
13 # endif
14 #include <ViennaRNA/params/basic.h>
15 #endif
16
17 #endif

```

## 18.147 ViennaRNA/params/1.8.4\_epars.h File Reference

Free energy parameters for parameter file conversion.

### 18.147.1 Detailed Description

Free energy parameters for parameter file conversion.

This file contains the free energy parameters used in ViennaRNAPackage 1.8.4. They are summarized in:  
D.H.Mathews, J. Sabina, M. Zuker, D.H. Turner "Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure" JMB, 288, pp 911-940, 1999

Enthalpies taken from:

A. Walter, D Turner, J Kim, M Lyttle, P M'uller, D Mathews, M Zuker "Coaxial stacking of helices enhances binding of oligoribonucleotides.." PNAS, 91, pp 9218-9222, 1994 D.H. Turner, N. Sugimoto, and S.M. Freier. "RNA Structure Prediction", Ann. Rev. Biophys. Biophys. Chem. 17, 167-192, 1988. John A.Jaeger, Douglas H.Turner, and Michael Zuker. "Improved predictions of secondary structures for RNA", PNAS, 86, 7706-7710, October 1989. L. He, R. Kierzek, J. SantaLucia, A.E. Walter, D.H. Turner "Nearest-Neighbor Parameters for GU Mismatches...." Biochemistry 1991, 30 11124-11132 A.E. Peritz, R. Kierzek, N. Sugimoto, D.H. Turner "Thermodynamic Study of Internal Loops in Oligoribonucleotides..." Biochemistry 1991, 30, 6428-6435

## 18.148 1.8.4\_epars.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_OLD_EPARS_

```

```

2 #define VIENNA_RNA_PACKAGE_OLD_EPARS_
39 #define K0 273.15
40 #ifdef INF
41 #undef INF
42 #endif
43 #define INF 1000000
44 #define NBPAIRS 7
45 #define NST 0 /* Energy for nonstandard stacked pairs */
46 #define DEF -50 /* Default terminal mismatch, used for I */
47 /* and any non_pairing bases */
48 #define NSM 0 /* terminal mismatch for non standard pairs */
49
50 PRIVATE double Tmeasure_184 = 37 + K0; /* temperature of param measurements */
51 PRIVATE double lxc37_184 = 107.856; /* parameter for logarithmic loop
52 energy extrapolation */
53
54 PRIVATE int stack37_184[NBPAIRS+1][NBPAIRS+1] =
55 /* CG GC GU UG AU UA */
56 { { INF, INF, INF, INF, INF, INF, INF },
57 { { INF, -240, -330, -210, -140, -210, -210, NST },
58 { { INF, -330, -340, -250, -150, -220, -240, NST },
59 { { INF, -210, -250, 130, -50, -140, -130, NST },
60 { { INF, -140, -150, -50, 30, -60, -100, NST },
61 { { INF, -210, -220, -140, -60, -110, -90, NST },
62 { { INF, -210, -240, -130, -100, -90, -130, NST },
63 { { INF, NST, NST, NST, NST, NST, NST } };
64
65 /* enthalpies (0.01*kcal/mol at 37 C) for stacked pairs */
66 /* different from mfold-2.3, which uses values from mfold-2.2 */
67 PRIVATE int enthalpies_184[NBPAIRS+1][NBPAIRS+1] =
68 /* CG GC GU UG AU UA */
69 { { INF, INF, INF, INF, INF, INF, INF },
70 { { INF, -1060, -1340, -1210, -560, -1050, -1040, NST },
71 { { INF, -1340, -1490, -1260, -830, -1140, -1240, NST },
72 { { INF, -1210, -1260, -1460, -1350, -880, -1280, NST },
73 { { INF, -560, -830, -1350, -930, -320, -700, NST },
74 { { INF, -1050, -1140, -880, -320, -940, -680, NST },
75 { { INF, -1040, -1240, -1280, -700, -680, -770, NST },
76 { { INF, NST, NST, NST, NST, NST, NST } };
77
78
79 /* old values are here just for comparison */
80 PRIVATE int oldhairpin37_184[31] = { /* from ViennaRNA 1.3 */
81 { INF, INF, INF, 410, 490, 440, 470, 500, 510, 520, 531,
82 542, 551, 560, 568, 575, 582, 589, 595, 601, 606,
83 611, 616, 621, 626, 630, 634, 638, 642, 646, 650 };
84
85 PRIVATE int hairpin37_184[31] = {
86 { INF, INF, INF, 570, 560, 560, 540, 590, 560, 640, 650,
87 660, 670, 678, 686, 694, 701, 707, 713, 719, 725,
88 730, 735, 740, 744, 749, 753, 757, 761, 765, 769 };
89
90 PRIVATE int oldbulge37_184[31] = {
91 { INF, 390, 310, 350, 420, 480, 500, 516, 531, 543, 555,
92 565, 574, 583, 591, 598, 605, 612, 618, 624, 630,
93 635, 640, 645, 649, 654, 658, 662, 666, 670, 673 };
94
95 PRIVATE int bulge37_184[31] = {
96 { INF, 380, 280, 320, 360, 400, 440, 459, 470, 480, 490,
97 500, 510, 519, 527, 534, 541, 548, 554, 560, 565,
98 571, 576, 580, 585, 589, 594, 598, 602, 605, 609 };
99
100 PRIVATE int oldinternal_loop37_184[31] = {
101 { INF, INF, 410, 510, 490, 530, 570, 587, 601, 614, 625,
102 635, 645, 653, 661, 669, 676, 682, 688, 694, 700,
103 705, 710, 715, 720, 724, 728, 732, 736, 740, 744 };
104
105 PRIVATE int internal_loop37_184[31] = {
106 { INF, INF, 410, 510, 170, 180, 200, 220, 230, 240, 250,
107 260, 270, 278, 286, 294, 301, 307, 313, 319, 325,
108 330, 335, 340, 345, 349, 353, 357, 361, 365, 369 };
109
110 /* terminal mismatches */
111 /* mismatch free energies for interior loops at 37C */
112 PRIVATE int mismatchI37_184[NBPAIRS+1][5][5] =
113 { /* @@ */
114 { {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0} },
115 { /* CG */
116 { {0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
117 { {0, 0, 0, -110, 0}, /* A@ AA AC AG AU */
118 { {0, 0, 0, 0, 0}, /* C@ CA CC CG CU */
119 { {0, -110, 0, 0, 0}, /* G@ GA GC GG GU */
120 { {0, 0, 0, 0, -70}}, /* U@ UA UC UG UU */
121 { /* GC */
122 { {0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
123 { {0, 0, 0, -110, 0}, /* A@ AA AC AG AU */
124 { {0, 0, 0, 0, 0}, /* C@ CA CC CG CU */

```

```

125 { 0, -110, 0, 0, 0}, /* G@ GA GC GG GU */
126 { 0, 0, 0, -70}, /* U@ UA UC UG UU */
127 { /* GU */
128 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
129 { 0, 70, 70, -40, 70}, /* A@ AA AC AG AU */
130 { 0, 70, 70, 70, 70}, /* C@ CA CC CG CU */
131 { 0, -40, 70, 70, 70}, /* G@ GA GC GG GU */
132 { 0, 70, 70, 70, 0}, /* U@ UA UC UG UU */
133 { /* UG */
134 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
135 { 0, 70, 70, -40, 70}, /* A@ AA AC AG AU */
136 { 0, 70, 70, 70, 70}, /* C@ CA CC CG CU */
137 { 0, -40, 70, 70, 70}, /* G@ GA GC GG GU */
138 { 0, 70, 70, 70, 0}, /* U@ UA UC UG UU */
139 { /* AU */
140 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
141 { 0, 70, 70, -40, 70}, /* A@ AA AC AG AU */
142 { 0, 70, 70, 70, 70}, /* C@ CA CC CG CU */
143 { 0, -40, 70, 70, 70}, /* G@ GA GC GG GU */
144 { 0, 70, 70, 70, 0}, /* U@ UA UC UG UU */
145 { /* UA */
146 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
147 { 0, 70, 70, -40, 70}, /* A@ AA AC AG AU */
148 { 0, 70, 70, 70, 70}, /* C@ CA CC CG CU */
149 { 0, -40, 70, 70, 70}, /* G@ GA GC GG GU */
150 { 0, 70, 70, 70, 0}, /* U@ UA UC UG UU */
151 { /* @@ */
152 { 90, 90, 90, 90, 90}, { 90, 90, 90, 90, -20}, { 90, 90, 90, 90, 90},
153 { 90, -20, 90, 90, 90}, { 90, 90, 90, 90, 20}
154 };
155
156 /* mismatch free energies for hairpins at 37C */
157 PRIVATE int mismatchH37_184[NBPAIRS+1][5][5] =
158 { /* @@ */
159 { {0,0,0,0,0}, {0,0,0,0,0}, {0,0,0,0,0}, {0,0,0,0,0}, {0,0,0,0,0}, /* CG */
160 { /* CG */
161 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
162 { -90, -150, -150, -140, -180}, /* A@ AA AC AG AU */
163 { -90, -100, -90, -290, -80}, /* C@ CA CC CG CU */
164 { -90, -220, -200, -160, -110}, /* G@ GA GC GG GU */
165 { -90, -170, -140, -180, -200}, /* U@ UA UC UG UU */
166 { /* GC */
167 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
168 { -70, -110, -150, -130, -210}, /* A@ AA AC AG AU */
169 { -70, -110, -70, -240, -50}, /* C@ CA CC CG CU */
170 { -70, -240, -290, -140, -120}, /* G@ GA GC GG GU */
171 { -70, -190, -100, -220, -150}, /* U@ UA UC UG UU */
172 { /* GU */
173 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
174 { 0, 20, -50, -30, -30}, /* A@ AA AC AG AU */
175 { 0, -10, -20, -150, -20}, /* C@ CA CC CG CU */
176 { 0, -90, -110, -30, 0}, /* G@ GA GC GG GU */
177 { 0, -30, -30, -40, -110}, /* U@ UA UC UG UU */
178 { /* UG */
179 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
180 { 0, -50, -30, -60, -50}, /* A@ AA AC AG AU */
181 { 0, -20, -10, -170, 0}, /* C@ CA CC CG CU */
182 { 0, -80, -120, -30, -70}, /* G@ GA GC GG GU */
183 { 0, -60, -10, -60, -80}, /* U@ UA UC UG UU */
184 { /* AU */
185 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
186 { 0, -30, -50, -30, -30}, /* A@ AA AC AG AU */
187 { 0, -10, -20, -150, -20}, /* C@ CA CC CG CU */
188 { 0, -110, -120, -20, 20}, /* G@ GA GC GG GU */
189 { 0, -30, -30, -60, -110}, /* U@ UA UC UG UU */
190 { /* UA */
191 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
192 { 0, -50, -30, -60, -50}, /* A@ AA AC AG AU */
193 { 0, -20, -10, -120, -0}, /* C@ CA CC CG CU */
194 { 0, -140, -120, -70, -20}, /* G@ GA GC GG GU */
195 { 0, -30, -10, -50, -80}, /* U@ UA UC UG UU */
196 { /* @@ */
197 { 0, 0, 0, 0, 0}, { 0, 0, 0, 0, 0}, { 0, 0, 0, 0, 0},
198 { 0, 0, 0, 0, 0}, { 0, 0, 0, 0, 0}
199 };
200
201 /* mismatch energies in multiloops */
202 PRIVATE int mismatchM37_184[NBPAIRS+1][5][5];
203
204 /* these are probably junk */
205 /* mismatch enthalpies for temperature scaling */
206 PRIVATE int mism_H_184[NBPAIRS+1][5][5] =
207 { /* no pair */
208 { {0,0,0,0,0}, {0,0,0,0,0}, {0,0,0,0,0}, {0,0,0,0,0}, {0,0,0,0,0}, /* CG */
209 { /* CG */
210 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
211 { DEF, -1030, -950, -1030, -1030}, /* A@ AA AC AG AU */

```

```

212 { DEF, -520, -450, -520, -670}, /* C@ CA CC CG CU */
213 { DEF, -940, -940, -940, -940}, /* G@ GA GC GG GU */
214 { DEF, -810, -740, -810, -860}, /* U@ UA UC UG UU */
215 { /* GC */
216 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
217 { DEF, -520, -880, -560, -880}, /* A@ AA AC AG AU */
218 { DEF, -720, -310, -310, -390}, /* C@ CA CC CG CU */
219 { DEF, -710, -740, -620, -740}, /* G@ GA GC GG GU */
220 { DEF, -500, -500, -500, -570}, /* U@ UA UC UG UU */
221 { /* GU */
222 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
223 { DEF, -430, -600, -600, -600}, /* A@ AA AC AG AU */
224 { DEF, -260, -240, -240, -240}, /* C@ CA CC CG CU */
225 { DEF, -340, -690, -690, -690}, /* G@ GA GC GG GU */
226 { DEF, -330, -330, -330, -330}, /* U@ UA UC UG UU */
227 { /* UG */
228 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
229 { DEF, -720, -790, -960, -810}, /* A@ AA AC AG AU */
230 { DEF, -480, -480, -360, -480}, /* C@ CA CC CG CU */
231 { DEF, -660, -810, -920, -810}, /* G@ GA GC GG GU */
232 { DEF, -550, -440, -550, -360}, /* U@ UA UC UG UU */
233 { /* AU */
234 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
235 { DEF, -430, -600, -600, -600}, /* A@ AA AC AG AU */
236 { DEF, -260, -240, -240, -240}, /* C@ CA CC CG CU */
237 { DEF, -340, -690, -690, -690}, /* G@ GA GC GG GU */
238 { DEF, -330, -330, -330, -330}, /* U@ UA UC UG UU */
239 { /* UA */
240 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
241 { DEF, -400, -630, -890, -590}, /* A@ AA AC AG AU */
242 { DEF, -430, -510, -200, -180}, /* C@ CA CC CG CU */
243 { DEF, -380, -680, -890, -680}, /* G@ GA GC GG GU */
244 { DEF, -280, -140, -280, -140}, /* U@ UA UC UG UU */
245 { /* nonstandard pair */
246 {DEF,DEF,DEF,DEF,DEF},{DEF,DEF,DEF,DEF,DEF},{DEF,DEF,DEF,DEF,DEF},
247 {DEF,DEF,DEF,DEF,DEF},{DEF,DEF,DEF,DEF,DEF}}
248 };
249
250 /* 5' dangling ends (unpaired base stacks on first paired base) */
251 PRIVATE int dangle5_37_184[NBPAIRS+1][5] =
252 {/*
253  @ A C G U */
254 { INF, INF, INF, INF, INF}, /* no pair */
255 { INF, -50, -30, -20, -10}, /* CG (stacks on C) */
256 { INF, -20, -30, -0, -0}, /* GC (stacks on G) */
257 { INF, -30, -30, -40, -20}, /* GU */
258 { INF, -30, -10, -20, -20}, /* UG */
259 { INF, -30, -30, -40, -20}, /* AU */
260 { 0, 0, 0, 0, 0} /* @ */
261 };
262
263 /* 3' dangling ends (unpaired base stacks on second paired base) */
264 PRIVATE int dangle3_37_184[NBPAIRS+1][5] =
265 {/*
266  @ A C G U */
267 { INF, INF, INF, INF, INF}, /* no pair */
268 { INF, -110, -40, -130, -60}, /* CG (stacks on G) */
269 { INF, -170, -80, -170, -120}, /* GC */
270 { INF, -70, -10, -70, -10}, /* GU */
271 { INF, -80, -50, -80, -60}, /* UG */
272 { INF, -80, -50, -80, -60}, /* AU */
273 { 0, 0, 0, 0, 0} /* @ */
274 };
275
276 /* enthalpies for temperature scaling */
277 PRIVATE int dangle3_H_184[NBPAIRS+1][5] =
278 {/*
279  @ A C G U */
280 { INF, INF, INF, INF, INF}, /* no pair */
281 { 0, -740, -280, -640, -360},
282 { 0, -900, -410, -860, -750},
283 { 0, -740, -240, -720, -490},
284 { 0, -490, -90, -550, -230},
285 { 0, -570, -70, -580, -220},
286 { 0, -490, -90, -550, -230},
287 { 0, 0, 0, 0, 0}
288
289 PRIVATE int dangle5_H_184[NBPAIRS+1][5] =
290 {/*
291  @ A C G U */
292 { INF, INF, INF, INF, INF}, /* no pair */
293 { 0, -240, 330, 80, -140},
294 { 0, -160, 70, -460, -40},
295 { 0, 160, 220, 70, 310},
296 { 0, -150, 510, 10, 100},
297 { 0, 160, 220, 70, 310},
298 { 0, -50, 690, -60, -60},
299 { 0, 0, 0, 0, 0}

```

```

299 };
300
301
302 /* constants for linearly destabilizing contributions for multi-loops
303   F = ML_closing + ML_intern*k + ML_BASE*u */
304 /* old versions erroneously used ML_intern*(k-1) */
305 PRIVATE int ML_BASE37_184 = 0;
306 PRIVATE int ML_closing37_184 = 340;
307 PRIVATE int ML_intern37_184 = 40;
308
309 /* Ninio-correction for asymmetric internal loops with branches n1 and n2 */
310 /*   ninio_energy = min{max_ninio, |n1-n2|*F_ninio[min{4.0, n1, n2}]} */
311 PRIVATE int MAX_NINIO_184 = 300; /* maximum correction */
312 PRIVATE int F_ninio37_184[5] = { 0, 40, 50, 20, 10 }; /* only F[2] used */
313
314 /* stabilizing contribution due to special hairpins of size 4 (tetraloops) */
315
316 PRIVATE char Tetraloops_184[1400] = /* place for up to 200 tetra loops */
317   "GGGGAC "
318   "GGUGAC "
319   "CGAAAG "
320   "GGAGAC "
321   "CGCAAG "
322   "GGAAAC "
323   "CGGAAG "
324   "CUUCGG "
325   "CGUGAG "
326   "CGAAGG "
327   "CUACGG "
328   "GGCAAC "
329   "CGCGAG "
330   "UGAGAG "
331   "CGAGAG "
332   "AGAAAU "
333   "CGUAAG "
334   "CUAACG "
335   "UGAAAG "
336   "GGAAGC "
337   "GGGAAC "
338   "UGAAAA "
339   "AGCAAU "
340   "AGUAAU "
341   "CGGGAG "
342   "AGUGAU "
343   "GGGCAC "
344   "GGGAGC "
345   "GUGAAC "
346   "UGGAAA "
347 ;
348
349 PRIVATE int TETRA_ENERGY37_184[200] = {
350   -300, -300, -300, -300, -300, -300, -300, -300, -250, -250, -250,
351   -250, -250, -200, -200, -200, -200, -200, -150, -150, -150, -150,
352   -150, -150, -150, -150, -150, -150};
353
354 PRIVATE int TETRA_ENTH37_184 = -400;
355
356 PRIVATE char Triloops_184[241] = "";
357
358 PRIVATE int Triloop_E37_184[40];
359
360 /* penalty for AU (or GU) terminating helix */
361 /* mismatches already contain these */
362 PRIVATE int TerminalAU_184 = 50;
363
364 /* penalty for forming a bi-molecular duplex */
365 PRIVATE int DuplexInit_184 = 410;
366
367 #endiff

```

## 18.149 ViennaRNA/params/1.8.4\_intloops.h File Reference

Free energy parameters for interior loop contributions needed by the parameter file conversion functions.

### 18.149.1 Detailed Description

Free energy parameters for interior loop contributions needed by the parameter file conversion functions.

## 18.150 1.8.4\_intloops.h

[Go to the documentation of this file.](#)

```
1 PRIVATE int int11_37_184[NBPAIRS+1][NBPAIRS+1][5][5] =
2 { /* noPair */ {{0}}, 
3 { /* noPair */ {{0}}, 
4 /* CG..CG */
5 {{ 110, 110, 110, 110, 110},
6 { 110, 110, 40, 40, 40},
7 { 110, 40, 40, 40, 40},
8 { 110, 40, 40, -140, 40},
9 { 110, 40, 40, 40, 40}
10 },
11 /* CG..GC */
12 {{ 110, 110, 110, 110, 110},
13 { 110, 40, -40, 40, 40},
14 { 110, 30, 50, 40, 50},
15 { 110, -10, 40, -170, 40},
16 { 110, 40, 0, 40, -30}
17 },
18 /* CG..GU */
19 {{ 110, 110, 110, 110, 110},
20 { 110, 110, 110, 110, 110},
21 { 110, 110, 110, 110, 110},
22 { 110, 110, 110, 110, 110}
23 },
24 /* CG..UG */
25 {{ 110, 110, 110, 110, 110},
26 { 110, 110, 110, 110, 110},
27 { 110, 110, 110, 110, 110},
28 { 110, 110, 110, -100, 110},
29 { 110, 110, 110, 110, 110}
30 },
31 /* CG..UG */
32 {{ 110, 110, 110, 110, 110},
33 { 110, 110, 110, 110, 110},
34 { 110, 110, 110, 110, 110},
35 { 110, 110, 110, -100, 110},
36 { 110, 110, 110, 110, 110}
37 },
38 /* CG..AU */
39 {{ 110, 110, 110, 110, 110},
40 { 110, 110, 110, 110, 110},
41 { 110, 110, 110, 110, 110},
42 { 110, 110, 110, -100, 110},
43 { 110, 110, 110, 110, 110}
44 },
45 /* CG..UA */
46 {{ 110, 110, 110, 110, 110},
47 { 110, 110, 110, 110, 110},
48 { 110, 110, 110, 110, 110},
49 { 110, 110, 110, -100, 110},
50 { 110, 110, 110, 110, 110}
51 },
52 /* CG..?? */
53 {{ 110, 110, 110, 110, 110},
54 { 110, 110, 110, 110, 110},
55 { 110, 110, 110, 110, 110},
56 { 110, 110, 110, 110, 110},
57 { 110, 110, 110, 110, 110}
58 },
59 },
60 { /* noPair */ {{0}},
61 /* GC..CG */
62 {{ 110, 110, 110, 110, 110},
63 { 110, 40, 30, -10, 40},
64 { 110, -40, 50, 40, 0},
65 { 110, 40, 40, -170, 40},
66 { 110, 40, 50, 40, -30}
67 },
68 /* GC..GC */
69 {{ 110, 110, 110, 110, 110},
70 { 110, 80, 40, 40, 40},
71 { 110, 40, 40, 40, 40},
72 { 110, 40, 40, -210, 40},
73 { 110, 40, 40, 40, -70}
74 },
75 /* GC..GU */
76 {{ 110, 110, 110, 110, 110},
77 { 110, 110, 110, 110, 110},
78 { 110, 110, 110, 110, 110},
79 { 110, 110, 110, -100, 110},
80 { 110, 110, 110, 110, 110}
81 },
82 /* GC..UG */
83 {{ 110, 110, 110, 110, 110},
84 { 110, 110, 110, 110, 110},
85 { 110, 110, 110, 110, 110},
86 { 110, 110, 110, -100, 110},
87 { 110, 110, 110, 110, 110}
88 },
```

```
89 /* GC..AU */
90 {{ 110, 110, 110, 110, 110},
91 { 110, 110, 110, 110, 110},
92 { 110, 110, 110, 110, 110},
93 { 110, 110, 110, -100, 110},
94 { 110, 110, 110, 110, 100}
95 },
96 /* GC..UA */
97 {{ 110, 110, 110, 110, 110},
98 { 110, 110, 110, 110, 110},
99 { 110, 110, 110, 110, 110},
100 { 110, 110, 110, -100, 110},
101 { 110, 110, 110, 110, 110}
102 },
103 /* GC..?? */
104 {{ 110, 110, 110, 110, 110},
105 { 110, 110, 110, 110, 110},
106 { 110, 110, 110, 110, 110},
107 { 110, 110, 110, 110, 110},
108 { 110, 110, 110, 110, 110}
109 },
110 },
111 { /* noPair */ {{0}},
112 /* GU..CG */
113 {{ 110, 110, 110, 110, 110},
114 { 110, 110, 110, 110, 110},
115 { 110, 110, 110, 110, 110},
116 { 110, 110, 110, -100, 110},
117 { 110, 110, 110, 110, 110}
118 },
119 /* GU..GC */
120 {{ 110, 110, 110, 110, 110},
121 { 110, 110, 110, 110, 110},
122 { 110, 110, 110, 110, 110},
123 { 110, 110, 110, -100, 110},
124 { 110, 110, 110, 110, 110}
125 },
126 /* GU..GU */
127 {{ 170, 170, 170, 170, 170},
128 { 170, 170, 170, 170, 170},
129 { 170, 170, 170, 170, 170},
130 { 170, 170, 170, -40, 170},
131 { 170, 170, 170, 170, 170}
132 },
133 /* GU..UG */
134 {{ 170, 170, 170, 170, 170},
135 { 170, 170, 170, 170, 170},
136 { 170, 170, 170, 170, 170},
137 { 170, 170, 170, -40, 170},
138 { 170, 170, 170, 170, 170}
139 },
140 /* GU..AU */
141 {{ 170, 170, 170, 170, 170},
142 { 170, 170, 170, 170, 170},
143 { 170, 170, 170, 170, 170},
144 { 170, 170, 170, -40, 170},
145 { 170, 170, 170, 170, 170}
146 },
147 /* GU..UA */
148 {{ 170, 170, 170, 170, 170},
149 { 170, 170, 170, 170, 170},
150 { 170, 170, 170, 170, 170},
151 { 170, 170, 170, -40, 170},
152 { 170, 170, 170, 170, 170}
153 },
154 /* GU..?? */
155 {{ 170, 170, 170, 170, 170},
156 { 170, 170, 170, 170, 170},
157 { 170, 170, 170, 170, 170},
158 { 170, 170, 170, 170, 170},
159 { 170, 170, 170, 170, 170}
160 },
161 },
162 { /* noPair */ {{0}},
163 /* UG..CG */
164 {{ 110, 110, 110, 110, 110},
165 { 110, 110, 110, 110, 110},
166 { 110, 110, 110, 110, 110},
167 { 110, 110, 110, -100, 110},
168 { 110, 110, 110, 110, 110}
169 },
170 /* UG..GC */
171 {{ 110, 110, 110, 110, 110},
172 { 110, 110, 110, 110, 110},
173 { 110, 110, 110, 110, 110},
174 { 110, 110, 110, -100, 110},
175 { 110, 110, 110, 110, 110}
```

```
176 },
177 /* UG..GU */
178 {{ 170, 170, 170, 170, 170},
179 { 170, 170, 170, 170, 170},
180 { 170, 170, 170, 170, 170},
181 { 170, 170, 170, -40, 170},
182 { 170, 170, 170, 170, 170}
183 },
184 /* UG..UG */
185 {{ 170, 170, 170, 170, 170},
186 { 170, 170, 170, 170, 170},
187 { 170, 170, 170, 170, 170},
188 { 170, 170, 170, -40, 170},
189 { 170, 170, 170, 170, 170}
190 },
191 /* UG..AU */
192 {{ 170, 170, 170, 170, 170},
193 { 170, 170, 170, 170, 170},
194 { 170, 170, 170, 170, 170},
195 { 170, 170, 170, -40, 170},
196 { 170, 170, 170, 170, 170}
197 },
198 /* UG..UA */
199 {{ 170, 170, 170, 170, 170},
200 { 170, 170, 170, 170, 170},
201 { 170, 170, 170, 170, 170},
202 { 170, 170, 170, -40, 170},
203 { 170, 170, 170, 170, 170}
204 },
205 /* UG..?? */
206 {{ 170, 170, 170, 170, 170},
207 { 170, 170, 170, 170, 170},
208 { 170, 170, 170, 170, 170},
209 { 170, 170, 170, 170, 170},
210 { 170, 170, 170, 170, 170}
211 },
212 },
213 { /* noPair */ {{0}}},
214 /* AU..CG */
215 {{ 110, 110, 110, 110, 110},
216 { 110, 110, 110, 110, 110},
217 { 110, 110, 110, 110, 110},
218 { 110, 110, 110, -100, 110},
219 { 110, 110, 110, 110, 110}
220 },
221 /* AU..GC */
222 {{ 110, 110, 110, 110, 110},
223 { 110, 110, 110, 110, 110},
224 { 110, 110, 110, 110, 110},
225 { 110, 110, 110, -100, 110},
226 { 110, 110, 110, 110, 100}
227 },
228 /* AU..GU */
229 {{ 170, 170, 170, 170, 170},
230 { 170, 170, 170, 170, 170},
231 { 170, 170, 170, 170, 170},
232 { 170, 170, 170, -40, 170},
233 { 170, 170, 170, 170, 170}
234 },
235 /* AU..UG */
236 {{ 170, 170, 170, 170, 170},
237 { 170, 170, 170, 170, 170},
238 { 170, 170, 170, 170, 170},
239 { 170, 170, 170, -40, 170},
240 { 170, 170, 170, 170, 170}
241 },
242 /* AU..AU */
243 {{ 170, 170, 170, 170, 170},
244 { 170, 170, 170, 170, 170},
245 { 170, 170, 170, 170, 170},
246 { 170, 170, 170, -40, 170},
247 { 170, 170, 170, 170, 120}
248 },
249 /* AU..UA */
250 {{ 170, 170, 170, 170, 170},
251 { 170, 170, 170, 170, 170},
252 { 170, 170, 170, 170, 170},
253 { 170, 170, 170, -40, 170},
254 { 170, 170, 170, 170, 150}
255 },
256 /* AU..?? */
257 {{ 170, 170, 170, 170, 170},
258 { 170, 170, 170, 170, 170},
259 { 170, 170, 170, 170, 170},
260 { 170, 170, 170, 170, 170},
261 { 170, 170, 170, 170, 170}
262 }
```

```
263 },
264 { /* noPair */ {{0}},
265 /* UA..CG */
266 {{ 110, 110, 110, 110, 110},
267 { 110, 110, 110, 110, 110},
268 { 110, 110, 110, 110, 110},
269 { 110, 110, 110, -100, 110},
270 { 110, 110, 110, 110, 110}
271 },
272 /* UA..GC */
273 {{ 110, 110, 110, 110, 110},
274 { 110, 110, 110, 110, 110},
275 { 110, 110, 110, 110, 110},
276 { 110, 110, 110, -100, 110},
277 { 110, 110, 110, 110, 110}
278 },
279 /* UA..GU */
280 {{ 170, 170, 170, 170, 170},
281 { 170, 170, 170, 170, 170},
282 { 170, 170, 170, 170, 170},
283 { 170, 170, 170, -40, 170},
284 { 170, 170, 170, 170, 170}
285 },
286 /* UA..UG */
287 {{ 170, 170, 170, 170, 170},
288 { 170, 170, 170, 170, 170},
289 { 170, 170, 170, 170, 170},
290 { 170, 170, 170, -40, 170},
291 { 170, 170, 170, 170, 170}
292 },
293 /* UA..AU */
294 {{ 170, 170, 170, 170, 170},
295 { 170, 170, 170, 170, 170},
296 { 170, 170, 170, 170, 170},
297 { 170, 170, 170, -40, 170},
298 { 170, 170, 170, 170, 150}
299 },
300 /* UA..UA */
301 {{ 170, 170, 170, 170, 170},
302 { 170, 170, 170, 170, 170},
303 { 170, 170, 170, 170, 170},
304 { 170, 170, 170, -40, 170},
305 { 170, 170, 170, 170, 180}
306 },
307 /* UA..?? */
308 {{ 170, 170, 170, 170, 170},
309 { 170, 170, 170, 170, 170},
310 { 170, 170, 170, 170, 170},
311 { 170, 170, 170, 170, 170},
312 { 170, 170, 170, 170, 170}
313 },
314 },
315 { /* noPair */ {{0}},
316 /* ??..CG */
317 {{ 110, 110, 110, 110, 110},
318 { 110, 110, 110, 110, 110},
319 { 110, 110, 110, 110, 110},
320 { 110, 110, 110, 110, 110},
321 { 110, 110, 110, 110, 110}
322 },
323 /* ??..GC */
324 {{ 110, 110, 110, 110, 110},
325 { 110, 110, 110, 110, 110},
326 { 110, 110, 110, 110, 110},
327 { 110, 110, 110, 110, 110},
328 { 110, 110, 110, 110, 110}
329 },
330 /* ??..GU */
331 {{ 170, 170, 170, 170, 170},
332 { 170, 170, 170, 170, 170},
333 { 170, 170, 170, 170, 170},
334 { 170, 170, 170, 170, 170},
335 { 170, 170, 170, 170, 170}
336 },
337 /* ??..UG */
338 {{ 170, 170, 170, 170, 170},
339 { 170, 170, 170, 170, 170},
340 { 170, 170, 170, 170, 170},
341 { 170, 170, 170, 170, 170},
342 { 170, 170, 170, 170, 170}
343 },
344 /* ??..AU */
345 {{ 170, 170, 170, 170, 170},
346 { 170, 170, 170, 170, 170},
347 { 170, 170, 170, 170, 170},
348 { 170, 170, 170, 170, 170},
349 { 170, 170, 170, 170, 170}
```

```
350 },
351 /* ??..UA */
352 {{ 170, 170, 170, 170, 170},
353 { 170, 170, 170, 170, 170},
354 { 170, 170, 170, 170, 170},
355 { 170, 170, 170, 170, 170},
356 { 170, 170, 170, 170, 170}
357 },
358 /* ??..?? */
359 {{ 170, 170, 170, 170, 170},
360 { 170, 170, 170, 170, 170},
361 { 170, 170, 170, 170, 170},
362 { 170, 170, 170, 170, 170},
363 { 170, 170, 170, 170, 170}
364 }
365 }
366 };
367
368 PRIVATE int int11_H_184[NBPAIRS+1][NBPAIRS+1][5][5] =
369 /* GC..GC */
370 { /* noPair */ {{0}}},
371 { /* noPair */ {0}},
372 { { 0, 0, 0, 0, 0},
373 { 0, 0, 0, 0, 0},
374 { 0, 0, 0, 0, 0},
375 { 0, 0, 0, 0, 0},
376 { 0, 0, 0, 0, 0},
377 /* GC..CG */
378 { { 0, 0, 0, 0, 0},
379 { 0, 0, 0, 0, 0},
380 { 0, 0, 0, 0, 0},
381 { 0, 0, 0, 0, 0},
382 { 0, 0, 0, 0, 0},
383 /* GC..GU */
384 { { 0, 0, 0, 0, 0},
385 { 0, 0, 0, 0, 0},
386 { 0, 0, 0, 0, 0},
387 { 0, 0, 0, 0, 0},
388 { 0, 0, 0, 0, 0},
389 /* GC..UG */
390 { { 0, 0, 0, 0, 0},
391 { 0, 0, 0, 0, 0},
392 { 0, 0, 0, 0, 0},
393 { 0, 0, 0, 0, 0},
394 { 0, 0, 0, 0, 0},
395 /* GC..AU */
396 { { 0, 0, 0, 0, 0},
397 { 0, 0, 0, 0, 0},
398 { 0, 0, 0, 0, 0},
399 { 0, 0, 0, 0, 0},
400 { 0, 0, 0, 0, 0},
401 /* GC..UA */
402 { { 0, 0, 0, 0, 0},
403 { 0, 0, 0, 0, 0},
404 { 0, 0, 0, 0, 0},
405 { 0, 0, 0, 0, 0},
406 { 0, 0, 0, 0, 0},
407 /* GC.. @ */
408 { { 0, 0, 0, 0, 0},
409 { 0, 0, 0, 0, 0},
410 { 0, 0, 0, 0, 0},
411 { 0, 0, 0, 0, 0},
412 { 0, 0, 0, 0, 0}},
413 /* CG..GC */
414 { /* noPair */ {0}},
415 { { 0, 0, 0, 0, 0},
416 { 0, 0, 0, 0, 0},
417 { 0, 0, 0, 0, 0},
418 { 0, 0, 0, 0, 0},
419 { 0, 0, 0, 0, 0},
420 /* CG..CG */
421 { { 0, 0, 0, 0, 0},
422 { 0, 0, 0, 0, 0},
423 { 0, 0, 0, 0, 0},
424 { 0, 0, 0, 0, 0},
425 { 0, 0, 0, 0, 0},
426 /* CG..GU */
427 { { 0, 0, 0, 0, 0},
428 { 0, 0, 0, 0, 0},
429 { 0, 0, 0, 0, 0},
430 { 0, 0, 0, 0, 0},
431 { 0, 0, 0, 0, 0},
432 /* CG..UG */
433 { { 0, 0, 0, 0, 0},
434 { 0, 0, 0, 0, 0},
435 { 0, 0, 0, 0, 0},
436 { 0, 0, 0, 0, 0},
```

```
437 { 0, 0, 0, 0, 0},  
438 /* CG..AU */  
439 { { 0, 0, 0, 0, 0},  
440 { 0, 0, 0, 0, 01},  
441 { 0, 0, 0, 0, 0},  
442 { 0, 0, 0, 0, 01},  
443 { 0, 0, 0, 0, 0}),  
444 /* CG..UA */  
445 { { 0, 0, 0, 0, 0},  
446 { 0, 0, 0, 0, 01},  
447 { 0, 0, 0, 0, 0},  
448 { 0, 0, 0, 0, 0},  
449 { 0, 0, 0, 0, 01}),  
450 /* CG.. @ */  
451 { { 0, 0, 0, 0, 0},  
452 { 0, 0, 0, 0, 0},  
453 { 0, 0, 0, 0, 0},  
454 { 0, 0, 0, 0, 0},  
455 { 0, 0, 0, 0, 0}),  
456 /* GU..GC */  
457 { /* noPair */ {{0}},  
458 { { 0, 0, 0, 0, 0},  
459 { 0, 0, 0, 0, 01},  
460 { 0, 0, 0, 0, 0},  
461 { 0, 0, 0, 0, 01},  
462 { 0, 0, 0, 0, 0}),  
463 /* GU..CG */  
464 { { 0, 0, 0, 0, 0},  
465 { 0, 0, 0, 0, 01},  
466 { 0, 0, 0, 0, 0},  
467 { 0, 0, 0, 0, 01},  
468 { 0, 0, 0, 0, 0}),  
469 /* GU..GU */  
470 { { 0, 0, 0, 0, 0},  
471 { 0, 0, 0, 0, 01},  
472 { 0, 0, 0, 0, 0},  
473 { 0, 0, 0, 0, 01},  
474 { 0, 0, 0, 0, 0}),  
475 /* GU..UG */  
476 { { 0, 0, 0, 0, 0},  
477 { 0, 0, 0, 0, 01},  
478 { 0, 0, 0, 0, 0},  
479 { 0, 0, 0, 0, 01},  
480 { 0, 0, 0, 0, 0}),  
481 /* GU..AU */  
482 { { 0, 0, 0, 0, 0},  
483 { 0, 0, 0, 0, 0},  
484 { 0, 0, 0, 0, 0},  
485 { 0, 0, 0, 0, 01},  
486 { 0, 0, 0, 0, 0}),  
487 /* GU..UA */  
488 { { 0, 0, 0, 0, 0},  
489 { 0, 0, 0, 0, 0},  
490 { 0, 0, 0, 0, 01},  
491 { 0, 0, 0, 0, 0},  
492 { 0, 0, 0, 0, 0}),  
493 /* GU.. @ */  
494 { { 0, 0, 0, 0, 0},  
495 { 0, 0, 0, 0, 0},  
496 { 0, 0, 0, 0, 01},  
497 { 0, 0, 0, 0, 0},  
498 { 0, 0, 0, 0, 0}),  
499 /* UG..GC */  
500 { /* noPair */ {{0}},  
501 { { 0, 0, 0, 0, 0},  
502 { 0, 0, 0, 0, 01},  
503 { 0, 0, 0, 0, 0},  
504 { 0, 0, 0, 0, 01},  
505 { 0, 0, 0, 0, 0}),  
506 /* UG..CG */  
507 { { 0, 0, 0, 0, 0},  
508 { 0, 0, 0, 0, 01},  
509 { 0, 0, 0, 0, 0},  
510 { 0, 0, 0, 0, 01},  
511 { 0, 0, 0, 0, 0}),  
512 /* UG..GU */  
513 { { 0, 0, 0, 0, 0},  
514 { 0, 0, 0, 0, 01},  
515 { 0, 0, 0, 0, 0},  
516 { 0, 0, 0, 0, 01},  
517 { 0, 0, 0, 0, 0}),  
518 /* UG..UG */  
519 { { 0, 0, 0, 0, 0},  
520 { 0, 0, 0, 0, 0},  
521 { 0, 0, 0, 0, 0},  
522 { 0, 0, 0, 0, 01},  
523 { 0, 0, 0, 0, 0}),
```

```
524 /* UG..AU */
525 { { 0, 0, 0, 0, 0}, 
526 { 0, 0, 0, 0, 0}, 
527 { 0, 0, 0, 0, 01}, 
528 { 0, 0, 0, 0, 0}, 
529 { 0, 0, 0, 0, 01}, 
530 /* UG..UA */
531 { { 0, 0, 0, 0, 0}, 
532 { 0, 0, 0, 0, 0}, 
533 { 0, 0, 0, 0, 01}, 
534 { 0, 0, 0, 0, 0}, 
535 { 0, 0, 0, 0, 01}, 
536 /* UG.. @ */
537 { { 0, 0, 0, 0, 0}, 
538 { 0, 0, 0, 0, 0}, 
539 { 0, 0, 0, 0, 0}, 
540 { 0, 0, 0, 0, 0}, 
541 { 0, 0, 0, 0, 0}}, 
542 /* AU..GC */
543 { /* noPair */ {{0}}, 
544 { { 0, 0, 0, 0, 0}, 
545 { 0, 0, 0, 0, 0}, 
546 { 0, 0, 0, 0, 01}, 
547 { 0, 0, 0, 0, 0}, 
548 { 0, 0, 0, 0, 0}, 
549 /* AU..CG */
550 { { 0, 0, 0, 0, 0}, 
551 { 0, 0, 0, 0, 0}, 
552 { 0, 0, 0, 0, 01}, 
553 { 0, 0, 0, 0, 0}, 
554 { 0, 0, 0, 0, 0}}, 
555 /* AU..GU */
556 { { 0, 0, 0, 0, 0}, 
557 { 0, 0, 0, 0, 0}, 
558 { 0, 0, 0, 0, 01}, 
559 { 0, 0, 0, 0, 0}, 
560 { 0, 0, 0, 0, 0}}, 
561 /* AU..UG */
562 { { 0, 0, 0, 0, 0}, 
563 { 0, 0, 0, 0, 0}, 
564 { 0, 0, 0, 0, 01}, 
565 { 0, 0, 0, 0, 0}, 
566 { 0, 0, 0, 0, 0}}, 
567 /* AU..AU */
568 { { 0, 0, 0, 0, 0}, 
569 { 0, 0, 0, 0, 0}, 
570 { 0, 0, 0, 0, 0}, 
571 { 0, 0, 0, 0, 0}, 
572 { 0, 0, 0, 0, 01}}, 
573 /* AU..UA */
574 { { 0, 0, 0, 0, 0}, 
575 { 0, 0, 0, 0, 0}, 
576 { 0, 0, 0, 0, 0}, 
577 { 0, 0, 0, 0, 01}, 
578 { 0, 0, 0, 0, 0}}, 
579 /* AU.. @ */
580 { { 0, 0, 0, 0, 0}, 
581 { 0, 0, 0, 0, 01}, 
582 { 0, 0, 0, 0, 0}, 
583 { 0, 0, 0, 0, 01}, 
584 { 0, 0, 0, 0, 0}}, 
585 /* UA..GC */
586 { /* noPair */ {{0}}, 
587 { { 0, 0, 0, 0, 0}, 
588 { 0, 0, 0, 0, 0}, 
589 { 0, 0, 0, 0, 01}, 
590 { 0, 0, 0, 0, 0}, 
591 { 0, 0, 0, 0, 0}}, 
592 /* UA..CG */
593 { { 0, 0, 0, 0, 0}, 
594 { 0, 0, 0, 0, 0}, 
595 { 0, 0, 0, 0, 01}, 
596 { 0, 0, 0, 0, 0}, 
597 { 0, 0, 0, 0, 01}}, 
598 /* UA..GU */
599 { { 0, 0, 0, 0, 0}, 
600 { 0, 0, 0, 0, 0}, 
601 { 0, 0, 0, 0, 0}, 
602 { 0, 0, 0, 0, 0}, 
603 { 0, 0, 0, 0, 0}}, 
604 /* UA..UG */
605 { { 0, 0, 0, 0, 0}, 
606 { 0, 0, 0, 0, 0}, 
607 { 0, 0, 0, 0, 0}, 
608 { 0, 0, 0, 0, 01}, 
609 { 0, 0, 0, 0, 01}}, 
610 /* UA..AU */
```



```

        400}, { 550, 400, 400, 400, 400}},  

692 /* CG.C..CG */  

693 {{ 550, 550, 550, 550, 550}, { 550, 230, 220, 400, 220}, { 550, 220, 250, 400, 220}, { 550, 400, 400, 400,  

    400}, { 550, 250, 190, 400, 220}},  

694 /* CG.G..CG */  

695 {{ 550, 550, 550, 550, 550}, { 550, 170, 400, 80, 400}, { 550, 400, 400, 400}, { 550, 80, 400, 220,  

    400}, { 550, 400, 400, 400, 400}},  

696 /* CG.U..CG */  

697 {{ 550, 550, 550, 550, 550}, { 550, 400, 400, 400, 400}, { 550, 400, 220, 400, 150}, { 550, 400, 400, 400,  

    400}, { 550, 400, 170, 400, 120}}}  

698 },  

699 {  

700 /* CG.@..UG */  

701 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

    550}, { 550, 550, 550, 550, 550}},  

702 /* CG.A..UG */  

703 {{ 550, 550, 550, 550, 550}, { 550, 320, 300, 240, 480}, { 550, 290, 250, 240, 480}, { 550, 180, 140, 120,  

    480}, { 550, 480, 480, 480, 480}},  

704 /* CG.C..UG */  

705 {{ 550, 550, 550, 550, 550}, { 550, 310, 300, 480, 300}, { 550, 300, 330, 480, 300}, { 550, 480, 480, 480,  

    480}, { 550, 330, 270, 480, 300}},  

706 /* CG.G..UG */  

707 {{ 550, 550, 550, 550, 550}, { 550, 250, 480, 160, 480}, { 550, 480, 480, 480}, { 550, 160, 480, 300,  

    480}, { 550, 480, 480, 480, 480}},  

708 /* CG.U..UG */  

709 {{ 550, 550, 550, 550, 550}, { 550, 480, 480, 480, 480}, { 550, 480, 300, 480, 210}, { 550, 480, 480, 480,  

    480}, { 550, 480, 250, 480, 200}}}  

710 },  

711 {  

712 /* CG.@..GU */  

713 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

    550}, { 550, 550, 550, 550, 550}},  

714 /* CG.A..GU */  

715 {{ 550, 550, 550, 550, 550}, { 550, 320, 300, 240, 480}, { 550, 290, 250, 240, 480}, { 550, 180, 140, 120,  

    480}, { 550, 480, 480, 480, 480}},  

716 /* CG.C..GU */  

717 {{ 550, 550, 550, 550, 550}, { 550, 310, 300, 480, 300}, { 550, 300, 330, 480, 300}, { 550, 480, 480, 480,  

    480}, { 550, 330, 270, 480, 300}},  

718 /* CG.G..GU */  

719 {{ 550, 550, 550, 550, 550}, { 550, 250, 480, 160, 480}, { 550, 480, 480, 480}, { 550, 160, 480, 300,  

    480}, { 550, 480, 480, 480, 480}},  

720 /* CG.U..GU */  

721 {{ 550, 550, 550, 550, 550}, { 550, 480, 480, 480, 480}, { 550, 480, 300, 480, 210}, { 550, 480, 480, 480,  

    480}, { 550, 480, 250, 480, 200}}}  

722 },  

723 {  

724 /* CG.@..UA */  

725 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

    550}, { 550, 550, 550, 550, 550}},  

726 /* CG.A..UA */  

727 {{ 550, 550, 550, 550, 550}, { 550, 320, 300, 240, 480}, { 550, 290, 250, 240, 480}, { 550, 180, 140, 120,  

    480}, { 550, 480, 480, 480, 480}},  

728 /* CG.C..UA */  

729 {{ 550, 550, 550, 550, 550}, { 550, 310, 300, 480, 300}, { 550, 300, 330, 480, 300}, { 550, 480, 480, 480,  

    480}, { 550, 330, 270, 480, 300}},  

730 /* CG.G..UA */  

731 {{ 550, 550, 550, 550, 550}, { 550, 250, 480, 160, 480}, { 550, 480, 480, 480}, { 550, 160, 480, 300,  

    480}, { 550, 480, 480, 480, 480}},  

732 /* CG.U..UA */  

733 {{ 550, 550, 550, 550, 550}, { 550, 480, 480, 480, 480}, { 550, 480, 300, 480, 210}, { 550, 480, 480, 480,  

    480}, { 550, 480, 250, 480, 200}}}  

734 },  

735 {  

736 /* CG.@..AU */  

737 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

    550}, { 550, 550, 550, 550, 550}},  

738 /* CG.A..AU */  

739 {{ 550, 550, 550, 550, 550}, { 550, 320, 300, 240, 480}, { 550, 290, 250, 240, 480}, { 550, 180, 140, 120,  

    480}, { 550, 480, 480, 480, 480}},  

740 /* CG.C..AU */  

741 {{ 550, 550, 550, 550, 550}, { 550, 310, 300, 480, 300}, { 550, 300, 330, 480, 300}, { 550, 480, 480, 480,  

    480}, { 550, 330, 270, 480, 300}},  

742 /* CG.G..AU */  

743 {{ 550, 550, 550, 550, 550}, { 550, 250, 480, 160, 480}, { 550, 480, 480, 480, 480}, { 550, 160, 480, 300,  

    480}, { 550, 480, 480, 480, 480}},  

744 /* CG.U..AU */  

745 {{ 550, 550, 550, 550, 550}, { 550, 480, 480, 480, 480}, { 550, 480, 300, 480, 210}, { 550, 480, 480, 480,  

    480}, { 550, 480, 250, 480, 200}}}  

746 },  

747 {  

748 /* CG.@..?? */  

749 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

    550}, { 550, 550, 550, 550, 550}},  

750 /* CG.A..?? */  

751 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

    550}, { 550, 550, 550, 550, 550}},  

752 /* CG.C..?? */

```



```

815 {{ 550, 550, 550, 550, 550}, { 550, 310, 300, 480, 300}, { 550, 300, 330, 480, 300}, { 550, 480, 480, 480,
816 /* GC.G..UA */
817 {{ 550, 550, 550, 550, 550}, { 550, 250, 480, 160, 480}, { 550, 480, 480, 480}, { 550, 160, 480, 300,
818 /* GC.U..UA */
819 {{ 550, 550, 550, 550, 550}, { 550, 480, 480, 480, 480}, { 550, 480, 300, 480, 210}, { 550, 480, 480, 480,
820 }, { 550, 480, 250, 480, 200}}
821 {
822 /* GC.G..AU */
823 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
824 /* GC.A..AU */
825 {{ 550, 550, 550, 550, 550}, { 550, 320, 300, 240, 480}, { 550, 290, 250, 240, 480}, { 550, 180, 140, 120,
826 /* GC.C..AU */
827 {{ 550, 550, 550, 550, 550}, { 550, 310, 300, 480, 300}, { 550, 300, 330, 480, 300}, { 550, 480, 480, 480,
828 /* GC.G..AU */
829 {{ 550, 550, 550, 550, 550}, { 550, 250, 480, 160, 480}, { 550, 480, 480, 480}, { 550, 160, 480, 300,
830 /* GC.U..AU */
831 {{ 550, 550, 550, 550, 550}, { 550, 480, 480, 480, 480}, { 550, 480, 300, 480, 210}, { 550, 480, 480, 480,
832 }, { 550, 480, 250, 480, 200}}
833 {
834 /* GC.G..?? */
835 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
836 /* GC.A..?? */
837 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
838 /* GC.C..?? */
839 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
840 /* GC.G..?? */
841 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
842 /* GC.U..?? */
843 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
844 },
845 },
846 { /* noPair */ {{ 0 }}},
847 {
848 /* GU.G..GC */
849 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
850 /* GU.A..GC */
851 {{ 550, 550, 550, 550, 550}, { 550, 320, 300, 240, 480}, { 550, 290, 250, 240, 480}, { 550, 180, 140, 120,
852 /* GU.C..GC */
853 {{ 550, 550, 550, 550, 550}, { 550, 310, 300, 480, 300}, { 550, 300, 330, 480, 300}, { 550, 480, 480,
854 /* GU.G..GC */
855 {{ 550, 550, 550, 550, 550}, { 550, 250, 480, 160, 480}, { 550, 480, 480, 480}, { 550, 160, 480, 300,
856 /* GU.U..GC */
857 {{ 550, 550, 550, 550, 550}, { 550, 480, 480, 480, 480}, { 550, 480, 300, 480, 210}, { 550, 480, 480, 480,
858 },
859 {
860 /* GU.G..CG */
861 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
862 /* GU.A..CG */
863 {{ 550, 550, 550, 550, 550}, { 550, 320, 300, 240, 480}, { 550, 290, 250, 240, 480}, { 550, 180, 140, 120,
864 /* GU.C..CG */
865 {{ 550, 550, 550, 550, 550}, { 550, 310, 300, 480, 300}, { 550, 300, 330, 480, 300}, { 550, 480, 480, 480,
866 /* GU.G..CG */
867 {{ 550, 550, 550, 550, 550}, { 550, 250, 480, 160, 480}, { 550, 480, 480, 480}, { 550, 160, 480, 300,
868 /* GU.U..CG */
869 {{ 550, 550, 550, 550, 550}, { 550, 480, 480, 480, 480}, { 550, 480, 300, 480, 210}, { 550, 480, 480, 480,
870 },
871 {
872 /* GU.G..UG */
873 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
874 /* GU.A..UG */
875 {{ 550, 550, 550, 550, 550}, { 550, 390, 370, 310, 550}, { 550, 360, 320, 310, 550}, { 550, 250, 210, 190,
876 /* GU.C..UG */

```

```

877 {{ 550, 550, 550, 550, 550}, { 550, 380, 370, 550, 370}, { 550, 370, 400, 550, 370}, { 550, 550, 550, 550,
550}, { 550, 400, 340, 550, 370}},  

878 /* GU.G..UG */  

879 {{ 550, 550, 550, 550, 550}, { 550, 320, 550, 230, 550}, { 550, 550, 550, 550, 550}, { 550, 230, 550, 370,
550}, { 550, 550, 550, 550, 550}},  

880 /* GU.U..UG */  

881 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 370, 550, 280}, { 550, 550, 550, 550,
550}, { 550, 550, 320, 550, 270}}}  

882 },  

883 {  

884 /* GU.Q..GU */  

885 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},  

886 /* GU.A..GU */  

887 {{ 550, 550, 550, 550, 550}, { 550, 390, 370, 310, 550}, { 550, 360, 320, 310, 550}, { 550, 250, 210, 190,
550}, { 550, 550, 550, 550, 550}},  

888 /* GU.C..GU */  

889 {{ 550, 550, 550, 550, 550}, { 550, 380, 370, 550, 370}, { 550, 370, 400, 550, 370}, { 550, 550, 550, 550,
550}, { 550, 400, 340, 550, 370}},  

890 /* GU.G..GU */  

891 {{ 550, 550, 550, 550, 550}, { 550, 320, 550, 230, 550}, { 550, 550, 550, 550, 550}, { 550, 230, 550, 370,
550}, { 550, 550, 550, 550, 550}},  

892 /* GU.U..GU */  

893 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 370, 550, 280}, { 550, 550, 550, 550,
550}, { 550, 550, 320, 550, 270}}}  

894 },  

895 {  

896 /* GU.Q..UA */  

897 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},  

898 /* GU.A..UA */  

899 {{ 550, 550, 550, 550, 550}, { 550, 390, 370, 310, 550}, { 550, 360, 320, 310, 550}, { 550, 250, 210, 190,
550}, { 550, 550, 550, 550, 550}},  

900 /* GU.C..UA */  

901 {{ 550, 550, 550, 550, 550}, { 550, 380, 370, 550, 370}, { 550, 370, 400, 550, 370}, { 550, 550, 550, 550,
550}, { 550, 400, 340, 550, 370}},  

902 /* GU.G..UA */  

903 {{ 550, 550, 550, 550, 550}, { 550, 320, 550, 230, 550}, { 550, 550, 550, 550, 550}, { 550, 230, 550, 370,
550}, { 550, 550, 550, 550, 550}},  

904 /* GU.U..UA */  

905 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 370, 550, 280}, { 550, 550, 550, 550,
550}, { 550, 550, 320, 550, 270}}}  

906 },  

907 {  

908 /* GU.Q..AU */  

909 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},  

910 /* GU.A..AU */  

911 {{ 550, 550, 550, 550, 550}, { 550, 390, 370, 310, 550}, { 550, 360, 320, 310, 550}, { 550, 250, 210, 190,
550}, { 550, 550, 550, 550, 550}},  

912 /* GU.C..AU */  

913 {{ 550, 550, 550, 550, 550}, { 550, 380, 370, 550, 370}, { 550, 370, 400, 550, 370}, { 550, 550, 550, 550,
550}, { 550, 400, 340, 550, 370}},  

914 /* GU.G..AU */  

915 {{ 550, 550, 550, 550, 550}, { 550, 320, 550, 230, 550}, { 550, 550, 550, 550, 550}, { 550, 230, 550, 370,
550}, { 550, 550, 550, 550, 550}},  

916 /* GU.U..AU */  

917 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 370, 550, 280}, { 550, 550, 550, 550,
550}, { 550, 550, 320, 550, 270}}}  

918 },  

919 {  

920 /* GU.Q..?? */  

921 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},  

922 /* GU.A..?? */  

923 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},  

924 /* GU.C..?? */  

925 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},  

926 /* GU.G..?? */  

927 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},  

928 /* GU.U..?? */  

929 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,
550}, { 550, 550, 550, 550, 550}}}  

930 },  

931 },  

932 { /* noPair */ {{0}}},  

933 {  

934 /* UG.Q..GC */  

935 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},  

936 /* UG.A..GC */  

937 {{ 550, 550, 550, 550, 550}, { 550, 320, 300, 240, 480}, { 550, 290, 250, 240, 480}, { 550, 180, 140, 120,
480}, { 550, 480, 480, 480, 480}}},  

938 /* UG.C..GC */

```

```

939 {{ 550, 550, 550, 550, 550}, { 550, 310, 300, 480, 300}, { 550, 300, 330, 480, 300}, { 550, 480, 480, 480,
480}, { 550, 330, 270, 480, 300}},  

940 /* UG.G..GC */  

941 {{ 550, 550, 550, 550, 550}, { 550, 250, 480, 160, 480}, { 550, 480, 480, 480}, { 550, 160, 480, 300,
480}, { 550, 480, 480, 480}},  

942 /* UG.U..GC */  

943 {{ 550, 550, 550, 550, 550}, { 550, 480, 480, 480, 480}, { 550, 480, 300, 480, 210}, { 550, 480, 480, 480,
480}, { 550, 480, 250, 480, 200}}}  

944 },  

945 {  

946 /* UG.Q..CG */  

947 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},  

948 /* UG.A..CG */  

949 {{ 550, 550, 550, 550, 550}, { 550, 320, 300, 240, 480}, { 550, 290, 250, 240, 480}, { 550, 180, 140, 120,
480}, { 550, 480, 480, 480, 480}},  

950 /* UG.C..CG */  

951 {{ 550, 550, 550, 550, 550}, { 550, 310, 300, 480, 300}, { 550, 300, 330, 480, 300}, { 550, 480, 480, 480,
480}, { 550, 330, 270, 480, 300}},  

952 /* UG.G..CG */  

953 {{ 550, 550, 550, 550, 550}, { 550, 250, 480, 160, 480}, { 550, 480, 480, 480}, { 550, 160, 480, 300,
480}, { 550, 480, 480, 480, 480}},  

954 /* UG.U..CG */  

955 {{ 550, 550, 550, 550, 550}, { 550, 480, 480, 480, 480}, { 550, 480, 300, 480, 210}, { 550, 480, 480, 480,
480}, { 550, 480, 250, 480, 200}}}  

956 },  

957 {  

958 /* UG.Q..UG */  

959 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},  

960 /* UG.A..UG */  

961 {{ 550, 550, 550, 550, 550}, { 550, 390, 370, 310, 550}, { 550, 360, 320, 310, 550}, { 550, 250, 210, 190,
550}, { 550, 550, 550, 550, 550}},  

962 /* UG.C..UG */  

963 {{ 550, 550, 550, 550, 550}, { 550, 380, 370, 550, 370}, { 550, 370, 400, 550, 370}, { 550, 550, 550,
550}, { 550, 400, 340, 550, 370}},  

964 /* UG.G..UG */  

965 {{ 550, 550, 550, 550, 550}, { 550, 320, 550, 230, 550}, { 550, 550, 550, 550, 550}, { 550, 230, 550, 370,
550}, { 550, 550, 550, 550, 550}},  

966 /* UG.U..UG */  

967 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 370, 550, 280}, { 550, 550, 550, 550,
550}, { 550, 550, 320, 550, 270}}}  

968 },  

969 {  

970 /* UG.Q..GU */  

971 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},  

972 /* UG.A..GU */  

973 {{ 550, 550, 550, 550, 550}, { 550, 390, 370, 310, 550}, { 550, 360, 320, 310, 550}, { 550, 250, 210, 190,
550}, { 550, 550, 550, 550, 550}},  

974 /* UG.C..GU */  

975 {{ 550, 550, 550, 550, 550}, { 550, 380, 370, 550, 370}, { 550, 370, 400, 550, 370}, { 550, 550, 550,
550}, { 550, 400, 340, 550, 370}},  

976 /* UG.G..GU */  

977 {{ 550, 550, 550, 550, 550}, { 550, 320, 550, 230, 550}, { 550, 550, 550, 550, 550}, { 550, 230, 550, 370,
550}, { 550, 550, 550, 550, 550}},  

978 /* UG.U..GU */  

979 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 370, 550, 280}, { 550, 550, 550, 550,
550}, { 550, 550, 320, 550, 270}}}  

980 },  

981 {  

982 /* UG.Q..UA */  

983 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},  

984 /* UG.A..UA */  

985 {{ 550, 550, 550, 550, 550}, { 550, 390, 370, 310, 550}, { 550, 360, 320, 310, 550}, { 550, 250, 210, 190,
550}, { 550, 550, 550, 550, 550}},  

986 /* UG.C..UA */  

987 {{ 550, 550, 550, 550, 550}, { 550, 380, 370, 550, 370}, { 550, 370, 400, 550, 370}, { 550, 550, 550,
550}, { 550, 400, 340, 550, 370}},  

988 /* UG.G..UA */  

989 {{ 550, 550, 550, 550, 550}, { 550, 320, 550, 230, 550}, { 550, 550, 550, 550, 550}, { 550, 230, 550, 370,
550}, { 550, 550, 550, 550, 550}},  

990 /* UG.U..UA */  

991 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 370, 550, 280}, { 550, 550, 550, 550,
550}, { 550, 550, 320, 550, 270}}}  

992 },  

993 {  

994 /* UG.Q..AU */  

995 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},  

996 /* UG.A..AU */  

997 {{ 550, 550, 550, 550, 550}, { 550, 390, 370, 310, 550}, { 550, 360, 320, 310, 550}, { 550, 250, 210, 190,
550}, { 550, 550, 550, 550, 550}},  

998 /* UG.C..AU */  

999 {{ 550, 550, 550, 550, 550}, { 550, 380, 370, 550, 370}, { 550, 370, 400, 550, 370}, { 550, 550, 550,
550}, { 550, 400, 340, 550, 370}},
```

```

1000 /* UG.G..AU */
1001 {{ 550, 550, 550, 550}, { 550, 320, 550, 230, 550}, { 550, 550, 550, 550}, { 550, 230, 550, 370,
550}, { 550, 550, 550, 550}, },
1002 /* UG.U..AU */
1003 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 370, 550, 280}, { 550, 550, 550, 550},
550}, { 550, 550, 320, 550, 270}},
1004 },
1005 {
1006 /* UG.@..?? */
1007 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1008 /* UG.A..?? */
1009 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1010 /* UG.C..?? */
1011 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1012 /* UG.G..?? */
1013 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1014 /* UG.U..?? */
1015 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1016 },
1017 },
1018 { /* noPair */ {{{0}}},
1019 {
1020 /* AU.@..GC */
1021 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1022 /* AU.A..GC */
1023 {{ 550, 550, 550, 550}, { 550, 320, 300, 240, 480}, { 550, 290, 250, 240, 480}, { 550, 180, 140, 120,
480}, { 550, 480, 480, 480, 480}}, ,
1024 /* AU.C..GC */
1025 {{ 550, 550, 550, 550}, { 550, 310, 300, 480, 300}, { 550, 300, 330, 480, 300}, { 550, 480, 480, 480,
480}, { 550, 330, 270, 480, 300}}, ,
1026 /* AU.G..GC */
1027 {{ 550, 550, 550, 550}, { 550, 250, 480, 160, 480}, { 550, 480, 480, 480, 480}, { 550, 160, 480, 300,
480}, { 550, 480, 480, 480, 480}}, ,
1028 /* AU.U..GC */
1029 {{ 550, 550, 550, 550}, { 550, 480, 480, 480, 480}, { 550, 480, 300, 480, 210}, { 550, 480, 480, 480,
480}, { 550, 480, 250, 480, 200}} ,
1030 },
1031 {
1032 /* AU.@..CG */
1033 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1034 /* AU.A..CG */
1035 {{ 550, 550, 550, 550}, { 550, 320, 300, 240, 480}, { 550, 290, 250, 240, 480}, { 550, 180, 140, 120,
480}, { 550, 480, 480, 480, 480}}, ,
1036 /* AU.C..CG */
1037 {{ 550, 550, 550, 550}, { 550, 310, 300, 480, 300}, { 550, 300, 330, 480, 300}, { 550, 480, 480, 480,
480}, { 550, 330, 270, 480, 300}}, ,
1038 /* AU.G..CG */
1039 {{ 550, 550, 550, 550}, { 550, 250, 480, 160, 480}, { 550, 480, 480, 480, 480}, { 550, 160, 480, 300,
480}, { 550, 480, 480, 480, 480}}, ,
1040 /* AU.U..CG */
1041 {{ 550, 550, 550, 550}, { 550, 480, 480, 480, 480}, { 550, 480, 300, 480, 210}, { 550, 480, 480, 480,
480}, { 550, 480, 250, 480, 200}} ,
1042 },
1043 {
1044 /* AU.@..UG */
1045 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1046 /* AU.A..UG */
1047 {{ 550, 550, 550, 550}, { 550, 390, 370, 310, 550}, { 550, 360, 320, 310, 550}, { 550, 250, 210, 190,
550}, { 550, 550, 550, 550}, },
1048 /* AU.C..UG */
1049 {{ 550, 550, 550, 550}, { 550, 380, 370, 550, 370}, { 550, 370, 400, 550, 370}, { 550, 550, 550, 550},
550}, { 550, 400, 340, 550, 370}}, ,
1050 /* AU.G..UG */
1051 {{ 550, 550, 550, 550}, { 550, 320, 550, 230, 550}, { 550, 550, 550, 550, 550}, { 550, 230, 550, 370,
550}, { 550, 550, 550, 550, 550}, },
1052 /* AU.U..UG */
1053 {{ 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 370, 550, 280}, { 550, 550, 550, 550,
550}, { 550, 550, 320, 550, 270}} ,
1054 },
1055 {
1056 /* AU.@..GU */
1057 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1058 /* AU.A..GU */
1059 {{ 550, 550, 550, 550}, { 550, 390, 370, 310, 550}, { 550, 360, 320, 310, 550}, { 550, 250, 210, 190,
550}, { 550, 550, 550, 550}, },
1060 /* AU.C..GU */
1061 {{ 550, 550, 550, 550}, { 550, 380, 370, 550, 370}, { 550, 370, 400, 550, 370}, { 550, 550, 550, 550},
550}, { 550, 400, 340, 550, 370}}, ,

```

```

1062 /* AU.G..GU */
1063 {{ 550, 550, 550, 550}, { 550, 320, 550, 230, 550}, { 550, 550, 550, 550}, { 550, 230, 550, 370,
550}, { 550, 550, 550, 550}, },
1064 /* AU.U..GU */
1065 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 370, 550, 280}, { 550, 550, 550, 550},
550}, { 550, 550, 320, 550, 270} }
1066 },
1067 {
1068 /* AU.Q..UA */
1069 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1070 /* AU.A..UA */
1071 {{ 550, 550, 550, 550}, { 550, 390, 370, 310, 550}, { 550, 360, 320, 310, 550}, { 550, 250, 210, 190,
550}, { 550, 550, 550, 550}, },
1072 /* AU.C..UA */
1073 {{ 550, 550, 550, 550}, { 550, 380, 370, 550, 370}, { 550, 370, 400, 550, 370}, { 550, 550, 550, 550},
550}, { 550, 400, 340, 550, 370} },
1074 /* AU.G..UA */
1075 {{ 550, 550, 550, 550}, { 550, 320, 550, 230, 550}, { 550, 550, 550, 550}, { 550, 230, 550, 370,
550}, { 550, 550, 550, 550}, },
1076 /* AU.U..UA */
1077 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 370, 550, 280}, { 550, 550, 550, 550},
550}, { 550, 550, 320, 550, 270} }
1078 },
1079 {
1080 /* AU.Q..AU */
1081 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1082 /* AU.A..AU */
1083 {{ 550, 550, 550, 550}, { 550, 390, 370, 310, 550}, { 550, 360, 320, 310, 550}, { 550, 250, 210, 190,
550}, { 550, 550, 550, 550}, },
1084 /* AU.C..AU */
1085 {{ 550, 550, 550, 550}, { 550, 380, 370, 550, 370}, { 550, 370, 400, 550, 370}, { 550, 550, 550, 550},
550}, { 550, 400, 340, 550, 370} },
1086 /* AU.G..AU */
1087 {{ 550, 550, 550, 550}, { 550, 320, 550, 230, 550}, { 550, 550, 550, 550}, { 550, 230, 550, 370,
550}, { 550, 550, 550, 550}, },
1088 /* AU.U..AU */
1089 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 370, 550, 280}, { 550, 550, 550, 550},
550}, { 550, 550, 320, 550, 270} }
1090 },
1091 {
1092 /* AU.Q..?? */
1093 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1094 /* AU.A..?? */
1095 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1096 /* AU.C..?? */
1097 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1098 /* AU.G..?? */
1099 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1100 /* AU.U..?? */
1101 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1102 },
1103 },
1104 /* noPair */ {{0}}, ,
1105 {
1106 /* UA.Q..GC */
1107 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1108 /* UA.A..GC */
1109 {{ 550, 550, 550, 550}, { 550, 320, 300, 240, 480}, { 550, 290, 250, 240, 480}, { 550, 180, 140, 120,
480}, { 550, 480, 480, 480, 480} },
1110 /* UA.C..GC */
1111 {{ 550, 550, 550, 550}, { 550, 310, 300, 480, 300}, { 550, 300, 330, 480, 300}, { 550, 480, 480, 480,
480}, { 550, 330, 270, 480, 300} },
1112 /* UA.G..GC */
1113 {{ 550, 550, 550, 550}, { 550, 250, 480, 160, 480}, { 550, 480, 480, 480, 480}, { 550, 160, 480, 300,
480}, { 550, 480, 480, 480, 480} },
1114 /* UA.U..GC */
1115 {{ 550, 550, 550, 550}, { 550, 480, 480, 480, 480}, { 550, 480, 300, 480, 210}, { 550, 480, 480, 480,
480}, { 550, 480, 250, 480, 200} }
1116 },
1117 {
1118 /* UA.Q..CG */
1119 {{ 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550}, { 550, 550, 550, 550},
550}, { 550, 550, 550, 550}, },
1120 /* UA.A..CG */
1121 {{ 550, 550, 550, 550}, { 550, 320, 300, 240, 480}, { 550, 290, 250, 240, 480}, { 550, 180, 140, 120,
480}, { 550, 480, 480, 480, 480} },
1122 /* UA.C..CG */
1123 {{ 550, 550, 550, 550}, { 550, 310, 300, 480, 300}, { 550, 300, 330, 480, 300}, { 550, 480, 480, 480,
480}, { 550, 330, 270, 480, 300} },

```

```

1124 /* UA.G..CG */
1125 {{ 550, 550, 550, 550}, { 550, 250, 480, 160, 480}, { 550, 480, 480, 480, 480}, { 550, 160, 480, 300,
480}, { 550, 480, 480, 480, 480}},
1126 /* UA.U..CG */
1127 {{ 550, 550, 550, 550, 550}, { 550, 480, 480, 480, 480}, { 550, 480, 300, 480, 210}, { 550, 480, 480,
480}, { 550, 480, 250, 480, 200}},
1128 },
1129 {
1130 /* UA.Q..UG */
1131 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},
1132 /* UA.A..UG */
1133 {{ 550, 550, 550, 550, 550}, { 550, 390, 370, 310, 550}, { 550, 360, 320, 310, 550}, { 550, 250, 210,
190, 550}, { 550, 550, 550, 550, 550}},
1134 /* UA.C..UG */
1135 {{ 550, 550, 550, 550, 550}, { 550, 380, 370, 350, 550}, { 550, 370, 400, 550, 370}, { 550, 550, 550,
550}, { 550, 400, 340, 550, 370}},
1136 /* UA.G..UG */
1137 {{ 550, 550, 550, 550, 550}, { 550, 320, 550, 230, 550}, { 550, 550, 550, 550, 550}, { 550, 230, 550,
370, 550}, { 550, 550, 550, 550, 550}},
1138 /* UA.U..UG */
1139 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 370, 550, 280}, { 550, 550, 550,
550}, { 550, 550, 320, 550, 270}},
1140 },
1141 {
1142 /* UA.Q..GU */
1143 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},
1144 /* UA.A..GU */
1145 {{ 550, 550, 550, 550, 550}, { 550, 390, 370, 310, 550}, { 550, 360, 320, 310, 550}, { 550, 250, 210,
190, 550}, { 550, 550, 550, 550, 550}},
1146 /* UA.C..GU */
1147 {{ 550, 550, 550, 550, 550}, { 550, 380, 370, 350, 550}, { 550, 370, 400, 550, 370}, { 550, 550, 550,
550}, { 550, 400, 340, 550, 370}},
1148 /* UA.G..GU */
1149 {{ 550, 550, 550, 550, 550}, { 550, 320, 550, 230, 550}, { 550, 550, 550, 550, 550}, { 550, 230, 550,
370, 550}, { 550, 550, 550, 550, 550}},
1150 /* UA.U..GU */
1151 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 370, 550, 280}, { 550, 550, 550,
550}, { 550, 550, 320, 550, 270}},
1152 },
1153 {
1154 /* UA.Q..UA */
1155 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},
1156 /* UA.A..UA */
1157 {{ 550, 550, 550, 550, 550}, { 550, 390, 370, 310, 550}, { 550, 360, 320, 310, 550}, { 550, 250, 210,
190, 550}, { 550, 550, 550, 550, 550}},
1158 /* UA.C..UA */
1159 {{ 550, 550, 550, 550, 550}, { 550, 380, 370, 350, 550}, { 550, 370, 400, 550, 370}, { 550, 550, 550,
550}, { 550, 400, 340, 550, 370}},
1160 /* UA.G..UA */
1161 {{ 550, 550, 550, 550, 550}, { 550, 320, 550, 230, 550}, { 550, 550, 550, 550, 550}, { 550, 230, 550,
370, 550}, { 550, 550, 550, 550, 550}},
1162 /* UA.U..UA */
1163 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 370, 550, 280}, { 550, 550, 550,
550}, { 550, 550, 320, 550, 270}},
1164 },
1165 {
1166 /* UA.Q..AU */
1167 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},
1168 /* UA.A..AU */
1169 {{ 550, 550, 550, 550, 550}, { 550, 390, 370, 310, 550}, { 550, 360, 320, 310, 550}, { 550, 250, 210,
190, 550}, { 550, 550, 550, 550, 550}},
1170 /* UA.C..AU */
1171 {{ 550, 550, 550, 550, 550}, { 550, 380, 370, 350, 550}, { 550, 370, 400, 550, 370}, { 550, 550, 550,
550}, { 550, 400, 340, 550, 370}},
1172 /* UA.G..AU */
1173 {{ 550, 550, 550, 550, 550}, { 550, 320, 550, 230, 550}, { 550, 550, 550, 550, 550}, { 550, 230, 550,
370, 550}, { 550, 550, 550, 550, 550}},
1174 /* UA.U..AU */
1175 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 370, 550, 280}, { 550, 550, 550,
550}, { 550, 550, 320, 550, 270}},
1176 },
1177 {
1178 /* UA.Q..?? */
1179 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},
1180 /* UA.A..?? */
1181 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},
1182 /* UA.C..?? */
1183 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
550}, { 550, 550, 550, 550, 550}},
1184 /* UA.G..?? */
1185 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550,
550}, { 550, 550, 550, 550, 550}}

```

```

      550}, { 550, 550, 550, 550, 550}},  

1186 /* UA.U..?? */  

1187 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}}}  

1188 }  

1189 },  

1190 { /* noPair */ {{{0}}},  

1191 {  

1192 /* ??@..GC */  

1193 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1194 /* ??A..GC */  

1195 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1196 /* ??C..GC */  

1197 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1198 /* ??G..GC */  

1199 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1200 /* ??U..GC */  

1201 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}}}  

1202 },  

1203 {  

1204 /* ??@..CG */  

1205 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1206 /* ??A..CG */  

1207 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1208 /* ??C..CG */  

1209 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1210 /* ??G..CG */  

1211 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1212 /* ??U..CG */  

1213 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}}}  

1214 },  

1215 {  

1216 /* ??@..UG */  

1217 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1218 /* ??A..UG */  

1219 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1220 /* ??C..UG */  

1221 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1222 /* ??G..UG */  

1223 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1224 /* ??U..UG */  

1225 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}}}  

1226 },  

1227 {  

1228 /* ??@..GU */  

1229 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1230 /* ??A..GU */  

1231 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1232 /* ??C..GU */  

1233 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1234 /* ??G..GU */  

1235 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1236 /* ??U..GU */  

1237 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}}}  

1238 },  

1239 {  

1240 /* ??@..UA */  

1241 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1242 /* ??A..UA */  

1243 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1244 /* ??C..UA */  

1245 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}},  

1246 /* ??G..UA */  

1247 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550},  

     { 550, 550, 550, 550, 550}}}

```

```

      550}, { 550, 550, 550, 550, 550}},  

1248 /* ??..U..UA */  

1249 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

      550}, { 550, 550, 550, 550, 550}}  

1250 },  

1251 {  

1252 /* ??..AU */  

1253 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

      550}, { 550, 550, 550, 550, 550}},  

1254 /* ??..A..AU */  

1255 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

      550}, { 550, 550, 550, 550, 550}},  

1256 /* ??..C..AU */  

1257 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

      550}, { 550, 550, 550, 550, 550}},  

1258 /* ??..G..AU */  

1259 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

      550}, { 550, 550, 550, 550, 550}},  

1260 /* ??..U..AU */  

1261 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

      550}, { 550, 550, 550, 550, 550}}  

1262 },  

1263 {  

1264 /* ??..@..?? */  

1265 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

      550}, { 550, 550, 550, 550, 550}},  

1266 /* ??..A..?? */  

1267 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

      550}, { 550, 550, 550, 550, 550}},  

1268 /* ??..C..?? */  

1269 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

      550}, { 550, 550, 550, 550, 550}},  

1270 /* ??..G..?? */  

1271 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

      550}, { 550, 550, 550, 550, 550}},  

1272 /* ??..U..?? */  

1273 {{ 550, 550, 550, 550, 550}, { 550, 550, 550, 550, 550}, { 550, 550, 550, 550,  

      550}, { 550, 550, 550, 550, 550}}  

1274 }  

1275 }  

1276 };  

1277  

1278 PRIVATE int int21_H_184[NBPAIRS+1][NBPAIRS+1][5][5][5] =  

1279 { /* noPair */ {{{0}}},  

1280 { /* noPair */ {{{0}}},  

1281 {  

1282 /* CG..@..CG */  

1283 {{ 0, 0, 0, 0, 0}, { DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}, {  

      DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}},  

1284 /* CG.A..CG */  

1285 {{ DEF, -1029, -949, -1029, -1029}, {-1079, -2058, -1978, -2058, -2058}, {-569, -1548, -1468, -1548, -1548}, {-  

      989, -1968, -1888, -1968, -1968}, {-859, -1838, -1758, -1838, -1838}},  

1286 /* CG.C..CG */  

1287 {{ DEF, -519, -449, -519, -669}, {-999, -1468, -1398, -1468, -1618}, {-499, -968, -898, -968, -1118}, {-  

      989, -1458, -1388, -1458, -1608}, {-789, -1258, -1188, -1258, -1408}},  

1288 /* CG.G..CG */  

1289 {{ DEF, -939, -939, -939, -939}, {-1079, -1968, -1968, -1968, -1968}, {-569, -1458, -1458, -1458, -1458}, {-  

      989, -1878, -1878, -1878, -1878}, {-859, -1748, -1748, -1748, -1748}},  

1290 /* CG.U..CG */  

1291 {{ DEF, -809, -739, -809, -859}, {-1079, -1838, -1768, -1888}, {-719, -1478, -1408, -1478, -1528}, {-  

      989, -1748, -1678, -1748, -1798}, {-909, -1668, -1598, -1668, -1718}}}  

1292 },  

1293 {  

1294 /* CG..@..GC */  

1295 {{ 0, 0, 0, 0, 0}, { DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}, {  

      DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}},  

1296 /* CG.A..GC */  

1297 {{ DEF, -1029, -949, -1029, -1029}, {-569, -1548, -1468, -1548, -1548}, {-769, -1748, -1668, -1748, -1748}, {-  

      759, -1738, -1658, -1738, -1738}, {-549, -1528, -1448, -1528, -1528}},  

1298 /* CG.C..GC */  

1299 {{ DEF, -519, -449, -519, -669}, {-929, -1398, -1328, -1398, -1548}, {-359, -828, -758, -828, -978}, {-  

      789, -1258, -1188, -1258, -1408}, {-549, -1018, -948, -1018, -1168}},  

1300 /* CG.G..GC */  

1301 {{ DEF, -939, -939, -939, -939}, {-609, -1498, -1498, -1498, -1498}, {-359, -1248, -1248, -1248, -1248}, {-  

      669, -1558, -1558, -1558, -1558}, {-549, -1438, -1438, -1438, -1438}},  

1302 /* CG.U..GC */  

1303 {{ DEF, -809, -739, -809, -859}, {-929, -1688, -1618, -1688, -1738}, {-439, -1198, -1128, -1198, -1248}, {-  

      789, -1548, -1478, -1548, -1598}, {-619, -1378, -1308, -1378, -1428}}}  

1304 },  

1305 {  

1306 /* CG..@..GU */  

1307 {{ 0, 0, 0, 0, 0}, { DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}, {  

      DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}},  

1308 /* CG.A..GU */  

1309 {{ DEF, -1029, -949, -1029, -1029}, {-479, -1458, -1378, -1458, -1458}, {-309, -1288, -1208, -1288, -1288}, {-  

      389, -1368, -1288, -1368, -1368}, {-379, -1358, -1278, -1358, -1358}},  

1310 /* CG.C..GU */}

```

```

1311 {{ DEF, -519, -449, -519, -669}, { -649, -1118, -1048, -1118, -1268}, { -289, -758, -688, -758, -908}, {  

    -739, -1208, -1138, -1208, -1358}, { -379, -848, -778, -848, -998}},  

1312 /* CG.G..GU */  

1313 {{ DEF, -939, -939, -939, -939}, { -649, -1538, -1538, -1538}, { -289, -1178, -1178, -1178}, {  

    -739, -1628, -1628, -1628}, { -379, -1268, -1268, -1268}, { -379, -1268, -1268, -1268}},  

1314 /* CG.U..GU */  

1315 {{ DEF, -809, -739, -809, -859}, { -649, -1408, -1338, -1408, -1458}, { -289, -1048, -978, -1048, -1098}, {  

    -739, -1498, -1428, -1548}, { -379, -1138, -1068, -1138, -1188}}}  

1316 },  

1317 {  

1318 /* CG.Q..UG */  

1319 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {  

    DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}},  

1320 /* CG.A..UG */  

1321 {{ DEF, -1029, -949, -1029, -1029}, { -769, -1748, -1668, -1748}, { -529, -1508, -1428, -1508}, {  

    -709, -1688, -1608, -1688}, { -599, -1578, -1498, -1578}, { -599, -1578, -1578}},  

1322 /* CG.C..UG */  

1323 {{ DEF, -519, -449, -519, -669}, { -839, -1308, -1238, -1308, -1458}, { -529, -998, -928, -998, -1148}, {  

    -859, -1328, -1258, -1328, -1478}, { -489, -958, -888, -958, -1108}},  

1324 /* CG.G..UG */  

1325 {{ DEF, -939, -939, -939, -939}, { -1009, -1898, -1898, -1898}, { -409, -1298, -1298, -1298}, {  

    -969, -1858, -1858, -1858}, { -599, -1488, -1488, -1488}, { -599, -1488, -1488, -1488}},  

1326 /* CG.U..UG */  

1327 {{ DEF, -809, -739, -809, -859}, { -859, -1618, -1548, -1618, -1668}, { -529, -1288, -1218, -1288, -1338}, {  

    -859, -1618, -1548, -1618, -1668}, { -409, -1168, -1098, -1168, -1218}}}  

1328 },  

1329 {  

1330 /* CG.Q..AU */  

1331 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {  

    DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}},  

1332 /* CG.A..AU */  

1333 {{ DEF, -1029, -949, -1029, -1029}, { -479, -1458, -1378, -1458}, { -309, -1288, -1208, -1288, -1288}, {  

    -389, -1368, -1288, -1368}, { -379, -1358, -1278, -1358}, { -379, -1358, -1358}},  

1334 /* CG.C..AU */  

1335 {{ DEF, -519, -449, -519, -669}, { -649, -1118, -1048, -1118, -1268}, { -289, -758, -688, -758, -908}, {  

    -739, -1208, -1138, -1208, -1358}, { -379, -848, -778, -848, -998}},  

1336 /* CG.G..AU */  

1337 {{ DEF, -939, -939, -939, -939}, { -649, -1538, -1538, -1538}, { -289, -1178, -1178, -1178}, {  

    -739, -1628, -1628, -1628}, { -379, -1268, -1268, -1268}, { -379, -1268, -1268, -1268}},  

1338 /* CG.U..AU */  

1339 {{ DEF, -809, -739, -809, -859}, { -649, -1408, -1338, -1408, -1458}, { -289, -1048, -978, -1048, -1098}, {  

    -739, -1498, -1428, -1548}, { -379, -1138, -1068, -1138, -1188}}}  

1340 },  

1341 {  

1342 /* CG.Q..UA */  

1343 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {  

    DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}},  

1344 /* CG.A..UA */  

1345 {{ DEF, -1029, -949, -1029, -1029}, { -449, -1428, -1348, -1428}, { -479, -1458, -1378, -1458}, {  

    -429, -1408, -1328, -1408}, { -329, -1308, -1228, -1308}, { -329, -1308, -1308}},  

1346 /* CG.C..UA */  

1347 {{ DEF, -519, -449, -519, -669}, { -679, -1148, -1078, -1148, -1298}, { -559, -1028, -958, -1028, -1178}, {  

    -729, -1198, -1128, -1198, -1348}, { -189, -658, -588, -658, -808}},  

1348 /* CG.G..UA */  

1349 {{ DEF, -939, -939, -939, -939}, { -939, -1828, -1828, -1828}, { -249, -1138, -1138, -1138, -1138}, {  

    -939, -1828, -1828, -1828}, { -329, -1218, -1218, -1218, -1218}},  

1350 /* CG.U..UA */  

1351 {{ DEF, -809, -739, -809, -859}, { -639, -1398, -1328, -1398, -1448}, { -229, -988, -918, -988, -1038}, {  

    -729, -1488, -1418, -1488, -1538}, { -190, -949, -879, -949, -999}}}  

1352 },  

1353 {  

1354 /* CG.Q..@ */  

1355 {{ DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {  

    DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}},  

1356 /* CG.A..@ */  

1357 {{ -100, -1079, -999, -1079, -1079}, { -100, -1079, -999, -1079, -1079}, { -100, -1079, -999, -1079, -1079}, {  

    -100, -1079, -999, -1079, -1079}, { -100, -1079, -999, -1079, -1079}},  

1358 /* CG.C..@ */  

1359 {{ -100, -569, -499, -569, -719}, { -100, -569, -499, -569, -719}, { -100, -569, -499, -569, -719}, {  

    -100, -569, -499, -569, -719}, { -100, -569, -499, -569, -719}},  

1360 /* CG.G..@ */  

1361 {{ -100, -989, -989, -989, -989}, { -100, -989, -989, -989, -989}, { -100, -989, -989, -989, -989}, {  

    -100, -989, -989, -989, -989}, { -100, -989, -989, -989, -989}},  

1362 /* CG.U..@ */  

1363 {{ -100, -859, -789, -859, -909}, { -100, -859, -789, -859, -909}, { -100, -859, -789, -859, -909}, {  

    -100, -859, -789, -859, -909}, { -100, -859, -789, -859, -909}}}  

1364 },  

1365 },  

1366 /* noPair */ {{0}}},  

1367 {  

1368 /* GC.Q..CG */  

1369 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {  

    DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}},  

1370 /* GC.A..CG */  

1371 {{ DEF, -519, -879, -559, -879}, { -1079, -1548, -1908, -1588, -1908}, { -569, -1038, -1398, -1078, -1398}, {  

    -989, -1458, -1818, -1498, -1818}, { -859, -1328, -1688, -1368, -1688}},  

1372 /* GC.C..CG */

```

```

1373 {{ DEF, -719, -309, -309, -389}, { -999, -1668, -1258, -1338}, { -499, -1168, -758, -758, -838}, {
    -989, -1658, -1248, -1248, -1328}, { -789, -1458, -1048, -1048, -1128}}, ,
1374 /* GC.G..CG */
1375 {{ DEF, -709, -739, -619, -739}, { -1079, -1738, -1768, -1648, -1768}, { -569, -1228, -1258, -1138, -1258}, {
    -989, -1648, -1678, -1558, -1678}, { -859, -1518, -1548, -1428, -1548}}, ,
1376 /* GC.U..CG */
1377 {{ DEF, -499, -499, -499, -569}, { -1079, -1528, -1528, -1598}, { -719, -1168, -1168, -1168, -1238}, {
    -989, -1438, -1438, -1438, -1508}, { -909, -1358, -1358, -1358, -1428}}}
1378 },
1379 {
1380 /* GC.@..GC */
1381 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {
    DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}}, ,
1382 /* GC.A..GC */
1383 {{ DEF, -519, -879, -559, -879}, { -569, -1038, -1398, -1078, -1398}, { -769, -1238, -1598, -1278, -1598}, {
    -759, -1228, -1588, -1268, -1588}, { -549, -1018, -1378, -1058, -1378}}, ,
1384 /* GC.C..GC */
1385 {{ DEF, -719, -309, -309, -389}, { -929, -1598, -1188, -1188, -1268}, { -359, -1028, -618, -618, -698}, {
    -789, -1458, -1048, -1048, -1128}, { -549, -1218, -808, -808, -888}}, ,
1386 /* GC.G..GC */
1387 {{ DEF, -709, -739, -619, -739}, { -609, -1268, -1298, -1178, -1298}, { -359, -1018, -1048, -928, -1048}, {
    -669, -1328, -1358, -1238, -1358}, { -549, -1208, -1238, -1118, -1238}}, ,
1388 /* GC.U..GC */
1389 {{ DEF, -499, -499, -499, -569}, { -929, -1378, -1378, -1378, -1448}, { -439, -888, -888, -888, -958}, {
    -789, -1238, -1238, -1238, -1308}, { -619, -1068, -1068, -1068, -1138}}}
1390 },
1391 {
1392 /* GC.@..GU */
1393 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {
    DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}}, ,
1394 /* GC.A..GU */
1395 {{ DEF, -519, -879, -559, -879}, { -479, -948, -1308, -988, -1308}, { -309, -778, -1138, -818, -1138}, {
    -389, -858, -1218, -898, -1218}, { -379, -848, -1208, -888, -1208}}, ,
1396 /* GC.C..GU */
1397 {{ DEF, -719, -309, -309, -389}, { -649, -1318, -908, -908, -988}, { -289, -958, -548, -548, -628}, {
    -739, -1408, -998, -998, -1078}, { -379, -1048, -638, -638, -718}}, ,
1398 /* GC.G..GU */
1399 {{ DEF, -709, -739, -619, -739}, { -649, -1308, -1338, -1218, -1338}, { -289, -948, -978, -858, -978}, {
    -739, -1398, -1428, -1308, -1428}, { -379, -1038, -1068, -948, -1068}}, ,
1400 /* GC.U..GU */
1401 {{ DEF, -499, -499, -499, -569}, { -649, -1098, -1098, -1098, -1168}, { -289, -738, -738, -738, -808}, {
    -739, -1188, -1188, -1188, -1258}, { -379, -828, -828, -828, -898}}}
1402 },
1403 {
1404 /* GC.@..UG */
1405 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {
    DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}}, ,
1406 /* GC.A..UG */
1407 {{ DEF, -519, -879, -559, -879}, { -769, -1238, -1598, -1278, -1598}, { -529, -998, -1358, -1038, -1358}, {
    -709, -1178, -1538, -1218, -1538}, { -599, -1068, -1428, -1108, -1428}}, ,
1408 /* GC.C..UG */
1409 {{ DEF, -719, -309, -309, -389}, { -839, -1508, -1098, -1098, -1178}, { -529, -1198, -788, -788, -868}, {
    -859, -1528, -1118, -1118, -1198}, { -489, -1158, -748, -748, -828}}, ,
1410 /* GC.G..UG */
1411 {{ DEF, -709, -739, -619, -739}, { -1009, -1668, -1698, -1578, -1698}, { -409, -1068, -1098, -978, -1098}, {
    -969, -1628, -1658, -1538, -1658}, { -599, -1258, -1288, -1168, -1288}}, ,
1412 /* GC.U..UG */
1413 {{ DEF, -499, -499, -499, -569}, { -859, -1308, -1308, -1308, -1378}, { -529, -978, -978, -978, -1048}, {
    -859, -1308, -1308, -1308, -1378}, { -409, -858, -858, -858, -928}}}
1414 },
1415 {
1416 /* GC.@..AU */
1417 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {
    DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}}, ,
1418 /* GC.A..AU */
1419 {{ DEF, -519, -879, -559, -879}, { -479, -948, -1308, -988, -1308}, { -309, -778, -1138, -818, -1138}, {
    -389, -858, -1218, -898, -1218}, { -379, -848, -1208, -888, -1208}}, ,
1420 /* GC.C..AU */
1421 {{ DEF, -719, -309, -309, -389}, { -649, -1318, -908, -908, -988}, { -289, -958, -548, -548, -628}, {
    -739, -1408, -998, -998, -1078}, { -379, -1048, -638, -638, -718}}, ,
1422 /* GC.G..AU */
1423 {{ DEF, -709, -739, -619, -739}, { -649, -1308, -1338, -1218, -1338}, { -289, -948, -978, -858, -978}, {
    -739, -1398, -1428, -1308, -1428}, { -379, -1038, -1068, -948, -1068}}, ,
1424 /* GC.U..AU */
1425 {{ DEF, -499, -499, -499, -569}, { -649, -1098, -1098, -1098, -1168}, { -289, -738, -738, -738, -808}, {
    -739, -1188, -1188, -1188, -1258}, { -379, -828, -828, -828, -898}}}
1426 },
1427 {
1428 /* GC.@..UA */
1429 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {
    DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}}, ,
1430 /* GC.A..UA */
1431 {{ DEF, -519, -879, -559, -879}, { -449, -918, -1278, -958, -1278}, { -479, -948, -1308, -988, -1308}, {
    -429, -898, -1258, -938, -1258}, { -329, -798, -1158, -838, -1158}}, ,
1432 /* GC.C..UA */
1433 {{ DEF, -719, -309, -309, -389}, { -679, -1348, -938, -938, -1018}, { -559, -1228, -818, -818, -898}, {
    -729, -1398, -988, -988, -1068}, { -189, -858, -448, -448, -528}}}

```

```

1434 /* GC.G..UA */
1435 {{ DEF, -709, -739, -619, -739}, { -939, -1598, -1628, -1508, -1628}, { -249, -908, -938, -818, -938}, {
   -939, -1598, -1628, -1508, -1628}, { -329, -988, -1018, -898, -1018}}, ,
1436 /* GC.U..UA */
1437 {{ DEF, -499, -499, -499, -569}, { -639, -1088, -1088, -1088, -1158}, { -229, -678, -678, -678, -748}, {
   -729, -1178, -1178, -1178, -1248}, { -190, -639, -639, -639, -709}}}
1438 },
1439 {
1440 /* GC.@.. @ */
1441 {{ DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}, {
   DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}}, ,
1442 /* GC.A.. @ */
1443 {{ -100, -569, -929, -609, -929}, { -100, -569, -929, -609, -929}, { -100, -569, -929, -609, -929}, {
   -100, -569, -929, -609, -929}, { -100, -569, -929, -609, -929}}, ,
1444 /* GC.C.. @ */
1445 {{ -100, -769, -359, -359, -439}, { -100, -769, -359, -359, -439}, { -100, -769, -359, -359, -439}, {
   -100, -769, -359, -359, -439}, { -100, -769, -359, -359, -439}}, ,
1446 /* GC.G.. @ */
1447 {{ -100, -759, -789, -669, -789}, { -100, -759, -789, -669, -789}, { -100, -759, -789, -669, -789}, {
   -100, -759, -789, -669, -789}, { -100, -759, -789, -669, -789}}, ,
1448 /* GC.U.. @ */
1449 {{ -100, -549, -549, -549, -619}, { -100, -549, -549, -549, -619}, { -100, -549, -549, -549, -619}, {
   -100, -549, -549, -549, -619}, { -100, -549, -549, -549, -619}}}
1450 },
1451 },
1452 { /* noPair */ {{{0}}}},
1453 {
1454 /* GU.@..CG */
1455 {{ 0, 0, 0, 0, 0}, { DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}, {
   DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}}, ,
1456 /* GU.A..CG */
1457 {{ DEF, -429, -599, -599, -599}, { -1079, -1458, -1628, -1628}, { -569, -948, -1118, -1118, -1118}, {
   -989, -1368, -1538, -1538}, { -859, -1238, -1408, -1408}, -1408}}, ,
1458 /* GU.C..CG */
1459 {{ DEF, -259, -239, -239, -239}, { -999, -1208, -1188, -1188}, { -499, -708, -688, -688, -688}, {
   -989, -1198, -1178, -1178}, { -789, -998, -978, -978, -978}}, ,
1460 /* GU.G..CG */
1461 {{ DEF, -339, -689, -689, -689}, { -1079, -1368, -1718, -1718}, { -569, -858, -1208, -1208}, {
   -989, -1278, -1628, -1628}, { -859, -1148, -1498, -1498}, -1498}}, ,
1462 /* GU.U..CG */
1463 {{ DEF, -329, -329, -329, -329}, { -1079, -1358, -1358, -1358}, { -719, -998, -998, -998, -998}, {
   -989, -1268, -1268, -1268}, { -909, -1188, -1188, -1188}}}
1464 },
1465 {
1466 /* GU.@..GC */
1467 {{ 0, 0, 0, 0, 0}, { DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}, {
   DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}}, ,
1468 /* GU.A..GC */
1469 {{ DEF, -429, -599, -599, -599}, { -569, -948, -1118, -1118}, { -769, -1148, -1318, -1318}, {
   -759, -1138, -1308, -1308}, { -549, -928, -1098, -1098}, -1098}}, ,
1470 /* GU.C..GC */
1471 {{ DEF, -259, -239, -239, -239}, { -929, -1138, -1118, -1118}, { -359, -568, -548, -548, -548}, {
   -789, -998, -978, -978}, { -549, -758, -738, -738, -738}}, ,
1472 /* GU.G..GC */
1473 {{ DEF, -339, -689, -689, -689}, { -609, -898, -1248, -1248}, { -359, -648, -998, -998, -998}, {
   -669, -958, -1308, -1308}, { -549, -838, -1188, -1188}, -1188}}, ,
1474 /* GU.U..GC */
1475 {{ DEF, -329, -329, -329, -329}, { -929, -1208, -1208, -1208}, { -439, -718, -718, -718, -718}, {
   -789, -1068, -1068, -1068}, { -619, -898, -898, -898, -898}}}
1476 },
1477 {
1478 /* GU.@..GU */
1479 {{ 0, 0, 0, 0, 0}, { DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}, {
   DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}}, ,
1480 /* GU.A..GU */
1481 {{ DEF, -429, -599, -599, -599}, { -479, -858, -1028, -1028, -1028}, { -309, -688, -858, -858, -858}, {
   -389, -768, -938, -938}, { -379, -758, -928, -928}, -928}}, ,
1482 /* GU.C..GU */
1483 {{ DEF, -259, -239, -239, -239}, { -649, -858, -838, -838, -838}, { -289, -498, -478, -478, -478}, {
   -739, -948, -928, -928}, { -379, -588, -568, -568, -568}}, ,
1484 /* GU.G..GU */
1485 {{ DEF, -339, -689, -689, -689}, { -649, -938, -1288, -1288, -1288}, { -289, -578, -928, -928, -928}, {
   -739, -1028, -1378, -1378, -1378}, { -379, -668, -1018, -1018, -1018}}, ,
1486 /* GU.U..GU */
1487 {{ DEF, -329, -329, -329, -329}, { -649, -928, -928, -928, -928}, { -289, -568, -568, -568, -568}, {
   -739, -1018, -1018, -1018}, { -379, -658, -658, -658, -658}}}
1488 },
1489 {
1490 /* GU.@..UG */
1491 {{ 0, 0, 0, 0, 0}, { DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}, {
   DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}}, ,
1492 /* GU.A..UG */
1493 {{ DEF, -429, -599, -599, -599}, { -769, -1148, -1318, -1318}, { -529, -908, -1078, -1078, -1078}, {
   -709, -1088, -1258, -1258}, { -599, -978, -1148, -1148}, -1148}}, ,
1494 /* GU.C..UG */
1495 {{ DEF, -259, -239, -239, -239}, { -839, -1048, -1028, -1028, -1028}, { -529, -738, -718, -718, -718}, {
   -859, -1048, -1048, -1048}, { -489, -698, -678, -678, -678}}},

```

```

1496 /* GU.G..UG */
1497 {{ DEF, -339, -689, -689, -689}, {-1009, -1298, -1648, -1648}, { -409, -698, -1048, -1048}, {
-969, -1258, -1608, -1608, -1608}, { -599, -888, -1238, -1238, -1238}}, ,
1498 /* GU.U..UG */
1499 {{ DEF, -329, -329, -329, -329}, { -859, -1138, -1138, -1138}, { -529, -808, -808, -808}, {
-859, -1138, -1138, -1138}, { -409, -688, -688, -688}}, ,
1500 },
1501 {
1502 /* GU.Q..AU */
1503 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {
DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}}, ,
1504 /* GU.A..AU */
1505 {{ DEF, -429, -599, -599, -599}, { -479, -858, -1028, -1028}, { -309, -688, -858, -858}, {
-389, -768, -938, -938, -938}, { -379, -758, -928, -928, -928}}, ,
1506 /* GU.C..AU */
1507 {{ DEF, -259, -239, -239, -239}, { -649, -858, -838, -838}, { -289, -498, -478, -478, -478}, {
-739, -948, -928, -928, -928}, { -379, -588, -568, -568, -568}}, ,
1508 /* GU.G..AU */
1509 {{ DEF, -339, -689, -689, -689}, { -649, -938, -1288, -1288, -1288}, { -289, -578, -928, -928, -928}, {
-739, -1028, -1378, -1378, -1378}, { -379, -668, -1018, -1018, -1018}}, ,
1510 /* GU.U..AU */
1511 {{ DEF, -329, -329, -329, -329}, { -649, -928, -928, -928}, { -289, -568, -568, -568, -568}, {
-739, -1018, -1018, -1018, -1018}, { -379, -658, -658, -658, -658}}, ,
1512 },
1513 {
1514 /* GU.Q..UA */
1515 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {
DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}}, ,
1516 /* GU.A..UA */
1517 {{ DEF, -429, -599, -599, -599}, { -449, -828, -998, -998}, { -479, -858, -1028, -1028, -1028}, {
-429, -808, -978, -978, -978}, { -329, -708, -878, -878, -878}}, ,
1518 /* GU.C..UA */
1519 {{ DEF, -259, -239, -239, -239}, { -679, -888, -868, -868, -868}, { -559, -768, -748, -748, -748}, {
-729, -938, -918, -918, -918}, { -189, -398, -378, -378, -378}}, ,
1520 /* GU.G..UA */
1521 {{ DEF, -339, -689, -689, -689}, { -939, -1228, -1578, -1578, -1578}, { -249, -538, -888, -888, -888}, {
-939, -1228, -1578, -1578, -1578}, { -329, -618, -968, -968, -968}}, ,
1522 /* GU.U..UA */
1523 {{ DEF, -329, -329, -329, -329}, { -639, -918, -918, -918}, { -229, -508, -508, -508, -508}, {
-729, -1008, -1008, -1008, -1008}, { -190, -469, -469, -469, -469}}, ,
1524 },
1525 {
1526 /* GU.Q..@ */
1527 {{ DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {
DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}}, ,
1528 /* GU.A..@ */
1529 {{ -100, -479, -649, -649, -649}, { -100, -479, -649, -649, -649}, { -100, -479, -649, -649, -649}, {
-100, -479, -649, -649, -649}, { -100, -479, -649, -649, -649}}, ,
1530 /* GU.C..@ */
1531 {{ -100, -309, -289, -289, -289}, { -100, -309, -289, -289, -289}, { -100, -309, -289, -289, -289}, {
-100, -309, -289, -289, -289}, { -100, -309, -289, -289, -289}}, ,
1532 /* GU.G..@ */
1533 {{ -100, -389, -739, -739, -739}, { -100, -389, -739, -739, -739}, { -100, -389, -739, -739, -739}, {
-100, -389, -739, -739, -739}, { -100, -389, -739, -739, -739}}, ,
1534 /* GU.U..@ */
1535 {{ -100, -379, -379, -379, -379}, { -100, -379, -379, -379, -379}, { -100, -379, -379, -379, -379}, {
-100, -379, -379, -379, -379}, { -100, -379, -379, -379, -379}}, ,
1536 },
1537 },
1538 /* noPair */ {{{0}}}, ,
1539 {
1540 /* UG.Q..CG */
1541 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {
DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}}, ,
1542 /* UG.A..CG */
1543 {{ DEF, -719, -789, -959, -809}, {-1079, -1748, -1818, -1988, -1838}, { -569, -1238, -1308, -1478, -1328}, {
-989, -1658, -1728, -1898, -1748}, { -859, -1528, -1598, -1768, -1618}}, ,
1544 /* UG.C..CG */
1545 {{ DEF, -479, -479, -359, -479}, { -999, -1428, -1428, -1308, -1428}, { -499, -928, -928, -808, -928}, {
-989, -1418, -1418, -1298, -1418}, { -789, -1218, -1218, -1098, -1218}}, ,
1546 /* UG.G..CG */
1547 {{ DEF, -659, -809, -919, -809}, {-1079, -1688, -1838, -1948, -1838}, { -569, -1178, -1328, -1438, -1328}, {
-989, -1598, -1748, -1858, -1748}, { -859, -1468, -1618, -1728, -1618}}, ,
1548 /* UG.U..CG */
1549 {{ DEF, -549, -439, -549, -359}, {-1079, -1578, -1468, -1578, -1388}, { -719, -1218, -1108, -1218, -1028}, {
-989, -1488, -1378, -1488, -1298}, { -909, -1408, -1298, -1408, -1218}}, ,
1550 },
1551 {
1552 /* UG.Q..GC */
1553 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {
DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}}, ,
1554 /* UG.A..GC */
1555 {{ DEF, -719, -789, -959, -809}, { -569, -1238, -1308, -1478, -1328}, { -769, -1438, -1508, -1678, -1528}, {
-759, -1428, -1498, -1668, -1518}, { -549, -1218, -1288, -1458, -1308}}, ,
1556 /* UG.C..GC */
1557 {{ DEF, -479, -479, -359, -479}, { -929, -1358, -1358, -1238, -1358}, { -359, -788, -788, -668, -788}, {
-789, -1218, -1218, -1098, -1218}, { -549, -978, -978, -858, -978}}, ,

```

```

1558 /* UG.G..GC */
1559 {{ DEF, -659, -809, -919, -809}, { -609, -1218, -1368, -1478, -1368}, { -359, -968, -1118, -1228, -1118}, {
   -669, -1278, -1428, -1538, -1428}, { -549, -1158, -1308, -1418, -1308}}, ,
1560 /* UG.U..GC */
1561 {{ DEF, -549, -439, -549, -359}, { -929, -1428, -1318, -1428, -1238}, { -439, -938, -828, -938, -748}, {
   -789, -1288, -1178, -1288, -1098}, { -619, -1118, -1008, -1118, -928}} ,
1562 },
1563 {
1564 /* UG.Q..GU */
1565 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {
   DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}}, ,
1566 /* UG.A..GU */
1567 {{ DEF, -719, -789, -959, -809}, { -479, -1148, -1218, -1388, -1238}, { -309, -978, -1048, -1218, -1068}, {
   -389, -1058, -1128, -1298, -1148}, { -379, -1048, -1118, -1288, -1138}}, ,
1568 /* UG.C..GU */
1569 {{ DEF, -479, -479, -359, -479}, { -649, -1078, -1078, -958, -1078}, { -289, -718, -718, -598, -718}, {
   -739, -1168, -1168, -1048, -1168}, { -379, -808, -808, -688, -808}}, ,
1570 /* UG.G..GU */
1571 {{ DEF, -659, -809, -919, -809}, { -649, -1258, -1408, -1518, -1408}, { -289, -898, -1048, -1158, -1048}, {
   -739, -1348, -1498, -1608, -1498}, { -379, -988, -1138, -1248, -1138}}, ,
1572 /* UG.U..GU */
1573 {{ DEF, -549, -439, -549, -359}, { -649, -1148, -1038, -1148, -958}, { -289, -788, -678, -788, -598}, {
   -739, -1238, -1128, -1238, -1048}, { -379, -878, -768, -878, -688}} ,
1574 },
1575 {
1576 /* UG.Q..UG */
1577 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {
   DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}}, ,
1578 /* UG.A..UG */
1579 {{ DEF, -719, -789, -959, -809}, { -769, -1438, -1508, -1678, -1528}, { -529, -1198, -1268, -1438, -1288}, {
   -709, -1378, -1448, -1618, -1468}, { -599, -1268, -1338, -1508, -1358}}, ,
1580 /* UG.C..UG */
1581 {{ DEF, -479, -479, -359, -479}, { -839, -1268, -1268, -1148, -1268}, { -529, -958, -958, -838, -958}, {
   -859, -1288, -1288, -1168, -1288}, { -489, -918, -918, -798, -918}}, ,
1582 /* UG.G..UG */
1583 {{ DEF, -659, -809, -919, -809}, { -1009, -1618, -1768, -1878, -1768}, { -409, -1018, -1168, -1278, -1168}, {
   -969, -1578, -1728, -1838, -1728}, { -599, -1208, -1358, -1468, -1358}}, ,
1584 /* UG.U..UG */
1585 {{ DEF, -549, -439, -549, -359}, { -859, -1358, -1248, -1358, -1168}, { -529, -1028, -918, -1028, -838}, {
   -859, -1358, -1248, -1358, -1168}, { -409, -908, -798, -908, -718}} ,
1586 },
1587 {
1588 /* UG.Q..AU */
1589 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {
   DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}}, ,
1590 /* UG.A..AU */
1591 {{ DEF, -719, -789, -959, -809}, { -479, -1148, -1218, -1388, -1238}, { -309, -978, -1048, -1218, -1068}, {
   -389, -1058, -1128, -1298, -1148}, { -379, -1048, -1118, -1288, -1138}}, ,
1592 /* UG.C..AU */
1593 {{ DEF, -479, -479, -359, -479}, { -649, -1078, -1078, -958, -1078}, { -289, -718, -718, -598, -718}, {
   -739, -1168, -1168, -1048, -1168}, { -379, -808, -808, -688, -808}}, ,
1594 /* UG.G..AU */
1595 {{ DEF, -659, -809, -919, -809}, { -649, -1258, -1408, -1518, -1408}, { -289, -898, -1048, -1158, -1048}, {
   -739, -1348, -1498, -1608, -1498}, { -379, -988, -1138, -1248, -1138}}, ,
1596 /* UG.U..AU */
1597 {{ DEF, -549, -439, -549, -359}, { -649, -1148, -1038, -1148, -958}, { -289, -788, -678, -788, -598}, {
   -739, -1238, -1128, -1238, -1048}, { -379, -878, -768, -878, -688}} ,
1598 },
1599 {
1600 /* UG.Q..UA */
1601 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {
   DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}}, ,
1602 /* UG.A..UA */
1603 {{ DEF, -719, -789, -959, -809}, { -449, -1118, -1188, -1358, -1208}, { -479, -1148, -1218, -1388, -1238}, {
   -429, -1098, -1168, -1338, -1188}, { -329, -998, -1068, -1238, -1088}}, ,
1604 /* UG.C..UA */
1605 {{ DEF, -479, -479, -359, -479}, { -679, -1108, -1108, -988, -1108}, { -559, -988, -988, -868, -988}, {
   -729, -1158, -1158, -1038, -1158}, { -189, -618, -618, -498, -618}}, ,
1606 /* UG.G..UA */
1607 {{ DEF, -659, -809, -919, -809}, { -939, -1548, -1698, -1808, -1698}, { -249, -858, -1008, -1118, -1008}, {
   -939, -1548, -1698, -1808, -1698}, { -329, -938, -1088, -1198, -1088}}, ,
1608 /* UG.U..UA */
1609 {{ DEF, -549, -439, -549, -359}, { -639, -1138, -1028, -1138, -948}, { -229, -728, -618, -728, -538}, {
   -729, -1228, -1118, -1228, -1038}, { -190, -689, -579, -689, -499}} ,
1610 },
1611 {
1612 /* UG.Q.. @ */
1613 {{ DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {
   DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}}, ,
1614 /* UG.A.. @ */
1615 {{ -100, -769, -839, -1009, -859}, { -100, -769, -839, -1009, -859}, { -100, -769, -839, -1009, -859}, {
   -100, -769, -839, -1009, -859}, { -100, -769, -839, -1009, -859}}, ,
1616 /* UG.C.. @ */
1617 {{ -100, -529, -529, -409, -529}, { -100, -529, -529, -409, -529}, { -100, -529, -529, -409, -529}, {
   -100, -529, -529, -409, -529}, { -100, -529, -529, -409, -529}}, ,
1618 /* UG.G.. @ */
1619 {{ -100, -709, -859, -969, -859}, { -100, -709, -859, -969, -859}, { -100, -709, -859, -969, -859}, {
   -100, -709, -859, -969, -859}}

```



```

-739,-1028,-1378,-1378,-1378}, { -379, -668,-1018,-1018,-1018}),  

1682 /* AU.U..AU */  

1683 {{ DEF, -329, -329, -329, -329}, { -649, -928, -928, -928}, { -289, -568, -568, -568, -568}, {  

    -739,-1018,-1018,-1018}, { -379, -658, -658, -658}}}  

1684 },  

1685 {  

1686 /* AU.@..UA */  

1687 {{ 0, 0, 0, 0, 0}, { DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}, {  

    DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}},  

1688 /* AU.A..UA */  

1689 {{ DEF, -429, -599, -599, -599}, { -449, -828, -998, -998}, { -479, -858,-1028,-1028,-1028}, {  

    -429, -808, -978, -978}, { -329, -708, -878, -878, -878}},  

1690 /* AU.C..UA */  

1691 {{ DEF, -259, -239, -239, -239}, { -679, -888, -868, -868}, { -559, -768, -748, -748, -748}, {  

    -729, -938, -918, -918}, { -189, -398, -378, -378, -378}},  

1692 /* AU.G..UA */  

1693 {{ DEF, -339, -689, -689, -689}, { -939,-1228,-1578,-1578}, { -249, -538, -888, -888, -888}, {  

    -939,-1228,-1578,-1578}, { -329, -618, -968, -968, -968}},  

1694 /* AU.U..UA */  

1695 {{ DEF, -329, -329, -329, -329}, { -639, -918, -918, -918}, { -229, -508, -508, -508, -508}, {  

    -729,-1008,-1008,-1008}, { -190, -469, -469, -469, -469}}}  

1696 },  

1697 {  

1698 /* AU.@.. @ */  

1699 {{ DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}, {  

    DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}},  

1700 /* AU.A.. @ */  

1701 {{ -100, -479, -649, -649, -649}, { -100, -479, -649, -649, -649}, { -100, -479, -649, -649, -649}, {  

    -100, -479, -649, -649, -649}, { -100, -479, -649, -649, -649}},  

1702 /* AU.C.. @ */  

1703 {{ -100, -309, -289, -289, -289}, { -100, -309, -289, -289, -289}, { -100, -309, -289, -289, -289}, {  

    -100, -309, -289, -289, -289}, { -100, -309, -289, -289, -289}},  

1704 /* AU.G.. @ */  

1705 {{ -100, -389, -739, -739, -739}, { -100, -389, -739, -739, -739}, { -100, -389, -739, -739, -739}, {  

    -100, -389, -739, -739, -739}, { -100, -389, -739, -739, -739}},  

1706 /* AU.U.. @ */  

1707 {{ -100, -379, -379, -379, -379}, { -100, -379, -379, -379, -379}, { -100, -379, -379, -379, -379}, {  

    -100, -379, -379, -379, -379}, { -100, -379, -379, -379, -379}}}  

1708 },  

1709 },  

1710 { /* noPair */ {{0}}},  

1711 {  

1712 /* UA.@..CG */  

1713 {{ 0, 0, 0, 0, 0}, { DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}, {  

    DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}},  

1714 /* UA.A..CG */  

1715 {{ DEF, -399, -629, -889, -589}, {-1079,-1428,-1658,-1918,-1618}, { -569, -918,-1148,-1408,-1108}, {  

    -989,-1338,-1568,-1828,-1528}, { -859,-1208,-1438,-1698,-1398}},  

1716 /* UA.C..CG */  

1717 {{ DEF, -429, -509, -199, -179}, { -999,-1378,-1458,-1148,-1128}, { -499, -878, -958, -648, -628}, {  

    -989,-1368,-1448,-1138,-1118}, { -789,-1168,-1248, -938, -918}},  

1718 /* UA.G..CG */  

1719 {{ DEF, -379, -679, -889, -679}, {-1079,-1408,-1708,-1918,-1708}, { -569, -898,-1198,-1408,-1198}, {  

    -989,-1318,-1618,-1828,-1618}, { -859,-1188,-1488,-1698,-1488}},  

1720 /* UA.U..CG */  

1721 {{ DEF, -279, -139, -279, -140}, {-1079,-1308,-1168,-1308,-1169}, { -719, -948, -808, -948, -809}, {  

    -989,-1218,-1078,-1218,-1079}, { -909,-1138, -998,-1138, -999}}}  

1722 },  

1723 {  

1724 /* UA.@..GC */  

1725 {{ 0, 0, 0, 0, 0}, { DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}, {  

    DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}},  

1726 /* UA.A..GC */  

1727 {{ DEF, -399, -629, -889, -589}, { -569, -918,-1148,-1408,-1108}, { -769,-1118,-1348,-1608,-1308}, {  

    -759,-1108,-1338,-1598,-1298}, { -549, -898,-1128,-1388,-1088}},  

1728 /* UA.C..GC */  

1729 {{ DEF, -429, -509, -199, -179}, { -929,-1308,-1388,-1078,-1058}, { -359, -738, -818, -508, -488}, {  

    -789,-1168,-1248, -938, -918}, { -549, -928,-1008, -698, -678}},  

1730 /* UA.G..GC */  

1731 {{ DEF, -379, -679, -889, -679}, { -609, -938,-1238,-1448,-1238}, { -359, -688, -988,-1198, -988}, {  

    -669, -998,-1298,-1508,-1298}, { -549, -878,-1178,-1388,-1178}},  

1732 /* UA.U..GC */  

1733 {{ DEF, -279, -139, -279, -140}, { -929,-1158,-1018,-1158,-1019}, { -439, -668, -528, -668, -529}, {  

    -789,-1018, -878,-1018, -879}, { -619, -848, -708, -848, -709}}}  

1734 },  

1735 {  

1736 /* UA.@..GU */  

1737 {{ 0, 0, 0, 0, 0}, { DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}, {  

    DEF, DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF, DEF}},  

1738 /* UA.A..GU */  

1739 {{ DEF, -399, -629, -889, -589}, { -479, -828,-1058,-1318,-1018}, { -309, -658, -888,-1148, -848}, {  

    -389, -738, -968,-1228, -928}, { -379, -728, -958,-1218, -918}},  

1740 /* UA.C..GU */  

1741 {{ DEF, -429, -509, -199, -179}, { -649,-1028,-1108, -798, -778}, { -289, -668, -748, -438, -418}, {  

    -739,-1118,-1198, -888, -868}, { -379, -758, -838, -528, -508}},  

1742 /* UA.G..GU */  

1743 {{ DEF, -379, -679, -889, -679}, { -649, -978,-1278,-1488,-1278}, { -289, -618, -918,-1128, -918}, {
```

```

    -739, -1068, -1368, -1578, -1368}, { -379, -708, -1008, -1218, -1008}},  

1744 /* UA.U.GU */  

1745 {{ DEF, -279, -139, -279, -140}, { -649, -878, -738, -878, -739}, { -289, -518, -378, -518, -379}, {  

    -739, -968, -828, -968, -829}, { -379, -608, -468, -608, -469}}}  

1746 },  

1747 {  

1748 /* UA.Q.UG */  

1749 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {  

    DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}},  

1750 /* UA.A.UG */  

1751 {{ DEF, -399, -629, -889, -589}, { -769, -1118, -1348, -1608, -1308}, { -529, -878, -1108, -1368, -1068}, {  

    -709, -1058, -1288, -1548, -1248}, { -599, -948, -1178, -1438, -1138}}},  

1752 /* UA.C.UG */  

1753 {{ DEF, -429, -509, -199, -179}, { -839, -1218, -1298, -988, -968}, { -529, -908, -988, -678, -658}, {  

    -859, -1238, -1318, -1008, -988}, { -489, -868, -948, -638, -618}}},  

1754 /* UA.G.UG */  

1755 {{ DEF, -379, -679, -889, -679}, { -1009, -1338, -1638, -1848, -1638}, { -409, -738, -1038, -1248, -1038}, {  

    -969, -1298, -1598, -1808, -1598}, { -599, -928, -1228, -1438, -1228}}},  

1756 /* UA.U.UG */  

1757 {{ DEF, -279, -139, -279, -140}, { -859, -1088, -948, -1088, -949}, { -529, -758, -618, -758, -619}, {  

    -859, -1088, -948, -1088, -949}, { -409, -638, -498, -638, -499}}}  

1758 },  

1759 {  

1760 /* UA.Q.AU */  

1761 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {  

    DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}},  

1762 /* UA.A.AU */  

1763 {{ DEF, -399, -629, -889, -589}, { -479, -828, -1058, -1318, -1018}, { -309, -658, -888, -1148, -848}, {  

    -389, -738, -968, -1228, -928}, { -379, -728, -958, -1218, -918}}},  

1764 /* UA.C.AU */  

1765 {{ DEF, -429, -509, -199, -179}, { -649, -1028, -1108, -798, -778}, { -289, -668, -748, -438, -418}, {  

    -739, -1118, -1198, -888, -868}, { -379, -758, -838, -528, -508}}},  

1766 /* UA.G.AU */  

1767 {{ DEF, -379, -679, -889, -679}, { -649, -978, -1278, -1488, -1278}, { -289, -618, -918, -1128, -918}, {  

    -739, -1068, -1368, -1578, -1368}, { -379, -708, -1008, -1218, -1008}}},  

1768 /* UA.U.AU */  

1769 {{ DEF, -279, -139, -279, -140}, { -649, -878, -738, -878, -739}, { -289, -518, -378, -518, -379}, {  

    -739, -968, -828, -968, -829}, { -379, -608, -468, -608, -469}}}  

1770 },  

1771 {  

1772 /* UA.Q.UA */  

1773 {{ 0, 0, 0, 0}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {  

    DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}},  

1774 /* UA.A.UA */  

1775 {{ DEF, -399, -629, -889, -589}, { -449, -798, -1028, -1288, -988}, { -479, -828, -1058, -1318, -1018}, {  

    -429, -778, -1008, -1268, -968}, { -329, -678, -908, -1168, -868}}},  

1776 /* UA.C.UA */  

1777 {{ DEF, -429, -509, -199, -179}, { -679, -1058, -1138, -828, -808}, { -559, -938, -1018, -708, -688}, {  

    -729, -1108, -1188, -878, -858}, { -189, -568, -648, -338, -318}}},  

1778 /* UA.G.UA */  

1779 {{ DEF, -379, -679, -889, -679}, { -939, -1268, -1568, -1778, -1568}, { -249, -578, -878, -1088, -878}, {  

    -939, -1268, -1568, -1778, -1568}, { -329, -658, -958, -1168, -958}}},  

1780 /* UA.U.UA */  

1781 {{ DEF, -279, -139, -279, -140}, { -639, -868, -728, -868, -729}, { -229, -458, -318, -458, -319}, {  

    -729, -958, -818, -958, -819}, { -190, -419, -279, -419, -280}}}  

1782 },  

1783 {  

1784 /* UA.Q.Q */  

1785 {{ DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}, {  

    DEF, DEF, DEF, DEF}, { DEF, DEF, DEF, DEF}},  

1786 /* UA.A.Q */  

1787 {{ -100, -449, -679, -939, -639}, { -100, -449, -679, -939, -639}, { -100, -449, -679, -939, -639}, {  

    -100, -449, -679, -939, -639}, { -100, -449, -679, -939, -639}}},  

1788 /* UA.C.Q */  

1789 {{ -100, -479, -559, -249, -229}, { -100, -479, -559, -249, -229}, { -100, -479, -559, -249, -229}, {  

    -100, -479, -559, -249, -229}, { -100, -479, -559, -249, -229}}},  

1790 /* UA.G.Q */  

1791 {{ -100, -429, -729, -939, -729}, { -100, -429, -729, -939, -729}, { -100, -429, -729, -939, -729}, {  

    -100, -429, -729, -939, -729}, { -100, -429, -729, -939, -729}}},  

1792 /* UA.U.Q */  

1793 {{ -100, -329, -189, -329, -190}, { -100, -329, -189, -329, -190}, { -100, -329, -189, -329, -190}, {  

    -100, -329, -189, -329, -190}, { -100, -329, -189, -329, -190}}}  

1794 },  

1795 },  

1796 /* noPair */ {{0}}},  

1797 {  

1798 /* @.Q.CG */  

1799 {{ DEF, DEF, DEF, DEF}, { -100, -100, -100, -100}, { -100, -100, -100, -100}, {  

    -100, -100, -100, -100}, { -100, -100, -100, -100}},  

1800 /* @.A.CG */  

1801 {{ DEF, DEF, DEF, DEF}, { -1079, -1079, -1079, -1079}, { -569, -569, -569, -569}, {  

    -989, -989, -989, -989}, { -859, -859, -859, -859}},  

1802 /* @.C.CG */  

1803 {{ DEF, DEF, DEF, DEF}, { -999, -999, -999, -999}, { -499, -499, -499, -499}, {  

    -989, -989, -989, -989}, { -789, -789, -789, -789}},  

1804 /* @.G.CG */  

1805 {{ DEF, DEF, DEF, DEF}, { -1079, -1079, -1079, -1079}, { -569, -569, -569, -569}, {
```

```

-989, -989, -989, -989, -989}, { -859, -859, -859, -859, -859}},  

1806 /* @.U.CG */  

1807 {{ DEF, DEF, DEF, DEF}, {-1079, -1079, -1079, -1079}, { -719, -719, -719, -719}, {  

-989, -989, -989, -989}, { -909, -909, -909, -909}}}  

1808 },  

1809 {  

1810 /* @.G.GC */  

1811 {{ DEF, DEF, DEF, DEF}, { -100, -100, -100, -100}, { -100, -100, -100, -100}, {  

-100, -100, -100, -100}, { -100, -100, -100, -100}},  

1812 /* @.A.GC */  

1813 {{ DEF, DEF, DEF, DEF}, { -569, -569, -569, -569}, { -769, -769, -769, -769}, {  

-759, -759, -759, -759}, { -549, -549, -549, -549}},  

1814 /* @.C.GC */  

1815 {{ DEF, DEF, DEF, DEF}, { -929, -929, -929, -929}, { -359, -359, -359, -359}, {  

-789, -789, -789, -789}, { -549, -549, -549, -549}},  

1816 /* @.G.GC */  

1817 {{ DEF, DEF, DEF, DEF}, { -609, -609, -609, -609}, { -359, -359, -359, -359}, {  

-669, -669, -669, -669}, { -549, -549, -549, -549}},  

1818 /* @.U.GC */  

1819 {{ DEF, DEF, DEF, DEF}, { -929, -929, -929, -929}, { -439, -439, -439, -439}, {  

-789, -789, -789, -789}, { -619, -619, -619, -619}}}  

1820 },  

1821 {  

1822 /* @.G.GU */  

1823 {{ DEF, DEF, DEF, DEF}, { -100, -100, -100, -100}, { -100, -100, -100, -100}, {  

-100, -100, -100, -100}, { -100, -100, -100, -100}},  

1824 /* @.A.GU */  

1825 {{ DEF, DEF, DEF, DEF}, { -479, -479, -479, -479}, { -309, -309, -309, -309}, {  

-389, -389, -389, -389}, { -379, -379, -379, -379}},  

1826 /* @.C.GU */  

1827 {{ DEF, DEF, DEF, DEF}, { -649, -649, -649, -649}, { -289, -289, -289, -289}, {  

-739, -739, -739, -739}, { -379, -379, -379, -379}},  

1828 /* @.G.GU */  

1829 {{ DEF, DEF, DEF, DEF}, { -649, -649, -649, -649}, { -289, -289, -289, -289}, {  

-739, -739, -739, -739}, { -379, -379, -379, -379}},  

1830 /* @.U.GU */  

1831 {{ DEF, DEF, DEF, DEF}, { -649, -649, -649, -649}, { -289, -289, -289, -289}, {  

-739, -739, -739, -739}, { -379, -379, -379, -379}}}  

1832 },  

1833 {  

1834 /* @.G.UG */  

1835 {{ DEF, DEF, DEF, DEF}, { -100, -100, -100, -100}, { -100, -100, -100, -100}, {  

-100, -100, -100, -100}, { -100, -100, -100, -100}},  

1836 /* @.A.UG */  

1837 {{ DEF, DEF, DEF, DEF}, { -769, -769, -769, -769}, { -529, -529, -529, -529}, {  

-709, -709, -709, -709}, { -599, -599, -599, -599}},  

1838 /* @.C.UG */  

1839 {{ DEF, DEF, DEF, DEF}, { -839, -839, -839, -839}, { -529, -529, -529, -529}, {  

-859, -859, -859, -859}, { -489, -489, -489, -489}},  

1840 /* @.G.UG */  

1841 {{ DEF, DEF, DEF, DEF}, { -1009, -1009, -1009, -1009}, { -409, -409, -409, -409}, {  

-969, -969, -969, -969}, { -599, -599, -599, -599}},  

1842 /* @.U.UG */  

1843 {{ DEF, DEF, DEF, DEF}, { -859, -859, -859, -859}, { -529, -529, -529, -529}, {  

-859, -859, -859, -859}, { -409, -409, -409, -409}}}  

1844 },  

1845 {  

1846 /* @.G.AU */  

1847 {{ DEF, DEF, DEF, DEF}, { -100, -100, -100, -100}, { -100, -100, -100, -100}, {  

-100, -100, -100, -100}, { -100, -100, -100, -100}},  

1848 /* @.A.AU */  

1849 {{ DEF, DEF, DEF, DEF}, { -479, -479, -479, -479}, { -309, -309, -309, -309}, {  

-389, -389, -389, -389}, { -379, -379, -379, -379}},  

1850 /* @.C.AU */  

1851 {{ DEF, DEF, DEF, DEF}, { -649, -649, -649, -649}, { -289, -289, -289, -289}, {  

-739, -739, -739, -739}, { -379, -379, -379, -379}},  

1852 /* @.G.AU */  

1853 {{ DEF, DEF, DEF, DEF}, { -649, -649, -649, -649}, { -289, -289, -289, -289}, {  

-739, -739, -739, -739}, { -379, -379, -379, -379}},  

1854 /* @.U.AU */  

1855 {{ DEF, DEF, DEF, DEF}, { -649, -649, -649, -649}, { -289, -289, -289, -289}, {  

-739, -739, -739, -739}, { -379, -379, -379, -379}}}  

1856 },  

1857 {  

1858 /* @.G.UA */  

1859 {{ DEF, DEF, DEF, DEF}, { -100, -100, -100, -100}, { -100, -100, -100, -100}, {  

-100, -100, -100, -100}, { -100, -100, -100, -100}},  

1860 /* @.A.UA */  

1861 {{ DEF, DEF, DEF, DEF}, { -449, -449, -449, -449}, { -479, -479, -479, -479}, {  

-429, -429, -429, -429}, { -329, -329, -329, -329}},  

1862 /* @.C.UA */  

1863 {{ DEF, DEF, DEF, DEF}, { -679, -679, -679, -679}, { -559, -559, -559, -559}, {  

-729, -729, -729, -729}, { -189, -189, -189, -189}},  

1864 /* @.G.UA */  

1865 {{ DEF, DEF, DEF, DEF}, { -939, -939, -939, -939}, { -249, -249, -249, -249}, {  

-939, -939, -939, -939}, { -329, -329, -329, -329}},  

1866 /* @.U.UA */

```

```

1867 {{ DEF, DEF, DEF, DEF, DEF},{ -639, -639, -639, -639, -639},{ -229, -229, -229, -229, -229},{ -729, -729, -729, -729, -729},{ -190, -190, -190, -190, -190}}
1868 },
1869 {
1870 /* @.@.. @ */
1871 {{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100}},
1872 /* @.A.. @ */
1873 {{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100}},
1874 /* @.C.. @ */
1875 {{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100}},
1876 /* @.G.. @ */
1877 {{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100}},
1878 /* @.U.. @ */
1879 {{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100}}
1880 }
1881 }
1882 };
1883
1884 PRIVATE int int22_37_184[NBPAIRS+1][NBPAIRS+1][5][5][5][5] =
1885 /* noPair */ {{{{0}}}}},
1886 /* noPair */ {{{{0}}}}},
1887 /* CG...CG */
1888 {
1889 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},
1890 { 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},
1891 { 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},
1892 { 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},
1893 { 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},
1894 },
1895 {
1896 { 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},
1897 { 340, 340, 340, 340, 340},{ 340, 130, 160, 30, 200},{ 340, 120, 150, 20, 200},{ 340, 30, 60, -70,
200},{ 340, 200, 200, 200, 200},
1898 { 340, 340, 340, 340},{ 340, 160, 200, 60, 200},{ 340, 210, 180, 150, 200},{ 340, 200, 200, 200,
200},{ 340, 190, 170, 130, 200},
1899 { 340, 340, 340, 340},{ 340, 30, 60, -70, 200},{ 340, 200, 200, 200, 200},{ 340, 100, 140, 0,
200},{ 340, -40, -110, -60, 200},
1900 { 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 190, 170, 130, 200},{ 340, 110, 40, 90,
200},{ 340, 140, 80, 130, 200}
1901 },
1902 {
1903 { 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},
1904 { 340, 340, 340, 340, 340},{ 340, 120, 210, 200, 190},{ 340, 110, 140, 200, 120},{ 340, 20, 150, 200,
130},{ 340, 200, 200, 200,
200},
1905 { 340, 340, 340, 340, 340},{ 340, 150, 180, 200, 170},{ 340, 140, 170, 200, 150},{ 340, 200, 200, 200,
200},{ 340, 120, 150, 200, 140},
1906 { 340, 340, 340, 340, 340},{ 340, 20, 150, 200, 130},{ 340, 200, 200, 200, 200},{ 340, 90, 180, 200,
170},{ 340, -150, -20, 200, -40},
1907 { 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 120, 150, 200, 140},{ 340, 0, 130, 200,
110},{ 340, 30, 60, 200, 50}}
1908 },
1909 {
1910 { 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},
1911 { 340, 340, 340, 340, 340},{ 340, 30, 200, 100, 110},{ 340, 20, 200, 90, 0},{ 340, -70, 200, 0,
90},{ 340, 200, 200, 200,
200},
1912 { 340, 340, 340, 340, 340},{ 340, 60, 200, 140, 40},{ 340, 150, 200, 180, 130},{ 340, 200, 200, 200,
200},{ 340, 130, 200, 170, 110},
1913 { 340, 340, 340, 340, 340},{ 340, -70, 200, 0, 90},{ 340, 200, 200, 200, 200},{ 340, 0, 200, 80,
90},{ 340, -60, 200, -70, -260},
1914 { 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 130, 200, 170, 110},{ 340, 90, 200, 90,
-110},{ 340, 130, 200, 120, 110}}
1915 },
1916 {
1917 { 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},
1918 { 340, 340, 340, 340, 340},{ 340, 200, 190, -40, 140},{ 340, 200, 120, -150, 30},{ 340, 200, 130,
-60, 130},{ 340, 200, 200, 200,
200},
1919 { 340, 340, 340, 340, 340},{ 340, 200, 170, -110, 80},{ 340, 200, 150, -20, 60},{ 340, 200, 200,
200, 200},{ 340, 200, 140, -40,
50},
1920 { 340, 340, 340, 340, 340},{ 340, 200, 130, -60, 130},{ 340, 200, 200, 200, 200},{ 340, 200, 170, -70,
120},{ 340, 200, -40, -420, -50},
1921 { 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 140, -40, 50},{ 340, 200, 110,
-260, 110},{ 340, 200, 50, -50, -40}}
1922 }

```







```

2079 },
2080 {
2081 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340}}, 
2082 {{ 340, 340, 340, 340}, { 340, 200, 240, 100, 200}, { 340, 160, 190, 60, 200}, { 340, 100, 130, 0,
2001}, { 340, 200, 200, 200, 200}}, 
2083 {{ 340, 340, 340, 340}, { 340, 200, 240, 100, 200}, { 340, 260, 240, 200, 200}, { 340, 200, 200, 200,
2001}, { 340, 260, 240, 200, 200}}, 
2084 {{ 340, 340, 340, 340}, { 340, 100, 130, 0, 200}, { 340, 200, 200, 200, 200}, { 340, 140, 170, 40,
2001}, { 340, 20, -40, 0, 200}}, 
2085 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 230, 210, 170, 200}, { 340, 150, 80, 130,
2001}, { 340, 220, 150, 200, 200}} 
2086 }, 
2087 { 
2088 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}}, 
2089 {{ 340, 340, 340, 340}, { 340, 190, 280, 200, 270}, { 340, 150, 180, 200, 160}, { 340, 90, 220, 200,
2001}, { 340, 200, 200, 200, 200}}, 
2090 {{ 340, 340, 340, 340}, { 340, 190, 220, 200, 210}, { 340, 190, 220, 200, 210}, { 340, 200, 200, 200,
2001}, { 340, 190, 220, 200, 210}}, 
2091 {{ 340, 340, 340, 340}, { 340, 90, 220, 200, 200}, { 340, 200, 200, 200, 200}, { 340, 130, 220, 200,
2001}, { 340, -90, 40, 200, 30}}, 
2092 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 160, 190, 200, 180}, { 340, 40, 170, 200,
150}, { 340, 110, 140, 200, 120}} 
2093 }, 
2094 { 
2095 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}}, 
2096 {{ 340, 340, 340, 340}, { 340, 100, 200, 180, 180}, { 340, 60, 200, 130, 40}, { 340, 0, 200, 70,
160}, { 340, 200, 200, 200, 200}}, 
2097 {{ 340, 340, 340, 340}, { 340, 100, 200, 180, 80}, { 340, 200, 200, 240, 180}, { 340, 200, 200, 200,
200}, { 340, 200, 240, 180}}, 
2098 {{ 340, 340, 340, 340}, { 340, 0, 200, 70, 160}, { 340, 200, 200, 200, 200}, { 340, 40, 200, 110,
120}, { 340, 0, 200, 0, -200}}, 
2099 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 170, 200, 210, 150}, { 340, 130, 200, 120,
-70}, { 340, 200, 200, 190, 180}} 
2100 }, 
2101 { 
2102 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}}, 
2103 {{ 340, 340, 340, 340}, { 340, 200, 270, 30, 220}, { 340, 200, 160, -110, 70}, { 340, 200, 200,
10, 190}, { 340, 200, 200, 200, 200}}, 
2104 {{ 340, 340, 340, 340}, { 340, 200, 210, -70, 120}, { 340, 200, 210, 30, 120}, { 340, 200, 200, 200,
200}, { 340, 200, 210, 30, 120}}, 
2105 {{ 340, 340, 340, 340}, { 340, 200, 200, 10, 190}, { 340, 200, 200, 200, 200}, { 340, 200, 200, -30,
150}, { 340, 200, 30, -350, 20}}, 
2106 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 180, 0, 90}, { 340, 200, 150,
-220, 150}, { 340, 200, 120, 30, 30}} 
2107 } 
2108 }, 
2109 /* CG....?? */ 
2110 { 
2111 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}}, 
2112 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}}, 
2113 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}}, 
2114 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}}, 
2115 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}} 
2116 }, 
2117 { 
2118 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}}, 
2119 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}}, 
2120 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}}, 
2121 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}}, 
2122 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}} 
2123 }, 
2124 { 
2125 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}}, 
2126 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}}, 
2127 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}}, 
2128 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}}, 
2129 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
340}, { 340, 340, 340, 340}} 
2130 },

```

```

2131 {
2132 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
2133 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
2134 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2135 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2136 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2137 }, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
2138 },
2139 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2140 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2141 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2142 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2143 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2144 },
2145 },
2146 },
2147 /* noPair */ {{{0}}}},
2148 /* GC...CG */
2149 {
2150 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2151 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2152 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2153 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2154 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2155 },
2156 {
2157 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2158 {{ 340, 340, 340, 340}, { 340, 50, 110, 40, 200}, { 340, 130, 100, 70, 200}, { 340, -20, 70, -50,
2159 {{ 340, 340, 340, 340}, { 340, 60, 110, 50, 200}, { 340, 220, 190, 70, 200}, { 340, 200, 200, 200,
2160 {{ 340, 340, 340, 340}, { 340, 0, -100, -70, 200}, { 340, 200, 200, 200, 200}, { 340, 110, 80,
2161 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 110, 100, 200}, { 340, 90, -10, 60,
2162 },
2163 {
2164 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2165 {{ 340, 340, 340, 340}, { 340, 110, 170, 200, 180}, { 340, 100, 100, 200, 110}, { 340, -40, 110, 200,
2166 {{ 340, 340, 340, 340}, { 340, 150, 150, 200, 150}, { 340, 130, 130, 200, 140}, { 340, 200, 200, 200,
2167 {{ 340, 340, 340, 340}, { 340, -70, -60, 200, 120}, { 340, 200, 200, 200, 200}, { 340, 90, 150, 200,
2168 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 120, 120, 200, 120}, { 340, 0, 100, 200,
2169 },
2170 {
2171 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2172 {{ 340, 340, 340, 340}, { 340, -30, 200, 100, -50}, { 340, -70, 200, 90, -150}, { 340, -170, 200,
2173 {{ 340, 340, 340, 340}, { 340, 10, 200, 140, -60}, { 340, 70, 200, 180, -20}, { 340, 200, 200, 200,
2174 {{ 340, 340, 340, 340}, { 340, -160, 200, 0, -60}, { 340, 200, 200, 200, 200}, { 340, -90, 200,
2175 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 40, 200, 170, -30}, { 340, 30, 200, 90,
2176 },
2177 {
2178 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
2179 {{ 340, 340, 340, 340}, { 340, 200, 70, 10, 150}, { 340, 200, 0, -190, -20}, { 340, 200, 20,
2180 {{ 340, 340, 340, 340}, { 340, 200, 50, -70, 0}, { 340, 200, 30, -30, -10}, { 340, 200, 200, 200,
2181 {{ 340, 340, 340, 340}, { 340, 200, 20, -80, 90}, { 340, 200, 200, 200, 200}, { 340, 200, 50,
2182 {{ 340, 340, 340, 340}, { 340, 200, 10, -100, 60}, { 340, 200, 170, -70, 20}, { 340, 200, 0,

```



```

200}, { 340, 200, 70, 150, 200} }

2236 },
2237 {
2238 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
      340}, { 340, 340, 340, 340, 340}},
2239 {{ 340, 340, 340, 340}, { 340, 190, 250, 200, 250}, { 340, 160, 160, 200, 170}, { 340, 60, 160, 200,
      170}, { 340, 200, 200, 200, 200}},
2240 {{ 340, 340, 340, 340, 340}, { 340, 170, 170, 200, 180}, { 340, 160, 160, 200, 160}, { 340, 200, 200, 200,
      200}, { 340, 160, 160, 200, 160}},
2241 {{ 340, 340, 340, 340, 340}, { 340, 60, 160, 200, 170}, { 340, 200, 200, 200, 200}, { 340, 120, 180, 200,
      180}, { 340, -50, 50, 200, 50}},
2242 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 160, 160, 200, 160}, { 340, 40, 140, 200,
      150}, { 340, 80, 80, 200, 80}}
2243 },
2244 {
2245 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340,
      340}, { 340, 340, 340, 340, 340}},
2246 {{ 340, 340, 340, 340, 340}, { 340, 10, 200, 180, 40}, { 340, -10, 200, 150, -90}, { 340, -110, 200,
      50, -10}, { 340, 200, 200, 200, 200}},
2247 {{ 340, 340, 340, 340, 340}, { 340, 0, 200, 160, -80}, { 340, 80, 200, 210, 10}, { 340, 200, 200, 200,
      200}, { 340, 80, 200, 210, 10}},
2248 {{ 340, 340, 340, 340, 340}, { 340, -110, 200, 50, -10}, { 340, 200, 200, 200, 200}, { 340, -60, 200,
      110, -30}, { 340, -50, 200, 40, -310}},
2249 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 80, 200, 210, 10}, { 340, 50, 200, 130,
      -210}, { 340, 80, 200, 170, 10}}
2250 },
2251 {
2252 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340,
      340}, { 340, 340, 340, 340, 340}},
2253 {{ 340, 340, 340, 340, 340}, { 340, 200, 150, 0, 210}, { 340, 200, 60, -130, 90}, { 340, 200, 70,
      -50, 170}, { 340, 200, 200, 200, 200}},
2254 {{ 340, 340, 340, 340, 340}, { 340, 200, 70, -120, 100}, { 340, 200, 60, -30, 80}, { 340, 200, 200,
      200}, { 340, 200, 60, -30, 80}},
2255 {{ 340, 340, 340, 340, 340}, { 340, 200, 70, -50, 170}, { 340, 200, 200, 200, 200}, { 340, 200, 80, -70,
      140}, { 340, 200, -50, -350, 50}},
2256 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 60, -30, 80}, { 340, 200, 50,
      -250, 150}, { 340, 200, -20, -30, 0}}
2257 },
2258 },
2259 /* GC....UG */
2260 {
2261 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340,
      340}, { 340, 340, 340, 340, 340}},
2262 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340,
      340}, { 340, 340, 340, 340, 340}},
2263 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340,
      340}, { 340, 340, 340, 340, 340}},
2264 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340,
      340}, { 340, 340, 340, 340, 340}},
2265 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340,
      340}, { 340, 340, 340, 340, 340}}
2266 },
2267 {
2268 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340,
      340}, { 340, 340, 340, 340, 340}},
2269 {{ 340, 340, 340, 340, 340}, { 340, 210, 180, 70, 200}, { 340, 170, 140, 30, 200}, { 340, 110, 80, -30,
      200}, { 340, 200, 200, 200, 200}},
2270 {{ 340, 340, 340, 340, 340}, { 340, 210, 180, 70, 200}, { 340, 270, 180, 170, 200}, { 340, 200, 200, 200,
      200}, { 340, 270, 180, 170, 200}},
2271 {{ 340, 340, 340, 340, 340}, { 340, 110, 80, -30, 200}, { 340, 200, 200, 200, 200}, { 340, 150, 120, 10,
      200}, { 340, 30, -100, -30, 200}},
2272 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 240, 150, 140, 200}, { 340, 160, 30, 100,
      200}, { 340, 230, 100, 170, 200}}
2273 },
2274 {
2275 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340,
      340}, { 340, 340, 340, 340, 340}},
2276 {{ 340, 340, 340, 340, 340}, { 340, 190, 250, 200, 250}, { 340, 140, 140, 200, 150}, { 340, 80, 180, 200,
      190}, { 340, 200, 200, 200, 200}},
2277 {{ 340, 340, 340, 340, 340}, { 340, 190, 190, 200, 190}, { 340, 190, 190, 200, 190}, { 340, 200, 200, 200,
      200}, { 340, 190, 190, 200, 190}},
2278 {{ 340, 340, 340, 340, 340}, { 340, 80, 180, 200, 190}, { 340, 200, 200, 200, 200}, { 340, 120, 180, 200,
      190}, { 340, -90, 10, 200, 10}},
2279 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 160, 160, 200, 160}, { 340, 30, 130, 200,
      140}, { 340, 100, 100, 200, 110}}
2280 },
2281 {
2282 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340,
      340}, { 340, 340, 340, 340, 340}},
2283 {{ 340, 340, 340, 340, 340}, { 340, 10, 200, 180, 40}, { 340, -30, 200, 130, -110}, { 340, -90, 200,
      70, 10}, { 340, 200, 200, 200, 200}},
2284 {{ 340, 340, 340, 340, 340}, { 340, 10, 200, 180, -60}, { 340, 110, 200, 240, 40}, { 340, 200, 200, 200,
      200}, { 340, 110, 200, 240, 40}},
2285 {{ 340, 340, 340, 340, 340}, { 340, -90, 200, 70, 10}, { 340, 200, 200, 200, 200}, { 340, -50, 200, 110,
      -30}, { 340, -90, 200, 0, -350}},
2286 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 80, 200, 210, 10}, { 340, 40, 200, 120,
      -220}, { 340, 110, 200, 190, 30}}}
```

```

2287 },
2288 {
2289 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
2290 { 340, 340, 340, 340}, { 340, 200, 150, 0, 210}, { 340, 200, 40, -150, 70}, { 340, 200, 90,
2291 -30, 190}, { 340, 200, 200, 200}, { 340, 340, 340}, { 340, 200, 90, -100, 110}, { 340, 200, 90, 0, 110}, { 340, 200, 200,
2292 200}, { 340, 200, 90, 0, 110}, { 340, 340, 340, 340}, { 340, 200, 90, -30, 190}, { 340, 200, 200, 200}, { 340, 200, 80, -70,
2293 150}, { 340, 200, -90, -390, 10}, { 340, 340, 340, 340}, { 340, 200, 200, 200}, { 340, 200, 60, -30, 80}, { 340, 200, 40,
2294 -260, 140}, { 340, 200, 0, -10, 30}}
2294 }
2295 },
2296 /* GC....AU */
2297 {
2298 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
2299 { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
2300 { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
2301 { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
2302 { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}
2303 },
2304 {
2305 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
2306 { 340, 340, 340, 340}, { 340, 210, 180, 70, 200}, { 340, 190, 160, 50, 200}, { 340, 90, 60, -50,
200}, { 340, 200, 200, 200}, { 340, 340, 340, 340}, { 340, 200, 170, 60, 200}, { 340, 240, 150, 140, 200}, { 340, 200, 200, 200},
2307 { 340, 240, 150, 140, 200}, { 340, 340, 340, 340}, { 340, 70, -60, 10, 200}, { 340, 200, 200, 200}, { 340, 140, 110, 0,
200}, { 340, 200, 70, 150, 200}, { 340, 340, 340, 340}, { 340, 200, 200, 200}, { 340, 170, 40, 110,
2308 200}, { 340, 200, 70, 150, 200}
2309 },
2310 },
2311 {
2312 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
2313 { 340, 340, 340, 340}, { 340, 190, 250, 200, 250}, { 340, 160, 160, 200, 170}, { 340, 60, 160, 200,
170}, { 340, 200, 200, 200}, { 340, 340, 340, 340}, { 340, 170, 170, 200, 180}, { 340, 160, 160, 200, 160}, { 340, 200, 200, 200},
2314 { 340, 160, 160, 200, 160}, { 340, 340, 340, 340}, { 340, 60, 160, 200, 170}, { 340, 200, 200, 200}, { 340, 120, 180, 200,
180}, { 340, -50, 50, 200, 50}, { 340, 340, 340, 340}, { 340, 200, 200, 200}, { 340, 160, 160, 200}, { 340, 40, 140, 200,
2316 150}, { 340, 80, 80, 200, 80}
2317 },
2318 {
2319 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
2320 { 340, 340, 340, 340}, { 340, 10, 200, 180, 40}, { 340, -10, 200, 150, -90}, { 340, -110, 200,
50, -10}, { 340, 200, 200, 200}, { 340, 340, 340, 340}, { 340, 0, 200, 160, -80}, { 340, 80, 200, 210, 10}, { 340, 200, 200, 200},
2321 { 340, 80, 200, 210, 10}, { 340, 340, 340, 340}, { 340, -110, 200, 50, -10}, { 340, 200, 200, 200}, { 340, -60, 200,
110, -30}, { 340, -50, 200, 40, -310}, { 340, 340, 340, 340}, { 340, 200, 200, 200}, { 340, 80, 200, 210, 10}, { 340, 50, 200, 130,
-210}, { 340, 80, 200, 170, 10}
2322 },
2323 },
2324 },
2325 {
2326 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
2327 { 340, 340, 340, 340}, { 340, 200, 150, 0, 210}, { 340, 200, 60, -130, 90}, { 340, 200, 70,
-50, 170}, { 340, 200, 200, 200}, { 340, 340, 340, 340}, { 340, 200, 70, -120, 100}, { 340, 200, 60, -30, 80}, { 340, 200, 200,
200}, { 340, 200, 60, -30, 80}, { 340, 340, 340, 340}, { 340, 200, 70, -50, 170}, { 340, 200, 200, 200}, { 340, 200, 80, -70,
140}, { 340, 200, -50, -350, 50}, { 340, 340, 340, 340}, { 340, 200, 200, 200}, { 340, 200, 60, -30, 80}, { 340, 200, 50,
-250, 150}, { 340, 200, -20, -30, 0}
2328 },
2329 },
2330 },
2331 },
2332 },
2333 /* GC....UA */
2334 {
2335 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
2336 { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
2337 { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
2338 { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
2339 { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}

```



```

2391 },
2392 {
2393 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2394   340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2395   340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2396   340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2397   340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2398   340}, { 340, 340, 340, 340}}
2399 },
2400 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2401   340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2402   340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2403   340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2404   340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2405   340}}
2406 },
2407 },
2408 /* noPair */ {{{0}}},
2409 /* GU...CG */
2410 {
2411 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2412   340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2413   340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2414   340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2415   340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2416   340}}
2417 },
2418 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2419   340}, { 340, 340, 340, 340}, { 340, 200, 190, 80, 200}, { 340, 190, 180, 70, 200}, { 340, 100,
2420   90, -20, 200}, { 340, 200, 200, 200}, { 340, 340, 340, 340}, { 340, 240, 220, 110, 200},
2421 { 340, 280, 210, 200, 200}, { 340, 200, 200, 200}, { 340, 180, 160, 50, 200}, { 340, 30, -80,
2422   -10, 200}, { 340, 200, 200, 200}, { 340, 270, 190, 180, 200}, { 340, 180, 70, 140, 200},
2423 { 340, 220, 100, 180, 200}}
2424 },
2425 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2426   340}, { 340, 340, 340, 340}, { 340, 180, 230, 200, 230}, { 340, 170, 160, 200, 160}, { 340,
2427   80, 170, 200, 170}, { 340, 200, 200, 200}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
2428 { 340, 210, 200, 200}, { 340, 190, 180, 190}, { 340, 200, 190, 190}, { 340, 200, 190, 190}, { 340,
2429   200, 190, 190}, { 340, 340, 340, 340}, { 340, 200, 200, 200}, { 340, 180, 180, 200}, { 340,
150, 150, 150}, { 340, 90, 90, 200}, { 340, 90, 90, 200}}
2430 },
2431 {
2432 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2433   340}, { 340, 340, 340, 340}, { 340, 80, 200, 130, 160}, { 340, 70, 200, 120, 50}, { 340,
2434   -20, 200, 30, 140}, { 340, 200, 200, 200}, { 340, 200, 200, 200}, { 340, 200, 200, 200},
2435 { 340, 200, 200, 200}, { 340, 180, 200, 200}, { 340, 180, 200, 200}, { 340, 180, 200, 200}, { 340,
2436   180, 200, 200}, { 340, 340, 340, 340}, { 340, 200, 200, 200}, { 340, 180, 200, 200}, { 340,
160, 160}, { 340, 180, 200, 150, 160}}
2437 },
2438 {
2439 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
2440   340}, { 340, 340, 340, 340}, { 340, 200, 230, 60, 190}, { 340, 200, 160, -50, 80}, { 340,
2441   200, 170, 40, 180}, { 340, 200, 200, 200}, { 340, 200, 200, 200}, { 340, 200, 200, 200},
2442 { 340, 200, 200, 200}, { 340, 170, 40, 180}, { 340, 200, 200, 200}, { 340, 200, 210, 40,
2443   170}, { 340, 200, 0, -310, 0}, { 340, 200, 200, 200}, { 340, 180, 70, 100}, { 340, 200, 150,
2444   150}}

```



```

2496 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 310, 230, 220, 200}, { 340, 230, 120, 190,
200}, { 340, 270, 150, 220, 200}}
2497 },
2498 {
2499 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, 
2500 {{ 340, 340, 340, 340}, { 340, 250, 310, 200, 310}, { 340, 230, 220, 200, 220}, { 340, 130, 220, 200,
220}, { 340, 200, 200, 200, 200}}, 
2501 {{ 340, 340, 340, 340}, { 340, 240, 230, 200, 230}, { 340, 220, 220, 200, 220}, { 340, 200, 200, 200,
200}, { 340, 220, 220, 200, 220}}, 
2502 {{ 340, 340, 340, 340}, { 340, 130, 220, 200, 220}, { 340, 200, 200, 200, 200}, { 340, 180, 240, 200,
240}, { 340, 10, 100, 200, 100}}, 
2503 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 220, 220, 200, 220}, { 340, 110, 200, 200,
200}, { 340, 140, 140, 200, 140}}
2504 },
2505 {
2506 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, 
2507 {{ 340, 340, 340, 340}, { 340, 150, 200, 210, 230}, { 340, 130, 200, 180, 110}, { 340, 30, 200, 80,
190}, { 340, 200, 200, 200, 200}}, 
2508 {{ 340, 340, 340, 340}, { 340, 140, 200, 190, 120}, { 340, 220, 200, 240, 200}, { 340, 200, 200, 200,
200}, { 340, 220, 200, 240, 200}}, 
2509 {{ 340, 340, 340, 340}, { 340, 30, 200, 80, 190}, { 340, 200, 200, 200, 200}, { 340, 80, 200, 140,
160}, { 340, 90, 200, 70, -110}}, 
2510 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 220, 200, 240, 200}, { 340, 190, 200, 160,
-10}, { 340, 220, 200, 200, 200}}
2511 },
2512 {
2513 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, 
2514 {{ 340, 340, 340, 340}, { 340, 200, 310, 130, 270}, { 340, 200, 220, 10, 140}, { 340, 200, 220, 90,
220}, { 340, 200, 200, 200, 200}}, 
2515 {{ 340, 340, 340, 340}, { 340, 200, 230, 20, 150}, { 340, 200, 220, 100, 140}, { 340, 200, 200, 200,
200}, { 340, 200, 220, 100, 140}}, 
2516 {{ 340, 340, 340, 340}, { 340, 200, 220, 90, 220}, { 340, 200, 200, 200, 200}, { 340, 200, 240, 70,
200}, { 340, 200, 100, -210, 110}}, 
2517 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 220, 100, 140}, { 340, 200, 200,
-110, 200}, { 340, 200, 140, 110, 60}}
2518 },
2519 },
2520 /* GU....UG */
2521 {{ 
2522 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, 
2523 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, 
2524 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, 
2525 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, 
2526 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}
2527 },
2528 {
2529 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, 
2530 {{ 340, 340, 340, 340}, { 340, 280, 260, 150, 200}, { 340, 230, 220, 110, 200}, { 340, 170, 160,
50, 200}, { 340, 200, 200, 200, 200}}, 
2531 {{ 340, 340, 340, 340}, { 340, 280, 260, 150, 200}, { 340, 340, 260, 250, 200}, { 340, 200, 200, 200,
200}, { 340, 340, 260, 250, 200}}, 
2532 {{ 340, 340, 340, 340}, { 340, 170, 160, 50, 200}, { 340, 200, 200, 200, 200}, { 340, 210, 200, 90,
200}, { 340, 100, -20, 50, 200}}, 
2533 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 310, 230, 220, 200}, { 340, 220, 110, 180,
200}, { 340, 290, 180, 250, 200}}
2534 },
2535 {
2536 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, 
2537 {{ 340, 340, 340, 340}, { 340, 250, 310, 200, 310}, { 340, 210, 200, 200, 200}, { 340, 150, 240, 200,
240}, { 340, 200, 200, 200, 200}}, 
2538 {{ 340, 340, 340, 340}, { 340, 250, 250, 200, 250}, { 340, 250, 250, 200, 250}, { 340, 200, 200, 200,
200}, { 340, 250, 250, 200, 250}}, 
2539 {{ 340, 340, 340, 340}, { 340, 150, 240, 200, 240}, { 340, 200, 200, 200, 200}, { 340, 190, 240, 200,
240}, { 340, 30, 70, 200, 70}}, 
2540 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 220, 220, 200, 220}, { 340, 100, 190, 200,
190}, { 340, 170, 160, 200, 160}}
2541 },
2542 {
2543 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, 
2544 {{ 340, 340, 340, 340}, { 340, 150, 200, 210, 230}, { 340, 110, 200, 160, 90}, { 340, 50, 200, 100,
210}, { 340, 200, 200, 200, 200}}, 
2545 {{ 340, 340, 340, 340}, { 340, 150, 200, 210, 130}, { 340, 250, 200, 270, 230}, { 340, 200, 200, 200,
200}, { 340, 250, 200, 270, 230}}, 
2546 {{ 340, 340, 340, 340}, { 340, 200, 100, 210}, { 340, 200, 200, 200, 200}, { 340, 90, 200, 140,
170}, { 340, 50, 200, 30, -150}}, 
2547 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 220, 200, 240, 200}, { 340, 180, 200, 150,
200}}

```

```

-20}, { 340, 250, 200, 220, 230} }

2548 },
2549 {
2550 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
      340}, { 340, 340, 340, 340, 340}},
2551 {{ 340, 340, 340, 340}, { 340, 200, 310, 130, 270}, { 340, 200, 200, -10, 120}, { 340, 200, 240, 110,
      240}, { 340, 200, 200, 200, 200}},
2552 {{ 340, 340, 340, 340}, { 340, 200, 250, 30, 170}, { 340, 200, 250, 130, 170}, { 340, 200, 200, 200,
      200}, { 340, 200, 250, 130, 170}},
2553 {{ 340, 340, 340, 340}, { 340, 200, 240, 110, 240}, { 340, 200, 200, 200, 200}, { 340, 200, 240, 70,
      200}, { 340, 200, 70, -250, 70}},
2554 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 220, 100, 140}, { 340, 200, 190,
      -120, 190}, { 340, 200, 160, 130, 80}}
2555 },
2556 },
2557 /* GU....AU */
2558 {
2559 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
      340}, { 340, 340, 340, 340, 340}},
2560 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
      340}, { 340, 340, 340, 340, 340}},
2561 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
      340}, { 340, 340, 340, 340, 340}},
2562 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
      340}, { 340, 340, 340, 340, 340}},
2563 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
      340}, { 340, 340, 340, 340, 340}}
2564 },
2565 {
2566 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
      340}, { 340, 340, 340, 340, 340}},
2567 {{ 340, 340, 340, 340}, { 340, 280, 260, 150, 200}, { 340, 250, 240, 130, 200}, { 340, 150, 140,
      30, 200}, { 340, 200, 200, 200, 200}},
2568 {{ 340, 340, 340, 340}, { 340, 260, 250, 140, 200}, { 340, 310, 230, 220, 200}, { 340, 200, 200, 200,
      200}, { 340, 310, 230, 220, 200}},
2569 {{ 340, 340, 340, 340}, { 340, 150, 140, 30, 200}, { 340, 200, 200, 200, 200}, { 340, 210, 190,
      80, 200}, { 340, 130, 20, 90, 200}},
2570 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 310, 230, 220, 200}, { 340, 230, 120, 190,
      200}, { 340, 270, 150, 220, 200}}
2571 },
2572 {
2573 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
      340}, { 340, 340, 340, 340, 340}},
2574 {{ 340, 340, 340, 340}, { 340, 250, 310, 200, 310}, { 340, 230, 220, 200, 220}, { 340, 130, 220, 200,
      220}, { 340, 200, 200, 200, 200}},
2575 {{ 340, 340, 340, 340}, { 340, 240, 230, 200, 230}, { 340, 220, 220, 200, 220}, { 340, 200, 200, 200,
      200}, { 340, 220, 220, 200, 220}},
2576 {{ 340, 340, 340, 340}, { 340, 130, 220, 200, 220}, { 340, 200, 200, 200, 200}, { 340, 180, 240, 200,
      240}, { 340, 10, 100, 200, 100}},
2577 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 220, 220, 200, 220}, { 340, 110, 200, 200,
      200}, { 340, 140, 140, 200, 140}}
2578 },
2579 {
2580 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
      340}, { 340, 340, 340, 340, 340}},
2581 {{ 340, 340, 340, 340}, { 340, 150, 200, 210, 230}, { 340, 130, 200, 180, 110}, { 340, 30, 200,
      80, 190}, { 340, 200, 200, 200, 200}},
2582 {{ 340, 340, 340, 340}, { 340, 140, 200, 190, 120}, { 340, 220, 200, 240, 200}, { 340, 200, 200, 200,
      200}, { 340, 220, 200, 240, 200}},
2583 {{ 340, 340, 340, 340}, { 340, 30, 200, 80, 190}, { 340, 200, 200, 200, 200}, { 340, 80, 200, 140,
      160}, { 340, 90, 200, 70, -110}},
2584 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 220, 200, 240, 200}, { 340, 190, 200, 160,
      -10}, { 340, 220, 200, 200, 200}}
2585 },
2586 {
2587 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
      340}, { 340, 340, 340, 340, 340}},
2588 {{ 340, 340, 340, 340}, { 340, 200, 310, 130, 270}, { 340, 200, 220, 10, 140}, { 340, 200, 220,
      90, 220}, { 340, 200, 200, 200, 200}},
2589 {{ 340, 340, 340, 340}, { 340, 200, 230, 20, 150}, { 340, 200, 220, 100, 140}, { 340, 200, 200, 200,
      200}, { 340, 200, 220, 100, 140}},
2590 {{ 340, 340, 340, 340}, { 340, 200, 220, 90, 220}, { 340, 200, 200, 200, 200}, { 340, 200, 240, 70,
      200}, { 340, 200, 100, -210, 110}},
2591 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 220, 100, 140}, { 340, 200, 200,
      -110, 200}, { 340, 200, 140, 110, 60}}
2592 },
2593 },
2594 /* GU....UA */
2595 {
2596 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
      340}, { 340, 340, 340, 340, 340}},
2597 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
      340}, { 340, 340, 340, 340, 340}},
2598 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
      340}, { 340, 340, 340, 340, 340}},
2599 {{ 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},
      340}, { 340, 340, 340, 340, 340}}}
```



```

    340}, { 340, 340, 340, 340, 340}},  

2652 },  

2653 {  

2654 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}},  

2655 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}},  

2656 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}},  

2657 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}},  

2658 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}}  

2659 },  

2660 {  

2661 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}},  

2662 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}},  

2663 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}},  

2664 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}},  

2665 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}}  

2666 }  

2667 }  

2668 },  

2669 /* noPair */ {{{0}}},  

2670 /* UG...CG */  

2671 {{  

2672 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}},  

2673 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}},  

2674 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}},  

2675 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}},  

2676 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}}  

2677 },  

2678 {  

2679 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}},  

2680 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 100, 200}, { 340, 190, 190, 90, 200}, { 340, 100, 100, 0,  

     200}, { 340, 200, 200, 200, 200}},  

2681 {{ 340, 340, 340, 340, 340}, { 340, 240, 240, 130, 200}, { 340, 280, 220, 220, 200}, { 340, 200, 200, 200,  

     200}, { 340, 270, 210, 200, 200}},  

2682 {{ 340, 340, 340, 340, 340}, { 340, 100, 100, 0, 200}, { 340, 200, 200, 200, 200}, { 340, 180, 180, 70,  

     200}, { 340, 30, -70, 10, 200}},  

2683 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 270, 210, 200, 200}, { 340, 180, 80, 160,  

     200}, { 340, 220, 120, 190, 200}}}  

2684 },  

2685 {  

2686 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}},  

2687 {{ 340, 340, 340, 340, 340}, { 340, 160, 260, 200, 230}, { 340, 150, 190, 200, 160}, { 340, 60, 200, 200,  

     170}, { 340, 200, 200, 200, 200}},  

2688 {{ 340, 340, 340, 340, 340}, { 340, 190, 240, 200, 210}, { 340, 180, 220, 200, 190}, { 340, 200, 200, 200,  

     200}, { 340, 160, 210, 200, 180}},  

2689 {{ 340, 340, 340, 340, 340}, { 340, 60, 200, 200, 170}, { 340, 200, 200, 200, 200}, { 340, 130, 240, 200,  

     210}, { 340, -110, 30, 200, 0}},  

2690 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 160, 210, 200, 180}, { 340, 40, 180, 200,  

     150}, { 340, 70, 120, 200, 90}}}  

2691 },  

2692 {  

2693 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}},  

2694 {{ 340, 340, 340, 340, 340}, { 340, 100, 200, 140, 150}, { 340, 90, 200, 130, 40}, { 340, 0, 200, 40,  

     130}, { 340, 200, 200, 200, 200}},  

2695 {{ 340, 340, 340, 340, 340}, { 340, 130, 200, 170, 80}, { 340, 220, 200, 220, 170}, { 340, 200, 200, 200,  

     200}, { 340, 200, 200, 200, 150}},  

2696 {{ 340, 340, 340, 340, 340}, { 340, 0, 200, 40, 130}, { 340, 200, 200, 200, 200}, { 340, 70, 200, 110,  

     120}, { 340, 10, 200, -30, -220}},  

2697 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 200, 200, 150}, { 340, 160, 200, 120,  

     -70}, { 340, 190, 200, 150, 150}}}  

2698 },  

2699 {  

2700 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

     { 340, 340, 340, 340, 340}},  

2701 {{ 340, 340, 340, 340, 340}, { 340, 200, 260, 20, 220}, { 340, 200, 190, -90, 110}, { 340, 200, 200, 0,  

     200}, { 340, 200, 200, 200, 200}},  

2702 {{ 340, 340, 340, 340, 340}, { 340, 200, 240, -40, 150}, { 340, 200, 220, 40, 140}, { 340, 200, 200, 200,  

     200}, { 340, 200, 210, 30, 120}},  

2703 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 0, 200}, { 340, 200, 200, 200, 200}, { 340, 200, 240, 0,  

     190}, { 340, 200, 30, -350, 30}}},
```

```

2704 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 210, 30, 120}, { 340, 200, 180,
-200, 180}, { 340, 200, 120, 20, 30}}
2705 }
2706 },
2707 /* UG....GC */
2708 {
2709 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, ,
2710 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, ,
2711 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, ,
2712 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, ,
2713 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340}}},
2714 },
2715 {
2716 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, ,
2717 {{ 340, 340, 340, 340, 340}, { 340, 210, 210, 110, 200}, { 340, 190, 190, 80, 200}, { 340, 10, 10, -90,
200}, { 340, 200, 200, 200, 200}}, ,
2718 {{ 340, 340, 340, 340, 340}, { 340, 180, 180, 80, 200}, { 340, 250, 190, 180, 200}, { 340, 200, 200, 200,
200}, { 340, 150, 90, 90, 200}}, ,
2719 {{ 340, 340, 340, 340, 340}, { 340, 70, 70, -30, 200}, { 340, 200, 200, 200, 200}, { 340, 180, 180, 70,
200}, { 340, 0, -100, -30, 200}}, ,
2720 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 250, 190, 190, 200}, { 340, 40, -60, 10,
200}, { 340, 210, 110, 190, 200}}},
2721 },
2722 {
2723 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, ,
2724 {{ 340, 340, 340, 340, 340}, { 340, 170, 270, 200, 240}, { 340, 140, 190, 200, 160}, { 340, -30, 110, 200,
80}, { 340, 200, 200, 200, 200}}, ,
2725 {{ 340, 340, 340, 340, 340}, { 340, 140, 180, 200, 150}, { 340, 140, 190, 200, 160}, { 340, 200, 200, 200,
200}, { 340, 40, 90, 200, 60}}, ,
2726 {{ 340, 340, 340, 340, 340}, { 340, 30, 170, 200, 140}, { 340, 200, 200, 200, 200}, { 340, 130, 240, 200,
210}, { 340, -150, 0, 200, -30}}, ,
2727 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 150, 190, 200, 160}, { 340, -110, 40,
200, 10}, { 340, 70, 110, 200, 80}}},
2728 },
2729 {
2730 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, ,
2731 {{ 340, 340, 340, 340, 340}, { 340, 110, 200, 150, 160}, { 340, 80, 200, 120, 30}, { 340, -90, 200, -50,
40}, { 340, 200, 200, 200, 200}}, ,
2732 {{ 340, 340, 340, 340, 340}, { 340, 80, 200, 120, 30}, { 340, 180, 200, 180, 130}, { 340, 200, 200, 200,
200}, { 340, 90, 200, 80, 40}}, ,
2733 {{ 340, 340, 340, 340, 340}, { 340, -30, 200, 10, 100}, { 340, 200, 200, 200, 200}, { 340, 70, 200, 110,
120}, { 340, -30, 200, -70, -260}}, ,
2734 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 190, 200, 190, 140}, { 340, 10, 200, -30,
-220}, { 340, 190, 200, 150, 140}}},
2735 },
2736 {
2737 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, ,
2738 {{ 340, 340, 340, 340, 340}, { 340, 200, 270, 30, 230}, { 340, 200, 190, -90, 100}, { 340, 200, 110, -90,
110}, { 340, 200, 200, 200, 200}}, ,
2739 {{ 340, 340, 340, 340, 340}, { 340, 200, 180, -100, 100}, { 340, 200, 190, 10, 100}, { 340, 200, 200,
200}, { 340, 200, 90, 90, 0}}, ,
2740 {{ 340, 340, 340, 340, 340}, { 340, 200, 170, -30, 170}, { 340, 200, 200, 200, 200}, { 340, 200, 240, 0,
190}, { 340, 200, 0, -390, -10}}, ,
2741 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 190, 10, 110}, { 340, 200, 40,
-350, 30}, { 340, 200, 110, 10, 30}}},
2742 },
2743 },
2744 /* UG....GU */
2745 {
2746 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, ,
2747 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, ,
2748 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, ,
2749 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, ,
2750 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}},
2751 },
2752 {
2753 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}, ,
2754 {{ 340, 340, 340, 340, 340}, { 340, 280, 280, 170, 200}, { 340, 250, 250, 150, 200}, { 340, 150, 150,
50, 200}, { 340, 200, 200, 200, 200}}, ,
2755 {{ 340, 340, 340, 340, 340}, { 340, 260, 260, 160, 200}, { 340, 310, 250, 240, 200}, { 340, 200, 200, 200,
200}, { 340, 310, 250, 240, 200}}, ,
2756 {{ 340, 340, 340, 340, 340}, { 340, 150, 150, 50, 200}, { 340, 200, 200, 200, 200}, { 340, 210, 210, 100,
100}, { 340, 210, 210, 100, 100}}}

```

```

    200}, { 340, 130, 30, 110, 200}},  

2757 {{ 340, 340, 340, 340}, { 340, 200, 200, 200}, { 340, 310, 250, 240, 200}, { 340, 230, 130, 210,  

    200}, { 340, 270, 170, 240, 200}}  

2758 },  

2759 {  

2760 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

2761 {{ 340, 340, 340, 340}, { 340, 230, 340, 200, 310}, { 340, 210, 250, 200, 220}, { 340, 110, 250, 200,  

    220}, { 340, 200, 200, 200, 200}},  

2762 {{ 340, 340, 340, 340}, { 340, 220, 260, 200, 230}, { 340, 200, 250, 200, 220}, { 340, 200, 200, 200,  

    200}, { 340, 200, 250, 200, 220}},  

2763 {{ 340, 340, 340, 340}, { 340, 110, 250, 200, 220}, { 340, 200, 200, 200, 200}, { 340, 160, 270, 200,  

    240}, { 340, -10, 130, 200, 100}},  

2764 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 250, 200, 220}, { 340, 90, 230, 200,  

    200}, { 340, 120, 170, 200, 140}}  

2765 },  

2766 {  

2767 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

2768 {{ 340, 340, 340, 340}, { 340, 170, 200, 210, 220}, { 340, 150, 200, 190, 100}, { 340, 50, 200, 90,  

    180}, { 340, 200, 200, 200, 200}},  

2769 {{ 340, 340, 340, 340}, { 340, 160, 200, 200, 110}, { 340, 240, 200, 240, 190}, { 340, 200, 200, 200,  

    200}, { 340, 240, 200, 240, 190}},  

2770 {{ 340, 340, 340, 340}, { 340, 50, 200, 90, 180}, { 340, 200, 200, 200, 200}, { 340, 100, 200, 140,  

    150}, { 340, 110, 200, 70, -120}},  

2771 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 240, 200, 240, 190}, { 340, 210, 200, 170,  

    -20}, { 340, 240, 200, 200, 190}}  

2772 },  

2773 {  

2774 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

2775 {{ 340, 340, 340, 340}, { 340, 200, 340, 100, 290}, { 340, 200, 250, -30, 170}, { 340, 200, 250, 50,  

    250}, { 340, 200, 200, 200, 200}},  

2776 {{ 340, 340, 340, 340}, { 340, 200, 260, -20, 180}, { 340, 200, 250, 70, 160}, { 340, 200, 200, 200,  

    200}, { 340, 200, 250, 70, 160}},  

2777 {{ 340, 340, 340, 340}, { 340, 200, 250, 50, 250}, { 340, 200, 200, 200, 200}, { 340, 200, 270, 30,  

    220}, { 340, 200, 130, -250, 130}},  

2778 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 250, 70, 160}, { 340, 200, 230,  

    -150, 230}, { 340, 200, 170, 70, 80}}  

2779 }  

2780 },  

2781 /* UG....UG */  

2782 {{  

2783 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

2784 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

2785 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

2786 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

2787 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

2788 },  

2789 {  

2790 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

2791 {{ 340, 340, 340, 340}, { 340, 280, 280, 170, 200}, { 340, 230, 230, 130, 200}, { 340, 170, 170, 70,  

    200}, { 340, 200, 200, 200, 200}},  

2792 {{ 340, 340, 340, 340}, { 340, 280, 280, 170, 200}, { 340, 340, 280, 270, 200}, { 340, 200, 200, 200,  

    200}, { 340, 340, 280, 270, 200}},  

2793 {{ 340, 340, 340, 340}, { 340, 170, 170, 70, 200}, { 340, 200, 200, 200, 200}, { 340, 210, 210, 110,  

    200}, { 340, 100, 0, 70, 200}},  

2794 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 310, 250, 240, 200}, { 340, 220, 120, 200,  

    200}, { 340, 290, 190, 270, 200}}  

2795 },  

2796 {  

2797 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

2798 {{ 340, 340, 340, 340}, { 340, 230, 340, 200, 310}, { 340, 190, 230, 200, 200}, { 340, 130, 270, 200,  

    240}, { 340, 200, 200, 200, 200}},  

2799 {{ 340, 340, 340, 340}, { 340, 230, 280, 200, 250}, { 340, 230, 280, 200, 250}, { 340, 200, 200, 200,  

    200}, { 340, 230, 280, 200, 250}},  

2800 {{ 340, 340, 340, 340}, { 340, 130, 270, 200, 240}, { 340, 200, 200, 200, 200}, { 340, 170, 270, 200,  

    240}, { 340, -50, 100, 200, 70}},  

2801 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 250, 200, 220}, { 340, 80, 220, 200,  

    190}, { 340, 150, 190, 200, 160}}  

2802 },  

2803 {  

2804 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

2805 {{ 340, 340, 340, 340}, { 340, 170, 200, 210, 220}, { 340, 130, 200, 170, 80}, { 340, 70, 200, 110,  

    200}, { 340, 200, 200, 200, 200}},  

2806 {{ 340, 340, 340, 340}, { 340, 170, 200, 210, 120}, { 340, 270, 200, 270, 220}, { 340, 200, 200, 200,  

    200}, { 340, 270, 200, 270, 220}},  

2807 {{ 340, 340, 340, 340}, { 340, 70, 200, 110, 200}, { 340, 200, 200, 200, 200}, { 340, 110, 200, 150,  

    160}, { 340, 70, 200, 30, -160}}},
```



```

    340}, { 340, 340, 340, 340, 340}},  

2861 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}}}  

2862 },  

2863 {  

2864 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}},  

2865 {{ 340, 340, 340, 340}, { 340, 280, 280, 170, 200}, { 340, 230, 230, 130, 200}, { 340, 170, 170,  

    70, 200}, { 340, 200, 200, 200, 200}},  

2866 {{ 340, 340, 340, 340}, { 340, 280, 280, 170, 200}, { 340, 340, 280, 270, 200}, { 340, 200, 200,  

    200}, { 340, 340, 280, 270, 200}},  

2867 {{ 340, 340, 340, 340}, { 340, 170, 170, 70, 200}, { 340, 200, 200, 200, 200}, { 340, 210, 210,  

    110, 200}, { 340, 100, 0, 70, 200}},  

2868 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 310, 250, 240, 200}, { 340, 220, 120,  

    200}, { 340, 290, 190, 270, 200}}}  

2869 },  

2870 {  

2871 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}},  

2872 {{ 340, 340, 340, 340}, { 340, 230, 230, 130, 200}, { 340, 190, 230, 200, 200}, { 340, 130, 270,  

    200, 240}, { 340, 200, 200, 200, 200}},  

2873 {{ 340, 340, 340, 340}, { 340, 230, 280, 200, 250}, { 340, 230, 280, 200, 250}, { 340, 200, 200,  

    200}, { 340, 230, 280, 200, 250}},  

2874 {{ 340, 340, 340, 340}, { 340, 130, 270, 200, 240}, { 340, 200, 200, 200, 200}, { 340, 170, 270,  

    200, 240}, { 340, -50, 100, 200, 70}},  

2875 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 250, 200, 220}, { 340, 80, 220,  

    200, 190}, { 340, 150, 190, 200, 160}}}  

2876 },  

2877 {  

2878 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}},  

2879 {{ 340, 340, 340, 340}, { 340, 170, 200, 210, 220}, { 340, 130, 200, 170, 80}, { 340, 70, 200,  

    110, 200}, { 340, 200, 200, 200, 200}},  

2880 {{ 340, 340, 340, 340}, { 340, 170, 200, 210, 120}, { 340, 270, 200, 270, 220}, { 340, 200, 200,  

    200}, { 340, 270, 200, 270, 220}},  

2881 {{ 340, 340, 340, 340}, { 340, 70, 200, 110, 200}, { 340, 200, 200, 200, 200}, { 340, 110, 200,  

    150, 160}, { 340, 70, 200, 30, -160}},  

2882 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 240, 200, 240, 190}, { 340, 200, 200,  

    160, -30}, { 340, 270, 200, 230, 220}}}  

2883 },  

2884 {  

2885 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}},  

2886 {{ 340, 340, 340, 340}, { 340, 200, 340, 100, 290}, { 340, 200, 230, -50, 150}, { 340, 200, 270,  

    70, 270}, { 340, 200, 200, 200, 200}},  

2887 {{ 340, 340, 340, 340}, { 340, 200, 280, 0, 190}, { 340, 200, 280, 100, 190}, { 340, 200, 200,  

    200}, { 340, 200, 280, 100, 190}},  

2888 {{ 340, 340, 340, 340}, { 340, 200, 270, 70, 270}, { 340, 200, 200, 200, 200}, { 340, 200, 270,  

    30, 230}, { 340, 200, 100, -290, 90}},  

2889 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 250, 70, 160}, { 340, 200, 220,  

    -160, 220}, { 340, 200, 190, 90, 110}}}  

2890 }  

2891 },  

2892 /* UG....?? */  

2893 {{  

2894 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}},  

2895 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}},  

2896 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}},  

2897 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}},  

2898 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}}}  

2899 },  

2900 {  

2901 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}},  

2902 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}},  

2903 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}},  

2904 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}},  

2905 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}}}  

2906 },  

2907 {  

2908 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}},  

2909 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}},  

2910 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}},  

2911 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

    340}, { 340, 340, 340, 340}}},
```



```

170}, { 340, 200, 0, -310, 0},  

2965 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 180, 70, 100}, { 340, 200, 150,  

-160, 160}, { 340, 200, 90, 60, 10}}}  

2966 },  

2967 },  

2968 /* AU....GC */  

2969 {{  

2970 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340,  

340}, { 340, 340, 340, 340, 340}},  

2971 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340,  

340}, { 340, 340, 340, 340, 340}},  

2972 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340,  

340}, { 340, 340, 340, 340, 340}},  

2973 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340,  

340}, { 340, 340, 340, 340, 340}},  

2974 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340,  

340}, { 340, 340, 340, 340, 340}}}  

2975 },  

2976 {  

2977 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340,  

340}, { 340, 340, 340, 340, 340}},  

2978 {{ 340, 340, 340, 340, 340}, { 340, 210, 200, 90, 200}, { 340, 190, 170, 60, 200}, { 340, 10, 0,  

-110, 200}, { 340, 200, 200, 200, 200}},  

2979 {{ 340, 340, 340, 340, 340}, { 340, 180, 170, 60, 200}, { 340, 250, 170, 160, 200}, { 340, 200, 200,  

200}, { 340, 150, 70, 70, 200}},  

2980 {{ 340, 340, 340, 340, 340}, { 340, 70, 60, -50, 200}, { 340, 200, 200, 200, 200}, { 340, 180, 160,  

50, 200}, { 340, 0, -120, -50, 200}},  

2981 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 250, 180, 170, 200}, { 340, 40, -80,  

-10, 200}, { 340, 210, 100, 170, 200}}}  

2982 },  

2983 {  

2984 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340,  

340}, { 340, 340, 340, 340, 340}},  

2985 {{ 340, 340, 340, 340, 340}, { 340, 190, 240, 200, 240}, { 340, 160, 160, 200, 160}, { 340, -10, 80,  

200, 80}, { 340, 200, 200, 200, 200}},  

2986 {{ 340, 340, 340, 340, 340}, { 340, 160, 150, 200, 150}, { 340, 160, 160, 200, 160}, { 340, 200, 200,  

200}, { 340, 60, 60, 200, 60}},  

2987 {{ 340, 340, 340, 340, 340}, { 340, 50, 140, 200, 140}, { 340, 200, 200, 200, 200}, { 340, 150, 210,  

200, 210}, { 340, -130, -30, 200, -30}},  

2988 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 170, 160, 200, 160}, { 340, -90,  

10, 200, 10}, { 340, 90, 80, 200, 80}}}  

2989 },  

2990 {  

2991 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340,  

340}, { 340, 340, 340, 340, 340}},  

2992 {{ 340, 340, 340, 340, 340}, { 340, 90, 200, 140, 170}, { 340, 60, 200, 120, 40}, { 340, -110, 200,  

-60, 50}, { 340, 200, 200, 200, 200}},  

2993 {{ 340, 340, 340, 340, 340}, { 340, 60, 200, 110, 40}, { 340, 160, 200, 180, 140}, { 340, 200, 200,  

200}, { 340, 70, 200, 80, 50}},  

2994 {{ 340, 340, 340, 340, 340}, { 340, -50, 200, 0, 110}, { 340, 200, 200, 200, 200}, { 340, 50, 200, 110,  

130}, { 340, -50, 200, -70, -250}},  

2995 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 170, 200, 180, 150}, { 340, -10, 200,  

-30, -210}, { 340, 170, 200, 140, 150}}}  

2996 },  

2997 {  

2998 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340,  

340}, { 340, 340, 340, 340, 340}},  

2999 {{ 340, 340, 340, 340, 340}, { 340, 200, 240, 70, 200}, { 340, 200, 160, -50, 80}, { 340, 200, 80,  

-50, 80}, { 340, 200, 200, 200, 200}},  

3000 {{ 340, 340, 340, 340, 340}, { 340, 200, 150, -60, 70}, { 340, 200, 160, 50, 80}, { 340, 200, 200,  

200}, { 340, 200, 60, -50, -20}},  

3001 {{ 340, 340, 340, 340, 340}, { 340, 200, 140, 10, 150}, { 340, 200, 200, 200, 200}, { 340, 200, 210,  

40, 170}, { 340, 200, -30, -350, -30}},  

3002 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 160, 50, 80}, { 340, 200, 10,  

-310, 10}, { 340, 200, 80, 50, 0}}}  

3003 }  

3004 },  

3005 /* AU....GU */  

3006 {{  

3007 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340,  

340}, { 340, 340, 340, 340, 340}},  

3008 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340,  

340}, { 340, 340, 340, 340, 340}},  

3009 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340,  

340}, { 340, 340, 340, 340, 340}},  

3010 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340,  

340}, { 340, 340, 340, 340, 340}},  

3011 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340,  

340}, { 340, 340, 340, 340, 340}},  

3012 },  

3013 {  

3014 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340,  

340}, { 340, 340, 340, 340, 340}},  

3015 {{ 340, 340, 340, 340, 340}, { 340, 280, 260, 150, 200}, { 340, 250, 240, 130, 200}, { 340, 150, 140,  

30, 200}, { 340, 200, 200, 200, 200}},  

3016 {{ 340, 340, 340, 340, 340}, { 340, 260, 250, 140, 200}, { 340, 310, 230, 220, 200}, { 340, 200, 200,  

200}, { 340, 310, 230, 220, 200}}}

```

```

3017 {{ 340, 340, 340, 340}, { 340, 150, 140, 30, 200}, { 340, 200, 200, 200, 200}, { 340, 210, 190, 80,
200}, { 340, 130, 20, 90, 200}},
3018 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 310, 230, 220, 200}, { 340, 230, 120, 190,
200}, { 340, 270, 150, 220, 200}}
3019 },
3020 {
3021 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},
3022 {{ 340, 340, 340, 340}, { 340, 250, 310, 200, 310}, { 340, 230, 220, 200, 220}, { 340, 130, 220, 200,
220}, { 340, 200, 200, 200, 200}},
3023 {{ 340, 340, 340, 340}, { 340, 240, 230, 200, 230}, { 340, 220, 220, 200, 220}, { 340, 200, 200, 200,
200}, { 340, 220, 220, 200, 220}},
3024 {{ 340, 340, 340, 340}, { 340, 130, 220, 200, 220}, { 340, 200, 200, 200, 200}, { 340, 180, 240, 200,
240}, { 340, 10, 100, 200, 100}},
3025 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 220, 220, 200, 220}, { 340, 110, 200, 200,
200}, { 340, 140, 140, 200, 140}}
3026 },
3027 {
3028 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},
3029 {{ 340, 340, 340, 340}, { 340, 150, 200, 210, 230}, { 340, 130, 200, 180, 110}, { 340, 30, 200, 80,
190}, { 340, 200, 200, 200, 200}},
3030 {{ 340, 340, 340, 340}, { 340, 140, 200, 190, 120}, { 340, 220, 200, 240, 200}, { 340, 200, 200, 200,
200}, { 340, 220, 200, 240, 200}},
3031 {{ 340, 340, 340, 340}, { 340, 30, 200, 80, 190}, { 340, 200, 200, 200, 200}, { 340, 80, 200, 140,
160}, { 340, 90, 200, 70, -110}},
3032 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 220, 200, 240, 200}, { 340, 190, 200, 160,
-10}, { 340, 220, 200, 200, 200}}
3033 },
3034 {
3035 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},
3036 {{ 340, 340, 340, 340}, { 340, 200, 310, 130, 270}, { 340, 200, 220, 10, 140}, { 340, 200, 220, 90,
220}, { 340, 200, 200, 200, 200}},
3037 {{ 340, 340, 340, 340}, { 340, 200, 230, 20, 150}, { 340, 200, 220, 100, 140}, { 340, 200, 200, 200,
200}, { 340, 200, 220, 100, 140}},
3038 {{ 340, 340, 340, 340}, { 340, 200, 220, 90, 220}, { 340, 200, 200, 200, 200}, { 340, 200, 240, 70,
200}, { 340, 200, 100, -210, 110}},
3039 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 220, 100, 140}, { 340, 200, 200,
-110, 200}, { 340, 200, 140, 110, 60}}
3040 }
3041 },
3042 /* AU....UG */
3043 {
3044 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},
3045 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},
3046 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},
3047 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},
3048 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}}
3049 },
3050 {
3051 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},
3052 {{ 340, 340, 340, 340}, { 340, 280, 260, 150, 200}, { 340, 230, 220, 110, 200}, { 340, 170, 160, 50,
200}, { 340, 200, 200, 200, 200}},
3053 {{ 340, 340, 340, 340}, { 340, 280, 260, 150, 200}, { 340, 340, 260, 250, 200}, { 340, 200, 200, 200,
200}, { 340, 340, 260, 250, 200}},
3054 {{ 340, 340, 340, 340}, { 340, 170, 160, 50, 200}, { 340, 200, 200, 200, 200}, { 340, 210, 200, 90,
200}, { 340, 100, -20, 50, 200}},
3055 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 310, 230, 220, 200}, { 340, 220, 110, 180,
200}, { 340, 290, 180, 250, 200}}
3056 },
3057 {
3058 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},
3059 {{ 340, 340, 340, 340}, { 340, 250, 310, 200, 310}, { 340, 210, 200, 200, 200}, { 340, 150, 240, 200,
240}, { 340, 200, 200, 200, 200}},
3060 {{ 340, 340, 340, 340}, { 340, 250, 250, 200, 250}, { 340, 250, 250, 200, 250}, { 340, 200, 200, 200,
200}, { 340, 250, 250, 200, 250}},
3061 {{ 340, 340, 340, 340}, { 340, 150, 240, 200, 240}, { 340, 200, 200, 200, 200}, { 340, 190, 240, 200,
240}, { 340, -30, 70, 200, 70}},
3062 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 220, 220, 200, 220}, { 340, 100, 190, 200,
190}, { 340, 170, 160, 200, 160}}
3063 },
3064 {
3065 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},
3066 {{ 340, 340, 340, 340}, { 340, 150, 200, 210, 230}, { 340, 110, 200, 160, 90}, { 340, 50, 200, 100,
210}, { 340, 200, 200, 200, 200}},
3067 {{ 340, 340, 340, 340}, { 340, 150, 200, 210, 130}, { 340, 250, 200, 270, 230}, { 340, 200, 200, 200,
200}, { 340, 250, 200, 270, 230}},
3068 {{ 340, 340, 340, 340}, { 340, 50, 200, 100, 210}, { 340, 200, 200, 200, 200}, { 340, 90, 200, 140,
200}

```

```

    170}, { 340, 50, 200, 30, -150}},
3069 {{ 340, 340, 340, 340}, { 340, 200, 200, 200}, { 340, 220, 200, 240, 200}, { 340, 180, 200, 150,
-20}, { 340, 250, 200, 220, 230}}
3070 },
3071 {
3072 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
3073 {{ 340, 340, 340, 340}, { 340, 200, 310, 130, 270}, { 340, 200, 200, -10, 120}, { 340, 200, 240, 110,
240}, { 340, 200, 200, 200}},
3074 {{ 340, 340, 340, 340}, { 340, 200, 250, 30, 170}, { 340, 200, 250, 130, 170}, { 340, 200, 200, 200,
200}, { 340, 200, 250, 130, 170}},
3075 {{ 340, 340, 340, 340}, { 340, 200, 240, 110, 240}, { 340, 200, 200, 200, 200}, { 340, 200, 240, 70,
200}, { 340, 200, 70, -250, 70}},
3076 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 220, 100, 140}, { 340, 200, 190,
-120, 190}, { 340, 200, 160, 130, 80}}
3077 },
3078 },
3079 /* AU....AU */
3080 {
3081 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
3082 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
3083 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
3084 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
3085 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
3086 },
3087 {
3088 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
3089 {{ 340, 340, 340, 340}, { 340, 280, 260, 150, 200}, { 340, 250, 240, 130, 200}, { 340, 150, 140, 30,
200}, { 340, 200, 200, 200, 200}},
3090 {{ 340, 340, 340, 340}, { 340, 260, 250, 140, 200}, { 340, 310, 230, 220, 200}, { 340, 200, 200, 200,
200}, { 340, 310, 230, 220, 200}},
3091 {{ 340, 340, 340, 340}, { 340, 150, 140, 30, 200}, { 340, 200, 200, 200, 200}, { 340, 210, 190, 80,
200}, { 340, 130, 20, 90, 200}},
3092 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 310, 230, 220, 200}, { 340, 230, 120, 190,
200}, { 340, 270, 150, 220, 200}}
3093 },
3094 {
3095 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
3096 {{ 340, 340, 340, 340}, { 340, 250, 310, 200, 310}, { 340, 230, 220, 200, 220}, { 340, 130, 220, 200,
220}, { 340, 200, 200, 200, 200}},
3097 {{ 340, 340, 340, 340}, { 340, 240, 230, 200, 230}, { 340, 220, 220, 200, 220}, { 340, 200, 200, 200,
200}, { 340, 220, 220, 200, 220}},
3098 {{ 340, 340, 340, 340}, { 340, 130, 220, 200, 220}, { 340, 200, 200, 200, 200}, { 340, 180, 240, 200,
240}, { 340, 10, 100, 200, 100}},
3099 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 220, 220, 200, 220}, { 340, 110, 200, 200,
200}, { 340, 140, 140, 200, 140}}
3100 },
3101 {
3102 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
3103 {{ 340, 340, 340, 340}, { 340, 150, 200, 210, 230}, { 340, 130, 200, 180, 110}, { 340, 30, 200, 80,
190}, { 340, 200, 200, 200, 200}},
3104 {{ 340, 340, 340, 340}, { 340, 140, 200, 190, 120}, { 340, 220, 200, 240, 200}, { 340, 200, 200, 200,
200}, { 340, 220, 200, 240, 200}},
3105 {{ 340, 340, 340, 340}, { 340, 30, 200, 80, 190}, { 340, 200, 200, 200, 200}, { 340, 80, 200, 140,
160}, { 340, 90, 200, 70, -110}},
3106 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 220, 200, 240, 200}, { 340, 190, 200, 160,
-10}, { 340, 220, 200, 200, 200}}
3107 },
3108 {
3109 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
3110 {{ 340, 340, 340, 340}, { 340, 200, 310, 130, 270}, { 340, 200, 220, 10, 140}, { 340, 200, 220, 90,
220}, { 340, 200, 200, 200, 200}},
3111 {{ 340, 340, 340, 340}, { 340, 200, 230, 20, 150}, { 340, 200, 220, 100, 140}, { 340, 200, 200, 200,
200}, { 340, 200, 220, 100, 140}},
3112 {{ 340, 340, 340, 340}, { 340, 200, 220, 90, 220}, { 340, 200, 200, 200, 200}, { 340, 200, 240, 70,
200}, { 340, 200, 100, -210, 110}},
3113 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 220, 100, 140}, { 340, 200, 200,
-110, 200}, { 340, 200, 140, 110, 60}}
3114 },
3115 },
3116 /* AU....UA */
3117 {
3118 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
3119 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
3120 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}}}
```



```

340}, { 340, 340, 340, 340, 340}},  

3173 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}}  

3174 },  

3175 {  

3176 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3177 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3178 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3179 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3180 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3181 },  

3182 {  

3183 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3184 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3185 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3186 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3187 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3188 }  

3189 }  

3190 },  

3191 /* noPair */ {{{0}}},  

3192 /* UA...CG */  

3193 {  

3194 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3195 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3196 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3197 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3198 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3199 },  

3200 {  

3201 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3202 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 100, 200}, { 340, 190, 190, 90, 200}, { 340, 100, 100, 0, 200},  

3203 {{ 340, 340, 340, 340, 340}, { 340, 240, 240, 130, 200}, { 340, 280, 220, 220, 200}, { 340, 200, 200, 200, 200},  

3204 {{ 340, 340, 340, 340, 340}, { 340, 100, 100, 0, 200}, { 340, 200, 200, 200, 200}, { 340, 180, 180, 70, 200},  

3205 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 270, 210, 200, 200}, { 340, 180, 80, 160, 200},  

3206 },  

3207 {  

3208 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3209 {{ 340, 340, 340, 340, 340}, { 340, 160, 260, 200, 230}, { 340, 150, 190, 200, 160}, { 340, 60, 200, 200, 170},  

3210 {{ 340, 340, 340, 340, 340}, { 340, 190, 240, 200, 210}, { 340, 180, 220, 200, 190}, { 340, 200, 200, 200, 200},  

3211 {{ 340, 340, 340, 340, 340}, { 340, 60, 200, 200, 170}, { 340, 200, 200, 200, 200}, { 340, 130, 240, 200, 210},  

3212 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 160, 210, 200, 180}, { 340, 40, 180, 200, 150},  

3213 },  

3214 {  

3215 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3216 {{ 340, 340, 340, 340, 340}, { 340, 100, 200, 140, 150}, { 340, 90, 200, 130, 40}, { 340, 0, 200, 40, 130},  

3217 {{ 340, 340, 340, 340, 340}, { 340, 130, 200, 170, 80}, { 340, 220, 200, 220, 170}, { 340, 200, 200, 200, 200},  

3218 {{ 340, 340, 340, 340, 340}, { 340, 0, 200, 40, 130}, { 340, 200, 200, 200, 200}, { 340, 70, 200, 110, 120},  

3219 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 200, 200, 150}, { 340, 160, 200, 120, 70},  

3220 },  

3221 {  

3222 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340},  

3223 {{ 340, 340, 340, 340, 340}, { 340, 200, 260, 20, 220}, { 340, 200, 190, -90, 110}, { 340, 200, 200, 0, 200},  

3224 {{ 340, 340, 340, 340, 340}, { 340, 200, 240, -40, 150}, { 340, 200, 220, 40, 140}, { 340, 200, 200, 200, 200},  

3225 {{ 340, 340, 340, 340, 340}, { 340, 200, 210, 30, 120}}

```

```

3225 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 0, 200}, { 340, 200, 200, 200, 200}, { 340, 200, 240, 0,
190}, { 340, 200, 30, -350, 30}},
3226 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 210, 30, 120}, { 340, 200, 180,
-200, 180}, { 340, 200, 120, 20, 30}}
3227 },
3228 },
3229 /* UA...GC */
3230 {
3231 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}},
3232 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}},
3233 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}},
3234 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}},
3235 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}
3236 },
3237 {
3238 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}},
3239 {{ 340, 340, 340, 340, 340}, { 340, 210, 210, 110, 200}, { 340, 190, 190, 80, 200}, { 340, 10, 10, -90,
200}, { 340, 200, 200, 200, 200}},
3240 {{ 340, 340, 340, 340, 340}, { 340, 180, 180, 80, 200}, { 340, 250, 190, 180, 200}, { 340, 200, 200, 200,
200}, { 340, 150, 90, 90, 200}},
3241 {{ 340, 340, 340, 340, 340}, { 340, 70, 70, -30, 200}, { 340, 200, 200, 200, 200}, { 340, 180, 180, 70,
200}, { 340, 0, -100, -30, 200}},
3242 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 250, 190, 190, 200}, { 340, 40, -60, 10,
200}, { 340, 210, 110, 190, 200}}
3243 },
3244 {
3245 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}},
3246 {{ 340, 340, 340, 340, 340}, { 340, 170, 270, 200, 240}, { 340, 140, 190, 200, 160}, { 340, -30, 110, 200,
80}, { 340, 200, 200, 200, 200}},
3247 {{ 340, 340, 340, 340, 340}, { 340, 140, 180, 200, 150}, { 340, 140, 190, 200, 160}, { 340, 200, 200, 200,
200}, { 340, 40, 90, 200, 60}},
3248 {{ 340, 340, 340, 340, 340}, { 340, 30, 170, 200, 140}, { 340, 200, 200, 200, 200}, { 340, 130, 240, 200,
210}, { 340, -150, 0, 200, -30}},
3249 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 150, 190, 200, 160}, { 340, -110, 40,
200, 10}, { 340, 70, 110, 200, 80}}
3250 },
3251 {
3252 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}},
3253 {{ 340, 340, 340, 340, 340}, { 340, 110, 200, 150, 160}, { 340, 80, 200, 120, 30}, { 340, -90, 200, -50,
40}, { 340, 200, 200, 200, 200}},
3254 {{ 340, 340, 340, 340, 340}, { 340, 80, 200, 120, 30}, { 340, 180, 200, 180, 130}, { 340, 200, 200, 200,
200}, { 340, 90, 200, 80, 40}},
3255 {{ 340, 340, 340, 340, 340}, { 340, -30, 200, 10, 100}, { 340, 200, 200, 200, 200}, { 340, 70, 200, 110,
120}, { 340, -30, 200, -70, -260}},
3256 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 190, 200, 190, 140}, { 340, 10, 200, -30,
-220}, { 340, 190, 200, 150, 140}}
3257 },
3258 {
3259 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}},
3260 {{ 340, 340, 340, 340, 340}, { 340, 200, 270, 30, 230}, { 340, 200, 190, -90, 100}, { 340, 200, 110, -90,
110}, { 340, 200, 200, 200, 200}},
3261 {{ 340, 340, 340, 340, 340}, { 340, 200, 180, -100, 100}, { 340, 200, 190, 10, 100}, { 340, 200, 200,
200, 200}, { 340, 200, 90, -90, 0}},
3262 {{ 340, 340, 340, 340, 340}, { 340, 200, 170, -30, 170}, { 340, 200, 200, 200, 200}, { 340, 200, 240, 0,
190}, { 340, 200, 0, -390, -10}},
3263 {{ 340, 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 190, 10, 110}, { 340, 200, 40,
-350, 30}, { 340, 200, 110, 10, 30}}
3264 },
3265 },
3266 /* UA...GU */
3267 {
3268 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}},
3269 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}},
3270 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}},
3271 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}},
3272 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}}
3273 },
3274 {
3275 {{ 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340, 340, 340}, { 340, 340, 340,
340}, { 340, 340, 340, 340, 340}},
3276 {{ 340, 340, 340, 340, 340}, { 340, 280, 280, 170, 200}, { 340, 250, 250, 150, 200}, { 340, 150, 150,
50, 200}, { 340, 200, 200, 200, 200}},
3277 {{ 340, 340, 340, 340, 340}, { 340, 260, 260, 160, 200}, { 340, 310, 250, 240, 200}, { 340, 200, 200, 200,
200}

```

```

    200}, { 340, 310, 250, 240, 200}},  

3278 {{ 340, 340, 340, 340}, { 340, 150, 150, 50, 200}, { 340, 200, 200, 200, 200}, { 340, 210, 210, 100,  

    200}, { 340, 130, 30, 110, 200}},  

3279 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 310, 250, 240, 200}, { 340, 230, 130, 210,  

    200}, { 340, 270, 170, 240, 200}}  

3280 },  

3281 {  

3282 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3283 {{ 340, 340, 340, 340}, { 340, 230, 340, 200, 310}, { 340, 210, 250, 200, 220}, { 340, 110, 250, 200,  

    220}, { 340, 200, 200, 200, 200}},  

3284 {{ 340, 340, 340, 340}, { 340, 220, 260, 200, 230}, { 340, 200, 250, 200, 220}, { 340, 200, 200, 200,  

    200}, { 340, 200, 250, 200, 220}},  

3285 {{ 340, 340, 340, 340}, { 340, 110, 250, 200, 220}, { 340, 200, 200, 200, 200}, { 340, 160, 270, 200,  

    240}, { 340, -10, 130, 200, 100}},  

3286 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 250, 200, 220}, { 340, 90, 230, 200,  

    200}, { 340, 120, 170, 200, 140}}  

3287 },  

3288 {  

3289 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3290 {{ 340, 340, 340, 340}, { 340, 170, 200, 210, 220}, { 340, 150, 200, 190, 100}, { 340, 50, 200, 90,  

    180}, { 340, 200, 200, 200, 200}},  

3291 {{ 340, 340, 340, 340}, { 340, 160, 200, 200, 110}, { 340, 240, 200, 240, 190}, { 340, 200, 200, 200,  

    200}, { 340, 240, 200, 240, 190}},  

3292 {{ 340, 340, 340, 340}, { 340, 50, 200, 90, 180}, { 340, 200, 200, 200, 200}, { 340, 100, 200, 140,  

    150}, { 340, 110, 200, 70, -120}},  

3293 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 240, 200, 240, 190}, { 340, 210, 200, 170,  

    -20}, { 340, 240, 200, 200, 190}}  

3294 },  

3295 {  

3296 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3297 {{ 340, 340, 340, 340}, { 340, 200, 340, 100, 290}, { 340, 200, 250, -30, 170}, { 340, 200, 250, 50,  

    250}, { 340, 200, 200, 200, 200}},  

3298 {{ 340, 340, 340, 340}, { 340, 200, 260, -20, 180}, { 340, 200, 250, 70, 160}, { 340, 200, 200, 200,  

    200}, { 340, 200, 250, 70, 160}},  

3299 {{ 340, 340, 340, 340}, { 340, 200, 250, 50, 250}, { 340, 200, 200, 200, 200}, { 340, 200, 270, 30,  

    220}, { 340, 200, 130, -250, 130}},  

3300 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 250, 70, 160}, { 340, 200, 230,  

    -150, 230}, { 340, 200, 170, 70, 80}}  

3301 },  

3302 },  

3303 /* UA...UG */  

3304 {  

3305 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3306 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3307 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3308 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3309 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    { 340, 340, 340, 340}, { 340, 340, 340, 340}}  

3310 },  

3311 {  

3312 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3313 {{ 340, 340, 340, 340}, { 340, 280, 280, 170, 200}, { 340, 230, 230, 130, 200}, { 340, 170, 170, 70,  

    200}, { 340, 200, 200, 200, 200}},  

3314 {{ 340, 340, 340, 340}, { 340, 280, 280, 170, 200}, { 340, 340, 280, 270, 200}, { 340, 200, 200, 200,  

    200}, { 340, 340, 280, 270, 200}},  

3315 {{ 340, 340, 340, 340}, { 340, 170, 170, 70, 200}, { 340, 200, 200, 200, 200}, { 340, 210, 210, 110,  

    200}, { 340, 100, 0, 70, 200}},  

3316 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 310, 250, 240, 200}, { 340, 220, 120, 200,  

    200}, { 340, 290, 190, 270, 200}}  

3317 },  

3318 {  

3319 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3320 {{ 340, 340, 340, 340}, { 340, 230, 340, 200, 310}, { 340, 190, 230, 200, 200}, { 340, 130, 270, 200,  

    240}, { 340, 200, 200, 200, 200}},  

3321 {{ 340, 340, 340, 340}, { 340, 230, 280, 200, 250}, { 340, 230, 280, 200, 250}, { 340, 200, 200, 200,  

    200}, { 340, 230, 280, 200, 250}},  

3322 {{ 340, 340, 340, 340}, { 340, 130, 270, 200, 240}, { 340, 200, 200, 200, 200}, { 340, 170, 270, 200,  

    240}, { 340, -50, 100, 200, 70}},  

3323 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 250, 200, 220}, { 340, 80, 220, 200,  

    190}, { 340, 150, 190, 200, 160}}  

3324 },  

3325 {  

3326 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},  

    { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3327 {{ 340, 340, 340, 340}, { 340, 170, 200, 210, 220}, { 340, 130, 200, 170, 80}, { 340, 70, 200, 110,  

    200}, { 340, 200, 200, 200, 200}},  

3328 {{ 340, 340, 340, 340}, { 340, 170, 200, 210, 120}, { 340, 270, 200, 270, 220}, { 340, 200, 200, 200,  

    200}, { 340, 270, 200, 270, 220}}}

```

```

3329 {{ 340, 340, 340, 340}, { 340, 70, 200, 110, 200}, { 340, 200, 200, 200, 200}, { 340, 110, 200, 150,
    160}, { 340, 70, 200, 30, -160}}, ,
3330 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 240, 200, 240, 190}, { 340, 200, 200, 160,
    -30}, { 340, 270, 200, 230, 220}}}
3331 },
3332 {
3333 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
    340}, { 340, 340, 340, 340}}, ,
3334 {{ 340, 340, 340, 340}, { 340, 200, 340, 100, 290}, { 340, 200, 230, -50, 150}, { 340, 200, 270, 70,
    270}, { 340, 200, 200, 200, 200}}, ,
3335 {{ 340, 340, 340, 340}, { 340, 200, 280, 0, 190}, { 340, 200, 280, 100, 190}, { 340, 200, 200, 200,
    200}, { 340, 200, 280, 100, 190}}, ,
3336 {{ 340, 340, 340, 340}, { 340, 200, 270, 70, 270}, { 340, 200, 200, 200, 200}, { 340, 200, 270, 30,
    230}, { 340, 200, 100, -290, 90}}, ,
3337 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 250, 70, 160}, { 340, 200, 220,
    -160, 220}, { 340, 200, 190, 90, 110}}}
3338 }
3339 },
3340 /* UA....AU */
3341 {
3342 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340,
    340}, { 340, 340, 340, 340}}, ,
3343 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
    340}, { 340, 340, 340, 340}, ,
3344 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
    340}, { 340, 340, 340, 340}, ,
3345 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
    340}, { 340, 340, 340, 340}, ,
3346 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
    340}, { 340, 340, 340, 340}}}
3347 },
3348 {
3349 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
    340}, { 340, 340, 340, 340}}, ,
3350 {{ 340, 340, 340, 340}, { 340, 280, 280, 170, 200}, { 340, 250, 250, 150, 200}, { 340, 150, 150,
    50, 200}, { 340, 200, 200, 200, 200}}, ,
3351 {{ 340, 340, 340, 340}, { 340, 260, 260, 160, 200}, { 340, 310, 250, 240, 200}, { 340, 200, 200,
    200}, { 340, 310, 250, 240, 200}}, ,
3352 {{ 340, 340, 340, 340}, { 340, 150, 150, 50, 200}, { 340, 200, 200, 200, 200}, { 340, 210, 210,
    100, 200}, { 340, 130, 30, 110, 200}}, ,
3353 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 310, 250, 240, 200}, { 340, 230, 130,
    210, 200}, { 340, 270, 170, 240, 200}}}
3354 },
3355 {
3356 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
    340}, { 340, 340, 340, 340}}, ,
3357 {{ 340, 340, 340, 340}, { 340, 230, 340, 200, 310}, { 340, 210, 250, 200, 220}, { 340, 110, 250,
    200, 220}, { 340, 200, 200, 200, 200}}, ,
3358 {{ 340, 340, 340, 340}, { 340, 220, 260, 200, 230}, { 340, 200, 250, 200, 220}, { 340, 200, 200,
    200}, { 340, 200, 250, 200, 220}}, ,
3359 {{ 340, 340, 340, 340}, { 340, 110, 250, 200, 220}, { 340, 200, 200, 200, 200}, { 340, 160, 270,
    200, 240}, { 340, -10, 130, 200, 100}}, ,
3360 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 250, 200, 220}, { 340, 90, 230,
    200, 200}, { 340, 120, 170, 200, 140}}}
3361 },
3362 {
3363 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
    340}, { 340, 340, 340, 340}}, ,
3364 {{ 340, 340, 340, 340}, { 340, 170, 200, 210, 220}, { 340, 150, 200, 190, 100}, { 340, 50, 200,
    90, 180}, { 340, 200, 200, 200, 200}}, ,
3365 {{ 340, 340, 340, 340}, { 340, 160, 200, 200, 110}, { 340, 240, 200, 240, 190}, { 340, 200, 200,
    200}, { 340, 240, 200, 240, 190}}, ,
3366 {{ 340, 340, 340, 340}, { 340, 50, 200, 90, 180}, { 340, 200, 200, 200, 200}, { 340, 100, 200,
    140, 150}, { 340, 110, 200, 70, -120}}, ,
3367 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 240, 200, 240, 190}, { 340, 210, 200,
    170, -20}, { 340, 240, 200, 200, 190}}}
3368 },
3369 {
3370 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
    340}, { 340, 340, 340, 340}}, ,
3371 {{ 340, 340, 340, 340}, { 340, 200, 340, 100, 290}, { 340, 200, 250, -30, 170}, { 340, 200, 250,
    50, 250}, { 340, 200, 200, 200, 200}}, ,
3372 {{ 340, 340, 340, 340}, { 340, 200, 260, -20, 180}, { 340, 200, 250, 70, 160}, { 340, 200, 200,
    200}, { 340, 200, 250, 70, 160}}, ,
3373 {{ 340, 340, 340, 340}, { 340, 200, 250, 50, 250}, { 340, 200, 200, 200, 200}, { 340, 200, 270,
    30, 220}, { 340, 200, 130, -250, 130}}, ,
3374 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 250, 70, 160}, { 340, 200, 230,
    -150, 230}, { 340, 200, 170, 70, 80}}}
3375 },
3376 },
3377 /* UA....UA */
3378 {
3379 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
    340}, { 340, 340, 340, 340}}, ,
3380 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
    340}, { 340, 340, 340, 340}}, ,
3381 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340},
    340}, { 340, 340, 340, 340}}}

```

```

340}, { 340, 340, 340, 340, 340}},  

3382 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3383 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}}}  

3384 },  

3385 {  

3386 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3387 {{ 340, 340, 340, 340}, { 340, 280, 280, 170, 200}, { 340, 230, 230, 130, 200}, { 340, 170, 170,  

70, 200}, { 340, 200, 200, 200, 200}},  

3388 {{ 340, 340, 340, 340}, { 340, 280, 280, 170, 200}, { 340, 340, 280, 270, 200}, { 340, 200, 200, 200,  

200}, { 340, 340, 280, 270, 200}},  

3389 {{ 340, 340, 340, 340}, { 340, 170, 170, 70, 200}, { 340, 200, 200, 200, 200}, { 340, 210, 210, 110,  

200}, { 340, 100, 0, 70, 200}},  

3390 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 310, 250, 240, 200}, { 340, 220, 120, 200,  

200}, { 340, 290, 190, 270, 200}}}  

3391 },  

3392 {  

3393 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3394 {{ 340, 340, 340, 340}, { 340, 230, 340, 200, 310}, { 340, 190, 230, 200, 200}, { 340, 130, 270, 200,  

240}, { 340, 200, 200, 200, 200}},  

3395 {{ 340, 340, 340, 340}, { 340, 230, 280, 200, 250}, { 340, 230, 280, 200, 250}, { 340, 200, 200, 200,  

200}, { 340, 230, 280, 200, 250}},  

3396 {{ 340, 340, 340, 340}, { 340, 130, 270, 200, 240}, { 340, 200, 200, 200, 200}, { 340, 170, 270, 200,  

240}, { 340, -50, 100, 200, 70}},  

3397 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 250, 200, 220}, { 340, 80, 220, 200,  

190}, { 340, 150, 190, 200, 160}}}  

3398 },  

3399 {  

3400 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3401 {{ 340, 340, 340, 340}, { 340, 170, 200, 210, 220}, { 340, 130, 200, 170, 80}, { 340, 70, 200, 110,  

200}, { 340, 200, 200, 200, 200}},  

3402 {{ 340, 340, 340, 340}, { 340, 170, 200, 210, 120}, { 340, 270, 200, 270, 220}, { 340, 200, 200, 200,  

200}, { 340, 270, 200, 270, 220}},  

3403 {{ 340, 340, 340, 340}, { 340, 70, 200, 110, 200}, { 340, 200, 200, 200, 200}, { 340, 110, 200, 150,  

160}, { 340, 70, 200, 30, -160}},  

3404 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 240, 200, 240, 190}, { 340, 200, 200, 160,  

-30}, { 340, 270, 200, 230, 220}}}  

3405 },  

3406 {  

3407 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3408 {{ 340, 340, 340, 340}, { 340, 200, 340, 100, 290}, { 340, 200, 230, -50, 150}, { 340, 200, 270, 70,  

270}, { 340, 200, 200, 200, 200}},  

3409 {{ 340, 340, 340, 340}, { 340, 200, 280, 0, 190}, { 340, 200, 280, 100, 190}, { 340, 200, 200, 200,  

200}, { 340, 200, 280, 100, 190}},  

3410 {{ 340, 340, 340, 340}, { 340, 200, 270, 70, 270}, { 340, 200, 200, 200, 200}, { 340, 200, 270, 30,  

230}, { 340, 200, 100, -290, 90}},  

3411 {{ 340, 340, 340, 340}, { 340, 200, 200, 200, 200}, { 340, 200, 250, 70, 160}, { 340, 200, 220,  

-160, 220}, { 340, 200, 190, 90, 110}}}  

3412 }  

3413 },  

3414 /* UA....?? */  

3415 {  

3416 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3417 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3418 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3419 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3420 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}}}  

3421 },  

3422 {  

3423 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3424 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3425 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3426 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3427 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}}}  

3428 },  

3429 {  

3430 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3431 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}},  

3432 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,  

340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}}},
```











```

    340}, { 340, 340, 340, 340, 340}}, ,
3694 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
    340}, { 340, 340, 340, 340, 340}}, ,
3695 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
    340}, { 340, 340, 340, 340, 340}} ,
3696 },
3697 {
3698 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
    340}, { 340, 340, 340, 340, 340}}, ,
3699 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
    340}, { 340, 340, 340, 340, 340}}, ,
3700 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
    340}, { 340, 340, 340, 340, 340}}, ,
3701 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
    340}, { 340, 340, 340, 340, 340}}, ,
3702 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
    340}, { 340, 340, 340, 340, 340}} ,
3703 },
3704 {
3705 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
    340}, { 340, 340, 340, 340, 340}}, ,
3706 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
    340}, { 340, 340, 340, 340, 340}}, ,
3707 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
    340}, { 340, 340, 340, 340, 340}}, ,
3708 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
    340}, { 340, 340, 340, 340, 340}}, ,
3709 {{ 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340, 340}, { 340, 340, 340,
    340}, { 340, 340, 340, 340, 340}} ,
3710 }
3711 }
3712 }
3713 };
3714
3715 PRIVATE int int22_H_184[NBPAIRS+1][NBPAIRS+1][5][5][5][5] =
3716 /* noPair */ {{{{0}}}}, ,
3717 /* noPair */ {{{{0}}}}, ,
3718 /* CG.@..CG */
3719 {{ {{ 0, 0, 0, 0, 0},
3720 { DEF, DEF, DEF, DEF, DEF},
3721 { DEF, DEF, DEF, DEF, DEF},
3722 { DEF, DEF, DEF, DEF, DEF},
3723 { DEF, DEF, DEF, DEF, DEF}}, ,
3724 /* CG.@A..CG */
3725 {{ 0, 0, 0, 0, 0},
3726 {-1029, -1029, -1029, -1029, -1029},
3727 { -519, -519, -519, -519, -519},
3728 { -939, -939, -939, -939, -939},
3729 { -809, -809, -809, -809, -809}},
3730 /* CG.@C..CG */
3731 {{ 0, 0, 0, 0, 0},
3732 { -949, -949, -949, -949, -949},
3733 { -449, -449, -449, -449, -449},
3734 { -939, -939, -939, -939, -939},
3735 { -739, -739, -739, -739, -739}},
3736 /* CG.@G..CG */
3737 {{ 0, 0, 0, 0, 0},
3738 {-1029, -1029, -1029, -1029, -1029},
3739 { -519, -519, -519, -519, -519},
3740 { -939, -939, -939, -939, -939},
3741 { -809, -809, -809, -809, -809}},
3742 /* CG.@U..CG */
3743 {{ 0, 0, 0, 0, 0},
3744 {-1029, -1029, -1029, -1029, -1029},
3745 { -669, -669, -669, -669, -669},
3746 { -939, -939, -939, -939, -939},
3747 { -859, -859, -859, -859, -859}},
3748 /* CG.A@..CG */
3749 {{ { DEF, -1029, -949, -1029, -1029},
3750 { -100, -1079, -999, -1079, -1079},
3751 { -100, -1079, -999, -1079, -1079},
3752 { -100, -1079, -999, -1079, -1079},
3753 { -100, -1079, -999, -1079, -1079}},
3754 /* CG.AA..CG */
3755 {{ { DEF, -1029, -949, -1029, -1029},
3756 {-1079, -2058, -1978, -2058, -2058},
3757 { -569, -1548, -1468, -1548, -1548},
3758 { -989, -1968, -1888, -1968, -1968},
3759 { -859, -1838, -1758, -1838, -1838}},
3760 /* CG.AC..CG */
3761 {{ { DEF, -1029, -949, -1029, -1029},
3762 { -999, -1978, -1898, -1978, -1978},
3763 { -499, -1478, -1398, -1478, -1478},
3764 { -989, -1968, -1888, -1968, -1968},
3765 { -789, -1768, -1688, -1768, -1768}},
3766 /* CG.AG..CG */
3767 {{ { DEF, -1029, -949, -1029, -1029},

```

```
3768 {-1079,-2058,-1978,-2058,-2058},  
3769 {-569,-1548,-1468,-1548,-1548},  
3770 {-989,-1968,-1888,-1968,-1968},  
3771 {-859,-1838,-1758,-1838,-1838}},  
3772 /* CG.AU..CG */  
3773 {{ DEF, -1029, -949, -1029, -1029},  
3774 {-1079,-2058,-1978,-2058,-2058},  
3775 {-719,-1698,-1618,-1698,-1698},  
3776 {-989,-1968,-1888,-1968,-1968},  
3777 {-909,-1888,-1808,-1888,-1888}}},  
3778 /* CG.C@..CG */  
3779 {{ DEF, -519, -449, -519, -669},  
3780 {-100, -569, -499, -569, -719},  
3781 {-100, -569, -499, -569, -719},  
3782 {-100, -569, -499, -569, -719},  
3783 {-100, -569, -499, -569, -719}}},  
3784 /* CG.CA..CG */  
3785 {{ DEF, -519, -449, -519, -669},  
3786 {-1079,-1548,-1478,-1548,-1698},  
3787 {-569,-1038, -968,-1038,-1188},  
3788 {-989,-1458,-1388,-1458,-1608},  
3789 {-859,-1328,-1258,-1328,-1478}}},  
3790 /* CG.CC..CG */  
3791 {{ DEF, -519, -449, -519, -669},  
3792 {-999,-1468,-1398,-1468,-1618},  
3793 {-499, -968, -898, -968,-1118},  
3794 {-989,-1458,-1388,-1458,-1608},  
3795 {-789,-1258,-1188,-1258,-1408}}},  
3796 /* CG.CG..CG */  
3797 {{ DEF, -519, -449, -519, -669},  
3798 {-1079,-1548,-1478,-1548,-1698},  
3799 {-569,-1038, -968,-1038,-1188},  
3800 {-989,-1458,-1388,-1458,-1608},  
3801 {-859,-1328,-1258,-1328,-1478}}},  
3802 /* CG.CU..CG */  
3803 {{ DEF, -519, -449, -519, -669},  
3804 {-1079,-1548,-1478,-1548,-1698},  
3805 {-719,-1188,-1118,-1188,-1338},  
3806 {-989,-1458,-1388,-1458,-1608},  
3807 {-909,-1378,-1308,-1378,-1528}}},  
3808 /* CG.G@..CG */  
3809 {{ DEF, -939, -939, -939, -939},  
3810 {-100, -989, -989, -989, -989},  
3811 {-100, -989, -989, -989, -989},  
3812 {-100, -989, -989, -989, -989},  
3813 {-100, -989, -989, -989, -989}}},  
3814 /* CG.GA..CG */  
3815 {{ DEF, -939, -939, -939, -939},  
3816 {-1079,-1968,-1968,-1968,-1968},  
3817 {-569,-1458,-1458,-1458,-1458},  
3818 {-989,-1878,-1878,-1878,-1878},  
3819 {-859,-1748,-1748,-1748,-1748}}},  
3820 /* CG.GC..CG */  
3821 {{ DEF, -939, -939, -939, -939},  
3822 {-999,-1888,-1888,-1888,-1888},  
3823 {-499,-1388,-1388,-1388,-1388},  
3824 {-989,-1878,-1878,-1878,-1878},  
3825 {-789,-1678,-1678,-1678,-1678}}},  
3826 /* CG.GG..CG */  
3827 {{ DEF, -939, -939, -939, -939},  
3828 {-1079,-1968,-1968,-1968,-1968},  
3829 {-569,-1458,-1458,-1458,-1458},  
3830 {-989,-1878,-1878,-1878,-1878},  
3831 {-859,-1748,-1748,-1748,-1748}}},  
3832 /* CG.GU..CG */  
3833 {{ DEF, -939, -939, -939, -939},  
3834 {-1079,-1968,-1968,-1968,-1968},  
3835 {-719,-1608,-1608,-1608,-1608},  
3836 {-989,-1878,-1878,-1878,-1878},  
3837 {-909,-1798,-1798,-1798,-1798}}},  
3838 /* CG.U@..CG */  
3839 {{ DEF, -809, -739, -809, -859},  
3840 {-100, -859, -789, -859, -909},  
3841 {-100, -859, -789, -859, -909},  
3842 {-100, -859, -789, -859, -909},  
3843 {-100, -859, -789, -859, -909}}},  
3844 /* CG.UA..CG */  
3845 {{ DEF, -809, -739, -809, -859},  
3846 {-1079,-1838,-1768,-1838,-1888},  
3847 {-569,-1328,-1258,-1328,-1378},  
3848 {-989,-1748,-1678,-1748,-1798},  
3849 {-859,-1618,-1548,-1618,-1668}}},  
3850 /* CG.UC..CG */  
3851 {{ DEF, -809, -739, -809, -859},  
3852 {-999,-1758,-1688,-1758,-1808},  
3853 {-499,-1258,-1188,-1258,-1308},  
3854 {-989,-1748,-1678,-1748,-1798},
```

```

3855 { -789, -1548, -1478, -1548, -1598}},  

3856 /* CG.UG..CG */  

3857 {{ DEF, -809, -739, -809, -859},  

3858 {-1079, -1838, -1768, -1838, -1888},  

3859 { -569, -1328, -1258, -1328, -1378},  

3860 { -989, -1748, -1678, -1748, -1798},  

3861 { -859, -1618, -1548, -1618, -1668}},  

3862 /* CG.UU..CG */  

3863 {{ DEF, -809, -739, -809, -859},  

3864 {-1079, -1838, -1768, -1838, -1888},  

3865 { -719, -1478, -1408, -1478, -1528},  

3866 { -989, -1748, -1678, -1748, -1798},  

3867 { -909, -1668, -1598, -1668, -1718}}}},  

3868 /* CG.@@..GC */  

3869 {{{ 0, 0, 0, 0, 0},  

3870 { DEF, DEF, DEF, DEF, DEF},  

3871 { DEF, DEF, DEF, DEF, DEF},  

3872 { DEF, DEF, DEF, DEF, DEF},  

3873 { DEF, DEF, DEF, DEF, DEF}}},  

3874 /* CG.@A..GC */  

3875 {{ 0, 0, 0, 0, 0},  

3876 { -519, -519, -519, -519, -519},  

3877 { -719, -719, -719, -719, -719},  

3878 { -709, -709, -709, -709, -709},  

3879 { -499, -499, -499, -499, -499}},  

3880 /* CG.@C..GC */  

3881 {{ 0, 0, 0, 0, 0},  

3882 { -879, -879, -879, -879, -879},  

3883 { -309, -309, -309, -309, -309},  

3884 { -739, -739, -739, -739, -739},  

3885 { -499, -499, -499, -499, -499}},  

3886 /* CG.@G..GC */  

3887 {{ 0, 0, 0, 0, 0},  

3888 { -559, -559, -559, -559, -559},  

3889 { -309, -309, -309, -309, -309},  

3890 { -619, -619, -619, -619, -619},  

3891 { -499, -499, -499, -499, -499}},  

3892 /* CG.@U..GC */  

3893 {{ 0, 0, 0, 0, 0},  

3894 { -879, -879, -879, -879, -879},  

3895 { -389, -389, -389, -389, -389},  

3896 { -739, -739, -739, -739, -739},  

3897 { -569, -569, -569, -569, -569}}},  

3898 /* CG.A@..GC */  

3899 {{ DEF, -1029, -949, -1029, -1029},  

3900 { -100, -1079, -999, -1079, -1079},  

3901 { -100, -1079, -999, -1079, -1079},  

3902 { -100, -1079, -999, -1079, -1079},  

3903 { -100, -1079, -999, -1079, -1079}},  

3904 /* CG.AA..GC */  

3905 {{ DEF, -1029, -949, -1029, -1029},  

3906 { -569, -1548, -1468, -1548, -1548},  

3907 { -769, -1748, -1668, -1748, -1748},  

3908 { -759, -1738, -1658, -1738, -1738},  

3909 { -549, -1528, -1448, -1528, -1528}},  

3910 /* CG.AC..GC */  

3911 {{ DEF, -1029, -949, -1029, -1029},  

3912 { -929, -1908, -1828, -1908, -1908},  

3913 { -359, -1338, -1258, -1338, -1338},  

3914 { -789, -1768, -1688, -1768, -1768},  

3915 { -549, -1528, -1448, -1528, -1528}},  

3916 /* CG.AG..GC */  

3917 {{ DEF, -1029, -949, -1029, -1029},  

3918 { -609, -1588, -1508, -1588, -1588},  

3919 { -359, -1338, -1258, -1338, -1338},  

3920 { -669, -1648, -1568, -1648, -1648},  

3921 { -549, -1528, -1448, -1528, -1528}},  

3922 /* CG.AU..GC */  

3923 {{ DEF, -1029, -949, -1029, -1029},  

3924 { -929, -1908, -1828, -1908, -1908},  

3925 { -439, -1418, -1338, -1418, -1418},  

3926 { -789, -1768, -1688, -1768, -1768},  

3927 { -619, -1598, -1518, -1598, -1598}}},  

3928 /* CG.C@..GC */  

3929 {{ DEF, -519, -449, -519, -669},  

3930 { -100, -569, -499, -569, -719},  

3931 { -100, -569, -499, -569, -719},  

3932 { -100, -569, -499, -569, -719},  

3933 { -100, -569, -499, -569, -719}},  

3934 /* CG.CA..GC */  

3935 {{ DEF, -519, -449, -519, -669},  

3936 { -569, -1038, -968, -1038, -1188},  

3937 { -769, -1238, -1168, -1238, -1388},  

3938 { -759, -1228, -1158, -1228, -1378},  

3939 { -549, -1018, -948, -1018, -1168}}},  

3940 /* CG.CC..GC */  

3941 {{ DEF, -519, -449, -519, -669},

```

```

3942 { -929, -1398, -1328, -1398, -1548},
3943 { -359, -828, -758, -828, -978},
3944 { -789, -1258, -1188, -1258, -1408},
3945 { -549, -1018, -948, -1018, -1168}},
3946 /* CG.CG..GC */
3947 {{ DEF, -519, -449, -519, -669},
3948 { -609, -1078, -1008, -1078, -1228},
3949 { -359, -828, -758, -828, -978},
3950 { -669, -1138, -1068, -1138, -1288},
3951 { -549, -1018, -948, -1018, -1168}},
3952 /* CG.CU..GC */
3953 {{ DEF, -519, -449, -519, -669},
3954 { -929, -1398, -1328, -1398, -1548},
3955 { -439, -908, -838, -908, -1058},
3956 { -789, -1258, -1188, -1258, -1408},
3957 { -619, -1088, -1018, -1088, -1238}},
3958 /* CG.G@..GC */
3959 {{ DEF, -939, -939, -939, -939},
3960 { -100, -989, -989, -989, -989},
3961 { -100, -989, -989, -989, -989},
3962 { -100, -989, -989, -989, -989},
3963 { -100, -989, -989, -989, -989}},
3964 /* CG.GA..GC */
3965 {{ DEF, -939, -939, -939, -939},
3966 { -569, -1458, -1458, -1458, -1458},
3967 { -769, -1658, -1658, -1658, -1658},
3968 { -759, -1648, -1648, -1648, -1648},
3969 { -549, -1438, -1438, -1438, -1438}},
3970 /* CG.GC..GC */
3971 {{ DEF, -939, -939, -939, -939},
3972 { -929, -1818, -1818, -1818, -1818},
3973 { -359, -1248, -1248, -1248, -1248},
3974 { -789, -1678, -1678, -1678, -1678},
3975 { -549, -1438, -1438, -1438, -1438}},
3976 /* CG.GG..GC */
3977 {{ DEF, -939, -939, -939, -939},
3978 { -609, -1498, -1498, -1498, -1498},
3979 { -359, -1248, -1248, -1248, -1248},
3980 { -669, -1558, -1558, -1558, -1558},
3981 { -549, -1438, -1438, -1438, -1438}},
3982 /* CG.GU..GC */
3983 {{ DEF, -939, -939, -939, -939},
3984 { -929, -1818, -1818, -1818, -1818},
3985 { -439, -1328, -1328, -1328, -1328},
3986 { -789, -1678, -1678, -1678, -3080},
3987 { -619, -1508, -1508, -1508, -1508}},
3988 /* CG.U@..GC */
3989 {{ DEF, -809, -739, -809, -859},
3990 { -100, -859, -789, -859, -909},
3991 { -100, -859, -789, -859, -909},
3992 { -100, -859, -789, -859, -909},
3993 { -100, -859, -789, -859, -909}},
3994 /* CG.UA..GC */
3995 {{ DEF, -809, -739, -809, -859},
3996 { -569, -1328, -1258, -1328, -1378},
3997 { -769, -1528, -1458, -1528, -1578},
3998 { -759, -1518, -1448, -1518, -1568},
3999 { -549, -1308, -1238, -1308, -1358}},
4000 /* CG.UC..GC */
4001 {{ DEF, -809, -739, -809, -859},
4002 { -929, -1688, -1618, -1688, -1738},
4003 { -359, -1118, -1048, -1118, -1168},
4004 { -789, -1548, -1478, -1548, -1598},
4005 { -549, -1308, -1238, -1308, -1358}},
4006 /* CG.UG..GC */
4007 {{ DEF, -809, -739, -809, -859},
4008 { -609, -1368, -1298, -1368, -1418},
4009 { -359, -1118, -1048, -1118, -1168},
4010 { -669, -1428, -1358, -1428, -1478},
4011 { -549, -1308, -1238, -1308, -1358}},
4012 /* CG.UU..GC */
4013 {{ DEF, -809, -739, -809, -859},
4014 { -929, -1688, -1618, -1688, -1738},
4015 { -439, -1198, -1128, -1198, -1248},
4016 { -789, -1548, -1478, -1548, -1598},
4017 { -619, -1378, -1308, -1378, -1428}}}},
4018 /* CG.@..GU */
4019 {{{ 0, 0, 0, 0, 0},
4020 { DEF, DEF, DEF, DEF, DEF},
4021 { DEF, DEF, DEF, DEF, DEF},
4022 { DEF, DEF, DEF, DEF, DEF},
4023 { DEF, DEF, DEF, DEF, DEF}},
4024 /* CG.@A..GU */
4025 {{ 0, 0, 0, 0, 0},
4026 { -429, -429, -429, -429, -429},
4027 { -259, -259, -259, -259, -259},
4028 { -339, -339, -339, -339, -339},

```

```

4029 { -329, -329, -329, -329, -329}},
4030 /* CG.@C..GU */
4031 {{ 0, 0, 0, 0, 0},
4032 { -599, -599, -599, -599, -599},
4033 { -239, -239, -239, -239, -239},
4034 { -689, -689, -689, -689, -689},
4035 { -329, -329, -329, -329, -329}},
4036 /* CG.@G..GU */
4037 {{ 0, 0, 0, 0, 0},
4038 { -599, -599, -599, -599, -599},
4039 { -239, -239, -239, -239, -239},
4040 { -689, -689, -689, -689, -689},
4041 { -329, -329, -329, -329, -329}},
4042 /* CG.@U..GU */
4043 {{ 0, 0, 0, 0, 0},
4044 { -599, -599, -599, -599, -599},
4045 { -239, -239, -239, -239, -239},
4046 { -689, -689, -689, -689, -689},
4047 { -329, -329, -329, -329, -329}}),
4048 /* CG.A@..GU */
4049 {{ DEF,-1029, -949,-1029,-1029},
4050 { -100,-1079, -999,-1079,-1079},
4051 { -100,-1079, -999,-1079,-1079},
4052 { -100,-1079, -999,-1079,-1079},
4053 { -100,-1079, -999,-1079,-1079}},
4054 /* CG.AA..GU */
4055 {{ DEF,-1029, -949,-1029,-1029},
4056 { -479,-1458,-1378,-1458,-1458},
4057 { -309,-1288,-1208,-1288,-1288},
4058 { -389,-1368,-1288,-1368,-1368},
4059 { -379,-1358,-1278,-1358,-1358}},
4060 /* CG.AC..GU */
4061 {{ DEF,-1029, -949,-1029,-1029},
4062 { -649,-1628,-1548,-1628,-1628},
4063 { -289,-1268,-1188,-1268,-1268},
4064 { -739,-1718,-1638,-1718,-1718},
4065 { -379,-1358,-1278,-1358,-1358}},
4066 /* CG.AG..GU */
4067 {{ DEF,-1029, -949,-1029,-1029},
4068 { -649,-1628,-1548,-1628,-1628},
4069 { -289,-1268,-1188,-1268,-1268},
4070 { -739,-1718,-1638,-1718,-1718},
4071 { -379,-1358,-1278,-1358,-1358}},
4072 /* CG.AU..GU */
4073 {{ DEF,-1029, -949,-1029,-1029},
4074 { -649,-1628,-1548,-1628,-1628},
4075 { -289,-1268,-1188,-1268,-1268},
4076 { -739,-1718,-1638,-1718,-1718},
4077 { -379,-1358,-1278,-1358,-1358}},
4078 /* CG.C@..GU */
4079 {{ DEF, -519, -449, -519, -669},
4080 { -100, -569, -499, -569, -719},
4081 { -100, -569, -499, -569, -719},
4082 { -100, -569, -499, -569, -719},
4083 { -100, -569, -499, -569, -719}},
4084 /* CG.CA..GU */
4085 {{ DEF, -519, -449, -519, -669},
4086 { -479, -948, -878, -948,-1098},
4087 { -309, -778, -708, -778, -928},
4088 { -389, -858, -788, -858,-1008},
4089 { -379, -848, -778, -848, -998}},
4090 /* CG.CC..GU */
4091 {{ DEF, -519, -449, -519, -669},
4092 { -649,-1118,-1048,-1118,-1268},
4093 { -289, -758, -688, -758, -908},
4094 { -739,-1208,-1138,-1208,-1358},
4095 { -379, -848, -778, -848, -998}},
4096 /* CG.CG..GU */
4097 {{ DEF, -519, -449, -519, -669},
4098 { -649,-1118,-1048,-1118,-1268},
4099 { -289, -758, -688, -758, -908},
4100 { -739,-1208,-1138,-1208,-1358},
4101 { -379, -848, -778, -848, -998}},
4102 /* CG.CU..GU */
4103 {{ DEF, -519, -449, -519, -669},
4104 { -649,-1118,-1048,-1118,-1268},
4105 { -289, -758, -688, -758, -908},
4106 { -739,-1208,-1138,-1208,-1358},
4107 { -379, -848, -778, -848, -998}},
4108 /* CG.G@..GU */
4109 {{ DEF, -939, -939, -939, -939},
4110 { -100, -989, -989, -989, -989},
4111 { -100, -989, -989, -989, -989},
4112 { -100, -989, -989, -989, -989},
4113 { -100, -989, -989, -989, -989}},
4114 /* CG.GA..GU */
4115 {{ DEF, -939, -939, -939, -939},

```

```

4116 { -479, -1368, -1368, -1368, -1368},
4117 { -309, -1198, -1198, -1198, -1198},
4118 { -389, -1278, -1278, -1278, -1278},
4119 { -379, -1268, -1268, -1268, -1268}},
4120 /* CG.GC..GU */
4121 {{ DEF, -939, -939, -939, -939},
4122 { -649, -1538, -1538, -1538, -1538},
4123 { -289, -1178, -1178, -1178, -1178},
4124 { -739, -1628, -1628, -1628, -1628},
4125 { -379, -1268, -1268, -1268, -1268}},
4126 /* CG.GG..GU */
4127 {{ DEF, -939, -939, -939, -939},
4128 { -649, -1538, -1538, -1538, -1538},
4129 { -289, -1178, -1178, -1178, -1178},
4130 { -739, -1628, -1628, -1628, -1628},
4131 { -379, -1268, -1268, -1268, -1268}},
4132 /* CG.GU..GU */
4133 {{ DEF, -939, -939, -939, -939},
4134 { -649, -1538, -1538, -1538, -1538},
4135 { -289, -1178, -1178, -1178, -1178},
4136 { -739, -1628, -1628, -1628, -1628},
4137 { -379, -1268, -1268, -1268, -1268}}}},
4138 /* CG.U@..GU */
4139 {{ DEF, -809, -739, -809, -859},
4140 { -100, -859, -789, -859, -909},
4141 { -100, -859, -789, -859, -909},
4142 { -100, -859, -789, -859, -909},
4143 { -100, -859, -789, -859, -909}},
4144 /* CG.UA..GU */
4145 {{ DEF, -809, -739, -809, -859},
4146 { -479, -1238, -1168, -1238, -1288},
4147 { -309, -1068, -998, -1068, -1118},
4148 { -389, -1148, -1078, -1148, -1198},
4149 { -379, -1138, -1068, -1138, -1188}},
4150 /* CG.UC..GU */
4151 {{ DEF, -809, -739, -809, -859},
4152 { -649, -1408, -1338, -1408, -1458},
4153 { -289, -1048, -978, -1048, -1098},
4154 { -739, -1498, -1428, -1498, -1548},
4155 { -379, -1138, -1068, -1138, -1188}},
4156 /* CG.UG..GU */
4157 {{ DEF, -809, -739, -809, -859},
4158 { -649, -1408, -1338, -1408, -1458},
4159 { -289, -1048, -978, -1048, -1098},
4160 { -739, -1498, -1428, -1498, -1548},
4161 { -379, -1138, -1068, -1138, -1188}},
4162 /* CG.UU..GU */
4163 {{ DEF, -809, -739, -809, -859},
4164 { -649, -1408, -1338, -1408, -1458},
4165 { -289, -1048, -978, -1048, -1098},
4166 { -739, -1498, -1428, -1498, -1548},
4167 { -379, -1138, -1068, -1138, -1188}}}},
4168 /* CG.@@..UG */
4169 {{{ 0, 0, 0, 0, 0},
4170 { DEF, DEF, DEF, DEF, DEF},
4171 { DEF, DEF, DEF, DEF, DEF},
4172 { DEF, DEF, DEF, DEF, DEF},
4173 { DEF, DEF, DEF, DEF, DEF}}},
4174 /* CG.@A..UG */
4175 {{ 0, 0, 0, 0, 0},
4176 { -719, -719, -719, -719, -719},
4177 { -479, -479, -479, -479, -479},
4178 { -659, -659, -659, -659, -659},
4179 { -549, -549, -549, -549, -549}},
4180 /* CG.@C..UG */
4181 {{ 0, 0, 0, 0, 0},
4182 { -789, -789, -789, -789, -789},
4183 { -479, -479, -479, -479, -479},
4184 { -809, -809, -809, -809, -809},
4185 { -439, -439, -439, -439, -439}},
4186 /* CG.@G..UG */
4187 {{ 0, 0, 0, 0, 0},
4188 { -959, -959, -959, -959, -959},
4189 { -359, -359, -359, -359, -359},
4190 { -919, -919, -919, -919, -919},
4191 { -549, -549, -549, -549, -549}},
4192 /* CG.@U..UG */
4193 {{ 0, 0, 0, 0, 0},
4194 { -809, -809, -809, -809, -809},
4195 { -479, -479, -479, -479, -479},
4196 { -809, -809, -809, -809, -809},
4197 { -359, -359, -359, -359, -359}}}},
4198 /* CG.A@..UG */
4199 {{ DEF, -1029, -949, -1029, -1029},
4200 { -100, -1079, -999, -1079, -1079},
4201 { -100, -1079, -999, -1079, -1079},
4202 { -100, -1079, -999, -1079, -1079},

```

```

4203 { -100,-1079, -999,-1079,-1079}},  

4204 /* CG.AA..UG */  

4205 {{ DEF,-1029, -949,-1029,-1029},  

4206 { -769,-1748,-1668,-1748,-1748},  

4207 { -529,-1508,-1428,-1508,-1508},  

4208 { -709,-1688,-1608,-1688,-1688},  

4209 { -599,-1578,-1498,-1578,-1578}},  

4210 /* CG.AC..UG */  

4211 {{ DEF,-1029, -949,-1029,-1029},  

4212 { -839,-1818,-1738,-1818,-1818},  

4213 { -529,-1508,-1428,-1508,-1508},  

4214 { -859,-1838,-1758,-1838,-1838},  

4215 { -489,-1468,-1388,-1468,-1468}},  

4216 /* CG.AG..UG */  

4217 {{ DEF,-1029, -949,-1029,-1029},  

4218 {-1009,-1988,-1908,-1988,-1988},  

4219 { -409,-1388,-1308,-1388,-1388},  

4220 { -969,-1948,-1868,-1948,-1948},  

4221 { -599,-1578,-1498,-1578,-1578}},  

4222 /* CG.AU..UG */  

4223 {{ DEF,-1029, -949,-1029,-1029},  

4224 { -859,-1838,-1758,-1838,-1838},  

4225 { -529,-1508,-1428,-1508,-1508},  

4226 { -859,-1838,-1758,-1838,-1838},  

4227 { -409,-1388,-1308,-1388,-1388}},  

4228 /* CG.C@..UG */  

4229 {{ DEF, -519, -449, -519, -669},  

4230 { -100, -569, -499, -569, -719},  

4231 { -100, -569, -499, -569, -719},  

4232 { -100, -569, -499, -569, -719},  

4233 { -100, -569, -499, -569, -719}},  

4234 /* CG.CA..UG */  

4235 {{ DEF, -519, -449, -519, -669},  

4236 { -769,-1238,-1168,-1238,-1388},  

4237 { -529, -998, -928, -998,-1148},  

4238 { -709,-1178,-1108,-1178,-1328},  

4239 { -599,-1068, -998,-1068,-1218}},  

4240 /* CG.CC..UG */  

4241 {{ DEF, -519, -449, -519, -669},  

4242 { -839,-1308,-1238,-1308,-1458},  

4243 { -529, -998, -928, -998,-1148},  

4244 { -859,-1328,-1258,-1328,-1478},  

4245 { -489, -958, -888, -958,-1108}},  

4246 /* CG.CG..UG */  

4247 {{ DEF, -519, -449, -519, -669},  

4248 {-1009,-1478,-1408,-1478,-1628},  

4249 { -409, -878, -808, -878,-1028},  

4250 { -969,-1438,-1368,-1438,-1588},  

4251 { -599,-1068, -998,-1068,-1218}},  

4252 /* CG.CU..UG */  

4253 {{ DEF, -519, -449, -519, -669},  

4254 { -859,-1328,-1258,-1328,-1478},  

4255 { -529, -998, -928, -998,-1148},  

4256 { -859,-1328,-1258,-1328,-1478},  

4257 { -409, -878, -808, -878,-1028}},  

4258 /* CG.G@..UG */  

4259 {{ DEF, -939, -939, -939, -939},  

4260 { -100, -989, -989, -989, -989},  

4261 { -100, -989, -989, -989, -989},  

4262 { -100, -989, -989, -989, -989},  

4263 { -100, -989, -989, -989, -989}},  

4264 /* CG.GA..UG */  

4265 {{ DEF, -939, -939, -939, -939},  

4266 { -769,-1658,-1658,-1658,-1658},  

4267 { -529,-1418,-1418,-1418,-1418},  

4268 { -709,-1598,-1598,-1598,-1598},  

4269 { -599,-1488,-1488,-1488,-1488}},  

4270 /* CG.GC..UG */  

4271 {{ DEF, -939, -939, -939, -939},  

4272 { -839,-1728,-1728,-1728,-1728},  

4273 { -529,-1418,-1418,-1418,-1418},  

4274 { -859,-1748,-1748,-1748,-1748},  

4275 { -489,-1378,-1378,-1378,-1378}},  

4276 /* CG.GG..UG */  

4277 {{ DEF, -939, -939, -939, -939},  

4278 {-1009,-1898,-1898,-1898,-1898},  

4279 { -409,-1298,-1298,-1298,-1298},  

4280 { -969,-1858,-1858,-1858,-1858},  

4281 { -599,-1488,-1488,-1488,-1488}},  

4282 /* CG.GU..UG */  

4283 {{ DEF, -939, -939, -939, -939},  

4284 { -859,-1748,-1748,-1748,-1748},  

4285 { -529,-1418,-1418,-1418,-1418},  

4286 { -859,-1748,-1748,-1748,-1748},  

4287 { -409,-1298,-1298,-1298,-1298}},  

4288 /* CG.U@..UG */  

4289 {{ DEF, -809, -739, -809, -859},

```

```

4290 { -100, -859, -789, -859, -909},
4291 { -100, -859, -789, -859, -909},
4292 { -100, -859, -789, -859, -909},
4293 { -100, -859, -789, -859, -909}},
4294 /* CG.UA..UG */
4295 {{ DEF, -809, -739, -809, -859},
4296 { -769,-1528,-1458,-1528,-1578},
4297 { -529,-1288,-1218,-1288,-1338},
4298 { -709,-1468,-1398,-1468,-1518},
4299 { -599,-1358,-1288,-1358,-1408}},
4300 /* CG.UC..UG */
4301 {{ DEF, -809, -739, -809, -859},
4302 { -839,-1598,-1528,-1598,-1648},
4303 { -529,-1288,-1218,-1288,-1338},
4304 { -859,-1618,-1548,-1618,-1668},
4305 { -489,-1248,-1178,-1248,-1298}},
4306 /* CG.UG..UG */
4307 {{ DEF, -809, -739, -809, -859},
4308 { -1009,-1768,-1698,-1768,-1818},
4309 { -409,-1168,-1098,-1168,-1218},
4310 { -969,-1728,-1658,-1728,-1778},
4311 { -599,-1358,-1288,-1358,-1408}},
4312 /* CG.UU..UG */
4313 {{ DEF, -809, -739, -809, -859},
4314 { -859,-1618,-1548,-1618,-1668},
4315 { -529,-1288,-1218,-1288,-1338},
4316 { -859,-1618,-1548,-1618,-1668},
4317 { -409,-1168,-1098,-1168,-1218}}}},
4318 /* CG.@@..AU */
4319 {{{ 0, 0, 0, 0, 0},
4320 { DEF, DEF, DEF, DEF, DEF},
4321 { DEF, DEF, DEF, DEF, DEF},
4322 { DEF, DEF, DEF, DEF, DEF},
4323 { DEF, DEF, DEF, DEF, DEF}},
4324 /* CG.@A..AU */
4325 {{ 0, 0, 0, 0, 0},
4326 { -429, -429, -429, -429, -429},
4327 { -259, -259, -259, -259, -259},
4328 { -339, -339, -339, -339, -339},
4329 { -329, -329, -329, -329, -329}},
4330 /* CG.@C..AU */
4331 {{ 0, 0, 0, 0, 0},
4332 { -599, -599, -599, -599, -599},
4333 { -239, -239, -239, -239, -239},
4334 { -689, -689, -689, -689, -689},
4335 { -329, -329, -329, -329, -329}},
4336 /* CG.@G..AU */
4337 {{ 0, 0, 0, 0, 0},
4338 { -599, -599, -599, -599, -599},
4339 { -239, -239, -239, -239, -239},
4340 { -689, -689, -689, -689, -689},
4341 { -329, -329, -329, -329, -329}},
4342 /* CG.@U..AU */
4343 {{ 0, 0, 0, 0, 0},
4344 { -599, -599, -599, -599, -599},
4345 { -239, -239, -239, -239, -239},
4346 { -689, -689, -689, -689, -689},
4347 { -329, -329, -329, -329, -329}}},
4348 /* CG.A@..AU */
4349 {{ DEF,-1029, -949,-1029,-1029},
4350 { -100,-1079, -999,-1079,-1079},
4351 { -100,-1079, -999,-1079,-1079},
4352 { -100,-1079, -999,-1079,-1079},
4353 { -100,-1079, -999,-1079,-1079}},
4354 /* CG.AA..AU */
4355 {{ DEF,-1029, -949,-1029,-1029},
4356 { -479,-1458,-1378,-1458,-1458},
4357 { -309,-1288,-1208,-1288,-1288},
4358 { -389,-1368,-1288,-1368,-1368},
4359 { -379,-1358,-1278,-1358,-1358}},
4360 /* CG.AC..AU */
4361 {{ DEF,-1029, -949,-1029,-1029},
4362 { -649,-1628,-1548,-1628,-1628},
4363 { -289,-1268,-1188,-1268,-1268},
4364 { -739,-1718,-1638,-1718,-1718},
4365 { -379,-1358,-1278,-1358,-1358}},
4366 /* CG.AG..AU */
4367 {{ DEF,-1029, -949,-1029,-1029},
4368 { -649,-1628,-1548,-1628,-1628},
4369 { -289,-1268,-1188,-1268,-1268},
4370 { -739,-1718,-1638,-1718,-1718},
4371 { -379,-1358,-1278,-1358,-1358}},
4372 /* CG.AU..AU */
4373 {{ DEF,-1029, -949,-1029,-1029},
4374 { -649,-1628,-1548,-1628,-1628},
4375 { -289,-1268,-1188,-1268,-1268},
4376 { -739,-1718,-1638,-1718,-1718},

```

```

4377 { -379, -1358, -1278, -1358, -1358}}},  

4378 /* CG.C@..AU */  

4379 {{ DEF, -519, -449, -519, -669},  

4380 { -100, -569, -499, -569, -719},  

4381 { -100, -569, -499, -569, -719},  

4382 { -100, -569, -499, -569, -719},  

4383 { -100, -569, -499, -569, -719}},  

4384 /* CG.CA..AU */  

4385 {{ DEF, -519, -449, -519, -669},  

4386 { -479, -948, -878, -948, -1098},  

4387 { -309, -778, -708, -778, -928},  

4388 { -389, -858, -788, -858, -1008},  

4389 { -379, -848, -778, -848, -998}},  

4390 /* CG.CC..AU */  

4391 {{ DEF, -519, -449, -519, -669},  

4392 { -649, -1118, -1048, -1118, -1268},  

4393 { -289, -758, -688, -758, -908},  

4394 { -739, -1208, -1138, -1208, -1358},  

4395 { -379, -848, -778, -848, -998}},  

4396 /* CG.CG..AU */  

4397 {{ DEF, -519, -449, -519, -669},  

4398 { -649, -1118, -1048, -1118, -1268},  

4399 { -289, -758, -688, -758, -908},  

4400 { -739, -1208, -1138, -1208, -1358},  

4401 { -379, -848, -778, -848, -998}},  

4402 /* CG.CU..AU */  

4403 {{ DEF, -519, -449, -519, -669},  

4404 { -649, -1118, -1048, -1118, -1268},  

4405 { -289, -758, -688, -758, -908},  

4406 { -739, -1208, -1138, -1208, -1358},  

4407 { -379, -848, -778, -848, -998}},  

4408 /* CG.G@..AU */  

4409 {{ DEF, -939, -939, -939, -939},  

4410 { -100, -989, -989, -989, -989},  

4411 { -100, -989, -989, -989, -989},  

4412 { -100, -989, -989, -989, -989},  

4413 { -100, -989, -989, -989, -989}},  

4414 /* CG.GA..AU */  

4415 {{ DEF, -939, -939, -939, -939},  

4416 { -479, -1368, -1368, -1368, -1368},  

4417 { -309, -1198, -1198, -1198, -1198},  

4418 { -389, -1278, -1278, -1278, -1278},  

4419 { -379, -1268, -1268, -1268, -1268}},  

4420 /* CG.GC..AU */  

4421 {{ DEF, -939, -939, -939, -939},  

4422 { -649, -1538, -1538, -1538, -1538},  

4423 { -289, -1178, -1178, -1178, -1178},  

4424 { -739, -1628, -1628, -1628, -1628},  

4425 { -379, -1268, -1268, -1268, -1268}},  

4426 /* CG.GG..AU */  

4427 {{ DEF, -939, -939, -939, -939},  

4428 { -649, -1538, -1538, -1538, -1538},  

4429 { -289, -1178, -1178, -1178, -1178},  

4430 { -739, -1628, -1628, -1628, -1628},  

4431 { -379, -1268, -1268, -1268, -1268}},  

4432 /* CG.GU..AU */  

4433 {{ DEF, -939, -939, -939, -939},  

4434 { -649, -1538, -1538, -1538, -1538},  

4435 { -289, -1178, -1178, -1178, -1178},  

4436 { -739, -1628, -1628, -1628, -1628},  

4437 { -379, -1268, -1268, -1268, -1268}},  

4438 /* CG.U@..AU */  

4439 {{ DEF, -809, -739, -809, -859},  

4440 { -100, -859, -789, -859, -909},  

4441 { -100, -859, -789, -859, -909},  

4442 { -100, -859, -789, -859, -909},  

4443 { -100, -859, -789, -859, -909}},  

4444 /* CG.UA..AU */  

4445 {{ DEF, -809, -739, -809, -859},  

4446 { -479, -1238, -1168, -1238, -1288},  

4447 { -309, -1068, -998, -1068, -1118},  

4448 { -389, -1148, -1078, -1148, -1198},  

4449 { -379, -1138, -1068, -1138, -1188}},  

4450 /* CG.UC..AU */  

4451 {{ DEF, -809, -739, -809, -859},  

4452 { -649, -1408, -1338, -1408, -1458},  

4453 { -289, -1048, -978, -1048, -1098},  

4454 { -739, -1498, -1428, -1498, -1548},  

4455 { -379, -1138, -1068, -1138, -1188}},  

4456 /* CG.UG..AU */  

4457 {{ DEF, -809, -739, -809, -859},  

4458 { -649, -1408, -1338, -1408, -1458},  

4459 { -289, -1048, -978, -1048, -1098},  

4460 { -739, -1498, -1428, -1498, -1548},  

4461 { -379, -1138, -1068, -1138, -1188}},  

4462 /* CG.UU..AU */  

4463 {{ DEF, -809, -739, -809, -859},

```

```

4464 { -649, -1408, -1338, -1408, -1458},
4465 { -289, -1048, -978, -1048, -1098},
4466 { -739, -1498, -1428, -1498, -1548},
4467 { -379, -1138, -1068, -1138, -1188} } },
4468 /* CG.@..UA */
4469 {{ { 0, 0, 0, 0, 0 },
4470 { DEF, DEF, DEF, DEF, DEF },
4471 { DEF, DEF, DEF, DEF, DEF },
4472 { DEF, DEF, DEF, DEF, DEF },
4473 { DEF, DEF, DEF, DEF, DEF } },
4474 /* CG.@A..UA */
4475 {{ { 0, 0, 0, 0, 0 },
4476 { -399, -399, -399, -399, -399 },
4477 { -429, -429, -429, -429, -429 },
4478 { -379, -379, -379, -379, -379 },
4479 { -279, -279, -279, -279, -279 } },
4480 /* CG.@C..UA */
4481 {{ { 0, 0, 0, 0, 0 },
4482 { -629, -629, -629, -629, -629 },
4483 { -509, -509, -509, -509, -509 },
4484 { -679, -679, -679, -679, -679 },
4485 { -139, -139, -139, -139, -139 } },
4486 /* CG.@G..UA */
4487 {{ { 0, 0, 0, 0, 0 },
4488 { -889, -889, -889, -889, -889 },
4489 { -199, -199, -199, -199, -199 },
4490 { -889, -889, -889, -889, -889 },
4491 { -279, -279, -279, -279, -279 } },
4492 /* CG.@U..UA */
4493 {{ { 0, 0, 0, 0, 0 },
4494 { -589, -589, -589, -589, -589 },
4495 { -179, -179, -179, -179, -179 },
4496 { -679, -679, -679, -679, -679 },
4497 { -140, -140, -140, -140, -140 } },
4498 /* CG.A@..UA */
4499 {{ { DEF, -1029, -949, -1029, -1029 },
4500 { -100, -1079, -999, -1079, -1079 },
4501 { -100, -1079, -999, -1079, -1079 },
4502 { -100, -1079, -999, -1079, -1079 },
4503 { -100, -1079, -999, -1079, -1079 } },
4504 /* CG.AA..UA */
4505 {{ { DEF, -1029, -949, -1029, -1029 },
4506 { -449, -1428, -1348, -1428, -1428 },
4507 { -479, -1458, -1378, -1458, -1458 },
4508 { -429, -1408, -1328, -1408, -1408 },
4509 { -329, -1308, -1228, -1308, -1308 } },
4510 /* CG.AC..UA */
4511 {{ { DEF, -1029, -949, -1029, -1029 },
4512 { -679, -1658, -1578, -1658, -1658 },
4513 { -559, -1538, -1458, -1538, -1538 },
4514 { -729, -1708, -1628, -1708, -1708 },
4515 { -189, -1168, -1088, -1168, -1168 } },
4516 /* CG.AG..UA */
4517 {{ { DEF, -1029, -949, -1029, -1029 },
4518 { -939, -1918, -1838, -1918, -1918 },
4519 { -249, -1228, -1148, -1228, -1228 },
4520 { -939, -1918, -1838, -1918, -1918 },
4521 { -329, -1308, -1228, -1308, -1308 } },
4522 /* CG.AU..UA */
4523 {{ { DEF, -1029, -949, -1029, -1029 },
4524 { -639, -1618, -1538, -1618, -1618 },
4525 { -229, -1208, -1128, -1208, -1208 },
4526 { -729, -1708, -1628, -1708, -1708 },
4527 { -190, -1169, -1089, -1169, -1169 } },
4528 /* CG.C@..UA */
4529 {{ { DEF, -519, -449, -519, -669 },
4530 { -100, -569, -499, -569, -719 },
4531 { -100, -569, -499, -569, -719 },
4532 { -100, -569, -499, -569, -719 },
4533 { -100, -569, -499, -569, -719 } },
4534 /* CG.CA..UA */
4535 {{ { DEF, -519, -449, -519, -669 },
4536 { -449, -918, -848, -918, -1068 },
4537 { -479, -948, -878, -948, -1098 },
4538 { -429, -898, -828, -898, -1048 },
4539 { -329, -798, -728, -798, -948 } },
4540 /* CG.CC..UA */
4541 {{ { DEF, -519, -449, -519, -669 },
4542 { -679, -1148, -1078, -1148, -1298 },
4543 { -559, -1028, -958, -1028, -1178 },
4544 { -729, -1198, -1128, -1198, -1348 },
4545 { -189, -658, -588, -658, -808 } },
4546 /* CG.CG..UA */
4547 {{ { DEF, -519, -449, -519, -669 },
4548 { -939, -1408, -1338, -1408, -1558 },
4549 { -249, -718, -648, -718, -868 },
4550 { -939, -1408, -1338, -1408, -1558 } }

```

```

4551 { -329, -798, -728, -798, -948}},
4552 /* CG.CU..UA */
4553 {{ DEF, -519, -449, -519, -669},
4554 { -639,-1108,-1038,-1108,-1258},
4555 { -229, -698, -628, -698, -848},
4556 { -729,-1198,-1128,-1198,-1348},
4557 { -190, -659, -589, -659, -809}},
4558 /* CG.G@..UA */
4559 {{ DEF, -939, -939, -939, -939},
4560 { -100, -989, -989, -989, -989},
4561 { -100, -989, -989, -989, -989},
4562 { -100, -989, -989, -989, -989},
4563 { -100, -989, -989, -989, -989},
4564 /* CG.GA..UA */
4565 {{ DEF, -939, -939, -939, -939},
4566 { -449,-1338,-1338,-1338,-1338},
4567 { -479,-1368,-1368,-1368,-1368},
4568 { -429,-1318,-1318,-1318,-1318},
4569 { -329,-1218,-1218,-1218,-1218}},
4570 /* CG.GC..UA */
4571 {{ DEF, -939, -939, -939, -939},
4572 { -679,-1568,-1568,-1568,-1568},
4573 { -559,-1448,-1448,-1448,-1448},
4574 { -729,-1618,-1618,-1618,-1618},
4575 { -189,-1078,-1078,-1078,-1078}},
4576 /* CG.GG..UA */
4577 {{ DEF, -939, -939, -939, -939},
4578 { -939,-1828,-1828,-1828,-1828},
4579 { -249,-1138,-1138,-1138,-1138},
4580 { -939,-1828,-1828,-1828,-1828},
4581 { -329,-1218,-1218,-1218,-1218}},
4582 /* CG.GU..UA */
4583 {{ DEF, -939, -939, -939, -939},
4584 { -639,-1528,-1528,-1528,-1528},
4585 { -229,-1118,-1118,-1118,-1118},
4586 { -729,-1618,-1618,-1618,-1618},
4587 { -190,-1079,-1079,-1079,-1079}},
4588 /* CG.U@..UA */
4589 {{ DEF, -809, -739, -809, -859},
4590 { -100, -859, -789, -859, -909},
4591 { -100, -859, -789, -859, -909},
4592 { -100, -859, -789, -859, -909},
4593 { -100, -859, -789, -859, -909}},
4594 /* CG.UA..UA */
4595 {{ DEF, -809, -739, -809, -859},
4596 { -449,-1208,-1138,-1208,-1258},
4597 { -479,-1238,-1168,-1238,-1288},
4598 { -429,-1188,-1118,-1188,-1238},
4599 { -329,-1088,-1018,-1088,-1138}},
4600 /* CG.UC..UA */
4601 {{ DEF, -809, -739, -809, -859},
4602 { -679,-1438,-1368,-1438,-1488},
4603 { -559,-1318,-1248,-1318,-1368},
4604 { -729,-1488,-1418,-1488,-1538},
4605 { -189, -948, -878, -948, -998}},
4606 /* CG.UG..UA */
4607 {{ DEF, -809, -739, -809, -859},
4608 { -939,-1698,-1628,-1698,-1748},
4609 { -249,-1008, -938,-1008,-1058},
4610 { -939,-1698,-1628,-1698,-1748},
4611 { -329,-1088,-1018,-1088,-1138}},
4612 /* CG.UU..UA */
4613 {{ DEF, -809, -739, -809, -859},
4614 { -639,-1398,-1328,-1398,-1448},
4615 { -229, -988, -918, -988,-1038},
4616 { -729,-1488,-1418,-1488,-1538},
4617 { -190, -949, -879, -949, -999}}},
4618 /* CG.@@.. @ */
4619 {{{ DEF, DEF, DEF, DEF, DEF},
4620 { DEF, DEF, DEF, DEF, DEF},
4621 { DEF, DEF, DEF, DEF, DEF},
4622 { DEF, DEF, DEF, DEF, DEF},
4623 { DEF, DEF, DEF, DEF, DEF}},
4624 /* CG.@A.. @ */
4625 {{ DEF, DEF, DEF, DEF, DEF},
4626 { DEF, DEF, DEF, DEF, DEF},
4627 { DEF, DEF, DEF, DEF, DEF},
4628 { DEF, DEF, DEF, DEF, DEF},
4629 { DEF, DEF, DEF, DEF, DEF}},
4630 /* CG.@C.. @ */
4631 {{ DEF, DEF, DEF, DEF, DEF},
4632 { DEF, DEF, DEF, DEF, DEF},
4633 { DEF, DEF, DEF, DEF, DEF},
4634 { DEF, DEF, DEF, DEF, DEF},
4635 { DEF, DEF, DEF, DEF, DEF}},
4636 /* CG.@G.. @ */
4637 {{ DEF, DEF, DEF, DEF, DEF},

```

```
4638 { DEF, DEF, DEF, DEF, DEF},
4639 { DEF, DEF, DEF, DEF, DEF},
4640 { DEF, DEF, DEF, DEF, DEF},
4641 { DEF, DEF, DEF, DEF, DEF}},
4642 /* CG.@.. @ */
4643 {{ DEF, DEF, DEF, DEF, DEF},
4644 { DEF, DEF, DEF, DEF, DEF},
4645 { DEF, DEF, DEF, DEF, DEF},
4646 { DEF, DEF, DEF, DEF, DEF},
4647 { DEF, DEF, DEF, DEF, DEF}}}},
4648 /* CG.A@.. @ */
4649 {{ -100,-1079, -999,-1079,-1079},
4650 { -100,-1079, -999,-1079,-1079},
4651 { -100,-1079, -999,-1079,-1079},
4652 { -100,-1079, -999,-1079,-1079},
4653 { -100,-1079, -999,-1079,-1079}},
4654 /* CG.AA.. @ */
4655 {{ -100,-1079, -999,-1079,-1079},
4656 { -100,-1079, -999,-1079,-1079},
4657 { -100,-1079, -999,-1079,-1079},
4658 { -100,-1079, -999,-1079,-1079},
4659 { -100,-1079, -999,-1079,-1079}},
4660 /* CG.AC.. @ */
4661 {{ -100,-1079, -999,-1079,-1079},
4662 { -100,-1079, -999,-1079,-1079},
4663 { -100,-1079, -999,-1079,-1079},
4664 { -100,-1079, -999,-1079,-1079},
4665 { -100,-1079, -999,-1079,-1079}},
4666 /* CG.AG.. @ */
4667 {{ -100,-1079, -999,-1079,-1079},
4668 { -100,-1079, -999,-1079,-1079},
4669 { -100,-1079, -999,-1079,-1079},
4670 { -100,-1079, -999,-1079,-1079},
4671 { -100,-1079, -999,-1079,-1079}},
4672 /* CG.AU.. @ */
4673 {{ -100,-1079, -999,-1079,-1079},
4674 { -100,-1079, -999,-1079,-1079},
4675 { -100,-1079, -999,-1079,-1079},
4676 { -100,-1079, -999,-1079,-1079},
4677 { -100,-1079, -999,-1079,-1079}}}},
4678 /* CG.C@.. @ */
4679 {{ -100, -569, -499, -569, -719},
4680 { -100, -569, -499, -569, -719},
4681 { -100, -569, -499, -569, -719},
4682 { -100, -569, -499, -569, -719},
4683 { -100, -569, -499, -569, -719}},
4684 /* CG.CA.. @ */
4685 {{ -100, -569, -499, -569, -719},
4686 { -100, -569, -499, -569, -719},
4687 { -100, -569, -499, -569, -719},
4688 { -100, -569, -499, -569, -719},
4689 { -100, -569, -499, -569, -719}},
4690 /* CG.CC.. @ */
4691 {{ -100, -569, -499, -569, -719},
4692 { -100, -569, -499, -569, -719},
4693 { -100, -569, -499, -569, -719},
4694 { -100, -569, -499, -569, -719},
4695 { -100, -569, -499, -569, -719}},
4696 /* CG.CG.. @ */
4697 {{ -100, -569, -499, -569, -719},
4698 { -100, -569, -499, -569, -719},
4699 { -100, -569, -499, -569, -719},
4700 { -100, -569, -499, -569, -719},
4701 { -100, -569, -499, -569, -719}},
4702 /* CG.CU.. @ */
4703 {{ -100, -569, -499, -569, -719},
4704 { -100, -569, -499, -569, -719},
4705 { -100, -569, -499, -569, -719},
4706 { -100, -569, -499, -569, -719},
4707 { -100, -569, -499, -569, -719}}}},
4708 /* CG.G@.. @ */
4709 {{ -100, -989, -989, -989, -989},
4710 { -100, -989, -989, -989, -989},
4711 { -100, -989, -989, -989, -989},
4712 { -100, -989, -989, -989, -989},
4713 { -100, -989, -989, -989, -989}},
4714 /* CG.GA.. @ */
4715 {{ -100, -989, -989, -989, -989},
4716 { -100, -989, -989, -989, -989},
4717 { -100, -989, -989, -989, -989},
4718 { -100, -989, -989, -989, -989},
4719 { -100, -989, -989, -989, -989}},
4720 /* CG.GC.. @ */
4721 {{ -100, -989, -989, -989, -989},
4722 { -100, -989, -989, -989, -989},
4723 { -100, -989, -989, -989, -989},
4724 { -100, -989, -989, -989, -989},
```

```

4725 { -100, -989, -989, -989, -989},  

4726 /* CG.GG.. @ */  

4727 {{ -100, -989, -989, -989, -989},  

4728 { -100, -989, -989, -989, -989},  

4729 { -100, -989, -989, -989, -989},  

4730 { -100, -989, -989, -989, -989},  

4731 { -100, -989, -989, -989, -989},  

4732 /* CG.GU.. @ */  

4733 {{ -100, -989, -989, -989, -989},  

4734 { -100, -989, -989, -989, -989},  

4735 { -100, -989, -989, -989, -989},  

4736 { -100, -989, -989, -989, -989},  

4737 { -100, -989, -989, -989, -989}}},  

4738 /* CG.U@.. @ */  

4739 {{ -100, -859, -789, -859, -909},  

4740 { -100, -859, -789, -859, -909},  

4741 { -100, -859, -789, -859, -909},  

4742 { -100, -859, -789, -859, -909},  

4743 { -100, -859, -789, -859, -909}}},  

4744 /* CG.UA.. @ */  

4745 {{ -100, -859, -789, -859, -909},  

4746 { -100, -859, -789, -859, -909},  

4747 { -100, -859, -789, -859, -909},  

4748 { -100, -859, -789, -859, -909},  

4749 { -100, -859, -789, -859, -909}}},  

4750 /* CG.UC.. @ */  

4751 {{ -100, -859, -789, -859, -909},  

4752 { -100, -859, -789, -859, -909},  

4753 { -100, -859, -789, -859, -909},  

4754 { -100, -859, -789, -859, -909},  

4755 { -100, -859, -789, -859, -909}}},  

4756 /* CG.UG.. @ */  

4757 {{ -100, -859, -789, -859, -909},  

4758 { -100, -859, -789, -859, -909},  

4759 { -100, -859, -789, -859, -909},  

4760 { -100, -859, -789, -859, -909},  

4761 { -100, -859, -789, -859, -909}}},  

4762 /* CG.UU.. @ */  

4763 {{ -100, -859, -789, -859, -909},  

4764 { -100, -859, -789, -859, -909},  

4765 { -100, -859, -789, -859, -909},  

4766 { -100, -859, -789, -859, -909},  

4767 { -100, -859, -789, -859, -909}}}},  

4768 /* noPair */ {{{{0}}}},  

4769 /* GC.@@..CG */  

4770 {{{ 0, 0, 0, 0, 0},  

4771 { DEF, DEF, DEF, DEF, DEF},  

4772 { DEF, DEF, DEF, DEF, DEF},  

4773 { DEF, DEF, DEF, DEF, DEF},  

4774 { DEF, DEF, DEF, DEF, DEF}},  

4775 /* GC.@A..CG */  

4776 {{ 0, 0, 0, 0, 0},  

4777 {-1029,-1029,-1029,-1029,-1029},  

4778 { -519, -519, -519, -519, -519},  

4779 { -939, -939, -939, -939, -939},  

4780 { -809, -809, -809, -809, -809}}},  

4781 /* GC.@C..CG */  

4782 {{ 0, 0, 0, 0, 0},  

4783 { -949, -949, -949, -949, -949},  

4784 { -449, -449, -449, -449, -449},  

4785 { -939, -939, -939, -939, -939},  

4786 { -739, -739, -739, -739, -739}}},  

4787 /* GC.@G..CG */  

4788 {{ 0, 0, 0, 0, 0},  

4789 {-1029,-1029,-1029,-1029,-1029},  

4790 { -519, -519, -519, -519, -519},  

4791 { -939, -939, -939, -939, -939},  

4792 { -809, -809, -809, -809, -809}}},  

4793 /* GC.@U..CG */  

4794 {{ 0, 0, 0, 0, 0},  

4795 {-1029,-1029,-1029,-1029,-1029},  

4796 { -669, -669, -669, -669, -669},  

4797 { -939, -939, -939, -939, -939},  

4798 { -859, -859, -859, -859, -859}}},  

4799 /* GC.A@..CG */  

4800 {{ DEF, -519, -879, -559, -879},  

4801 { -100, -569, -929, -609, -929},  

4802 { -100, -569, -929, -609, -929},  

4803 { -100, -569, -929, -609, -929},  

4804 { -100, -569, -929, -609, -929}}},  

4805 /* GC.AA..CG */  

4806 {{ DEF, -519, -879, -559, -879},  

4807 {-1079,-1548,-1908,-1588,-1908},  

4808 { -569,-1038,-1398,-1078,-1398},  

4809 { -989,-1458,-1818,-1498,-1818},  

4810 { -859,-1328,-1688,-1368,-1688}},  

4811 /* GC.AC..CG */

```

```
4812 {{ DEF, -519, -879, -559, -879},  
4813 { -999,-1468,-1828,-1508,-1828},  
4814 { -499, -968,-1328,-1008,-1328},  
4815 { -989,-1458,-1818,-1498,-1818},  
4816 { -789,-1258,-1618,-1298,-1618}},  
4817 /* GC.AG..CG */  
4818 {{ DEF, -519, -879, -559, -879},  
4819 {-1079,-1548,-1908,-1588,-1908},  
4820 { -569,-1038,-1398,-1078,-1398},  
4821 { -989,-1458,-1818,-1498,-1818},  
4822 { -859,-1328,-1688,-1368,-1688}},  
4823 /* GC.AU..CG */  
4824 {{ DEF, -519, -879, -559, -879},  
4825 {-1079,-1548,-1908,-1588,-1908},  
4826 { -719,-1188,-1548,-1228,-1548},  
4827 { -989,-1458,-1818,-1498,-1818},  
4828 { -909,-1378,-1738,-1418,-1738}}},  
4829 /* GC.C@..CG */  
4830 {{{{ DEF, -719, -309, -309, -389},  
4831 { -100, -769, -359, -359, -439},  
4832 { -100, -769, -359, -359, -439},  
4833 { -100, -769, -359, -359, -439},  
4834 { -100, -769, -359, -359, -439}}},  
4835 /* GC.CA..CG */  
4836 {{ DEF, -719, -309, -309, -389},  
4837 {-1079,-1748,-1338,-1338,-1418},  
4838 { -569,-1238, -828, -828, -908},  
4839 { -989,-1658,-1248,-1248,-1328},  
4840 { -859,-1528,-1118,-1118,-1198}}},  
4841 /* GC.CC..CG */  
4842 {{{{ DEF, -719, -309, -309, -389},  
4843 { -999,-1668,-1258,-1258,-1338},  
4844 { -499,-1168, -758, -758, -838},  
4845 { -989,-1658,-1248,-1248,-1328},  
4846 { -789,-1458,-1048,-1048,-1128}}},  
4847 /* GC.CG..CG */  
4848 {{{{ DEF, -719, -309, -309, -389},  
4849 {-1079,-1748,-1338,-1338,-1418},  
4850 { -569,-1238, -828, -828, -908},  
4851 { -989,-1658,-1248,-1248,-1328},  
4852 { -859,-1528,-1118,-1118,-1198}}},  
4853 /* GC.CU..CG */  
4854 {{{{ DEF, -719, -309, -309, -389},  
4855 {-1079,-1748,-1338,-1338,-1418},  
4856 { -719,-1388, -978, -978,-1058},  
4857 { -989,-1658,-1248,-1248,-1328},  
4858 { -909,-1578,-1168,-1168,-1248}}},  
4859 /* GC.G@..CG */  
4860 {{{{ DEF, -709, -739, -619, -739},  
4861 { -100, -759, -789, -669, -789},  
4862 { -100, -759, -789, -669, -789},  
4863 { -100, -759, -789, -669, -789},  
4864 { -100, -759, -789, -669, -789}}},  
4865 /* GC.GA..CG */  
4866 {{{{ DEF, -709, -739, -619, -739},  
4867 {-1079,-1738,-1768,-1648,-1768},  
4868 { -569,-1228,-1258,-1138,-1258},  
4869 { -989,-1648,-1678,-1558,-1678},  
4870 { -859,-1518,-1548,-1428,-1548}}},  
4871 /* GC.GC..CG */  
4872 {{{{ DEF, -709, -739, -619, -739},  
4873 { -999,-1658,-1688,-1568,-1688},  
4874 { -499,-1158,-1188,-1068,-1188},  
4875 { -989,-1648,-1678,-1558,-1678},  
4876 { -789,-1448,-1478,-1358,-1478}}},  
4877 /* GC.GG..CG */  
4878 {{{{ DEF, -709, -739, -619, -739},  
4879 {-1079,-1738,-1768,-1648,-1768},  
4880 { -569,-1228,-1258,-1138,-1258},  
4881 { -989,-1648,-1678,-1558,-1678},  
4882 { -859,-1518,-1548,-1428,-1548}}},  
4883 /* GC.GU..CG */  
4884 {{{{ DEF, -709, -739, -619, -739},  
4885 {-1079,-1738,-1768,-1648,-1768},  
4886 { -719,-1378,-1408,-1288,-1408},  
4887 { -989,-1648,-1678,-1558,-3080},  
4888 { -909,-1568,-1598,-1478,-1598}}},  
4889 /* GC.U@..CG */  
4890 {{{{ DEF, -499, -499, -499, -569},  
4891 { -100, -549, -549, -549, -619},  
4892 { -100, -549, -549, -549, -619},  
4893 { -100, -549, -549, -549, -619},  
4894 { -100, -549, -549, -549, -619}}},  
4895 /* GC.UA..CG */  
4896 {{{{ DEF, -499, -499, -499, -569},  
4897 {-1079,-1528,-1528,-1528,-1598},  
4898 { -569,-1018,-1018,-1018,-1088},
```

```

4899 { -989, -1438, -1438, -1438, -1508},
4900 { -859, -1308, -1308, -1308, -1378}},
4901 /* GC.UC..CG */
4902 {{ DEF, -499, -499, -499, -569},
4903 { -999, -1448, -1448, -1448, -1518},
4904 { -499, -948, -948, -948, -1018},
4905 { -989, -1438, -1438, -1438, -1508},
4906 { -789, -1238, -1238, -1238, -1308}},
4907 /* GC.UG..CG */
4908 {{ DEF, -499, -499, -499, -569},
4909 { -1079, -1528, -1528, -1528, -1598},
4910 { -569, -1018, -1018, -1018, -1088},
4911 { -989, -1438, -1438, -1438, -1508},
4912 { -859, -1308, -1308, -1308, -1378}},
4913 /* GC.UU..CG */
4914 {{ DEF, -499, -499, -499, -569},
4915 { -1079, -1528, -1528, -1528, -1598},
4916 { -719, -1168, -1168, -1168, -1238},
4917 { -989, -1438, -1438, -1438, -1508},
4918 { -909, -1358, -1358, -1358, -1428}}},
4919 /* GC.@@..GC */
4920 {{{ 0, 0, 0, 0, 0},
4921 { DEF, DEF, DEF, DEF, DEF},
4922 { DEF, DEF, DEF, DEF, DEF},
4923 { DEF, DEF, DEF, DEF, DEF},
4924 { DEF, DEF, DEF, DEF, DEF}},
4925 /* GC.@A..GC */
4926 {{ 0, 0, 0, 0, 0},
4927 { -519, -519, -519, -519, -519},
4928 { -719, -719, -719, -719, -719},
4929 { -709, -709, -709, -709, -709},
4930 { -499, -499, -499, -499, -499}},
4931 /* GC.@C..GC */
4932 {{ 0, 0, 0, 0, 0},
4933 { -879, -879, -879, -879, -879},
4934 { -309, -309, -309, -309, -309},
4935 { -739, -739, -739, -739, -739},
4936 { -499, -499, -499, -499, -499}},
4937 /* GC.@G..GC */
4938 {{ 0, 0, 0, 0, 0},
4939 { -559, -559, -559, -559, -559},
4940 { -309, -309, -309, -309, -309},
4941 { -619, -619, -619, -619, -619},
4942 { -499, -499, -499, -499, -499}},
4943 /* GC.@U..GC */
4944 {{ 0, 0, 0, 0, 0},
4945 { -879, -879, -879, -879, -879},
4946 { -389, -389, -389, -389, -389},
4947 { -739, -739, -739, -739, -739},
4948 { -569, -569, -569, -569, -569}}},
4949 /* GC.A@..GC */
4950 {{ DEF, -519, -879, -559, -879},
4951 { -100, -569, -929, -609, -929},
4952 { -100, -569, -929, -609, -929},
4953 { -100, -569, -929, -609, -929},
4954 { -100, -569, -929, -609, -929}},
4955 /* GC.AA..GC */
4956 {{ DEF, -519, -879, -559, -879},
4957 { -569, -1038, -1398, -1078, -1398},
4958 { -769, -1238, -1598, -1278, -1598},
4959 { -759, -1228, -1588, -1268, -1588},
4960 { -549, -1018, -1378, -1058, -1378}},
4961 /* GC.AC..GC */
4962 {{ DEF, -519, -879, -559, -879},
4963 { -929, -1398, -1758, -1438, -1758},
4964 { -359, -828, -1188, -868, -1188},
4965 { -789, -1258, -1618, -1298, -1618},
4966 { -549, -1018, -1378, -1058, -1378}},
4967 /* GC.AG..GC */
4968 {{ DEF, -519, -879, -559, -879},
4969 { -609, -1078, -1438, -1118, -1438},
4970 { -359, -828, -1188, -868, -1188},
4971 { -669, -1138, -1498, -1178, -1498},
4972 { -549, -1018, -1378, -1058, -1378}},
4973 /* GC.AU..GC */
4974 {{ DEF, -519, -879, -559, -879},
4975 { -929, -1398, -1758, -1438, -1758},
4976 { -439, -908, -1268, -948, -1268},
4977 { -789, -1258, -1618, -1298, -1618},
4978 { -619, -1088, -1448, -1128, -1448}}},
4979 /* GC.C@..GC */
4980 {{ DEF, -719, -309, -309, -309, -389},
4981 { -100, -769, -359, -359, -439},
4982 { -100, -769, -359, -359, -439},
4983 { -100, -769, -359, -359, -439},
4984 { -100, -769, -359, -359, -439}},
4985 /* GC.CA..GC */

```

```

4986 {{ DEF, -719, -309, -309, -389},
4987 { -569,-1238, -828, -828, -908},
4988 { -769,-1438,-1028,-1028,-1108},
4989 { -759,-1428,-1018,-1018,-1098},
4990 { -549,-1218, -808, -808, -888}},
4991 /* GC.CC..GC */
4992 {{ DEF, -719, -309, -309, -389},
4993 { -929,-1598,-1188,-1188,-1268},
4994 { -359,-1028, -618, -618, -698},
4995 { -789,-1458,-1048,-1048,-1128},
4996 { -549,-1218, -808, -808, -888}},
4997 /* GC.CG..GC */
4998 {{ DEF, -719, -309, -309, -389},
4999 { -609,-1278, -868, -868, -948},
5000 { -359,-1028, -618, -618, -698},
5001 { -669,-1338, -928, -928,-1008},
5002 { -549,-1218, -808, -808, -888}},
5003 /* GC.CU..GC */
5004 {{ DEF, -719, -309, -309, -389},
5005 { -929,-1598,-1188,-1188,-1268},
5006 { -439,-1108, -698, -698, -778},
5007 { -789,-1458,-1048,-1048,-1128},
5008 { -619,-1288, -878, -878, -958}},
5009 /* GC.G@..GC */
5010 {{{{ DEF, -709, -739, -619, -739},
5011 { -100, -759, -789, -669, -789},
5012 { -100, -759, -789, -669, -789},
5013 { -100, -759, -789, -669, -789},
5014 { -100, -759, -789, -669, -789}},
5015 /* GC.GA..GC */
5016 {{{ DEF, -709, -739, -619, -739},
5017 { -569,-1228,-1258,-1138,-1258},
5018 { -769,-1428,-1458,-1338,-1458},
5019 { -759,-1418,-1448,-1328,-1448},
5020 { -549,-1208,-1238,-1118,-1238}},
5021 /* GC.GC..GC */
5022 {{{ DEF, -709, -739, -619, -739},
5023 { -929,-1588,-1618,-1498,-1618},
5024 { -359,-1018,-1048, -928,-1048},
5025 { -789,-1448,-1478,-1358,-1478},
5026 { -549,-1208,-1238,-1118,-1238}},
5027 /* GC.GG..GC */
5028 {{{ DEF, -709, -739, -619, -739},
5029 { -609,-1268,-1298,-1178,-1298},
5030 { -359,-1018,-1048, -928,-1048},
5031 { -669,-1328,-1358,-1238,-1358},
5032 { -549,-1208,-1238,-1118,-1238}},
5033 /* GC.GU..GC */
5034 {{{ DEF, -709, -739, -619, -739},
5035 { -929,-1588,-1618,-1498,-1618},
5036 { -439,-1098,-1128,-1008,-1128},
5037 { -789,-1448,-1478,-1358,-3080},
5038 { -619,-1278,-1308,-1188,-1308}},
5039 /* GC.U@..GC */
5040 {{{{{ DEF, -499, -499, -499, -499, -569},
5041 { -100, -549, -549, -549, -619},
5042 { -100, -549, -549, -549, -619},
5043 { -100, -549, -549, -549, -619},
5044 { -100, -549, -549, -549, -619}},
5045 /* GC.UA..GC */
5046 {{{ DEF, -499, -499, -499, -499, -569},
5047 { -569,-1018,-1018,-1018,-1088},
5048 { -769,-1218,-1218,-1218,-1288},
5049 { -759,-1208,-1208,-1208,-1278},
5050 { -549, -998, -998, -998, -1068}},
5051 /* GC.UC..GC */
5052 {{{ DEF, -499, -499, -499, -499, -569},
5053 { -929,-1378,-1378,-1378,-1448},
5054 { -359, -808, -808, -808, -878},
5055 { -789,-1238,-1238,-1238,-1308},
5056 { -549, -998, -998, -998, -1068}},
5057 /* GC.UG..GC */
5058 {{{ DEF, -499, -499, -499, -499, -569},
5059 { -609,-1058,-1058,-1058,-1128},
5060 { -359, -808, -808, -808, -878},
5061 { -669,-1118,-1118,-1118,-1188},
5062 { -549, -998, -998, -998, -1068}},
5063 /* GC.UU..GC */
5064 {{{ DEF, -499, -499, -499, -499, -569},
5065 { -929,-1378,-1378,-1378,-1448},
5066 { -439, -888, -888, -888, -958},
5067 { -789,-1238,-1238,-1238,-1308},
5068 { -619,-1068,-1068,-1068,-1138}}},
5069 /* GC.@@..GU */
5070 {{{{ 0, 0, 0, 0, 0},
5071 { DEF, DEF, DEF, DEF, DEF},
5072 { DEF, DEF, DEF, DEF},

```

```

5073 {{ DEF, DEF, DEF, DEF, DEF},
5074 {{ DEF, DEF, DEF, DEF, DEF}},
5075 /* GC.@..GU */
5076 {{ 0, 0, 0, 0, 0},
5077 {-429, -429, -429, -429, -429},
5078 {-259, -259, -259, -259, -259},
5079 {-339, -339, -339, -339, -339},
5080 {-329, -329, -329, -329, -329}},
5081 /* GC.@C..GU */
5082 {{ 0, 0, 0, 0, 0},
5083 {-599, -599, -599, -599, -599},
5084 {-239, -239, -239, -239, -239},
5085 {-689, -689, -689, -689, -689},
5086 {-329, -329, -329, -329, -329}},
5087 /* GC.@G..GU */
5088 {{ 0, 0, 0, 0, 0},
5089 {-599, -599, -599, -599, -599},
5090 {-239, -239, -239, -239, -239},
5091 {-689, -689, -689, -689, -689},
5092 {-329, -329, -329, -329, -329}},
5093 /* GC.@U..GU */
5094 {{ 0, 0, 0, 0, 0},
5095 {-599, -599, -599, -599, -599},
5096 {-239, -239, -239, -239, -239},
5097 {-689, -689, -689, -689, -689},
5098 {-329, -329, -329, -329, -329}},
5099 /* GC.A@..GU */
5100 {{ { DEF, -519, -879, -559, -879},
5101 {-100, -569, -929, -609, -929},
5102 {-100, -569, -929, -609, -929},
5103 {-100, -569, -929, -609, -929},
5104 {-100, -569, -929, -609, -929}},
5105 /* GC.AA..GU */
5106 {{ { DEF, -519, -879, -559, -879},
5107 {-479, -948, -1308, -988, -1308},
5108 {-309, -778, -1138, -818, -1138},
5109 {-389, -858, -1218, -898, -1218},
5110 {-379, -848, -1208, -888, -1208}},
5111 /* GC.AC..GU */
5112 {{ { DEF, -519, -879, -559, -879},
5113 {-649, -1118, -1478, -1158, -1478},
5114 {-289, -758, -1118, -798, -1118},
5115 {-739, -1208, -1568, -1248, -1568},
5116 {-379, -848, -1208, -888, -1208}},
5117 /* GC.AG..GU */
5118 {{ { DEF, -519, -879, -559, -879},
5119 {-649, -1118, -1478, -1158, -1478},
5120 {-289, -758, -1118, -798, -1118},
5121 {-739, -1208, -1568, -1248, -1568},
5122 {-379, -848, -1208, -888, -1208}},
5123 /* GC.AU..GU */
5124 {{ { DEF, -519, -879, -559, -879},
5125 {-649, -1118, -1478, -1158, -1478},
5126 {-289, -758, -1118, -798, -1118},
5127 {-739, -1208, -1568, -1248, -1568},
5128 {-379, -848, -1208, -888, -1208}},
5129 /* GC.C@..GU */
5130 {{ { DEF, -719, -309, -309, -389},
5131 {-100, -769, -359, -359, -439},
5132 {-100, -769, -359, -359, -439},
5133 {-100, -769, -359, -359, -439},
5134 {-100, -769, -359, -359, -439}},
5135 /* GC.CA..GU */
5136 {{ { DEF, -719, -309, -309, -389},
5137 {-479, -1148, -738, -738, -818},
5138 {-309, -978, -568, -568, -648},
5139 {-389, -1058, -648, -648, -728},
5140 {-379, -1048, -638, -638, -718}},
5141 /* GC.CC..GU */
5142 {{ { DEF, -719, -309, -309, -389},
5143 {-649, -1318, -908, -908, -988},
5144 {-289, -958, -548, -548, -628},
5145 {-739, -1408, -998, -998, -1078},
5146 {-379, -1048, -638, -638, -718}},
5147 /* GC.CG..GU */
5148 {{ { DEF, -719, -309, -309, -389},
5149 {-649, -1318, -908, -908, -988},
5150 {-289, -958, -548, -548, -628},
5151 {-739, -1408, -998, -998, -1078},
5152 {-379, -1048, -638, -638, -718}},
5153 /* GC.CU..GU */
5154 {{ { DEF, -719, -309, -309, -389},
5155 {-649, -1318, -908, -908, -988},
5156 {-289, -958, -548, -548, -628},
5157 {-739, -1408, -998, -998, -1078},
5158 {-379, -1048, -638, -638, -718}},
5159 /* GC.G@..GU */

```

```

5160 {{ { DEF, -709, -739, -619, -739},
5161 { -100, -759, -789, -669, -789},
5162 { -100, -759, -789, -669, -789},
5163 { -100, -759, -789, -669, -789}},
5164 { -100, -759, -789, -669, -789}},
5165 /* GC.GA..GU */
5166 {{ { DEF, -709, -739, -619, -739},
5167 { -479,-1138,-1168,-1048,-1168},
5168 { -309, -968, -998, -878, -998},
5169 { -389,-1048,-1078, -958,-1078},
5170 { -379,-1038,-1068, -948,-1068}},
5171 /* GC.GC..GU */
5172 {{ { DEF, -709, -739, -619, -739},
5173 { -649,-1308,-1338,-1218,-1338},
5174 { -289, -948, -978, -858, -978},
5175 { -739,-1398,-1428,-1308,-1428},
5176 { -379,-1038,-1068, -948,-1068}},
5177 /* GC.GG..GU */
5178 {{ { DEF, -709, -739, -619, -739},
5179 { -649,-1308,-1338,-1218,-1338},
5180 { -289, -948, -978, -858, -978},
5181 { -739,-1398,-1428,-1308,-1428},
5182 { -379,-1038,-1068, -948,-1068}},
5183 /* GC.GU..GU */
5184 {{ { DEF, -709, -739, -619, -739},
5185 { -649,-1308,-1338,-1218,-1338},
5186 { -289, -948, -978, -858, -978},
5187 { -739,-1398,-1428,-1308,-1428},
5188 { -379,-1038,-1068, -948,-1068}},
5189 /* GC.U@..GU */
5190 {{ { DEF, -499, -499, -499, -569},
5191 { -100, -549, -549, -549, -619},
5192 { -100, -549, -549, -549, -619},
5193 { -100, -549, -549, -549, -619},
5194 { -100, -549, -549, -549, -619}},
5195 /* GC.UA..GU */
5196 {{ { DEF, -499, -499, -499, -569},
5197 { -479, -928, -928, -928, -998},
5198 { -309, -758, -758, -758, -828},
5199 { -389, -838, -838, -838, -908},
5200 { -379, -828, -828, -828, -898}},
5201 /* GC.UC..GU */
5202 {{ { DEF, -499, -499, -499, -569},
5203 { -649,-1098,-1098,-1098,-1168},
5204 { -289, -738, -738, -738, -808},
5205 { -739,-1188,-1188,-1188,-1258},
5206 { -379, -828, -828, -828, -898}},
5207 /* GC.UG..GU */
5208 {{ { DEF, -499, -499, -499, -569},
5209 { -649,-1098,-1098,-1098,-1168},
5210 { -289, -738, -738, -738, -808},
5211 { -739,-1188,-1188,-1188,-1258},
5212 { -379, -828, -828, -828, -898}},
5213 /* GC.UU..GU */
5214 {{ { DEF, -499, -499, -499, -569},
5215 { -649,-1098,-1098,-1098,-1168},
5216 { -289, -738, -738, -738, -808},
5217 { -739,-1188,-1188,-1188,-1258},
5218 { -379, -828, -828, -828, -898}}},
5219 /* GC.@@..UG */
5220 {{ { 0, 0, 0, 0, 0},
5221 { DEF, DEF, DEF, DEF, DEF},
5222 { DEF, DEF, DEF, DEF, DEF},
5223 { DEF, DEF, DEF, DEF, DEF},
5224 { DEF, DEF, DEF, DEF, DEF}},
5225 /* GC.@A..UG */
5226 {{ { 0, 0, 0, 0, 0},
5227 { -719, -719, -719, -719, -719},
5228 { -479, -479, -479, -479, -479},
5229 { -659, -659, -659, -659, -659},
5230 { -549, -549, -549, -549, -549}},
5231 /* GC.@C..UG */
5232 {{ { 0, 0, 0, 0, 0},
5233 { -789, -789, -789, -789, -789},
5234 { -479, -479, -479, -479, -479},
5235 { -809, -809, -809, -809, -809},
5236 { -439, -439, -439, -439, -439}},
5237 /* GC.@G..UG */
5238 {{ { 0, 0, 0, 0, 0},
5239 { -959, -959, -959, -959, -959},
5240 { -359, -359, -359, -359, -359},
5241 { -919, -919, -919, -919, -919},
5242 { -549, -549, -549, -549, -549}},
5243 /* GC.@U..UG */
5244 {{ { 0, 0, 0, 0, 0},
5245 { -809, -809, -809, -809, -809},
5246 { -479, -479, -479, -479, -479}},

```

```

5247 { -809, -809, -809, -809, -809},  

5248 { -359, -359, -359, -359, -359}}},  

5249 /* GC.A@..UG */  

5250 {{ DEF, -519, -879, -559, -879},  

5251 { -100, -569, -929, -609, -929},  

5252 { -100, -569, -929, -609, -929},  

5253 { -100, -569, -929, -609, -929},  

5254 { -100, -569, -929, -609, -929}}},  

5255 /* GC.AA..UG */  

5256 {{ DEF, -519, -879, -559, -879},  

5257 { -769,-1238,-1598,-1278,-1598},  

5258 { -529, -998,-1358,-1038,-1358},  

5259 { -709,-1178,-1538,-1218,-1538},  

5260 { -599,-1068,-1428,-1108,-1428}}},  

5261 /* GC.AC..UG */  

5262 {{ DEF, -519, -879, -559, -879},  

5263 { -839,-1308,-1668,-1348,-1668},  

5264 { -529, -998,-1358,-1038,-1358},  

5265 { -859,-1328,-1688,-1368,-1688},  

5266 { -489, -958,-1318, -998,-1318}}},  

5267 /* GC.AG..UG */  

5268 {{ DEF, -519, -879, -559, -879},  

5269 {-1009,-1478,-1838,-1518,-1838},  

5270 { -409, -878,-1238, -918,-1238},  

5271 { -969,-1438,-1798,-1478,-1798},  

5272 { -599,-1068,-1428,-1108,-1428}}},  

5273 /* GC.AU..UG */  

5274 {{ DEF, -519, -879, -559, -879},  

5275 { -859,-1328,-1688,-1368,-1688},  

5276 { -529, -998,-1358,-1038,-1358},  

5277 { -859,-1328,-1688,-1368,-1688},  

5278 { -409, -878,-1238, -918,-1238}}},  

5279 /* GC.C@..UG */  

5280 {{ DEF, -719, -309, -309, -389},  

5281 { -100, -769, -359, -359, -439},  

5282 { -100, -769, -359, -359, -439},  

5283 { -100, -769, -359, -359, -439},  

5284 { -100, -769, -359, -359, -439}}},  

5285 /* GC.CA..UG */  

5286 {{ DEF, -719, -309, -309, -389},  

5287 { -769,-1438,-1028,-1028,-1108},  

5288 { -529,-1198, -788, -788, -868},  

5289 { -709,-1378, -968, -968,-1048},  

5290 { -599,-1268, -858, -858, -938}}},  

5291 /* GC.CC..UG */  

5292 {{ DEF, -719, -309, -309, -389},  

5293 { -839,-1508,-1098,-1098,-1178},  

5294 { -529,-1198, -788, -788, -868},  

5295 { -859,-1528,-1118,-1118,-1198},  

5296 { -489,-1158, -748, -748, -828}}},  

5297 /* GC.CG..UG */  

5298 {{ DEF, -719, -309, -309, -389},  

5299 {-1009,-1678,-1268,-1268,-1348},  

5300 { -409,-1078, -668, -668, -748},  

5301 { -969,-1638,-1228,-1228,-1308},  

5302 { -599,-1268, -858, -858, -938}}},  

5303 /* GC.CU..UG */  

5304 {{ DEF, -719, -309, -309, -389},  

5305 { -859,-1528,-1118,-1118,-1198},  

5306 { -529,-1198, -788, -788, -868},  

5307 { -859,-1528,-1118,-1118,-1198},  

5308 { -409,-1078, -668, -668, -748}}},  

5309 /* GC.G@..UG */  

5310 {{ DEF, -709, -739, -619, -739},  

5311 { -100, -759, -789, -669, -789},  

5312 { -100, -759, -789, -669, -789},  

5313 { -100, -759, -789, -669, -789},  

5314 { -100, -759, -789, -669, -789}}},  

5315 /* GC.GA..UG */  

5316 {{ DEF, -709, -739, -619, -739},  

5317 { -769,-1428,-1458,-1338,-1458},  

5318 { -529,-1188,-1218,-1098,-1218},  

5319 { -709,-1368,-1398,-1278,-1398},  

5320 { -599,-1258,-1288,-1168,-1288}}},  

5321 /* GC.GC..UG */  

5322 {{ DEF, -709, -739, -619, -739},  

5323 { -839,-1498,-1528,-1408,-1528},  

5324 { -529,-1188,-1218,-1098,-1218},  

5325 { -859,-1518,-1548,-1428,-1548},  

5326 { -489,-1148,-1178,-1058,-1178}}},  

5327 /* GC.GG..UG */  

5328 {{ DEF, -709, -739, -619, -739},  

5329 {-1009,-1668,-1698,-1578,-1698},  

5330 { -409,-1068,-1098, -978,-1098},  

5331 { -969,-1628,-1658,-1538,-1658},  

5332 { -599,-1258,-1288,-1168,-1288}}},  

5333 /* GC.GU..UG */

```

```

5334 {{ DEF, -709, -739, -619, -739},
5335 { -859,-1518,-1548,-1428,-1548},
5336 { -529,-1188,-1218,-1098,-1218},
5337 { -859,-1518,-1548,-1428,-1548},
5338 { -409,-1068,-1098, -978,-1098}}}},
5339 /* GC.U@..UG */
5340 {{ DEF, -499, -499, -499, -569},
5341 { -100, -549, -549, -549, -619},
5342 { -100, -549, -549, -549, -619},
5343 { -100, -549, -549, -549, -619},
5344 { -100, -549, -549, -549, -619}}},
5345 /* GC.UA..UG */
5346 {{ DEF, -499, -499, -499, -569},
5347 { -769,-1218,-1218,-1218,-1288},
5348 { -529, -978, -978, -978,-1048},
5349 { -709,-1158,-1158,-1158,-1228},
5350 { -599,-1048,-1048,-1048,-1118}}},
5351 /* GC.UC..UG */
5352 {{ DEF, -499, -499, -499, -569},
5353 { -839,-1288,-1288,-1288,-1358},
5354 { -529, -978, -978, -978,-1048},
5355 { -859,-1308,-1308,-1308,-1378},
5356 { -489, -938, -938, -938,-1008}}},
5357 /* GC.UG..UG */
5358 {{ DEF, -499, -499, -499, -569},
5359 {-1009,-1458,-1458,-1458,-1528},
5360 { -409, -858, -858, -858, -928},
5361 { -969,-1418,-1418,-1418,-1488},
5362 { -599,-1048,-1048,-1048,-1118}}},
5363 /* GC.UU..UG */
5364 {{ DEF, -499, -499, -499, -569},
5365 { -859,-1308,-1308,-1308,-1378},
5366 { -529, -978, -978, -978,-1048},
5367 { -859,-1308,-1308,-1308,-1378},
5368 { -409, -858, -858, -858, -928}}}},
5369 /* GC.@@..AU */
5370 {{{ 0, 0, 0, 0, 0},
5371 { DEF, DEF, DEF, DEF, DEF},
5372 { DEF, DEF, DEF, DEF, DEF},
5373 { DEF, DEF, DEF, DEF, DEF},
5374 { DEF, DEF, DEF, DEF, DEF}}},
5375 /* GC.@A..AU */
5376 {{ 0, 0, 0, 0, 0},
5377 { -429, -429, -429, -429, -429},
5378 { -259, -259, -259, -259, -259},
5379 { -339, -339, -339, -339, -339},
5380 { -329, -329, -329, -329, -329}}},
5381 /* GC.@C..AU */
5382 {{ 0, 0, 0, 0, 0},
5383 { -599, -599, -599, -599, -599},
5384 { -239, -239, -239, -239, -239},
5385 { -689, -689, -689, -689, -689},
5386 { -329, -329, -329, -329, -329}}},
5387 /* GC.@G..AU */
5388 {{ 0, 0, 0, 0, 0},
5389 { -599, -599, -599, -599, -599},
5390 { -239, -239, -239, -239, -239},
5391 { -689, -689, -689, -689, -689},
5392 { -329, -329, -329, -329, -329}}},
5393 /* GC.@U..AU */
5394 {{ 0, 0, 0, 0, 0},
5395 { -599, -599, -599, -599, -599},
5396 { -239, -239, -239, -239, -239},
5397 { -689, -689, -689, -689, -689},
5398 { -329, -329, -329, -329, -329}}},
5399 /* GC.A@..AU */
5400 {{ DEF, -519, -879, -559, -879},
5401 { -100, -569, -929, -609, -929},
5402 { -100, -569, -929, -609, -929},
5403 { -100, -569, -929, -609, -929},
5404 { -100, -569, -929, -609, -929}}},
5405 /* GC.AA..AU */
5406 {{ DEF, -519, -879, -559, -879},
5407 { -479, -948,-1308, -988,-1308},
5408 { -309, -778,-1138, -818,-1138},
5409 { -389, -858,-1218, -898,-1218},
5410 { -379, -848,-1208, -888,-1208}}},
5411 /* GC.AC..AU */
5412 {{ DEF, -519, -879, -559, -879},
5413 { -649,-1118,-1478,-1158,-1478},
5414 { -289, -758,-1118, -798,-1118},
5415 { -739,-1208,-1568,-1248,-1568},
5416 { -379, -848,-1208, -888,-1208}}},
5417 /* GC.AG..AU */
5418 {{ DEF, -519, -879, -559, -879},
5419 { -649,-1118,-1478,-1158,-1478},
5420 { -289, -758,-1118, -798,-1118}},

```

```

5421 { -739, -1208, -1568, -1248, -1568},
5422 { -379, -848, -1208, -888, -1208}},
5423 /* GC.AU..AU */
5424 {{ DEF, -519, -879, -559, -879},
5425 { -649, -1118, -1478, -1158, -1478},
5426 { -289, -758, -1118, -798, -1118},
5427 { -739, -1208, -1568, -1248, -1568},
5428 { -379, -848, -1208, -888, -1208}}},
5429 /* GC.C@..AU */
5430 {{ DEF, -719, -309, -309, -389},
5431 { -100, -769, -359, -359, -439},
5432 { -100, -769, -359, -359, -439},
5433 { -100, -769, -359, -359, -439},
5434 { -100, -769, -359, -359, -439}},
5435 /* GC.CA..AU */
5436 {{ DEF, -719, -309, -309, -389},
5437 { -479, -1148, -738, -738, -818},
5438 { -309, -978, -568, -568, -648},
5439 { -389, -1058, -648, -648, -728},
5440 { -379, -1048, -638, -638, -718}}},
5441 /* GC.CC..AU */
5442 {{ DEF, -719, -309, -309, -389},
5443 { -649, -1318, -908, -908, -988},
5444 { -289, -958, -548, -548, -628},
5445 { -739, -1408, -998, -998, -1078},
5446 { -379, -1048, -638, -638, -718}},
5447 /* GC.CG..AU */
5448 {{ DEF, -719, -309, -309, -389},
5449 { -649, -1318, -908, -908, -988},
5450 { -289, -958, -548, -548, -628},
5451 { -739, -1408, -998, -998, -1078},
5452 { -379, -1048, -638, -638, -718}}},
5453 /* GC.CU..AU */
5454 {{ DEF, -719, -309, -309, -389},
5455 { -649, -1318, -908, -908, -988},
5456 { -289, -958, -548, -548, -628},
5457 { -739, -1408, -998, -998, -1078},
5458 { -379, -1048, -638, -638, -718}}},
5459 /* GC.G@..AU */
5460 {{ DEF, -709, -739, -619, -739},
5461 { -100, -759, -789, -669, -789},
5462 { -100, -759, -789, -669, -789},
5463 { -100, -759, -789, -669, -789},
5464 { -100, -759, -789, -669, -789}},
5465 /* GC.GA..AU */
5466 {{ DEF, -709, -739, -619, -739},
5467 { -479, -1138, -1168, -1048, -1168},
5468 { -309, -968, -998, -878, -998},
5469 { -389, -1048, -1078, -958, -1078},
5470 { -379, -1038, -1068, -948, -1068}}},
5471 /* GC.GC..AU */
5472 {{ DEF, -709, -739, -619, -739},
5473 { -649, -1308, -1338, -1218, -1338},
5474 { -289, -948, -978, -858, -978},
5475 { -739, -1398, -1428, -1308, -1428},
5476 { -379, -1038, -1068, -948, -1068}},
5477 /* GC.GG..AU */
5478 {{ DEF, -709, -739, -619, -739},
5479 { -649, -1308, -1338, -1218, -1338},
5480 { -289, -948, -978, -858, -978},
5481 { -739, -1398, -1428, -1308, -1428},
5482 { -379, -1038, -1068, -948, -1068}},
5483 /* GC.GU..AU */
5484 {{ DEF, -709, -739, -619, -739},
5485 { -649, -1308, -1338, -1218, -1338},
5486 { -289, -948, -978, -858, -978},
5487 { -739, -1398, -1428, -1308, -1428},
5488 { -379, -1038, -1068, -948, -1068}}},
5489 /* GC.U@..AU */
5490 {{ DEF, -499, -499, -499, -569},
5491 { -100, -549, -549, -549, -619},
5492 { -100, -549, -549, -549, -619},
5493 { -100, -549, -549, -549, -619},
5494 { -100, -549, -549, -549, -619}},
5495 /* GC.UA..AU */
5496 {{ DEF, -499, -499, -499, -569},
5497 { -479, -928, -928, -928, -998},
5498 { -309, -758, -758, -758, -828},
5499 { -389, -838, -838, -838, -908},
5500 { -379, -828, -828, -828, -898}},
5501 /* GC.UC..AU */
5502 {{ DEF, -499, -499, -499, -569},
5503 { -649, -1098, -1098, -1098, -1168},
5504 { -289, -738, -738, -738, -808},
5505 { -739, -1188, -1188, -1188, -1258},
5506 { -379, -828, -828, -828, -898}},
5507 /* GC.UG..AU */

```

```

5508 {{ DEF, -499, -499, -499, -569},
5509 { -649,-1098,-1098,-1098,-1168},
5510 { -289, -738, -738, -738, -808},
5511 { -739,-1188,-1188,-1188,-1258},
5512 { -379, -828, -828, -828, -898}},
5513 /* GC.UU..UA */
5514 {{ DEF, -499, -499, -499, -569},
5515 { -649,-1098,-1098,-1098,-1168},
5516 { -289, -738, -738, -738, -808},
5517 { -739,-1188,-1188,-1188,-1258},
5518 { -379, -828, -828, -828, -898}}}),
5519 /* GC.@@..UA */
5520 {{{ 0, 0, 0, 0, 0},
5521 { DEF, DEF, DEF, DEF, DEF},
5522 { DEF, DEF, DEF, DEF, DEF},
5523 { DEF, DEF, DEF, DEF, DEF},
5524 { DEF, DEF, DEF, DEF, DEF}},
5525 /* GC.@A..UA */
5526 {{ 0, 0, 0, 0, 0},
5527 { -399, -399, -399, -399, -399},
5528 { -429, -429, -429, -429, -429},
5529 { -379, -379, -379, -379, -379},
5530 { -279, -279, -279, -279, -279}},
5531 /* GC.@C..UA */
5532 {{ 0, 0, 0, 0, 0},
5533 { -629, -629, -629, -629, -629},
5534 { -509, -509, -509, -509, -509},
5535 { -679, -679, -679, -679, -679},
5536 { -139, -139, -139, -139, -139}},
5537 /* GC.@G..UA */
5538 {{ 0, 0, 0, 0, 0},
5539 { -889, -889, -889, -889, -889},
5540 { -199, -199, -199, -199, -199},
5541 { -889, -889, -889, -889, -889},
5542 { -279, -279, -279, -279, -279}},
5543 /* GC.@U..UA */
5544 {{ 0, 0, 0, 0, 0},
5545 { -589, -589, -589, -589, -589},
5546 { -179, -179, -179, -179, -179},
5547 { -679, -679, -679, -679, -679},
5548 { -140, -140, -140, -140, -140}},
5549 /* GC.A@..UA */
5550 {{ DEF, -519, -879, -559, -879},
5551 { -100, -569, -929, -609, -929},
5552 { -100, -569, -929, -609, -929},
5553 { -100, -569, -929, -609, -929},
5554 { -100, -569, -929, -609, -929}},
5555 /* GC.AA..UA */
5556 {{ DEF, -519, -879, -559, -879},
5557 { -449, -918,-1278, -958,-1278},
5558 { -479, -948,-1308, -988,-1308},
5559 { -429, -898,-1258, -938,-1258},
5560 { -329, -798,-1158, -838,-1158}},
5561 /* GC.AC..UA */
5562 {{ DEF, -519, -879, -559, -879},
5563 { -679,-1148,-1508,-1188,-1508},
5564 { -559,-1028,-1388,-1068,-1388},
5565 { -729,-1198,-1558,-1238,-1558},
5566 { -189, -658,-1018, -698,-1018}},
5567 /* GC.AG..UA */
5568 {{ DEF, -519, -879, -559, -879},
5569 { -939,-1408,-1768,-1448,-1768},
5570 { -249, -718,-1078, -758,-1078},
5571 { -939,-1408,-1768,-1448,-1768},
5572 { -329, -798,-1158, -838,-1158}},
5573 /* GC.AU..UA */
5574 {{ DEF, -519, -879, -559, -879},
5575 { -639,-1108,-1468,-1148,-1468},
5576 { -229, -698,-1058, -738,-1058},
5577 { -729,-1198,-1558,-1238,-1558},
5578 { -190, -659,-1019, -699,-1019}},
5579 /* GC.C@..UA */
5580 {{ DEF, -719, -309, -309, -389},
5581 { -100, -769, -359, -359, -439},
5582 { -100, -769, -359, -359, -439},
5583 { -100, -769, -359, -359, -439},
5584 { -100, -769, -359, -359, -439}},
5585 /* GC.CA..UA */
5586 {{ DEF, -719, -309, -309, -389},
5587 { -449,-1118, -708, -708, -788},
5588 { -479,-1148, -738, -738, -818},
5589 { -429,-1098, -688, -688, -768},
5590 { -329, -998, -588, -588, -668}},
5591 /* GC.CC..UA */
5592 {{ DEF, -719, -309, -309, -389},
5593 { -679,-1348, -938, -938, -1018},
5594 { -559,-1228, -818, -818, -898},

```

```

5595 { -729, -1398, -988, -988, -1068},
5596 { -189, -858, -448, -448, -528}},
5597 /* GC.CG..UA */
5598 {{ DEF, -719, -309, -309, -389},
5599 { -939, -1608, -1198, -1198, -1278},
5600 { -249, -918, -508, -508, -588},
5601 { -939, -1608, -1198, -1198, -1278},
5602 { -329, -998, -588, -588, -668}},
5603 /* GC.CU..UA */
5604 {{ DEF, -719, -309, -309, -389},
5605 { -639, -1308, -898, -898, -978},
5606 { -229, -898, -488, -488, -568},
5607 { -729, -1398, -988, -988, -1068},
5608 { -190, -859, -449, -449, -529}}},
5609 /* GC.G@..UA */
5610 {{ DEF, -709, -739, -619, -739},
5611 { -100, -759, -789, -669, -789},
5612 { -100, -759, -789, -669, -789},
5613 { -100, -759, -789, -669, -789},
5614 { -100, -759, -789, -669, -789}},
5615 /* GC.GA..UA */
5616 {{ DEF, -709, -739, -619, -739},
5617 { -449, -1108, -1138, -1018, -1138},
5618 { -479, -1138, -1168, -1048, -1168},
5619 { -429, -1088, -1118, -998, -1118},
5620 { -329, -988, -1018, -898, -1018}},
5621 /* GC.GC..UA */
5622 {{ DEF, -709, -739, -619, -739},
5623 { -679, -1338, -1368, -1248, -1368},
5624 { -559, -1218, -1248, -1128, -1248},
5625 { -729, -1388, -1418, -1298, -1418},
5626 { -189, -848, -878, -758, -878}},
5627 /* GC.GG..UA */
5628 {{ DEF, -709, -739, -619, -739},
5629 { -939, -1598, -1628, -1508, -1628},
5630 { -249, -908, -938, -818, -938},
5631 { -939, -1598, -1628, -1508, -1628},
5632 { -329, -988, -1018, -898, -1018}},
5633 /* GC.GU..UA */
5634 {{ DEF, -709, -739, -619, -739},
5635 { -639, -1298, -1328, -1208, -1328},
5636 { -229, -888, -918, -798, -918},
5637 { -729, -1388, -1418, -1298, -1418},
5638 { -190, -849, -879, -759, -879}}},
5639 /* GC.U@..UA */
5640 {{ DEF, -499, -499, -499, -569},
5641 { -100, -549, -549, -549, -619},
5642 { -100, -549, -549, -549, -619},
5643 { -100, -549, -549, -549, -619},
5644 { -100, -549, -549, -549, -619}},
5645 /* GC.UA..UA */
5646 {{ DEF, -499, -499, -499, -569},
5647 { -449, -898, -898, -898, -968},
5648 { -479, -928, -928, -928, -998},
5649 { -429, -878, -878, -878, -948},
5650 { -329, -778, -778, -778, -848}},
5651 /* GC.UC..UA */
5652 {{ DEF, -499, -499, -499, -569},
5653 { -679, -1128, -1128, -1128, -1198},
5654 { -559, -1008, -1008, -1008, -1078},
5655 { -729, -1178, -1178, -1178, -1248},
5656 { -189, -638, -638, -638, -708}},
5657 /* GC.UG..UA */
5658 {{ DEF, -499, -499, -499, -569},
5659 { -939, -1388, -1388, -1388, -1458},
5660 { -249, -698, -698, -698, -768},
5661 { -939, -1388, -1388, -1388, -1458},
5662 { -329, -778, -778, -778, -848}}},
5663 /* GC.UU..UA */
5664 {{ DEF, -499, -499, -499, -569},
5665 { -639, -1088, -1088, -1088, -1158},
5666 { -229, -678, -678, -678, -748},
5667 { -729, -1178, -1178, -1178, -1248},
5668 { -190, -639, -639, -639, -709}}},
5669 /* GC.@@.. @ */
5670 {{{ DEF, DEF, DEF, DEF, DEF},
5671 { DEF, DEF, DEF, DEF, DEF},
5672 { DEF, DEF, DEF, DEF, DEF},
5673 { DEF, DEF, DEF, DEF, DEF},
5674 { DEF, DEF, DEF, DEF, DEF}},
5675 /* GC.@@.. @ */
5676 {{ DEF, DEF, DEF, DEF, DEF},
5677 { DEF, DEF, DEF, DEF, DEF},
5678 { DEF, DEF, DEF, DEF, DEF},
5679 { DEF, DEF, DEF, DEF, DEF},
5680 { DEF, DEF, DEF, DEF, DEF}},
5681 /* GC.@@.. @ */

```

```
5682 {{ DEF, DEF, DEF, DEF, DEF},
5683 { DEF, DEF, DEF, DEF, DEF},
5684 { DEF, DEF, DEF, DEF, DEF},
5685 { DEF, DEF, DEF, DEF, DEF},
5686 { DEF, DEF, DEF, DEF, DEF}},
5687 /* GC.@G.. @ */
5688 {{ DEF, DEF, DEF, DEF, DEF},
5689 { DEF, DEF, DEF, DEF, DEF},
5690 { DEF, DEF, DEF, DEF, DEF},
5691 { DEF, DEF, DEF, DEF, DEF},
5692 { DEF, DEF, DEF, DEF, DEF}},
5693 /* GC.@U.. @ */
5694 {{ DEF, DEF, DEF, DEF, DEF},
5695 { DEF, DEF, DEF, DEF, DEF},
5696 { DEF, DEF, DEF, DEF, DEF},
5697 { DEF, DEF, DEF, DEF, DEF},
5698 { DEF, DEF, DEF, DEF, DEF}}),
5699 /* GC.A@.. @ */
5700 {{{-100, -569, -929, -609, -929},
5701 {-100, -569, -929, -609, -929},
5702 {-100, -569, -929, -609, -929},
5703 {-100, -569, -929, -609, -929},
5704 {-100, -569, -929, -609, -929}}},
5705 /* GC.AA.. @ */
5706 {{-100, -569, -929, -609, -929},
5707 {-100, -569, -929, -609, -929},
5708 {-100, -569, -929, -609, -929},
5709 {-100, -569, -929, -609, -929},
5710 {-100, -569, -929, -609, -929}},
5711 /* GC.AC.. @ */
5712 {{-100, -569, -929, -609, -929},
5713 {-100, -569, -929, -609, -929},
5714 {-100, -569, -929, -609, -929},
5715 {-100, -569, -929, -609, -929},
5716 {-100, -569, -929, -609, -929}}},
5717 /* GC.AG.. @ */
5718 {{-100, -569, -929, -609, -929},
5719 {-100, -569, -929, -609, -929},
5720 {-100, -569, -929, -609, -929},
5721 {-100, -569, -929, -609, -929},
5722 {-100, -569, -929, -609, -929}},
5723 /* GC.AU.. @ */
5724 {{-100, -569, -929, -609, -929},
5725 {-100, -569, -929, -609, -929},
5726 {-100, -569, -929, -609, -929},
5727 {-100, -569, -929, -609, -929},
5728 {-100, -569, -929, -609, -929}}},
5729 /* GC.C@.. @ */
5730 {{{-100, -769, -359, -359, -439},
5731 {-100, -769, -359, -359, -439},
5732 {-100, -769, -359, -359, -439},
5733 {-100, -769, -359, -359, -439},
5734 {-100, -769, -359, -359, -439}}},
5735 /* GC.CA.. @ */
5736 {{-100, -769, -359, -359, -439},
5737 {-100, -769, -359, -359, -439},
5738 {-100, -769, -359, -359, -439},
5739 {-100, -769, -359, -359, -439},
5740 {-100, -769, -359, -359, -439}},
5741 /* GC.CC.. @ */
5742 {{-100, -769, -359, -359, -439},
5743 {-100, -769, -359, -359, -439},
5744 {-100, -769, -359, -359, -439},
5745 {-100, -769, -359, -359, -439},
5746 {-100, -769, -359, -359, -439}}},
5747 /* GC.CG.. @ */
5748 {{-100, -769, -359, -359, -439},
5749 {-100, -769, -359, -359, -439},
5750 {-100, -769, -359, -359, -439},
5751 {-100, -769, -359, -359, -439},
5752 {-100, -769, -359, -359, -439}}},
5753 /* GC.CU.. @ */
5754 {{-100, -769, -359, -359, -439},
5755 {-100, -769, -359, -359, -439},
5756 {-100, -769, -359, -359, -439},
5757 {-100, -769, -359, -359, -439},
5758 {-100, -769, -359, -359, -439}}},
5759 /* GC.G@.. @ */
5760 {{{-100, -759, -789, -669, -789},
5761 {-100, -759, -789, -669, -789},
5762 {-100, -759, -789, -669, -789},
5763 {-100, -759, -789, -669, -789},
5764 {-100, -759, -789, -669, -789}}},
5765 /* GC.GA.. @ */
5766 {{-100, -759, -789, -669, -789},
5767 {-100, -759, -789, -669, -789},
5768 {-100, -759, -789, -669, -789}},
```

```

5769 { -100, -759, -789, -669, -789},
5770 { -100, -759, -789, -669, -789}},
5771 /* GC.GC.. @ */
5772 {{ -100, -759, -789, -669, -789},
5773 { -100, -759, -789, -669, -789},
5774 { -100, -759, -789, -669, -789},
5775 { -100, -759, -789, -669, -789},
5776 { -100, -759, -789, -669, -789}},
5777 /* GC.GG.. @ */
5778 {{ -100, -759, -789, -669, -789},
5779 { -100, -759, -789, -669, -789},
5780 { -100, -759, -789, -669, -789},
5781 { -100, -759, -789, -669, -789},
5782 { -100, -759, -789, -669, -789}},
5783 /* GC.GU.. @ */
5784 {{ -100, -759, -789, -669, -789},
5785 { -100, -759, -789, -669, -789},
5786 { -100, -759, -789, -669, -789},
5787 { -100, -759, -789, -669, -789},
5788 { -100, -759, -789, -669, -789}},
5789 /* GC.U@.. @ */
5790 {{{ -100, -549, -549, -549, -619},
5791 { -100, -549, -549, -549, -619},
5792 { -100, -549, -549, -549, -619},
5793 { -100, -549, -549, -549, -619},
5794 { -100, -549, -549, -549, -619}},
5795 /* GC.UA.. @ */
5796 {{ -100, -549, -549, -549, -619},
5797 { -100, -549, -549, -549, -619},
5798 { -100, -549, -549, -549, -619},
5799 { -100, -549, -549, -549, -619},
5800 { -100, -549, -549, -549, -619}},
5801 /* GC.UC.. @ */
5802 {{ -100, -549, -549, -549, -619},
5803 { -100, -549, -549, -549, -619},
5804 { -100, -549, -549, -549, -619},
5805 { -100, -549, -549, -549, -619},
5806 { -100, -549, -549, -549, -619}},
5807 /* GC.UG.. @ */
5808 {{ -100, -549, -549, -549, -619},
5809 { -100, -549, -549, -549, -619},
5810 { -100, -549, -549, -549, -619},
5811 { -100, -549, -549, -549, -619},
5812 { -100, -549, -549, -549, -619}},
5813 /* GC.UU.. @ */
5814 {{ -100, -549, -549, -549, -619},
5815 { -100, -549, -549, -549, -619},
5816 { -100, -549, -549, -549, -619},
5817 { -100, -549, -549, -549, -619},
5818 { -100, -549, -549, -549, -619}}}},
5819 /* noPair */ {{{{0}}}},
5820 /* GU.@@.CG */
5821 {{{{ 0, 0, 0, 0, 0 },
5822 { DEF, DEF, DEF, DEF, DEF },
5823 { DEF, DEF, DEF, DEF, DEF },
5824 { DEF, DEF, DEF, DEF, DEF },
5825 { DEF, DEF, DEF, DEF, DEF }},
5826 /* GU.@A..CG */
5827 {{ 0, 0, 0, 0, 0 },
5828 {-1029,-1029,-1029,-1029,-1029},
5829 { -519, -519, -519, -519, -519 },
5830 { -939, -939, -939, -939, -939 },
5831 { -809, -809, -809, -809, -809 } },
5832 /* GU.@C..CG */
5833 {{ 0, 0, 0, 0, 0 },
5834 { -949, -949, -949, -949, -949 },
5835 { -449, -449, -449, -449, -449 },
5836 { -939, -939, -939, -939, -939 },
5837 { -739, -739, -739, -739, -739 } },
5838 /* GU.@G..CG */
5839 {{ 0, 0, 0, 0, 0 },
5840 {-1029,-1029,-1029,-1029,-1029},
5841 { -519, -519, -519, -519, -519 },
5842 { -939, -939, -939, -939, -939 },
5843 { -809, -809, -809, -809, -809 } },
5844 /* GU.@U..CG */
5845 {{ 0, 0, 0, 0, 0 },
5846 {-1029,-1029,-1029,-1029,-1029},
5847 { -669, -669, -669, -669, -669 },
5848 { -939, -939, -939, -939, -939 },
5849 { -859, -859, -859, -859, -859 } },
5850 /* GU.A@..CG */
5851 {{{ DEF, -429, -599, -599, -599 },
5852 { -100, -479, -649, -649, -649 },
5853 { -100, -479, -649, -649, -649 },
5854 { -100, -479, -649, -649, -649 },
5855 { -100, -479, -649, -649, -649 } },

```

```

5856 /* GU.AA..CG */
5857 {{ DEF, -429, -599, -599, -599,
5858 {-1079,-1458,-1628,-1628,-1628},
5859 { -569, -948,-1118,-1118,-1118},
5860 { -989,-1368,-1538,-1538,-1538},
5861 { -859,-1238,-1408,-1408,-1408}},
5862 /* GU.AC..CG */
5863 {{ DEF, -429, -599, -599, -599},
5864 { -999,-1378,-1548,-1548,-1548},
5865 { -499, -878,-1048,-1048,-1048},
5866 { -989,-1368,-1538,-1538,-1538},
5867 { -789,-1168,-1338,-1338,-1338}},
5868 /* GU.AG..CG */
5869 {{ DEF, -429, -599, -599, -599},
5870 {-1079,-1458,-1628,-1628,-1628},
5871 { -569, -948,-1118,-1118,-1118},
5872 { -989,-1368,-1538,-1538,-1538},
5873 { -859,-1238,-1408,-1408,-1408}},
5874 /* GU.AU..CG */
5875 {{ DEF, -429, -599, -599, -599},
5876 {-1079,-1458,-1628,-1628,-1628},
5877 { -719,-1098,-1268,-1268,-1268},
5878 { -989,-1368,-1538,-1538,-1538},
5879 { -909,-1288,-1458,-1458,-1458}},
5880 /* GU.C@..CG */
5881 {{{ DEF, -259, -239, -239, -239},
5882 { -100, -309, -289, -289, -289},
5883 { -100, -309, -289, -289, -289},
5884 { -100, -309, -289, -289, -289},
5885 { -100, -309, -289, -289, -289}},
5886 /* GU.CA..CG */
5887 {{ DEF, -259, -239, -239, -239},
5888 {-1079,-1288,-1268,-1268,-1268},
5889 { -569, -778, -758, -758, -758},
5890 { -989,-1198,-1178,-1178,-1178},
5891 { -859,-1068,-1048,-1048,-1048}},
5892 /* GU.CC..CG */
5893 {{ DEF, -259, -239, -239, -239},
5894 { -999,-1208,-1188,-1188,-1188},
5895 { -499, -708, -688, -688, -688},
5896 { -989,-1198,-1178,-1178,-1178},
5897 { -789, -998, -978, -978, -978}},
5898 /* GU.CG..CG */
5899 {{ DEF, -259, -239, -239, -239},
5900 {-1079,-1288,-1268,-1268,-1268},
5901 { -569, -778, -758, -758, -758},
5902 { -989,-1198,-1178,-1178,-1178},
5903 { -859,-1068,-1048,-1048,-1048}},
5904 /* GU.CU..CG */
5905 {{ DEF, -259, -239, -239, -239},
5906 {-1079,-1288,-1268,-1268,-1268},
5907 { -719, -928, -908, -908, -908},
5908 { -989,-1198,-1178,-1178,-1178},
5909 { -909,-1118,-1098,-1098,-1098}},
5910 /* GU.G@..CG */
5911 {{{ DEF, -339, -689, -689, -689},
5912 { -100, -389, -739, -739, -739},
5913 { -100, -389, -739, -739, -739},
5914 { -100, -389, -739, -739, -739},
5915 { -100, -389, -739, -739, -739}},
5916 /* GU.GA..CG */
5917 {{ DEF, -339, -689, -689, -689},
5918 {-1079,-1368,-1718,-1718,-1718},
5919 { -569, -858,-1208,-1208,-1208},
5920 { -989,-1278,-1628,-1628,-1628},
5921 { -859,-1148,-1498,-1498,-1498}},
5922 /* GU.GC..CG */
5923 {{ DEF, -339, -689, -689, -689},
5924 { -999,-1288,-1638,-1638,-1638},
5925 { -499, -788,-1138,-1138,-1138},
5926 { -989,-1278,-1628,-1628,-1628},
5927 { -789,-1078,-1428,-1428,-1428}},
5928 /* GU.GG..CG */
5929 {{ DEF, -339, -689, -689, -689},
5930 {-1079,-1368,-1718,-1718,-1718},
5931 { -569, -858,-1208,-1208,-1208},
5932 { -989,-1278,-1628,-1628,-1628},
5933 { -859,-1148,-1498,-1498,-1498}},
5934 /* GU.GU..CG */
5935 {{ DEF, -339, -689, -689, -689},
5936 {-1079,-1368,-1718,-1718,-1718},
5937 { -719,-1008,-1358,-1358,-1358},
5938 { -989,-1278,-1628,-1628,-1628},
5939 { -909,-1198,-1548,-1548,-1548}},
5940 /* GU.U@..CG */
5941 {{{ DEF, -329, -329, -329, -329},
5942 { -100, -379, -379, -379, -379},

```

```

5943 { -100, -379, -379, -379, -379},
5944 { -100, -379, -379, -379, -379},
5945 { -100, -379, -379, -379, -379}},
5946 /* GU.UA..CG */
5947 {{ DEF, -329, -329, -329, -329},
5948 {-1079,-1358,-1358,-1358,-1358},
5949 { -569, -848, -848, -848, -848},
5950 { -989,-1268,-1268,-1268,-1268},
5951 { -859,-1138,-1138,-1138,-1138}},
5952 /* GU.UC..CG */
5953 {{ DEF, -329, -329, -329, -329},
5954 { -999,-1278,-1278,-1278,-1278},
5955 { -499, -778, -778, -778, -778},
5956 { -989,-1268,-1268,-1268,-1268},
5957 { -789,-1068,-1068,-1068,-1068}},
5958 /* GU.UG..CG */
5959 {{ DEF, -329, -329, -329, -329},
5960 {-1079,-1358,-1358,-1358,-1358},
5961 { -569, -848, -848, -848, -848},
5962 { -989,-1268,-1268,-1268,-1268},
5963 { -859,-1138,-1138,-1138,-1138}},
5964 /* GU.UU..CG */
5965 {{ DEF, -329, -329, -329, -329},
5966 {-1079,-1358,-1358,-1358,-1358},
5967 { -719, -998, -998, -998, -998},
5968 { -989,-1268,-1268,-1268,-1268},
5969 { -909,-1188,-1188,-1188,-1188}}}},
5970 /* GU.@@..GC */
5971 {{{ 0, 0, 0, 0, 0},
5972 { DEF, DEF, DEF, DEF, DEF},
5973 { DEF, DEF, DEF, DEF, DEF},
5974 { DEF, DEF, DEF, DEF, DEF},
5975 { DEF, DEF, DEF, DEF, DEF}},
5976 /* GU.@A..GC */
5977 {{ 0, 0, 0, 0, 0},
5978 { -519, -519, -519, -519, -519},
5979 { -719, -719, -719, -719, -719},
5980 { -709, -709, -709, -709, -709},
5981 { -499, -499, -499, -499, -499}},
5982 /* GU.@C..GC */
5983 {{ 0, 0, 0, 0, 0},
5984 { -879, -879, -879, -879, -879},
5985 { -309, -309, -309, -309, -309},
5986 { -739, -739, -739, -739, -739},
5987 { -499, -499, -499, -499, -499}},
5988 /* GU.@G..GC */
5989 {{ 0, 0, 0, 0, 0},
5990 { -559, -559, -559, -559, -559},
5991 { -309, -309, -309, -309, -309},
5992 { -619, -619, -619, -619, -619},
5993 { -499, -499, -499, -499, -499}},
5994 /* GU.@U..GC */
5995 {{ 0, 0, 0, 0, 0},
5996 { -879, -879, -879, -879, -879},
5997 { -389, -389, -389, -389, -389},
5998 { -739, -739, -739, -739, -739},
5999 { -569, -569, -569, -569, -569}}}},
6000 /* GU.A@..GC */
6001 {{{ DEF, -429, -599, -599, -599},
6002 { -100, -479, -649, -649, -649},
6003 { -100, -479, -649, -649, -649},
6004 { -100, -479, -649, -649, -649},
6005 { -100, -479, -649, -649, -649}},
6006 /* GU.AA..GC */
6007 {{ DEF, -429, -599, -599, -599},
6008 { -569, -948,-1118,-1118,-1118},
6009 { -769,-1148,-1318,-1318,-1318},
6010 { -759,-1138,-1308,-1308,-1308},
6011 { -549, -928,-1098,-1098,-1098}},
6012 /* GU.AC..GC */
6013 {{ DEF, -429, -599, -599, -599},
6014 { -929,-1308,-1478,-1478,-1478},
6015 { -359, -738, -908, -908, -908},
6016 { -789,-1168,-1338,-1338,-1338},
6017 { -549, -928,-1098,-1098,-1098}},
6018 /* GU.AG..GC */
6019 {{ DEF, -429, -599, -599, -599},
6020 { -609, -988,-1158,-1158,-1158},
6021 { -359, -738, -908, -908, -908},
6022 { -669,-1048,-1218,-1218,-1218},
6023 { -549, -928,-1098,-1098,-1098}},
6024 /* GU.AU..GC */
6025 {{ DEF, -429, -599, -599, -599},
6026 { -929,-1308,-1478,-1478,-1478},
6027 { -439, -818, -988, -988, -988},
6028 { -789,-1168,-1338,-1338,-1338},
6029 { -619, -998,-1168,-1168,-1168}}}},

```

```
6030 /* GU.C@..GC */
6031 {{ DEF, -259, -239, -239, -239},
6032 { -100, -309, -289, -289, -289},
6033 { -100, -309, -289, -289, -289},
6034 { -100, -309, -289, -289, -289},
6035 { -100, -309, -289, -289, -289}},
6036 /* GU.CA..GC */
6037 {{ DEF, -259, -239, -239, -239},
6038 { -569, -778, -758, -758, -758},
6039 { -769, -978, -958, -958, -958},
6040 { -759, -968, -948, -948, -948},
6041 { -549, -758, -738, -738, -738}},
6042 /* GU.CC..GC */
6043 {{ DEF, -259, -239, -239, -239},
6044 { -929,-1138,-1118,-1118,-1118},
6045 { -359, -568, -548, -548, -548},
6046 { -789, -998, -978, -978, -978},
6047 { -549, -758, -738, -738, -738}},
6048 /* GU.CG..GC */
6049 {{ DEF, -259, -239, -239, -239},
6050 { -609, -818, -798, -798, -798},
6051 { -359, -568, -548, -548, -548},
6052 { -669, -878, -858, -858, -858},
6053 { -549, -758, -738, -738, -738}},
6054 /* GU.CU..GC */
6055 {{ DEF, -259, -239, -239, -239},
6056 { -929,-1138,-1118,-1118,-1118},
6057 { -439, -648, -628, -628, -628},
6058 { -789, -998, -978, -978, -978},
6059 { -619, -828, -808, -808, -808}},
6060 /* GU.G@..GC */
6061 {{ DEF, -339, -689, -689, -689},
6062 { -100, -389, -739, -739, -739},
6063 { -100, -389, -739, -739, -739},
6064 { -100, -389, -739, -739, -739},
6065 { -100, -389, -739, -739, -739}},
6066 /* GU.GA..GC */
6067 {{ DEF, -339, -689, -689, -689},
6068 { -569, -858,-1208,-1208,-1208},
6069 { -769,-1058,-1408,-1408,-1408},
6070 { -759,-1048,-1398,-1398,-1398},
6071 { -549, -838,-1188,-1188,-1188}},
6072 /* GU.GC..GC */
6073 {{ DEF, -339, -689, -689, -689},
6074 { -929,-1218,-1568,-1568,-1568},
6075 { -359, -648, -998, -998, -998},
6076 { -789,-1078,-1428,-1428,-1428},
6077 { -549, -838,-1188,-1188,-1188}},
6078 /* GU.GG..GC */
6079 {{ DEF, -339, -689, -689, -689},
6080 { -609, -898,-1248,-1248,-1248},
6081 { -359, -648, -998, -998, -998},
6082 { -669, -958,-1308,-1308,-1308},
6083 { -549, -838,-1188,-1188,-1188}},
6084 /* GU.GU..GC */
6085 {{ DEF, -339, -689, -689, -689},
6086 { -929,-1218,-1568,-1568,-1568},
6087 { -439, -728,-1078,-1078,-1078},
6088 { -789,-1078,-1428,-1428,-1428},
6089 { -619, -908,-1258,-1258,-1258}},
6090 /* GU.U@..GC */
6091 {{ DEF, -329, -329, -329, -329},
6092 { -100, -379, -379, -379, -379},
6093 { -100, -379, -379, -379, -379},
6094 { -100, -379, -379, -379, -379},
6095 { -100, -379, -379, -379, -379}},
6096 /* GU.UA..GC */
6097 {{ DEF, -329, -329, -329, -329},
6098 { -569, -848, -848, -848, -848},
6099 { -769,-1048,-1048,-1048,-1048},
6100 { -759,-1038,-1038,-1038,-1038},
6101 { -549, -828, -828, -828, -828}},
6102 /* GU.UC..GC */
6103 {{ DEF, -329, -329, -329, -329},
6104 { -929,-1208,-1208,-1208,-1208},
6105 { -359, -638, -638, -638, -638},
6106 { -789,-1068,-1068,-1068,-1068},
6107 { -549, -828, -828, -828, -828}},
6108 /* GU.UG..GC */
6109 {{ DEF, -329, -329, -329, -329},
6110 { -609, -888, -888, -888, -888},
6111 { -359, -638, -638, -638, -638},
6112 { -669, -948, -948, -948, -948},
6113 { -549, -828, -828, -828, -828}},
6114 /* GU.UU..GC */
6115 {{ DEF, -329, -329, -329, -329},
6116 { -929,-1208,-1208,-1208,-1208},
```

```

6117 { -439, -718, -718, -718, -718},
6118 { -789,-1068,-1068,-1068,-1068},},
6119 { -619, -898, -898, -898, -898}}}},,
6120 /* GU.C@..GU */
6121 {{{ 0, 0, 0, 0, 0},
6122 { DEF, DEF, DEF, DEF, DEF},
6123 { DEF, DEF, DEF, DEF, DEF},
6124 { DEF, DEF, DEF, DEF, DEF},
6125 { DEF, DEF, DEF, DEF, DEF}},,
6126 /* GU.@A..GU */
6127 {{ 0, 0, 0, 0, 0},
6128 { -429, -429, -429, -429, -429},
6129 { -259, -259, -259, -259, -259},
6130 { -339, -339, -339, -339, -339},
6131 { -329, -329, -329, -329, -329}},,
6132 /* GU.@C..GU */
6133 {{ 0, 0, 0, 0, 0},
6134 { -599, -599, -599, -599, -599},
6135 { -239, -239, -239, -239, -239},
6136 { -689, -689, -689, -689, -689},
6137 { -329, -329, -329, -329, -329}},,
6138 /* GU.@G..GU */
6139 {{ 0, 0, 0, 0, 0},
6140 { -599, -599, -599, -599, -599},
6141 { -239, -239, -239, -239, -239},
6142 { -689, -689, -689, -689, -689},
6143 { -329, -329, -329, -329, -329}},,
6144 /* GU.@U..GU */
6145 {{ 0, 0, 0, 0, 0},
6146 { -599, -599, -599, -599, -599},
6147 { -239, -239, -239, -239, -239},
6148 { -689, -689, -689, -689, -689},
6149 { -329, -329, -329, -329, -329}},,
6150 /* GU.A@..GU */
6151 {{{ DEF, -429, -599, -599, -599},
6152 { -100, -479, -649, -649, -649},
6153 { -100, -479, -649, -649, -649},
6154 { -100, -479, -649, -649, -649},
6155 { -100, -479, -649, -649, -649}},,
6156 /* GU.AA..GU */
6157 {{ DEF, -429, -599, -599, -599},
6158 { -479, -858,-1028,-1028,-1028},
6159 { -309, -688, -858, -858, -858},
6160 { -389, -768, -938, -938, -938},
6161 { -379, -758, -928, -928, -928}},,
6162 /* GU.AC..GU */
6163 {{ DEF, -429, -599, -599, -599},
6164 { -649,-1028,-1198,-1198,-1198},
6165 { -289, -668, -838, -838, -838},
6166 { -739,-1118,-1288,-1288,-1288},
6167 { -379, -758, -928, -928, -928}},,
6168 /* GU.AG..GU */
6169 {{ DEF, -429, -599, -599, -599},
6170 { -649,-1028,-1198,-1198,-1198},
6171 { -289, -668, -838, -838, -838},
6172 { -739,-1118,-1288,-1288,-1288},
6173 { -379, -758, -928, -928, -928}},,
6174 /* GU.AU..GU */
6175 {{ DEF, -429, -599, -599, -599},
6176 { -649,-1028,-1198,-1198,-1198},
6177 { -289, -668, -838, -838, -838},
6178 { -739,-1118,-1288,-1288,-1288},
6179 { -379, -758, -928, -928, -928}},,
6180 /* GU.C@..GU */
6181 {{ DEF, -259, -239, -239, -239},
6182 { -100, -309, -289, -289, -289},
6183 { -100, -309, -289, -289, -289},
6184 { -100, -309, -289, -289, -289},
6185 { -100, -309, -289, -289, -289}},,
6186 /* GU.CA..GU */
6187 {{ DEF, -259, -239, -239, -239},
6188 { -479, -688, -668, -668, -668},
6189 { -309, -518, -498, -498, -498},
6190 { -389, -598, -578, -578, -578},
6191 { -379, -588, -568, -568, -568}},,
6192 /* GU.CC..GU */
6193 {{ DEF, -259, -239, -239, -239},
6194 { -649, -858, -838, -838, -838},
6195 { -289, -498, -478, -478, -478},
6196 { -739, -948, -928, -928, -928},
6197 { -379, -588, -568, -568, -568}},,
6198 /* GU.CG..GU */
6199 {{ DEF, -259, -239, -239, -239},
6200 { -649, -858, -838, -838, -838},
6201 { -289, -498, -478, -478, -478},
6202 { -739, -948, -928, -928, -928},
6203 { -379, -588, -568, -568, -568}},,

```

```

6204 /* GU.CU..GU */
6205 {{ DEF, -259, -239, -239, -239},
6206 { -649, -858, -838, -838, -838},
6207 { -289, -498, -478, -478, -478},
6208 { -739, -948, -928, -928, -928},
6209 { -379, -588, -568, -568, -568}}}},
6210 /* GU.G@..GU */
6211 {{{ DEF, -339, -689, -689, -689},
6212 { -100, -389, -739, -739, -739},
6213 { -100, -389, -739, -739, -739},
6214 { -100, -389, -739, -739, -739},
6215 { -100, -389, -739, -739, -739}}},
6216 /* GU.GA..GU */
6217 {{ DEF, -339, -689, -689, -689},
6218 { -479, -768, -1118, -1118, -1118},
6219 { -309, -598, -948, -948, -948},
6220 { -389, -678, -1028, -1028, -1028},
6221 { -379, -668, -1018, -1018, -1018}}},
6222 /* GU.GC..GU */
6223 {{ DEF, -339, -689, -689, -689},
6224 { -649, -938, -1288, -1288, -1288},
6225 { -289, -578, -928, -928, -928},
6226 { -739, -1028, -1378, -1378, -1378},
6227 { -379, -668, -1018, -1018, -1018}}},
6228 /* GU.GG..GU */
6229 {{ DEF, -339, -689, -689, -689},
6230 { -649, -938, -1288, -1288, -1288},
6231 { -289, -578, -928, -928, -928},
6232 { -739, -1028, -1378, -1378, -1378},
6233 { -379, -668, -1018, -1018, -1018}}},
6234 /* GU.GU..GU */
6235 {{ DEF, -339, -689, -689, -689},
6236 { -649, -938, -1288, -1288, -1288},
6237 { -289, -578, -928, -928, -928},
6238 { -739, -1028, -1378, -1378, -1378},
6239 { -379, -668, -1018, -1018, -1018}}},
6240 /* GU.U@..GU */
6241 {{{ DEF, -329, -329, -329, -329},
6242 { -100, -379, -379, -379, -379},
6243 { -100, -379, -379, -379, -379},
6244 { -100, -379, -379, -379, -379},
6245 { -100, -379, -379, -379, -379}}},
6246 /* GU.UA..GU */
6247 {{ DEF, -329, -329, -329, -329},
6248 { -479, -758, -758, -758, -758},
6249 { -309, -588, -588, -588, -588},
6250 { -389, -668, -668, -668, -668},
6251 { -379, -658, -658, -658, -658}}},
6252 /* GU.UC..GU */
6253 {{ DEF, -329, -329, -329, -329},
6254 { -649, -928, -928, -928, -928},
6255 { -289, -568, -568, -568, -568},
6256 { -739, -1018, -1018, -1018, -1018},
6257 { -379, -658, -658, -658, -658}}},
6258 /* GU.UG..GU */
6259 {{ DEF, -329, -329, -329, -329},
6260 { -649, -928, -928, -928, -928},
6261 { -289, -568, -568, -568, -568},
6262 { -739, -1018, -1018, -1018, -1018},
6263 { -379, -658, -658, -658, -658}}},
6264 /* GU.UU..GU */
6265 {{ DEF, -329, -329, -329, -329},
6266 { -649, -928, -928, -928, -928},
6267 { -289, -568, -568, -568, -568},
6268 { -739, -1018, -1018, -1018, -1018},
6269 { -379, -658, -658, -658, -658}}},
6270 /* GU.@@..UG */
6271 {{{{ 0, 0, 0, 0, 0},
6272 { DEF, DEF, DEF, DEF, DEF},
6273 { DEF, DEF, DEF, DEF, DEF},
6274 { DEF, DEF, DEF, DEF, DEF},
6275 { DEF, DEF, DEF, DEF, DEF}}},
6276 /* GU.@A..UG */
6277 {{ 0, 0, 0, 0, 0},
6278 { -719, -719, -719, -719, -719},
6279 { -479, -479, -479, -479, -479},
6280 { -659, -659, -659, -659, -659},
6281 { -549, -549, -549, -549, -549}}},
6282 /* GU.@C..UG */
6283 {{ 0, 0, 0, 0, 0},
6284 { -789, -789, -789, -789, -789},
6285 { -479, -479, -479, -479, -479},
6286 { -809, -809, -809, -809, -809},
6287 { -439, -439, -439, -439, -439}}},
6288 /* GU.@G..UG */
6289 {{ 0, 0, 0, 0, 0},
6290 { -959, -959, -959, -959, -959}},

```

```

6291 { -359, -359, -359, -359, -359},
6292 { -919, -919, -919, -919, -919},
6293 { -549, -549, -549, -549, -549}},
6294 /* GU.@U..UG */
6295 {{ 0, 0, 0, 0, 0},
6296 { -809, -809, -809, -809, -809},
6297 { -479, -479, -479, -479, -479},
6298 { -809, -809, -809, -809, -809},
6299 { -359, -359, -359, -359, -359}}}},
6300 /* GU.A@..UG */
6301 {{ DEF, -429, -599, -599, -599},
6302 { -100, -479, -649, -649, -649},
6303 { -100, -479, -649, -649, -649},
6304 { -100, -479, -649, -649, -649},
6305 { -100, -479, -649, -649, -649}}},
6306 /* GU.AA..UG */
6307 {{ DEF, -429, -599, -599, -599},
6308 { -769, -1148, -1318, -1318, -1318},
6309 { -529, -908, -1078, -1078, -1078},
6310 { -709, -1088, -1258, -1258, -1258},
6311 { -599, -978, -1148, -1148, -1148}}},
6312 /* GU.AC..UG */
6313 {{ DEF, -429, -599, -599, -599},
6314 { -839, -1218, -1388, -1388, -1388},
6315 { -529, -908, -1078, -1078, -1078},
6316 { -859, -1238, -1408, -1408, -1408},
6317 { -489, -868, -1038, -1038, -1038}}},
6318 /* GU.AG..UG */
6319 {{ DEF, -429, -599, -599, -599},
6320 { -1009, -1388, -1558, -1558, -1558},
6321 { -409, -788, -958, -958, -958},
6322 { -969, -1348, -1518, -1518, -1518},
6323 { -599, -978, -1148, -1148, -1148}}},
6324 /* GU.AU..UG */
6325 {{ DEF, -429, -599, -599, -599},
6326 { -859, -1238, -1408, -1408, -1408},
6327 { -529, -908, -1078, -1078, -1078},
6328 { -859, -1238, -1408, -1408, -1408},
6329 { -409, -788, -958, -958, -958}}},
6330 /* GU.C@..UG */
6331 {{ DEF, -259, -239, -239, -239},
6332 { -100, -309, -289, -289, -289},
6333 { -100, -309, -289, -289, -289},
6334 { -100, -309, -289, -289, -289},
6335 { -100, -309, -289, -289, -289}}},
6336 /* GU.CA..UG */
6337 {{ DEF, -259, -239, -239, -239},
6338 { -769, -978, -958, -958, -958},
6339 { -529, -738, -718, -718, -718},
6340 { -709, -918, -898, -898, -898},
6341 { -599, -808, -788, -788, -788}}},
6342 /* GU.CC..UG */
6343 {{ DEF, -259, -239, -239, -239},
6344 { -839, -1048, -1028, -1028, -1028},
6345 { -529, -738, -718, -718, -718},
6346 { -859, -1068, -1048, -1048, -1048},
6347 { -489, -698, -678, -678, -678}}},
6348 /* GU.CG..UG */
6349 {{ DEF, -259, -239, -239, -239},
6350 { -1009, -1218, -1198, -1198, -1198},
6351 { -409, -618, -598, -598, -598},
6352 { -969, -1178, -1158, -1158, -1158},
6353 { -599, -808, -788, -788, -788}}},
6354 /* GU.CU..UG */
6355 {{ DEF, -259, -239, -239, -239},
6356 { -859, -1068, -1048, -1048, -1048},
6357 { -529, -738, -718, -718, -718},
6358 { -859, -1068, -1048, -1048, -1048},
6359 { -409, -618, -598, -598, -598}}},
6360 /* GU.G@..UG */
6361 {{ DEF, -339, -689, -689, -689},
6362 { -100, -389, -739, -739, -739},
6363 { -100, -389, -739, -739, -739},
6364 { -100, -389, -739, -739, -739},
6365 { -100, -389, -739, -739, -739}}},
6366 /* GU.GA..UG */
6367 {{ DEF, -339, -689, -689, -689},
6368 { -769, -1058, -1408, -1408, -1408},
6369 { -529, -818, -1168, -1168, -1168},
6370 { -709, -998, -1348, -1348, -1348},
6371 { -599, -888, -1238, -1238, -1238}}},
6372 /* GU.GC..UG */
6373 {{ DEF, -339, -689, -689, -689},
6374 { -839, -1128, -1478, -1478, -1478},
6375 { -529, -818, -1168, -1168, -1168},
6376 { -859, -1148, -1498, -1498, -1498},
6377 { -489, -778, -1128, -1128, -1128}}},

```

```

6378 /* GU.GG..UG */
6379 {{ DEF, -339, -689, -689, -689},
6380 {-1009,-1298,-1648,-1648,-1648},
6381 { -409, -698,-1048,-1048,-1048},
6382 { -969,-1258,-1608,-1608,-1608},
6383 { -599, -888,-1238,-1238,-1238}},
6384 /* GU.GU..UG */
6385 {{ DEF, -339, -689, -689, -689},
6386 { -859,-1148,-1498,-1498,-1498},
6387 { -529, -818,-1168,-1168,-1168},
6388 { -859,-1148,-1498,-1498,-1498},
6389 { -409, -698,-1048,-1048,-1048}},
6390 /* GU.U@..UG */
6391 {{{ DEF, -329, -329, -329, -329},
6392 { -100, -379, -379, -379, -379},
6393 { -100, -379, -379, -379, -379},
6394 { -100, -379, -379, -379, -379},
6395 { -100, -379, -379, -379, -379}},
6396 /* GU.UA..UG */
6397 {{ DEF, -329, -329, -329, -329},
6398 { -769,-1048,-1048,-1048,-1048},
6399 { -529, -808, -808, -808, -808},
6400 { -709, -988, -988, -988, -988},
6401 { -599, -878, -878, -878, -878}},
6402 /* GU.UC..UG */
6403 {{ DEF, -329, -329, -329, -329},
6404 { -839,-1118,-1118,-1118,-1118},
6405 { -529, -808, -808, -808, -808},
6406 { -859,-1138,-1138,-1138,-1138},
6407 { -489, -768, -768, -768, -768}},
6408 /* GU.UG..UG */
6409 {{ DEF, -329, -329, -329, -329},
6410 {-1009,-1288,-1288,-1288,-1288},
6411 { -409, -688, -688, -688, -688},
6412 { -969,-1248,-1248,-1248,-1248},
6413 { -599, -878, -878, -878, -878}},
6414 /* GU.UU..UG */
6415 {{ DEF, -329, -329, -329, -329},
6416 { -859,-1138,-1138,-1138,-1138},
6417 { -529, -808, -808, -808, -808},
6418 { -859,-1138,-1138,-1138,-1138},
6419 { -409, -688, -688, -688, -688}}},
6420 /* GU.@@..AU */
6421 {{{{ 0, 0, 0, 0, 0},
6422 { DEF, DEF, DEF, DEF, DEF},
6423 { DEF, DEF, DEF, DEF, DEF},
6424 { DEF, DEF, DEF, DEF, DEF},
6425 { DEF, DEF, DEF, DEF, DEF}},
6426 /* GU.@A..AU */
6427 {{ 0, 0, 0, 0, 0},
6428 { -429, -429, -429, -429, -429},
6429 { -259, -259, -259, -259, -259},
6430 { -339, -339, -339, -339, -339},
6431 { -329, -329, -329, -329, -329}},
6432 /* GU.@C..AU */
6433 {{ 0, 0, 0, 0, 0},
6434 { -599, -599, -599, -599, -599},
6435 { -239, -239, -239, -239, -239},
6436 { -689, -689, -689, -689, -689},
6437 { -329, -329, -329, -329, -329}},
6438 /* GU.@G..AU */
6439 {{ 0, 0, 0, 0, 0},
6440 { -599, -599, -599, -599, -599},
6441 { -239, -239, -239, -239, -239},
6442 { -689, -689, -689, -689, -689},
6443 { -329, -329, -329, -329, -329}},
6444 /* GU.@U..AU */
6445 {{ 0, 0, 0, 0, 0},
6446 { -599, -599, -599, -599, -599},
6447 { -239, -239, -239, -239, -239},
6448 { -689, -689, -689, -689, -689},
6449 { -329, -329, -329, -329, -329}},
6450 /* GU.A@..AU */
6451 {{{ DEF, -429, -599, -599, -599},
6452 { -100, -479, -649, -649, -649},
6453 { -100, -479, -649, -649, -649},
6454 { -100, -479, -649, -649, -649},
6455 { -100, -479, -649, -649, -649}},
6456 /* GU.AA..AU */
6457 {{ DEF, -429, -599, -599, -599},
6458 { -479, -858,-1028,-1028,-1028},
6459 { -309, -688, -858, -858, -858},
6460 { -389, -768, -938, -938, -938},
6461 { -379, -758, -928, -928, -928}},
6462 /* GU.AC..AU */
6463 {{ DEF, -429, -599, -599, -599},
6464 { -649,-1028,-1198,-1198,-1198},

```

```

6465 { -289, -668, -838, -838, -838},
6466 { -739,-1118,-1288,-1288,-1288},
6467 { -379, -758, -928, -928, -928}},
6468 /* GU.AG..AU */
6469 {{ DEF, -429, -599, -599, -599},
6470 { -649,-1028,-1198,-1198,-1198},
6471 { -289, -668, -838, -838, -838},
6472 { -739,-1118,-1288,-1288,-1288},
6473 { -379, -758, -928, -928, -928}},
6474 /* GU.AU..AU */
6475 {{ DEF, -429, -599, -599, -599},
6476 { -649,-1028,-1198,-1198,-1198},
6477 { -289, -668, -838, -838, -838},
6478 { -739,-1118,-1288,-1288,-1288},
6479 { -379, -758, -928, -928, -928}}},
6480 /* GU.C@..AU */
6481 {{{ DEF, -259, -239, -239, -239},
6482 { -100, -309, -289, -289, -289},
6483 { -100, -309, -289, -289, -289},
6484 { -100, -309, -289, -289, -289},
6485 { -100, -309, -289, -289, -289}},
6486 /* GU.CA..AU */
6487 {{ DEF, -259, -239, -239, -239},
6488 { -479, -688, -668, -668, -668},
6489 { -309, -518, -498, -498, -498},
6490 { -389, -598, -578, -578, -578},
6491 { -379, -588, -568, -568, -568}},
6492 /* GU.CC..AU */
6493 {{ DEF, -259, -239, -239, -239},
6494 { -649, -858, -838, -838, -838},
6495 { -289, -498, -478, -478, -478},
6496 { -739, -948, -928, -928, -928},
6497 { -379, -588, -568, -568, -568}},
6498 /* GU.CG..AU */
6499 {{ DEF, -259, -239, -239, -239},
6500 { -649, -858, -838, -838, -838},
6501 { -289, -498, -478, -478, -478},
6502 { -739, -948, -928, -928, -928},
6503 { -379, -588, -568, -568, -568}},
6504 /* GU.CU..AU */
6505 {{ DEF, -259, -239, -239, -239},
6506 { -649, -858, -838, -838, -838},
6507 { -289, -498, -478, -478, -478},
6508 { -739, -948, -928, -928, -928},
6509 { -379, -588, -568, -568, -568}},
6510 /* GU.G@..AU */
6511 {{{ DEF, -339, -689, -689, -689},
6512 { -100, -389, -739, -739, -739},
6513 { -100, -389, -739, -739, -739},
6514 { -100, -389, -739, -739, -739},
6515 { -100, -389, -739, -739, -739}},
6516 /* GU.GA..AU */
6517 {{ DEF, -339, -689, -689, -689},
6518 { -479, -768,-1118,-1118,-1118},
6519 { -309, -598, -948, -948, -948},
6520 { -389, -678,-1028,-1028,-1028},
6521 { -379, -668,-1018,-1018,-1018}},
6522 /* GU.GC..AU */
6523 {{ DEF, -339, -689, -689, -689},
6524 { -649, -938,-1288,-1288,-1288},
6525 { -289, -578, -928, -928, -928},
6526 { -739,-1028,-1378,-1378,-1378},
6527 { -379, -668,-1018,-1018,-1018}},
6528 /* GU.GG..AU */
6529 {{ DEF, -339, -689, -689, -689},
6530 { -649, -938,-1288,-1288,-1288},
6531 { -289, -578, -928, -928, -928},
6532 { -739,-1028,-1378,-1378,-1378},
6533 { -379, -668,-1018,-1018,-1018}},
6534 /* GU.GU..AU */
6535 {{ DEF, -339, -689, -689, -689},
6536 { -649, -938,-1288,-1288,-1288},
6537 { -289, -578, -928, -928, -928},
6538 { -739,-1028,-1378,-1378,-1378},
6539 { -379, -668,-1018,-1018,-1018}},
6540 /* GU.U@..AU */
6541 {{{ DEF, -329, -329, -329, -329},
6542 { -100, -379, -379, -379, -379},
6543 { -100, -379, -379, -379, -379},
6544 { -100, -379, -379, -379, -379},
6545 { -100, -379, -379, -379, -379}},
6546 /* GU.UA..AU */
6547 {{ DEF, -329, -329, -329, -329},
6548 { -479, -758, -758, -758, -758},
6549 { -309, -588, -588, -588, -588},
6550 { -389, -668, -668, -668, -668},
6551 { -379, -658, -658, -658, -658}},

```

```

6552 /* GU.UC..AU */
6553 {{ DEF, -329, -329, -329, -329},
6554 { -649, -928, -928, -928, -928},
6555 { -289, -568, -568, -568, -568},
6556 { -739,-1018,-1018,-1018,-1018},
6557 { -379, -658, -658, -658, -658}},
6558 /* GU.UG..AU */
6559 {{ DEF, -329, -329, -329, -329},
6560 { -649, -928, -928, -928, -928},
6561 { -289, -568, -568, -568, -568},
6562 { -739,-1018,-1018,-1018,-1018},
6563 { -379, -658, -658, -658, -658}},
6564 /* GU.UU..AU */
6565 {{ DEF, -329, -329, -329, -329},
6566 { -649, -928, -928, -928, -928},
6567 { -289, -568, -568, -568, -568},
6568 { -739,-1018,-1018,-1018,-1018},
6569 { -379, -658, -658, -658, -658}}}},
6570 /* GU.@@.UA */
6571 {{{ 0, 0, 0, 0, 0},
6572 { DEF, DEF, DEF, DEF, DEF},
6573 { DEF, DEF, DEF, DEF, DEF},
6574 { DEF, DEF, DEF, DEF, DEF},
6575 { DEF, DEF, DEF, DEF, DEF}},
6576 /* GU.@A..UA */
6577 {{ 0, 0, 0, 0, 0},
6578 { -399, -399, -399, -399, -399},
6579 { -429, -429, -429, -429, -429},
6580 { -379, -379, -379, -379, -379},
6581 { -279, -279, -279, -279, -279}},
6582 /* GU.@C..UA */
6583 {{ 0, 0, 0, 0, 0},
6584 { -629, -629, -629, -629, -629},
6585 { -509, -509, -509, -509, -509},
6586 { -679, -679, -679, -679, -679},
6587 { -139, -139, -139, -139, -139}},
6588 /* GU.@G..UA */
6589 {{ 0, 0, 0, 0, 0},
6590 { -889, -889, -889, -889, -889},
6591 { -199, -199, -199, -199, -199},
6592 { -889, -889, -889, -889, -889},
6593 { -279, -279, -279, -279, -279}},
6594 /* GU.@U..UA */
6595 {{ 0, 0, 0, 0, 0},
6596 { -589, -589, -589, -589, -589},
6597 { -179, -179, -179, -179, -179},
6598 { -679, -679, -679, -679, -679},
6599 { -140, -140, -140, -140, -140}}},
6600 /* GU.A@..UA */
6601 {{{ DEF, -429, -599, -599, -599},
6602 { -100, -479, -649, -649, -649},
6603 { -100, -479, -649, -649, -649},
6604 { -100, -479, -649, -649, -649},
6605 { -100, -479, -649, -649, -649}},
6606 /* GU.AA..UA */
6607 {{ DEF, -429, -599, -599, -599},
6608 { -449, -828, -998, -998, -998},
6609 { -479, -858,-1028,-1028,-1028},
6610 { -429, -808, -978, -978, -978},
6611 { -329, -708, -878, -878, -878}},
6612 /* GU.AC..UA */
6613 {{ DEF, -429, -599, -599, -599},
6614 { -679,-1058,-1228,-1228,-1228},
6615 { -559, -938,-1108,-1108,-1108},
6616 { -729,-1108,-1278,-1278,-1278},
6617 { -189, -568, -738, -738, -738}},
6618 /* GU.AG..UA */
6619 {{ DEF, -429, -599, -599, -599},
6620 { -939,-1318,-1488,-1488,-1488},
6621 { -249, -628, -798, -798, -798},
6622 { -939,-1318,-1488,-1488,-1488},
6623 { -329, -708, -878, -878, -878}},
6624 /* GU.AU..UA */
6625 {{ DEF, -429, -599, -599, -599},
6626 { -639,-1018,-1188,-1188,-1188},
6627 { -229, -608, -778, -778, -778},
6628 { -729,-1108,-1278,-1278,-1278},
6629 { -190, -569, -739, -739, -739}},
6630 /* GU.C@..UA */
6631 {{{ DEF, -259, -239, -239, -239},
6632 { -100, -309, -289, -289, -289},
6633 { -100, -309, -289, -289, -289},
6634 { -100, -309, -289, -289, -289},
6635 { -100, -309, -289, -289, -289}},
6636 /* GU.CA..UA */
6637 {{ DEF, -259, -239, -239, -239},
6638 { -449, -658, -638, -638, -638}},

```

```

6639 { -479, -688, -668, -668, -668},
6640 { -429, -638, -618, -618, -618},
6641 { -329, -538, -518, -518, -518}},
6642 /* GU.CC..UA */
6643 {{ DEF, -259, -239, -239, -239},
6644 { -679, -888, -868, -868, -868},
6645 { -559, -768, -748, -748, -748},
6646 { -729, -938, -918, -918, -918},
6647 { -189, -398, -378, -378, -378}},
6648 /* GU.CG..UA */
6649 {{ DEF, -259, -239, -239, -239},
6650 { -939,-1148,-1128,-1128,-1128},
6651 { -249, -458, -438, -438, -438},
6652 { -939,-1148,-1128,-1128,-1128},
6653 { -329, -538, -518, -518, -518}},
6654 /* GU.CU..UA */
6655 {{ DEF, -259, -239, -239, -239},
6656 { -639, -848, -828, -828, -828},
6657 { -229, -438, -418, -418, -418},
6658 { -729, -938, -918, -918, -918},
6659 { -190, -399, -379, -379, -379}},
6660 /* GU.G@..UA */
6661 {{ DEF, -339, -689, -689, -689},
6662 { -100, -389, -739, -739, -739},
6663 { -100, -389, -739, -739, -739},
6664 { -100, -389, -739, -739, -739},
6665 { -100, -389, -739, -739, -739}},
6666 /* GU.GA..UA */
6667 {{ DEF, -339, -689, -689, -689},
6668 { -449, -738,-1088,-1088,-1088},
6669 { -479, -768,-1118,-1118,-1118},
6670 { -429, -718,-1068,-1068,-1068},
6671 { -329, -618, -968, -968, -968}},
6672 /* GU.GC..UA */
6673 {{ DEF, -339, -689, -689, -689},
6674 { -679, -968,-1318,-1318,-1318},
6675 { -559, -848,-1198,-1198,-1198},
6676 { -729,-1018,-1368,-1368,-1368},
6677 { -189, -478, -828, -828, -828}},
6678 /* GU.GG..UA */
6679 {{ DEF, -339, -689, -689, -689},
6680 { -939,-1228,-1578,-1578,-1578},
6681 { -249, -538, -888, -888, -888},
6682 { -939,-1228,-1578,-1578,-1578},
6683 { -329, -618, -968, -968, -968}},
6684 /* GU.GU..UA */
6685 {{ DEF, -339, -689, -689, -689},
6686 { -639, -928,-1278,-1278,-1278},
6687 { -229, -518, -868, -868, -868},
6688 { -729,-1018,-1368,-1368,-1368},
6689 { -190, -479, -829, -829, -829}},
6690 /* GU.U@..UA */
6691 {{ DEF, -329, -329, -329, -329},
6692 { -100, -379, -379, -379, -379},
6693 { -100, -379, -379, -379, -379},
6694 { -100, -379, -379, -379, -379}},
6695 { -100, -379, -379, -379, -379}},
6696 /* GU.UA..UA */
6697 {{ DEF, -329, -329, -329, -329},
6698 { -449, -728, -728, -728, -728},
6699 { -479, -758, -758, -758, -758},
6700 { -429, -708, -708, -708, -708},
6701 { -329, -608, -608, -608, -608}},
6702 /* GU.UC..UA */
6703 {{ DEF, -329, -329, -329, -329},
6704 { -679, -958, -958, -958, -958},
6705 { -559, -838, -838, -838, -838},
6706 { -729,-1008,-1008,-1008,-1008},
6707 { -189, -468, -468, -468, -468}},
6708 /* GU.UG..UA */
6709 {{ DEF, -329, -329, -329, -329},
6710 { -939,-1218,-1218,-1218,-1218},
6711 { -249, -528, -528, -528, -528},
6712 { -939,-1218,-1218,-1218,-1218},
6713 { -329, -608, -608, -608, -608}},
6714 /* GU.UU..UA */
6715 {{ DEF, -329, -329, -329, -329},
6716 { -639, -918, -918, -918, -918},
6717 { -229, -508, -508, -508, -508},
6718 { -729,-1008,-1008,-1008,-1008},
6719 { -190, -469, -469, -469, -469}}},
6720 /* GU.@@.. @ */
6721 {{{ DEF, DEF, DEF, DEF, DEF},
6722 { DEF, DEF, DEF, DEF, DEF},
6723 { DEF, DEF, DEF, DEF, DEF},
6724 { DEF, DEF, DEF, DEF, DEF},
6725 { DEF, DEF, DEF, DEF, DEF}},

```

```

6726 /* GU.@A.. @ */
6727 {{ DEF, DEF, DEF, DEF, DEF},
6728 { DEF, DEF, DEF, DEF, DEF},
6729 { DEF, DEF, DEF, DEF, DEF},
6730 { DEF, DEF, DEF, DEF, DEF},
6731 { DEF, DEF, DEF, DEF, DEF}},
6732 /* GU.@C.. @ */
6733 {{ DEF, DEF, DEF, DEF, DEF},
6734 { DEF, DEF, DEF, DEF, DEF},
6735 { DEF, DEF, DEF, DEF, DEF},
6736 { DEF, DEF, DEF, DEF, DEF},
6737 { DEF, DEF, DEF, DEF, DEF}},
6738 /* GU.@G.. @ */
6739 {{ DEF, DEF, DEF, DEF, DEF},
6740 { DEF, DEF, DEF, DEF, DEF},
6741 { DEF, DEF, DEF, DEF, DEF},
6742 { DEF, DEF, DEF, DEF, DEF},
6743 { DEF, DEF, DEF, DEF, DEF}},
6744 /* GU.@U.. @ */
6745 {{ DEF, DEF, DEF, DEF, DEF},
6746 { DEF, DEF, DEF, DEF, DEF},
6747 { DEF, DEF, DEF, DEF, DEF},
6748 { DEF, DEF, DEF, DEF, DEF},
6749 { DEF, DEF, DEF, DEF, DEF}}},
6750 /* GU.A@.. @ */
6751 {{{-100, -479, -649, -649, -649},
6752 {-100, -479, -649, -649, -649},
6753 {-100, -479, -649, -649, -649},
6754 {-100, -479, -649, -649, -649},
6755 {-100, -479, -649, -649, -649}}},
6756 /* GU.AA.. @ */
6757 {{-100, -479, -649, -649, -649},
6758 {-100, -479, -649, -649, -649},
6759 {-100, -479, -649, -649, -649},
6760 {-100, -479, -649, -649, -649},
6761 {-100, -479, -649, -649, -649}}},
6762 /* GU.AC.. @ */
6763 {{-100, -479, -649, -649, -649},
6764 {-100, -479, -649, -649, -649},
6765 {-100, -479, -649, -649, -649},
6766 {-100, -479, -649, -649, -649},
6767 {-100, -479, -649, -649, -649}}},
6768 /* GU.AG.. @ */
6769 {{-100, -479, -649, -649, -649},
6770 {-100, -479, -649, -649, -649},
6771 {-100, -479, -649, -649, -649},
6772 {-100, -479, -649, -649, -649},
6773 {-100, -479, -649, -649, -649}}},
6774 /* GU.AU.. @ */
6775 {{-100, -479, -649, -649, -649},
6776 {-100, -479, -649, -649, -649},
6777 {-100, -479, -649, -649, -649},
6778 {-100, -479, -649, -649, -649},
6779 {-100, -479, -649, -649, -649}}},
6780 /* GU.C@.. @ */
6781 {{{-100, -309, -289, -289, -289},
6782 {-100, -309, -289, -289, -289},
6783 {-100, -309, -289, -289, -289},
6784 {-100, -309, -289, -289, -289},
6785 {-100, -309, -289, -289, -289}}},
6786 /* GU.CA.. @ */
6787 {{-100, -309, -289, -289, -289},
6788 {-100, -309, -289, -289, -289},
6789 {-100, -309, -289, -289, -289},
6790 {-100, -309, -289, -289, -289},
6791 {-100, -309, -289, -289, -289}}},
6792 /* GU.CC.. @ */
6793 {{-100, -309, -289, -289, -289},
6794 {-100, -309, -289, -289, -289},
6795 {-100, -309, -289, -289, -289},
6796 {-100, -309, -289, -289, -289},
6797 {-100, -309, -289, -289, -289}}},
6798 /* GU.CG.. @ */
6799 {{-100, -309, -289, -289, -289},
6800 {-100, -309, -289, -289, -289},
6801 {-100, -309, -289, -289, -289},
6802 {-100, -309, -289, -289, -289},
6803 {-100, -309, -289, -289, -289}}},
6804 /* GU.CU.. @ */
6805 {{-100, -309, -289, -289, -289},
6806 {-100, -309, -289, -289, -289},
6807 {-100, -309, -289, -289, -289},
6808 {-100, -309, -289, -289, -289},
6809 {-100, -309, -289, -289, -289}}},
6810 /* GU.G@.. @ */
6811 {{{-100, -389, -739, -739, -739},
6812 {-100, -389, -739, -739, -739}}},

```

```

6813 { -100, -389, -739, -739, -739},
6814 { -100, -389, -739, -739, -739},
6815 { -100, -389, -739, -739, -739}},
6816 /* GU.GA.. @ */
6817 {{ -100, -389, -739, -739, -739},
6818 { -100, -389, -739, -739, -739},
6819 { -100, -389, -739, -739, -739},
6820 { -100, -389, -739, -739, -739},
6821 { -100, -389, -739, -739, -739}},
6822 /* GU.GC.. @ */
6823 {{ -100, -389, -739, -739, -739},
6824 { -100, -389, -739, -739, -739},
6825 { -100, -389, -739, -739, -739},
6826 { -100, -389, -739, -739, -739},
6827 { -100, -389, -739, -739, -739}},
6828 /* GU.GG.. @ */
6829 {{ -100, -389, -739, -739, -739},
6830 { -100, -389, -739, -739, -739},
6831 { -100, -389, -739, -739, -739},
6832 { -100, -389, -739, -739, -739},
6833 { -100, -389, -739, -739, -739}},
6834 /* GU.GU.. @ */
6835 {{ -100, -389, -739, -739, -739},
6836 { -100, -389, -739, -739, -739},
6837 { -100, -389, -739, -739, -739},
6838 { -100, -389, -739, -739, -739},
6839 { -100, -389, -739, -739, -739}}),
6840 /* GU.U@.. @ */
6841 {{{ -100, -379, -379, -379, -379},
6842 { -100, -379, -379, -379, -379},
6843 { -100, -379, -379, -379, -379},
6844 { -100, -379, -379, -379, -379},
6845 { -100, -379, -379, -379, -379}},
6846 /* GU.UA.. @ */
6847 {{ -100, -379, -379, -379, -379},
6848 { -100, -379, -379, -379, -379},
6849 { -100, -379, -379, -379, -379},
6850 { -100, -379, -379, -379, -379},
6851 { -100, -379, -379, -379, -379}},
6852 /* GU.UC.. @ */
6853 {{ -100, -379, -379, -379, -379},
6854 { -100, -379, -379, -379, -379},
6855 { -100, -379, -379, -379, -379},
6856 { -100, -379, -379, -379, -379},
6857 { -100, -379, -379, -379, -379}},
6858 /* GU.UG.. @ */
6859 {{ -100, -379, -379, -379, -379},
6860 { -100, -379, -379, -379, -379},
6861 { -100, -379, -379, -379, -379},
6862 { -100, -379, -379, -379, -379},
6863 { -100, -379, -379, -379, -379}},
6864 /* GU.UU.. @ */
6865 {{ -100, -379, -379, -379, -379},
6866 { -100, -379, -379, -379, -379},
6867 { -100, -379, -379, -379, -379},
6868 { -100, -379, -379, -379, -379},
6869 { -100, -379, -379, -379, -379}}}),
6870 { /* noPair */ {{{{0}}}}},
6871 /* UG.@@..CG */
6872 {{{ 0, 0, 0, 0, 0},
6873 { DEF, DEF, DEF, DEF, DEF},
6874 { DEF, DEF, DEF, DEF, DEF},
6875 { DEF, DEF, DEF, DEF, DEF},
6876 { DEF, DEF, DEF, DEF, DEF}},
6877 /* UG.@A..CG */
6878 {{ 0, 0, 0, 0, 0},
6879 {-1029, -1029, -1029, -1029, -1029},
6880 { -519, -519, -519, -519, -519},
6881 { -939, -939, -939, -939, -939},
6882 { -809, -809, -809, -809, -809}},
6883 /* UG.@C..CG */
6884 {{ 0, 0, 0, 0, 0},
6885 { -949, -949, -949, -949, -949},
6886 { -449, -449, -449, -449, -449},
6887 { -939, -939, -939, -939, -939},
6888 { -739, -739, -739, -739, -739}},
6889 /* UG.@G..CG */
6890 {{ 0, 0, 0, 0, 0},
6891 {-1029, -1029, -1029, -1029, -1029},
6892 { -519, -519, -519, -519, -519},
6893 { -939, -939, -939, -939, -939},
6894 { -809, -809, -809, -809, -809}},
6895 /* UG.@U..CG */
6896 {{ 0, 0, 0, 0, 0},
6897 {-1029, -1029, -1029, -1029, -1029},
6898 { -669, -669, -669, -669, -669},
6899 { -939, -939, -939, -939, -939},

```

```
6900 {{ -859, -859, -859, -859, -859}}},  
6901 /* UG.A@..CG */  
6902 {{ { DEF, -719, -789, -959, -809},  
6903 { -100, -769, -839,-1009, -859},  
6904 { -100, -769, -839,-1009, -859},  
6905 { -100, -769, -839,-1009, -859},  
6906 { -100, -769, -839,1009, -859}},  
6907 /* UG.AA..CG */  
6908 {{ { DEF, -719, -789, -959, -809},  
6909 {-1079,-1748,-1818,-1988,-1838},  
6910 { -569,-1238,-1308,-1478,-1328},  
6911 { -989,-1658,-1728,-1898,-1748},  
6912 { -859,-1528,-1598,-1768,-1618}},  
6913 /* UG.AC..CG */  
6914 {{ { DEF, -719, -789, -959, -809},  
6915 { -999,-1668,-1738,-1908,-1758},  
6916 { -499,-1168,-1238,-1408,-1258},  
6917 { -989,-1658,-1728,-1898,-1748},  
6918 { -789,-1458,-1528,-1698,-1548}},  
6919 /* UG.AG..CG */  
6920 {{ { DEF, -719, -789, -959, -809},  
6921 {-1079,-1748,-1818,-1988,-1838},  
6922 { -569,-1238,-1308,-1478,-1328},  
6923 { -989,-1658,-1728,-1898,-1748},  
6924 { -859,-1528,-1598,-1768,-1618}},  
6925 /* UG.AU..CG */  
6926 {{ { DEF, -719, -789, -959, -809},  
6927 {-1079,-1748,-1818,-1988,-1838},  
6928 { -719,-1388,-1458,-1628,-1478},  
6929 { -989,-1658,-1728,-1898,-1748},  
6930 { -909,-1578,-1648,-1818,-1668}}},  
6931 /* UG.C@..CG */  
6932 {{ { DEF, -479, -479, -359, -479},  
6933 { -100, -529, -529, -409, -529},  
6934 { -100, -529, -529, -409, -529},  
6935 { -100, -529, -529, -409, -529},  
6936 { -100, -529, -529, -409, -529}},  
6937 /* UG.CA..CG */  
6938 {{ { DEF, -479, -479, -359, -479},  
6939 {-1079,-1508,-1508,-1388,-1508},  
6940 { -569, -998, -998, -878, -998},  
6941 { -989,-1418,-1418,-1298,-1418},  
6942 { -859,-1288,-1288,-1168,-1288}},  
6943 /* UG.CC..CG */  
6944 {{ { DEF, -479, -479, -359, -479},  
6945 { -999,-1428,-1428,-1308,-1428},  
6946 { -499, -928, -928, -808, -928},  
6947 { -989,-1418,-1418,-1298,-1418},  
6948 { -789,-1218,-1218,-1098,-1218}},  
6949 /* UG.CG..CG */  
6950 {{ { DEF, -479, -479, -359, -479},  
6951 {-1079,-1508,-1508,-1388,-1508},  
6952 { -569, -998, -998, -878, -998},  
6953 { -989,-1418,-1418,-1298,-1418},  
6954 { -859,-1288,-1288,-1168,-1288}},  
6955 /* UG.CU..CG */  
6956 {{ { DEF, -479, -479, -359, -479},  
6957 {-1079,-1508,-1508,-1388,-1508},  
6958 { -719,-1148,-1148,-1028,-1148},  
6959 { -989,-1418,-1418,-1298,-1418},  
6960 { -909,-1338,-1338,-1218,-1338}}},  
6961 /* UG.G@..CG */  
6962 {{ { DEF, -659, -809, -919, -809},  
6963 { -100, -709, -859, -969, -859},  
6964 { -100, -709, -859, -969, -859},  
6965 { -100, -709, -859, -969, -859},  
6966 { -100, -709, -859, -969, -859}},  
6967 /* UG.GA..CG */  
6968 {{ { DEF, -659, -809, -919, -809},  
6969 {-1079,-1688,-1838,-1948,-1838},  
6970 { -569,-1178,-1328,-1438,-1328},  
6971 { -989,-1598,-1748,-1858,-1748},  
6972 { -859,-1468,-1618,-1728,-1618}},  
6973 /* UG.GC..CG */  
6974 {{ { DEF, -659, -809, -919, -809},  
6975 { -999,-1608,-1758,-1868,-1758},  
6976 { -499,-1108,-1258,-1368,-1258},  
6977 { -989,-1598,-1748,-1858,-1748},  
6978 { -789,-1398,-1548,-1658,-1548}},  
6979 /* UG.GG..CG */  
6980 {{ { DEF, -659, -809, -919, -809},  
6981 {-1079,-1688,-1838,-1948,-1838},  
6982 { -569,-1178,-1328,-1438,-1328},  
6983 { -989,-1598,-1748,-1858,-1748},  
6984 { -859,-1468,-1618,-1728,-1618}},  
6985 /* UG.GU..CG */  
6986 {{ { DEF, -659, -809, -919, -809},
```

```

6987 {-1079,-1688,-1838,-1948,-1838},
6988 {-719,-1328,-1478,-1588,-1478},
6989 {-989,-1598,-1748,-1858,-1748},
6990 {-909,-1518,-1668,-1778,-1668}}}},
6991 /* UG.U@..CG */
6992 {{ DEF, -549, -439, -549, -359},
6993 {-100, -599, -489, -599, -409},
6994 {-100, -599, -489, -599, -409},
6995 {-100, -599, -489, -599, -409},
6996 {-100, -599, -489, -599, -409}},
6997 /* UG.UA..CG */
6998 {{ DEF, -549, -439, -549, -359},
6999 {-1079,-1578,-1468,-1578,-1388},
7000 {-569,-1068, -958,-1068, -878},
7001 {-989,-1488,-1378,-1488,-1298},
7002 {-859,-1358,-1248,-1358,-1168}}},
7003 /* UG.UC..CG */
7004 {{ DEF, -549, -439, -549, -359},
7005 {-999,-1498,-1388,-1498,-1308},
7006 {-499, -998, -888, -998, -808},
7007 {-989,-1488,-1378,-1488,-1298},
7008 {-789,-1288,-1178,-1288,-1098}}},
7009 /* UG.UG..CG */
7010 {{ DEF, -549, -439, -549, -359},
7011 {-1079,-1578,-1468,-1578,-1388},
7012 {-569,-1068, -958,-1068, -878},
7013 {-989,-1488,-1378,-1488,-1298},
7014 {-859,-1358,-1248,-1358,-1168}}},
7015 /* UG.UU..CG */
7016 {{ DEF, -549, -439, -549, -359},
7017 {-1079,-1578,-1468,-1578,-1388},
7018 {-719,-1218,-1108,-1218,-1028},
7019 {-989,-1488,-1378,-1488,-1298},
7020 {-909,-1408,-1298,-1408,-1218}}}},
7021 /* UG.@@..GC */
7022 {{{ 0, 0, 0, 0, 0},
7023 { DEF, DEF, DEF, DEF, DEF},
7024 { DEF, DEF, DEF, DEF, DEF},
7025 { DEF, DEF, DEF, DEF, DEF},
7026 { DEF, DEF, DEF, DEF, DEF}},
7027 /* UG.@A..GC */
7028 {{ 0, 0, 0, 0, 0},
7029 {-519, -519, -519, -519, -519},
7030 {-719, -719, -719, -719, -719},
7031 {-709, -709, -709, -709, -709},
7032 {-499, -499, -499, -499, -499}}},
7033 /* UG.@C..GC */
7034 {{ 0, 0, 0, 0, 0},
7035 {-879, -879, -879, -879, -879},
7036 {-309, -309, -309, -309, -309},
7037 {-739, -739, -739, -739, -739},
7038 {-499, -499, -499, -499, -499}}},
7039 /* UG.@G..GC */
7040 {{ 0, 0, 0, 0, 0},
7041 {-559, -559, -559, -559, -559},
7042 {-309, -309, -309, -309, -309},
7043 {-619, -619, -619, -619, -619},
7044 {-499, -499, -499, -499, -499}}},
7045 /* UG.@U..GC */
7046 {{ 0, 0, 0, 0, 0},
7047 {-879, -879, -879, -879, -879},
7048 {-389, -389, -389, -389, -389},
7049 {-739, -739, -739, -739, -739},
7050 {-569, -569, -569, -569, -569}}},
7051 /* UG.A@..GC */
7052 {{ DEF, -719, -789, -959, -809},
7053 {-100, -769, -839,-1009, -859},
7054 {-100, -769, -839,-1009, -859},
7055 {-100, -769, -839,-1009, -859},
7056 {-100, -769, -839,-1009, -859}},
7057 /* UG.AA..GC */
7058 {{ DEF, -719, -789, -959, -809},
7059 {-569,-1238,-1308,-1478,-1328},
7060 {-769,-1438,-1508,-1678,-1528},
7061 {-759,-1428,-1498,-1668,-1518},
7062 {-549,-1218,-1288,-1458,-1308}}},
7063 /* UG.AC..GC */
7064 {{ DEF, -719, -789, -959, -809},
7065 {-929,-1598,-1668,-1838,-1688},
7066 {-359,-1028,-1098,-1268,-1118},
7067 {-789,-1458,-1528,-1698,-1548},
7068 {-549,-1218,-1288,-1458,-1308}}},
7069 /* UG.AG..GC */
7070 {{ DEF, -719, -789, -959, -809},
7071 {-609,-1278,-1348,-1518,-1368},
7072 {-359,-1028,-1098,-1268,-1118},
7073 {-669,-1338,-1408,-1578,-1428},

```

```
7074 {{ -549, -1218, -1288, -1458, -1308}},  
7075 /* UG.AU..GC */  
7076 {{ DEF, -719, -789, -959, -809},  
7077 { -929, -1598, -1668, -1838, -1688},  
7078 { -439, -1108, -1178, -1348, -1198},  
7079 { -789, -1458, -1528, -1698, -1548},  
7080 { -619, -1288, -1358, -1528, -1378}},  
7081 /* UG.C@..GC */  
7082 {{ DEF, -479, -479, -359, -479},  
7083 { -100, -529, -529, -409, -529},  
7084 { -100, -529, -529, -409, -529},  
7085 { -100, -529, -529, -409, -529},  
7086 { -100, -529, -529, -409, -529}},  
7087 /* UG.CA..GC */  
7088 {{ DEF, -479, -479, -359, -479},  
7089 { -569, -998, -998, -878, -998},  
7090 { -769, -1198, -1198, -1078, -1198},  
7091 { -759, -1188, -1188, -1068, -1188},  
7092 { -549, -978, -978, -858, -978}},  
7093 /* UG.CC..GC */  
7094 {{ DEF, -479, -479, -359, -479},  
7095 { -929, -1358, -1358, -1238, -1358},  
7096 { -359, -788, -788, -668, -788},  
7097 { -789, -1218, -1218, -1098, -1218},  
7098 { -549, -978, -978, -858, -978}},  
7099 /* UG.CG..GC */  
7100 {{ DEF, -479, -479, -359, -479},  
7101 { -609, -1038, -1038, -918, -1038},  
7102 { -359, -788, -788, -668, -788},  
7103 { -669, -1098, -1098, -978, -1098},  
7104 { -549, -978, -978, -858, -978}},  
7105 /* UG.CU..GC */  
7106 {{ DEF, -479, -479, -359, -479},  
7107 { -929, -1358, -1358, -1238, -1358},  
7108 { -439, -868, -868, -748, -868},  
7109 { -789, -1218, -1218, -1098, -1218},  
7110 { -619, -1048, -1048, -928, -1048}},  
7111 /* UG.G@..GC */  
7112 {{ DEF, -659, -809, -919, -809},  
7113 { -100, -709, -859, -969, -859},  
7114 { -100, -709, -859, -969, -859},  
7115 { -100, -709, -859, -969, -859},  
7116 { -100, -709, -859, -969, -859}},  
7117 /* UG.GA..GC */  
7118 {{ DEF, -659, -809, -919, -809},  
7119 { -569, -1178, -1328, -1438, -1328},  
7120 { -769, -1378, -1528, -1638, -1528},  
7121 { -759, -1368, -1518, -1628, -1518},  
7122 { -549, -1158, -1308, -1418, -1308}},  
7123 /* UG.GC..GC */  
7124 {{ DEF, -659, -809, -919, -809},  
7125 { -929, -1538, -1688, -1798, -1688},  
7126 { -359, -968, -1118, -1228, -1118},  
7127 { -789, -1398, -1548, -1658, -1548},  
7128 { -549, -1158, -1308, -1418, -1308}},  
7129 /* UG.GG..GC */  
7130 {{ DEF, -659, -809, -919, -809},  
7131 { -609, -1218, -1368, -1478, -1368},  
7132 { -359, -968, -1118, -1228, -1118},  
7133 { -669, -1278, -1428, -1538, -1428},  
7134 { -549, -1158, -1308, -1418, -1308}},  
7135 /* UG.GU..GC */  
7136 {{ DEF, -659, -809, -919, -809},  
7137 { -929, -1538, -1688, -1798, -1688},  
7138 { -439, -1048, -1198, -1308, -1198},  
7139 { -789, -1398, -1548, -1658, -1548},  
7140 { -619, -1228, -1378, -1488, -1378}},  
7141 /* UG.U@..GC */  
7142 {{ DEF, -549, -439, -549, -359},  
7143 { -100, -599, -489, -599, -409},  
7144 { -100, -599, -489, -599, -409},  
7145 { -100, -599, -489, -599, -409},  
7146 { -100, -599, -489, -599, -409}},  
7147 /* UG.UA..GC */  
7148 {{ DEF, -549, -439, -549, -359},  
7149 { -569, -1068, -958, -1068, -878},  
7150 { -769, -1268, -1158, -1268, -1078},  
7151 { -759, -1258, -1148, -1258, -1068},  
7152 { -549, -1048, -938, -1048, -858}},  
7153 /* UG.UC..GC */  
7154 {{ DEF, -549, -439, -549, -359},  
7155 { -929, -1428, -1318, -1428, -1238},  
7156 { -359, -858, -748, -858, -668},  
7157 { -789, -1288, -1178, -1288, -1098},  
7158 { -549, -1048, -938, -1048, -858}},  
7159 /* UG.UG..GC */  
7160 {{ DEF, -549, -439, -549, -359},
```

```

7161 { -609, -1108, -998, -1108, -918},
7162 { -359, -858, -748, -858, -668},
7163 { -669, -1168, -1058, -1168, -978},
7164 { -549, -1048, -938, -1048, -858}),
7165 /* UG.UU..GC */
7166 {{ DEF, -549, -439, -549, -359},
7167 { -929, -1428, -1318, -1428, -1238},
7168 { -439, -938, -828, -938, -748},
7169 { -789, -1288, -1178, -1288, -1098},
7170 { -619, -1118, -1008, -1118, -928}}}},
7171 /* UG.@@..GU */
7172 {{{ 0, 0, 0, 0, 0},
7173 { DEF, DEF, DEF, DEF, DEF},
7174 { DEF, DEF, DEF, DEF, DEF},
7175 { DEF, DEF, DEF, DEF, DEF},
7176 { DEF, DEF, DEF, DEF, DEF}}},
7177 /* UG.@A..GU */
7178 {{ 0, 0, 0, 0, 0},
7179 { -429, -429, -429, -429, -429},
7180 { -259, -259, -259, -259, -259},
7181 { -339, -339, -339, -339, -339},
7182 { -329, -329, -329, -329, -329}},
7183 /* UG.@C..GU */
7184 {{ 0, 0, 0, 0, 0},
7185 { -599, -599, -599, -599, -599},
7186 { -239, -239, -239, -239, -239},
7187 { -689, -689, -689, -689, -689},
7188 { -329, -329, -329, -329, -329}},
7189 /* UG.@G..GU */
7190 {{ 0, 0, 0, 0, 0},
7191 { -599, -599, -599, -599, -599},
7192 { -239, -239, -239, -239, -239},
7193 { -689, -689, -689, -689, -689},
7194 { -329, -329, -329, -329, -329}},
7195 /* UG.@U..GU */
7196 {{ 0, 0, 0, 0, 0},
7197 { -599, -599, -599, -599, -599},
7198 { -239, -239, -239, -239, -239},
7199 { -689, -689, -689, -689, -689},
7200 { -329, -329, -329, -329, -329}}}},
7201 /* UG.A@..GU */
7202 {{{{ DEF, -719, -789, -959, -809},
7203 { -100, -769, -839, -1009, -859},
7204 { -100, -769, -839, -1009, -859},
7205 { -100, -769, -839, -1009, -859},
7206 { -100, -769, -839, -1009, -859}},
7207 /* UG.AA..GU */
7208 {{ DEF, -719, -789, -959, -809},
7209 { -479, -1148, -1218, -1388, -1238},
7210 { -309, -978, -1048, -1218, -1068},
7211 { -389, -1058, -1128, -1298, -1148},
7212 { -379, -1048, -1118, -1288, -1138}},
7213 /* UG.AC..GU */
7214 {{ DEF, -719, -789, -959, -809},
7215 { -649, -1318, -1388, -1558, -1408},
7216 { -289, -958, -1028, -1198, -1048},
7217 { -739, -1408, -1478, -1648, -1498},
7218 { -379, -1048, -1118, -1288, -1138}},
7219 /* UG.AG..GU */
7220 {{ DEF, -719, -789, -959, -809},
7221 { -649, -1318, -1388, -1558, -1408},
7222 { -289, -958, -1028, -1198, -1048},
7223 { -739, -1408, -1478, -1648, -1498},
7224 { -379, -1048, -1118, -1288, -1138}},
7225 /* UG.AU..GU */
7226 {{ DEF, -719, -789, -959, -809},
7227 { -649, -1318, -1388, -1558, -1408},
7228 { -289, -958, -1028, -1198, -1048},
7229 { -739, -1408, -1478, -1648, -1498},
7230 { -379, -1048, -1118, -1288, -1138}}}},
7231 /* UG.C@..GU */
7232 {{{ DEF, -479, -479, -359, -479},
7233 { -100, -529, -529, -409, -529},
7234 { -100, -529, -529, -409, -529},
7235 { -100, -529, -529, -409, -529},
7236 { -100, -529, -529, -409, -529}}},
7237 /* UG.CA..GU */
7238 {{ DEF, -479, -479, -359, -479},
7239 { -479, -908, -908, -788, -908},
7240 { -309, -738, -738, -618, -738},
7241 { -389, -818, -818, -698, -818},
7242 { -379, -808, -808, -688, -808}},
7243 /* UG.CC..GU */
7244 {{ DEF, -479, -479, -359, -479},
7245 { -649, -1078, -1078, -958, -1078},
7246 { -289, -718, -718, -598, -718},
7247 { -739, -1168, -1168, -1048, -1168},

```

```

7248 { -379, -808, -808, -688, -808}},
7249 /* UG.CG..GU */
7250 {{ DEF, -479, -479, -359, -479},
7251 { -649,-1078,-1078, -958,-1078},
7252 { -289, -718, -718, -598, -718},
7253 { -739,-1168,-1168,-1048,-1168},
7254 { -379, -808, -808, -688, -808}},
7255 /* UG.CU..GU */
7256 {{ DEF, -479, -479, -359, -479},
7257 { -649,-1078,-1078, -958,-1078},
7258 { -289, -718, -718, -598, -718},
7259 { -739,-1168,-1168,-1048,-1168},
7260 { -379, -808, -808, -688, -808}}),
7261 /* UG.G@..GU */
7262 {{ DEF, -659, -809, -919, -809},
7263 { -100, -709, -859, -969, -859},
7264 { -100, -709, -859, -969, -859},
7265 { -100, -709, -859, -969, -859},
7266 { -100, -709, -859, -969, -859}},
7267 /* UG.GA..GU */
7268 {{ DEF, -659, -809, -919, -809},
7269 { -479,-1088,-1238,-1348,-1238},
7270 { -309, -918,-1068,-1178,-1068},
7271 { -389, -998,-1148,-1258,-1148},
7272 { -379, -988,-1138,-1248,-1138}},
7273 /* UG.GC..GU */
7274 {{ DEF, -659, -809, -919, -809},
7275 { -649, -1258, -1408, -1518, -1408},
7276 { -289, -898,-1048,-1158,-1048},
7277 { -739,-1348,-1498,-1608,-1498},
7278 { -379, -988,-1138,-1248,-1138}},
7279 /* UG.GG..GU */
7280 {{ DEF, -659, -809, -919, -809},
7281 { -649,-1258,-1408,-1518,-1408},
7282 { -289, -898,-1048,-1158,-1048},
7283 { -739,-1348,-1498,-1608,-1498},
7284 { -379, -988,-1138,-1248,-1138}},
7285 /* UG.GU..GU */
7286 {{ DEF, -659, -809, -919, -809},
7287 { -649,-1258,-1408,-1518,-1408},
7288 { -289, -898,-1048,-1158,-1048},
7289 { -739,-1348,-1498,-1608,-1498},
7290 { -379, -988,-1138,-1248,-1138}}),
7291 /* UG.U@..GU */
7292 {{ DEF, -549, -439, -549, -359},
7293 { -100, -599, -489, -599, -409},
7294 { -100, -599, -489, -599, -409},
7295 { -100, -599, -489, -599, -409},
7296 { -100, -599, -489, -599, -409}},
7297 /* UG.UA..GU */
7298 {{ DEF, -549, -439, -549, -359},
7299 { -479, -978, -868, -978, -788},
7300 { -309, -808, -698, -808, -618},
7301 { -389, -888, -778, -888, -698},
7302 { -379, -878, -768, -878, -688}},
7303 /* UG.UC..GU */
7304 {{ DEF, -549, -439, -549, -359},
7305 { -649,-1148,-1038,-1148, -958},
7306 { -289, -788, -678, -788, -598},
7307 { -739,-1238,-1128,-1238,-1048},
7308 { -379, -878, -768, -878, -688}},
7309 /* UG.UG..GU */
7310 {{ DEF, -549, -439, -549, -359},
7311 { -649,-1148,-1038,-1148, -958},
7312 { -289, -788, -678, -788, -598},
7313 { -739,-1238,-1128,-1238,-1048},
7314 { -379, -878, -768, -878, -688}},
7315 /* UG.UU..GU */
7316 {{ DEF, -549, -439, -549, -359},
7317 { -649,-1148,-1038,-1148, -958},
7318 { -289, -788, -678, -788, -598},
7319 { -739,-1238,-1128,-1238,-1048},
7320 { -379, -878, -768, -878, -688}}),
7321 /* UG.@@..UG */
7322 {{{ 0, 0, 0, 0, 0},
7323 { DEF, DEF, DEF, DEF, DEF},
7324 { DEF, DEF, DEF, DEF, DEF},
7325 { DEF, DEF, DEF, DEF, DEF},
7326 { DEF, DEF, DEF, DEF, DEF}},
7327 /* UG.@A..UG */
7328 {{ 0, 0, 0, 0, 0},
7329 { -719, -719, -719, -719, -719},
7330 { -479, -479, -479, -479, -479},
7331 { -659, -659, -659, -659, -659},
7332 { -549, -549, -549, -549, -549}},
7333 /* UG.@C..UG */
7334 {{ 0, 0, 0, 0, 0},

```

```

7335 { -789, -789, -789, -789, -789},
7336 { -479, -479, -479, -479, -479},
7337 { -809, -809, -809, -809, -809},
7338 { -439, -439, -439, -439, -439}},
7339 /* UG.@G..UG */
7340 {{ 0, 0, 0, 0, 0},
7341 { -959, -959, -959, -959, -959},
7342 { -359, -359, -359, -359, -359},
7343 { -919, -919, -919, -919, -919},
7344 { -549, -549, -549, -549, -549}},
7345 /* UG.@U..UG */
7346 {{ 0, 0, 0, 0, 0},
7347 { -809, -809, -809, -809, -809},
7348 { -479, -479, -479, -479, -479},
7349 { -809, -809, -809, -809, -809},
7350 { -359, -359, -359, -359, -359}},
7351 /* UG.A@..UG */
7352 {{ DEF, -719, -789, -959, -809},
7353 { -100, -769, -839,-1009, -859},
7354 { -100, -769, -839,-1009, -859},
7355 { -100, -769, -839,-1009, -859},
7356 { -100, -769, -839,-1009, -859}},
7357 /* UG.AA..UG */
7358 {{ DEF, -719, -789, -959, -809},
7359 { -769,-1438,-1508,-1678,-1528},
7360 { -529,-1198,-1268,-1438,-1288},
7361 { -709,-1378,-1448,-1618,-1468},
7362 { -599,-1268,-1338,-1508,-1358}},
7363 /* UG.AC..UG */
7364 {{ DEF, -719, -789, -959, -809},
7365 { -839,-1508,-1578,-1748,-1598},
7366 { -529,-1198,-1268,-1438,-1288},
7367 { -859,-1528,-1598,-1768,-1618},
7368 { -489,-1158,-1228,-1398,-1248}},
7369 /* UG.AG..UG */
7370 {{ DEF, -719, -789, -959, -809},
7371 { -1009,-1678,-1748,-1918,-1768},
7372 { -409,-1078,-1148,-1318,-1168},
7373 { -969,-1638,-1708,-1878,-1728},
7374 { -599,-1268,-1338,-1508,-1358}},
7375 /* UG.AU..UG */
7376 {{ DEF, -719, -789, -959, -809},
7377 { -859,-1528,-1598,-1768,-1618},
7378 { -529,-1198,-1268,-1438,-1288},
7379 { -859,-1528,-1598,-1768,-1618},
7380 { -409,-1078,-1148,-1318,-1168}},
7381 /* UG.C@..UG */
7382 {{ DEF, -479, -479, -359, -479},
7383 { -100, -529, -529, -409, -529},
7384 { -100, -529, -529, -409, -529},
7385 { -100, -529, -529, -409, -529},
7386 { -100, -529, -529, -409, -529}},
7387 /* UG.CA..UG */
7388 {{ DEF, -479, -479, -359, -479},
7389 { -769,-1198,-1198,-1078,-1198},
7390 { -529, -958, -958, -838, -958},
7391 { -709,-1138,-1138,-1018,-1138},
7392 { -599,-1028,-1028, -908,-1028}},
7393 /* UG.CC..UG */
7394 {{ DEF, -479, -479, -359, -479},
7395 { -839,-1268,-1268,-1148,-1268},
7396 { -529, -958, -958, -838, -958},
7397 { -859,-1288,-1288,-1168,-1288},
7398 { -489, -918, -918, -798, -918}},
7399 /* UG.CG..UG */
7400 {{ DEF, -479, -479, -359, -479},
7401 { -1009,-1438,-1438,-1318,-1438},
7402 { -409, -838, -838, -718, -838},
7403 { -969,-1398,-1398,-1278,-1398},
7404 { -599,-1028,-1028, -908,-1028}},
7405 /* UG.CU..UG */
7406 {{ DEF, -479, -479, -359, -479},
7407 { -859,-1288,-1288,-1168,-1288},
7408 { -529, -958, -958, -838, -958},
7409 { -859,-1288,-1288,-1168,-1288},
7410 { -409, -838, -838, -718, -838}},
7411 /* UG.G@..UG */
7412 {{ DEF, -659, -809, -919, -809},
7413 { -100, -709, -859, -969, -859},
7414 { -100, -709, -859, -969, -859},
7415 { -100, -709, -859, -969, -859},
7416 { -100, -709, -859, -969, -859}},
7417 /* UG.GA..UG */
7418 {{ DEF, -659, -809, -919, -809},
7419 { -769,-1378,-1528,-1638,-1528},
7420 { -529,-1138,-1288,-1398,-1288},
7421 { -709,-1318,-1468,-1578,-1468},

```

```

7422 { -599, -1208, -1358, -1468, -1358}},
7423 /* UG.GC..UG */
7424 {{ DEF, -659, -809, -919, -809},
7425 { -839, -1448, -1598, -1708, -1598},
7426 { -529, -1138, -1288, -1398, -1288},
7427 { -859, -1468, -1618, -1728, -1618},
7428 { -489, -1098, -1248, -1358, -1248}},
7429 /* UG.GG..UG */
7430 {{ DEF, -659, -809, -919, -809},
7431 { -1009, -1618, -1768, -1878, -1768},
7432 { -409, -1018, -1168, -1278, -1168},
7433 { -969, -1578, -1728, -1838, -1728},
7434 { -599, -1208, -1358, -1468, -1358}},
7435 /* UG.GU..UG */
7436 {{ DEF, -659, -809, -919, -809},
7437 { -859, -1468, -1618, -1728, -1618},
7438 { -529, -1138, -1288, -1398, -1288},
7439 { -859, -1468, -1618, -1728, -1618},
7440 { -409, -1018, -1168, -1278, -1168}}}},
7441 /* UG.U@..UG */
7442 {{{{ DEF, -549, -439, -549, -359},
7443 { -100, -599, -489, -599, -409},
7444 { -100, -599, -489, -599, -409},
7445 { -100, -599, -489, -599, -409},
7446 { -100, -599, -489, -599, -409}},
7447 /* UG.UA..UG */
7448 {{ DEF, -549, -439, -549, -359},
7449 { -769, -1268, -1158, -1268, -1078},
7450 { -529, -1028, -918, -1028, -838},
7451 { -709, -1208, -1098, -1208, -1018},
7452 { -599, -1098, -988, -1098, -908}},
7453 /* UG.UC..UG */
7454 {{ DEF, -549, -439, -549, -359},
7455 { -839, -1338, -1228, -1338, -1148},
7456 { -529, -1028, -918, -1028, -838},
7457 { -859, -1358, -1248, -1358, -1168},
7458 { -489, -988, -878, -988, -798}},
7459 /* UG.UG..UG */
7460 {{ DEF, -549, -439, -549, -359},
7461 { -1009, -1508, -1398, -1508, -1318},
7462 { -409, -908, -798, -908, -718},
7463 { -969, -1468, -1358, -1468, -1278},
7464 { -599, -1098, -988, -1098, -908}},
7465 /* UG.UU..UG */
7466 {{ DEF, -549, -439, -549, -359},
7467 { -859, -1358, -1248, -1358, -1168},
7468 { -529, -1028, -918, -1028, -838},
7469 { -859, -1358, -1248, -1358, -1168},
7470 { -409, -908, -798, -908, -718}}}},
7471 /* UG.@A..AU */
7472 {{{{ 0, 0, 0, 0, 0},
7473 { DEF, DEF, DEF, DEF, DEF},
7474 { DEF, DEF, DEF, DEF, DEF},
7475 { DEF, DEF, DEF, DEF, DEF},
7476 { DEF, DEF, DEF, DEF, DEF}},
7477 /* UG.@A..AU */
7478 {{ 0, 0, 0, 0, 0},
7479 { -429, -429, -429, -429, -429},
7480 { -259, -259, -259, -259, -259},
7481 { -339, -339, -339, -339, -339},
7482 { -329, -329, -329, -329, -329}},
7483 /* UG.@C..AU */
7484 {{ 0, 0, 0, 0, 0},
7485 { -599, -599, -599, -599, -599},
7486 { -239, -239, -239, -239, -239},
7487 { -689, -689, -689, -689, -689},
7488 { -329, -329, -329, -329, -329}},
7489 /* UG.@G..AU */
7490 {{ 0, 0, 0, 0, 0},
7491 { -599, -599, -599, -599, -599},
7492 { -239, -239, -239, -239, -239},
7493 { -689, -689, -689, -689, -689},
7494 { -329, -329, -329, -329, -329}},
7495 /* UG.@U..AU */
7496 {{ 0, 0, 0, 0, 0},
7497 { -599, -599, -599, -599, -599},
7498 { -239, -239, -239, -239, -239},
7499 { -689, -689, -689, -689, -689},
7500 { -329, -329, -329, -329, -329}}},
7501 /* UG.A@..AU */
7502 {{{ DEF, -719, -789, -959, -809},
7503 { -100, -769, -839, -1009, -859},
7504 { -100, -769, -839, -1009, -859},
7505 { -100, -769, -839, -1009, -859},
7506 { -100, -769, -839, -1009, -859}}},
7507 /* UG.AA..AU */
7508 {{ DEF, -719, -789, -959, -809},

```

```

7509 { -479, -1148, -1218, -1388, -1238},
7510 { -309, -978, -1048, -1218, -1068},
7511 { -389, -1058, -1128, -1298, -1148},
7512 { -379, -1048, -1118, -1288, -1138}},
7513 /* UG.AC..AU */
7514 {{ DEF, -719, -789, -959, -809},
7515 { -649, -1318, -1388, -1558, -1408},
7516 { -289, -958, -1028, -1198, -1048},
7517 { -739, -1408, -1478, -1648, -1498},
7518 { -379, -1048, -1118, -1288, -1138}},
7519 /* UG.AG..AU */
7520 {{ DEF, -719, -789, -959, -809},
7521 { -649, -1318, -1388, -1558, -1408},
7522 { -289, -958, -1028, -1198, -1048},
7523 { -739, -1408, -1478, -1648, -1498},
7524 { -379, -1048, -1118, -1288, -1138}},
7525 /* UG.AU..AU */
7526 {{ DEF, -719, -789, -959, -809},
7527 { -649, -1318, -1388, -1558, -1408},
7528 { -289, -958, -1028, -1198, -1048},
7529 { -739, -1408, -1478, -1648, -1498},
7530 { -379, -1048, -1118, -1288, -1138}},
7531 /* UG.C@..AU */
7532 {{ DEF, -479, -479, -359, -479},
7533 { -100, -529, -529, -409, -529},
7534 { -100, -529, -529, -409, -529},
7535 { -100, -529, -529, -409, -529},
7536 { -100, -529, -529, -409, -529}},
7537 /* UG.CA..AU */
7538 {{ DEF, -479, -479, -359, -479},
7539 { -479, -908, -908, -788, -908},
7540 { -309, -738, -738, -618, -738},
7541 { -389, -818, -818, -698, -818},
7542 { -379, -808, -808, -688, -808}},
7543 /* UG.CC..AU */
7544 {{ DEF, -479, -479, -359, -479},
7545 { -649, -1078, -1078, -958, -1078},
7546 { -289, -718, -718, -598, -718},
7547 { -739, -1168, -1168, -1048, -1168},
7548 { -379, -808, -808, -688, -808}},
7549 /* UG.CG..AU */
7550 {{ DEF, -479, -479, -359, -479},
7551 { -649, -1078, -1078, -958, -1078},
7552 { -289, -718, -718, -598, -718},
7553 { -739, -1168, -1168, -1048, -1168},
7554 { -379, -808, -808, -688, -808}},
7555 /* UG.CU..AU */
7556 {{ DEF, -479, -479, -359, -479},
7557 { -649, -1078, -1078, -958, -1078},
7558 { -289, -718, -718, -598, -718},
7559 { -739, -1168, -1168, -1048, -1168},
7560 { -379, -808, -808, -688, -808}},
7561 /* UG.G@..AU */
7562 {{ DEF, -659, -809, -919, -809},
7563 { -100, -709, -859, -969, -859},
7564 { -100, -709, -859, -969, -859},
7565 { -100, -709, -859, -969, -859},
7566 { -100, -709, -859, -969, -859}},
7567 /* UG.GA..AU */
7568 {{ DEF, -659, -809, -919, -809},
7569 { -479, -1088, -1238, -1348, -1238},
7570 { -309, -918, -1068, -1178, -1068},
7571 { -389, -998, -1148, -1258, -1148},
7572 { -379, -988, -1138, -1248, -1138}},
7573 /* UG.GC..AU */
7574 {{ DEF, -659, -809, -919, -809},
7575 { -649, -1258, -1408, -1518, -1408},
7576 { -289, -898, -1048, -1158, -1048},
7577 { -739, -1348, -1498, -1608, -1498},
7578 { -379, -988, -1138, -1248, -1138}},
7579 /* UG.GG..AU */
7580 {{ DEF, -659, -809, -919, -809},
7581 { -649, -1258, -1408, -1518, -1408},
7582 { -289, -898, -1048, -1158, -1048},
7583 { -739, -1348, -1498, -1608, -1498},
7584 { -379, -988, -1138, -1248, -1138}},
7585 /* UG.GU..AU */
7586 {{ DEF, -659, -809, -919, -809},
7587 { -649, -1258, -1408, -1518, -1408},
7588 { -289, -898, -1048, -1158, -1048},
7589 { -739, -1348, -1498, -1608, -1498},
7590 { -379, -988, -1138, -1248, -1138}},
7591 /* UG.U@..AU */
7592 {{ DEF, -549, -439, -549, -359},
7593 { -100, -599, -489, -599, -409},
7594 { -100, -599, -489, -599, -409},
7595 { -100, -599, -489, -599, -409},

```

```

7596 { -100, -599, -489, -599, -409}},
7597 /* UG.UA..AU */
7598 {{ DEF, -549, -439, -549, -359},
7599 { -479, -978, -868, -978, -788},
7600 { -309, -808, -698, -808, -618},
7601 { -389, -888, -778, -888, -698},
7602 { -379, -878, -768, -878, -688}},
7603 /* UG.UC..AU */
7604 {{ DEF, -549, -439, -549, -359},
7605 { -649,-1148,-1038,-1148, -958},
7606 { -289, -788, -678, -788, -598},
7607 { -739,-1238,-1128,-1238,-1048},
7608 { -379, -878, -768, -878, -688}},
7609 /* UG.UG..AU */
7610 {{ DEF, -549, -439, -549, -359},
7611 { -649,-1148,-1038,-1148, -958},
7612 { -289, -788, -678, -788, -598},
7613 { -739,-1238,-1128,-1238,-1048},
7614 { -379, -878, -768, -878, -688}}}},
7615 /* UG.UU..AU */
7616 {{ DEF, -549, -439, -549, -359},
7617 { -649,-1148,-1038,-1148, -958},
7618 { -289, -788, -678, -788, -598},
7619 { -739,-1238,-1128,-1238,-1048},
7620 { -379, -878, -768, -878, -688}}}},
7621 /* UG.@A..UA */
7622 {{{ 0, 0, 0, 0, 0},
7623 { DEF, DEF, DEF, DEF, DEF},
7624 { DEF, DEF, DEF, DEF, DEF},
7625 { DEF, DEF, DEF, DEF, DEF},
7626 { DEF, DEF, DEF, DEF, DEF}}},
7627 /* UG.@A..UA */
7628 {{ 0, 0, 0, 0, 0},
7629 { -399, -399, -399, -399, -399},
7630 { -429, -429, -429, -429, -429},
7631 { -379, -379, -379, -379, -379},
7632 { -279, -279, -279, -279, -279}},
7633 /* UG.@C..UA */
7634 {{ 0, 0, 0, 0, 0},
7635 { -629, -629, -629, -629, -629},
7636 { -509, -509, -509, -509, -509},
7637 { -679, -679, -679, -679, -679},
7638 { -139, -139, -139, -139, -139}},
7639 /* UG.@G..UA */
7640 {{ 0, 0, 0, 0, 0},
7641 { -889, -889, -889, -889, -889},
7642 { -199, -199, -199, -199, -199},
7643 { -889, -889, -889, -889, -889},
7644 { -279, -279, -279, -279, -279}},
7645 /* UG.@U..UA */
7646 {{ 0, 0, 0, 0, 0},
7647 { -589, -589, -589, -589, -589},
7648 { -179, -179, -179, -179, -179},
7649 { -679, -679, -679, -679, -679},
7650 { -140, -140, -140, -140, -140}}}},
7651 /* UG.A@..UA */
7652 {{ DEF, -719, -789, -959, -809},
7653 { -100, -769, -839,-1009, -859},
7654 { -100, -769, -839,-1009, -859},
7655 { -100, -769, -839,-1009, -859},
7656 { -100, -769, -839,-1009, -859}},
7657 /* UG.AA..UA */
7658 {{ DEF, -719, -789, -959, -809},
7659 { -449,-1118,-1188,-1358,-1208},
7660 { -479,-1148,-1218,-1388,-1238},
7661 { -429,-1098,-1168,-1338,-1188},
7662 { -329, -998, -1068,-1238,-1088}},
7663 /* UG.AC..UA */
7664 {{ DEF, -719, -789, -959, -809},
7665 { -679,-1348,-1418,-1588,-1438},
7666 { -559,-1228,-1298,-1468,-1318},
7667 { -729,-1398,-1468,-1638,-1488},
7668 { -189, -858, -928,-1098, -948}},
7669 /* UG.AG..UA */
7670 {{ DEF, -719, -789, -959, -809},
7671 { -939,-1608,-1678,-1848,-1698},
7672 { -249, -918, -988,-1158,-1008},
7673 { -939,-1608,-1678,-1848,-1698},
7674 { -329, -998,-1068,-1238,-1088}},
7675 /* UG.AU..UA */
7676 {{ DEF, -719, -789, -959, -809},
7677 { -639,-1308,-1378,-1548,-1398},
7678 { -229, -898, -968,-1138, -988},
7679 { -729,-1398,-1468,-1638,-1488},
7680 { -190, -859, -929,-1099, -949}}}},
7681 /* UG.C@..UA */
7682 {{ DEF, -479, -479, -359, -479},

```

```

7683 { -100, -529, -529, -409, -529},
7684 { -100, -529, -529, -409, -529},
7685 { -100, -529, -529, -409, -529},
7686 { -100, -529, -529, -409, -529}},
7687 /* UG.CA..UA */
7688 {{ DEF, -479, -479, -359, -479},
7689 { -449, -878, -878, -758, -878},
7690 { -479, -908, -908, -788, -908},
7691 { -429, -858, -858, -738, -858},
7692 { -329, -758, -758, -638, -758}},
7693 /* UG.CC..UA */
7694 {{ DEF, -479, -479, -359, -479},
7695 { -679, -1108, -1108, -988, -1108},
7696 { -559, -988, -988, -868, -988},
7697 { -729, -1158, -1158, -1038, -1158},
7698 { -189, -618, -618, -498, -618}},
7699 /* UG.CG..UA */
7700 {{ DEF, -479, -479, -359, -479},
7701 { -939, -1368, -1368, -1248, -1368},
7702 { -249, -678, -678, -558, -678},
7703 { -939, -1368, -1368, -1248, -1368},
7704 { -329, -758, -758, -638, -758}},
7705 /* UG.CU..UA */
7706 {{ DEF, -479, -479, -359, -479},
7707 { -639, -1068, -1068, -948, -1068},
7708 { -229, -658, -658, -538, -658},
7709 { -729, -1158, -1158, -1038, -1158},
7710 { -190, -619, -619, -499, -619}},
7711 /* UG.G@..UA */
7712 {{ DEF, -659, -809, -919, -809},
7713 { -100, -709, -859, -969, -859},
7714 { -100, -709, -859, -969, -859},
7715 { -100, -709, -859, -969, -859},
7716 { -100, -709, -859, -969, -859}},
7717 /* UG.GA..UA */
7718 {{ DEF, -659, -809, -919, -809},
7719 { -449, -1058, -1208, -1318, -1208},
7720 { -479, -1088, -1238, -1348, -1238},
7721 { -429, -1038, -1188, -1298, -1188},
7722 { -329, -938, -1088, -1198, -1088}},
7723 /* UG.GC..UA */
7724 {{ DEF, -659, -809, -919, -809},
7725 { -679, -1288, -1438, -1548, -1438},
7726 { -559, -1168, -1318, -1428, -1318},
7727 { -729, -1338, -1488, -1598, -1488},
7728 { -189, -798, -948, -1058, -948}},
7729 /* UG.GG..UA */
7730 {{ DEF, -659, -809, -919, -809},
7731 { -939, -1548, -1698, -1808, -1698},
7732 { -249, -858, -1008, -1118, -1008},
7733 { -939, -1548, -1698, -1808, -1698},
7734 { -329, -938, -1088, -1198, -1088}},
7735 /* UG.GU..UA */
7736 {{ DEF, -659, -809, -919, -809},
7737 { -639, -1248, -1398, -1508, -1398},
7738 { -229, -838, -988, -1098, -988},
7739 { -729, -1338, -1488, -1598, -1488},
7740 { -190, -799, -949, -1059, -949}},
7741 /* UG.U@..UA */
7742 {{ DEF, -549, -439, -549, -359},
7743 { -100, -599, -489, -599, -409},
7744 { -100, -599, -489, -599, -409},
7745 { -100, -599, -489, -599, -409},
7746 { -100, -599, -489, -599, -409}},
7747 /* UG.UA..UA */
7748 {{ DEF, -549, -439, -549, -359},
7749 { -449, -948, -838, -948, -758},
7750 { -479, -978, -868, -978, -788},
7751 { -429, -928, -818, -928, -738},
7752 { -329, -828, -718, -828, -638}},
7753 /* UG.UC..UA */
7754 {{ DEF, -549, -439, -549, -359},
7755 { -679, -1178, -1068, -1178, -988},
7756 { -559, -1058, -948, -1058, -868},
7757 { -729, -1228, -1118, -1228, -1038},
7758 { -189, -688, -578, -688, -498}},
7759 /* UG.UG..UA */
7760 {{ DEF, -549, -439, -549, -359},
7761 { -939, -1438, -1328, -1438, -1248},
7762 { -249, -748, -638, -748, -558},
7763 { -939, -1438, -1328, -1438, -1248},
7764 { -329, -828, -718, -828, -638}},
7765 /* UG.UU..UA */
7766 {{ DEF, -549, -439, -549, -359},
7767 { -639, -1138, -1028, -1138, -948},
7768 { -229, -728, -618, -728, -538},
7769 { -729, -1228, -1118, -1228, -1038},

```

```

7770 { -190, -689, -579, -689, -499}}},  

7771 /* UG.@@.. @ */  

7772 {{{ DEF, DEF, DEF, DEF, DEF},  

7773 { DEF, DEF, DEF, DEF, DEF},  

7774 { DEF, DEF, DEF, DEF, DEF},  

7775 { DEF, DEF, DEF, DEF, DEF},  

7776 { DEF, DEF, DEF, DEF, DEF}},  

7777 /* UG.@A.. @ */  

7778 {{ DEF, DEF, DEF, DEF, DEF},  

7779 { DEF, DEF, DEF, DEF, DEF},  

7780 { DEF, DEF, DEF, DEF, DEF},  

7781 { DEF, DEF, DEF, DEF, DEF},  

7782 { DEF, DEF, DEF, DEF, DEF}},  

7783 /* UG.@C.. @ */  

7784 {{ DEF, DEF, DEF, DEF, DEF},  

7785 { DEF, DEF, DEF, DEF, DEF},  

7786 { DEF, DEF, DEF, DEF, DEF},  

7787 { DEF, DEF, DEF, DEF, DEF},  

7788 { DEF, DEF, DEF, DEF, DEF}},  

7789 /* UG.@G.. @ */  

7790 {{ DEF, DEF, DEF, DEF, DEF},  

7791 { DEF, DEF, DEF, DEF, DEF},  

7792 { DEF, DEF, DEF, DEF, DEF},  

7793 { DEF, DEF, DEF, DEF, DEF},  

7794 { DEF, DEF, DEF, DEF, DEF}},  

7795 /* UG.@U.. @ */  

7796 {{ DEF, DEF, DEF, DEF, DEF},  

7797 { DEF, DEF, DEF, DEF, DEF},  

7798 { DEF, DEF, DEF, DEF, DEF},  

7799 { DEF, DEF, DEF, DEF, DEF},  

7800 { DEF, DEF, DEF, DEF, DEF}}},  

7801 /* UG.A@.. @ */  

7802 {{ -100, -769, -839, -1009, -859},  

7803 { -100, -769, -839, -1009, -859},  

7804 { -100, -769, -839, -1009, -859},  

7805 { -100, -769, -839, -1009, -859},  

7806 { -100, -769, -839, -1009, -859}},  

7807 /* UG.AA.. @ */  

7808 {{ -100, -769, -839, -1009, -859},  

7809 { -100, -769, -839, -1009, -859},  

7810 { -100, -769, -839, -1009, -859},  

7811 { -100, -769, -839, -1009, -859},  

7812 { -100, -769, -839, -1009, -859}},  

7813 /* UG.AC.. @ */  

7814 {{ -100, -769, -839, -1009, -859},  

7815 { -100, -769, -839, -1009, -859},  

7816 { -100, -769, -839, -1009, -859},  

7817 { -100, -769, -839, -1009, -859},  

7818 { -100, -769, -839, -1009, -859}},  

7819 /* UG.AG.. @ */  

7820 {{ -100, -769, -839, -1009, -859},  

7821 { -100, -769, -839, -1009, -859},  

7822 { -100, -769, -839, -1009, -859},  

7823 { -100, -769, -839, -1009, -859},  

7824 { -100, -769, -839, -1009, -859}},  

7825 /* UG.AU.. @ */  

7826 {{ -100, -769, -839, -1009, -859},  

7827 { -100, -769, -839, -1009, -859},  

7828 { -100, -769, -839, -1009, -859},  

7829 { -100, -769, -839, -1009, -859},  

7830 { -100, -769, -839, -1009, -859}}},  

7831 /* UG.C@.. @ */  

7832 {{ -100, -529, -529, -409, -529},  

7833 { -100, -529, -529, -409, -529},  

7834 { -100, -529, -529, -409, -529},  

7835 { -100, -529, -529, -409, -529},  

7836 { -100, -529, -529, -409, -529}},  

7837 /* UG.CA.. @ */  

7838 {{ -100, -529, -529, -409, -529},  

7839 { -100, -529, -529, -409, -529},  

7840 { -100, -529, -529, -409, -529},  

7841 { -100, -529, -529, -409, -529},  

7842 { -100, -529, -529, -409, -529}},  

7843 /* UG.CC.. @ */  

7844 {{ -100, -529, -529, -409, -529},  

7845 { -100, -529, -529, -409, -529},  

7846 { -100, -529, -529, -409, -529},  

7847 { -100, -529, -529, -409, -529},  

7848 { -100, -529, -529, -409, -529}},  

7849 /* UG.CG.. @ */  

7850 {{ -100, -529, -529, -409, -529},  

7851 { -100, -529, -529, -409, -529},  

7852 { -100, -529, -529, -409, -529},  

7853 { -100, -529, -529, -409, -529},  

7854 { -100, -529, -529, -409, -529}},  

7855 /* UG.CU.. @ */  

7856 {{ -100, -529, -529, -409, -529},

```

```

7857 { -100, -529, -529, -409, -529},
7858 { -100, -529, -529, -409, -529},
7859 { -100, -529, -529, -409, -529},
7860 { -100, -529, -529, -409, -529} } },
7861 /* UG.G@.. @ */
7862 {{ -100, -709, -859, -969, -859},
7863 { -100, -709, -859, -969, -859},
7864 { -100, -709, -859, -969, -859},
7865 { -100, -709, -859, -969, -859},
7866 { -100, -709, -859, -969, -859} } },
7867 /* UG.GA.. @ */
7868 {{ -100, -709, -859, -969, -859},
7869 { -100, -709, -859, -969, -859},
7870 { -100, -709, -859, -969, -859},
7871 { -100, -709, -859, -969, -859},
7872 { -100, -709, -859, -969, -859} } },
7873 /* UG.GC.. @ */
7874 {{ -100, -709, -859, -969, -859},
7875 { -100, -709, -859, -969, -859},
7876 { -100, -709, -859, -969, -859},
7877 { -100, -709, -859, -969, -859},
7878 { -100, -709, -859, -969, -859} } },
7879 /* UG.GG.. @ */
7880 {{ -100, -709, -859, -969, -859},
7881 { -100, -709, -859, -969, -859},
7882 { -100, -709, -859, -969, -859},
7883 { -100, -709, -859, -969, -859},
7884 { -100, -709, -859, -969, -859} } },
7885 /* UG.GU.. @ */
7886 {{ -100, -709, -859, -969, -859},
7887 { -100, -709, -859, -969, -859},
7888 { -100, -709, -859, -969, -859},
7889 { -100, -709, -859, -969, -859},
7890 { -100, -709, -859, -969, -859} } },
7891 /* UG.U@.. @ */
7892 {{ -100, -599, -489, -599, -409},
7893 { -100, -599, -489, -599, -409},
7894 { -100, -599, -489, -599, -409},
7895 { -100, -599, -489, -599, -409},
7896 { -100, -599, -489, -599, -409} } },
7897 /* UG.UA.. @ */
7898 {{ -100, -599, -489, -599, -409},
7899 { -100, -599, -489, -599, -409},
7900 { -100, -599, -489, -599, -409},
7901 { -100, -599, -489, -599, -409},
7902 { -100, -599, -489, -599, -409} } },
7903 /* UG.UC.. @ */
7904 {{ -100, -599, -489, -599, -409},
7905 { -100, -599, -489, -599, -409},
7906 { -100, -599, -489, -599, -409},
7907 { -100, -599, -489, -599, -409},
7908 { -100, -599, -489, -599, -409} } },
7909 /* UG.UG.. @ */
7910 {{ -100, -599, -489, -599, -409},
7911 { -100, -599, -489, -599, -409},
7912 { -100, -599, -489, -599, -409},
7913 { -100, -599, -489, -599, -409},
7914 { -100, -599, -489, -599, -409} } },
7915 /* UG.UU.. @ */
7916 {{ -100, -599, -489, -599, -409},
7917 { -100, -599, -489, -599, -409},
7918 { -100, -599, -489, -599, -409},
7919 { -100, -599, -489, -599, -409},
7920 { -100, -599, -489, -599, -409} } },
7921 { /* noPair */ {{{0}}}},
7922 /* AU.@@..CG */
7923 {{{ 0, 0, 0, 0, 0 },
7924 { DEF, DEF, DEF, DEF, DEF },
7925 { DEF, DEF, DEF, DEF, DEF },
7926 { DEF, DEF, DEF, DEF, DEF },
7927 { DEF, DEF, DEF, DEF, DEF } },
7928 /* AU.@A..CG */
7929 {{ 0, 0, 0, 0, 0 },
7930 {-1029,-1029,-1029,-1029,-1029},
7931 { -519, -519, -519, -519, -519 },
7932 { -939, -939, -939, -939, -939 },
7933 { -809, -809, -809, -809, -809 },
7934 /* AU.@C..CG */
7935 {{ 0, 0, 0, 0, 0 },
7936 { -949, -949, -949, -949, -949 },
7937 { -449, -449, -449, -449, -449 },
7938 { -939, -939, -939, -939, -939 },
7939 { -739, -739, -739, -739, -739 } },
7940 /* AU.@G..CG */
7941 {{ 0, 0, 0, 0, 0 },
7942 {-1029,-1029,-1029,-1029,-1029},
7943 { -519, -519, -519, -519, -519 },

```

```

7944 { -939, -939, -939, -939, -939},
7945 { -809, -809, -809, -809, -809}},
7946 /* AU.@..CG */
7947 {{ 0, 0, 0, 0, 0},
7948 {-1029,-1029,-1029,-1029,-1029},
7949 { -669, -669, -669, -669, -669},
7950 { -939, -939, -939, -939, -939},
7951 { -859, -859, -859, -859, -859}},
7952 /* AU.A@..CG */
7953 {{ DEF, -429, -599, -599, -599},
7954 { -100, -479, -649, -649, -649},
7955 { -100, -479, -649, -649, -649},
7956 { -100, -479, -649, -649, -649},
7957 { -100, -479, -649, -649, -649}},
7958 /* AU.AA..CG */
7959 {{ DEF, -429, -599, -599, -599},
7960 {-1079,-1458,-1628,-1628,-1628},
7961 { -569, -948,-1118,-1118,-1118},
7962 { -989,-1368,-1538,-1538,-1538},
7963 { -859,-1238,-1408,-1408,-1408}},
7964 /* AU.AC..CG */
7965 {{ DEF, -429, -599, -599, -599},
7966 { -999,-1378,-1548,-1548,-1548},
7967 { -499, -878,-1048,-1048,-1048},
7968 { -989,-1368,-1538,-1538,-1538},
7969 { -789,-1168,-1338,-1338,-1338}},
7970 /* AU.AG..CG */
7971 {{ DEF, -429, -599, -599, -599},
7972 {-1079,-1458,-1628,-1628,-1628},
7973 { -569, -948,-1118,-1118,-1118},
7974 { -989,-1368,-1538,-1538,-1538},
7975 { -859,-1238,-1408,-1408,-1408}},
7976 /* AU.AU..CG */
7977 {{ DEF, -429, -599, -599, -599},
7978 {-1079,-1458,-1628,-1628,-1628},
7979 { -719,-1098,-1268,-1268,-1268},
7980 { -989,-1368,-1538,-1538,-1538},
7981 { -909,-1288,-1458,-1458,-1458}},
7982 /* AU.C@..CG */
7983 {{ DEF, -259, -239, -239, -239},
7984 { -100, -309, -289, -289, -289},
7985 { -100, -309, -289, -289, -289},
7986 { -100, -309, -289, -289, -289},
7987 { -100, -309, -289, -289, -289}},
7988 /* AU.CA..CG */
7989 {{ DEF, -259, -239, -239, -239},
7990 {-1079,-1288,-1268,-1268,-1268},
7991 { -569, -778, -758, -758, -758},
7992 { -989,-1198,-1178,-1178,-1178},
7993 { -859,-1068,-1048,-1048,-1048}},
7994 /* AU.CC..CG */
7995 {{ DEF, -259, -239, -239, -239},
7996 { -999,-1208,-1188,-1188,-1188},
7997 { -499, -708, -688, -688, -688},
7998 { -989,-1198,-1178,-1178,-1178},
7999 { -789, -998, -978, -978, -978}},
8000 /* AU.CG..CG */
8001 {{ DEF, -259, -239, -239, -239},
8002 {-1079,-1288,-1268,-1268,-1268},
8003 { -569, -778, -758, -758, -758},
8004 { -989,-1198,-1178,-1178,-1178},
8005 { -859,-1068,-1048,-1048,-1048}},
8006 /* AU.CU..CG */
8007 {{ DEF, -259, -239, -239, -239},
8008 {-1079,-1288,-1268,-1268,-1268},
8009 { -719, -928, -908, -908, -908},
8010 { -989,-1198,-1178,-1178,-1178},
8011 { -909,-1118,-1098,-1098,-1098}},
8012 /* AU.G@..CG */
8013 {{ DEF, -339, -689, -689, -689},
8014 { -100, -389, -739, -739, -739},
8015 { -100, -389, -739, -739, -739},
8016 { -100, -389, -739, -739, -739},
8017 { -100, -389, -739, -739, -739}},
8018 /* AU.GA..CG */
8019 {{ DEF, -339, -689, -689, -689},
8020 {-1079,-1368,-1718,-1718,-1718},
8021 { -569, -858,-1208,-1208,-1208},
8022 { -989,-1278,-1628,-1628,-1628},
8023 { -859,-1148,-1498,-1498,-1498}},
8024 /* AU.GC..CG */
8025 {{ DEF, -339, -689, -689, -689},
8026 { -999,-1288,-1638,-1638,-1638},
8027 { -499, -788,-1138,-1138,-1138},
8028 { -989,-1278,-1628,-1628,-1628},
8029 { -789,-1078,-1428,-1428,-1428}},
8030 /* AU.GG..CG */

```

```

8031 {{ DEF, -339, -689, -689, -689},
8032 {-1079,-1368,-1718,-1718,-1718},
8033 { -569, -858, -1208, -1208, -1208},
8034 { -989, -1278, -1628, -1628, -1628},
8035 { -859, -1148, -1498, -1498, -1498}},
8036 /* AU.GU..CG */
8037 {{ DEF, -339, -689, -689, -689},
8038 {-1079,-1368,-1718,-1718,-1718},
8039 { -719, -1008, -1358, -1358, -1358},
8040 { -989, -1278, -1628, -1628, -1628},
8041 { -909, -1198, -1548, -1548, -1548}}),
8042 /* AU.U@..CG */
8043 {{ DEF, -329, -329, -329, -329},
8044 { -100, -379, -379, -379, -379},
8045 { -100, -379, -379, -379, -379},
8046 { -100, -379, -379, -379, -379},
8047 { -100, -379, -379, -379, -379}},
8048 /* AU.UA..CG */
8049 {{ DEF, -329, -329, -329, -329},
8050 {-1079,-1358,-1358,-1358,-1358},
8051 { -569, -848, -848, -848, -848},
8052 { -989, -1268, -1268, -1268, -1268},
8053 { -859, -1138, -1138, -1138, -1138}},
8054 /* AU.UC..CG */
8055 {{ DEF, -329, -329, -329, -329},
8056 { -999, -1278, -1278, -1278, -1278},
8057 { -499, -778, -778, -778, -778},
8058 { -989, -1268, -1268, -1268, -1268},
8059 { -789, -1068, -1068, -1068, -1068}},
8060 /* AU.UG..CG */
8061 {{ DEF, -329, -329, -329, -329},
8062 {-1079,-1358,-1358,-1358,-1358},
8063 { -569, -848, -848, -848, -848},
8064 { -989, -1268, -1268, -1268, -1268},
8065 { -859, -1138, -1138, -1138, -1138}},
8066 /* AU.UU..CG */
8067 {{ DEF, -329, -329, -329, -329},
8068 {-1079,-1358,-1358,-1358,-1358},
8069 { -719, -998, -998, -998, -998},
8070 { -989, -1268, -1268, -1268, -1268},
8071 { -909, -1188, -1188, -1188, -1188}}),
8072 /* AU.@@..GC */
8073 {{{ 0, 0, 0, 0, 0},
8074 { DEF, DEF, DEF, DEF, DEF},
8075 { DEF, DEF, DEF, DEF, DEF},
8076 { DEF, DEF, DEF, DEF, DEF},
8077 { DEF, DEF, DEF, DEF, DEF}},
8078 /* AU.@A..GC */
8079 {{ 0, 0, 0, 0, 0},
8080 { -519, -519, -519, -519, -519},
8081 { -719, -719, -719, -719, -719},
8082 { -709, -709, -709, -709, -709},
8083 { -499, -499, -499, -499, -499}},
8084 /* AU.@C..GC */
8085 {{ 0, 0, 0, 0, 0},
8086 { -879, -879, -879, -879, -879},
8087 { -309, -309, -309, -309, -309},
8088 { -739, -739, -739, -739, -739},
8089 { -499, -499, -499, -499, -499}},
8090 /* AU.@G..GC */
8091 {{ 0, 0, 0, 0, 0},
8092 { -559, -559, -559, -559, -559},
8093 { -309, -309, -309, -309, -309},
8094 { -619, -619, -619, -619, -619},
8095 { -499, -499, -499, -499, -499}},
8096 /* AU.@U..GC */
8097 {{ 0, 0, 0, 0, 0},
8098 { -879, -879, -879, -879, -879},
8099 { -389, -389, -389, -389, -389},
8100 { -739, -739, -739, -739, -739},
8101 { -569, -569, -569, -569, -569}}),
8102 /* AU.A@..GC */
8103 {{ DEF, -429, -599, -599, -599},
8104 { -100, -479, -649, -649, -649},
8105 { -100, -479, -649, -649, -649},
8106 { -100, -479, -649, -649, -649},
8107 { -100, -479, -649, -649, -649}},
8108 /* AU.AA..GC */
8109 {{ DEF, -429, -599, -599, -599},
8110 { -569, -948, -1118, -1118, -1118},
8111 { -769, -1148, -1318, -1318, -1318},
8112 { -759, -1138, -1308, -1308, -1308},
8113 { -549, -928, -1098, -1098, -1098}},
8114 /* AU.AC..GC */
8115 {{ DEF, -429, -599, -599, -599},
8116 { -929, -1308, -1478, -1478, -1478},
8117 { -359, -738, -908, -908, -908},

```

```
8118 { -789, -1168, -1338, -1338, -1338},  
8119 { -549, -928, -1098, -1098, -1098}},  
8120 /* AU.AG..GC */  
8121 {{ DEF, -429, -599, -599, -599},  
8122 { -609, -988, -1158, -1158, -1158},  
8123 { -359, -738, -908, -908, -908},  
8124 { -669, -1048, -1218, -1218, -1218},  
8125 { -549, -928, -1098, -1098, -1098}},  
8126 /* AU.AU..GC */  
8127 {{ DEF, -429, -599, -599, -599},  
8128 { -929, -1308, -1478, -1478, -1478},  
8129 { -439, -818, -988, -988, -988},  
8130 { -789, -1168, -1338, -1338, -1338},  
8131 { -619, -998, -1168, -1168, -1168}}},  
8132 /* AU.C@..GC */  
8133 {{ DEF, -259, -239, -239, -239},  
8134 { -100, -309, -289, -289, -289},  
8135 { -100, -309, -289, -289, -289},  
8136 { -100, -309, -289, -289, -289},  
8137 { -100, -309, -289, -289, -289}}},  
8138 /* AU.CA..GC */  
8139 {{ DEF, -259, -239, -239, -239},  
8140 { -569, -778, -758, -758, -758},  
8141 { -769, -978, -958, -958, -958},  
8142 { -759, -968, -948, -948, -948},  
8143 { -549, -758, -738, -738, -738}}},  
8144 /* AU.CC..GC */  
8145 {{ DEF, -259, -239, -239, -239},  
8146 { -929, -1138, -1118, -1118, -1118},  
8147 { -359, -568, -548, -548, -548},  
8148 { -789, -998, -978, -978, -978},  
8149 { -549, -758, -738, -738, -738}}},  
8150 /* AU.CG..GC */  
8151 {{ DEF, -259, -239, -239, -239},  
8152 { -609, -818, -798, -798, -798},  
8153 { -359, -568, -548, -548, -548},  
8154 { -669, -878, -858, -858, -858},  
8155 { -549, -758, -738, -738, -738}}},  
8156 /* AU.CU..GC */  
8157 {{ DEF, -259, -239, -239, -239},  
8158 { -929, -1138, -1118, -1118, -1118},  
8159 { -439, -648, -628, -628, -628},  
8160 { -789, -998, -978, -978, -978},  
8161 { -619, -828, -808, -808, -808}}},  
8162 /* AU.G@..GC */  
8163 {{ DEF, -339, -689, -689, -689},  
8164 { -100, -389, -739, -739, -739},  
8165 { -100, -389, -739, -739, -739},  
8166 { -100, -389, -739, -739, -739},  
8167 { -100, -389, -739, -739, -739}}},  
8168 /* AU.GA..GC */  
8169 {{ DEF, -339, -689, -689, -689},  
8170 { -569, -858, -1208, -1208, -1208},  
8171 { -769, -1058, -1408, -1408, -1408},  
8172 { -759, -1048, -1398, -1398, -1398},  
8173 { -549, -838, -1188, -1188, -1188}}},  
8174 /* AU.GC..GC */  
8175 {{ DEF, -339, -689, -689, -689},  
8176 { -929, -1218, -1568, -1568, -1568},  
8177 { -359, -648, -998, -998, -998},  
8178 { -789, -1078, -1428, -1428, -1428},  
8179 { -549, -838, -1188, -1188, -1188}}},  
8180 /* AU.GG..GC */  
8181 {{ DEF, -339, -689, -689, -689},  
8182 { -609, -898, -1248, -1248, -1248},  
8183 { -359, -648, -998, -998, -998},  
8184 { -669, -958, -1308, -1308, -1308},  
8185 { -549, -838, -1188, -1188, -1188}}},  
8186 /* AU.GU..GC */  
8187 {{ DEF, -339, -689, -689, -689},  
8188 { -929, -1218, -1568, -1568, -1568},  
8189 { -439, -728, -1078, -1078, -1078},  
8190 { -789, -1078, -1428, -1428, -1428},  
8191 { -619, -908, -1258, -1258, -1258}}},  
8192 /* AU.U@..GC */  
8193 {{ DEF, -329, -329, -329, -329},  
8194 { -100, -379, -379, -379, -379},  
8195 { -100, -379, -379, -379, -379},  
8196 { -100, -379, -379, -379, -379},  
8197 { -100, -379, -379, -379, -379}}},  
8198 /* AU.UA..GC */  
8199 {{ DEF, -329, -329, -329, -329},  
8200 { -569, -848, -848, -848, -848},  
8201 { -769, -1048, -1048, -1048, -1048},  
8202 { -759, -1038, -1038, -1038, -1038},  
8203 { -549, -828, -828, -828, -828}}},  
8204 /* AU.UC..GC */
```

```

8205 {{ DEF, -329, -329, -329, -329, -329},
8206 { -929,-1208,-1208,-1208,-1208,-1208},
8207 { -359, -638, -638, -638, -638, -638},
8208 { -789,-1068,-1068,-1068,-1068,-1068},
8209 { -549, -828, -828, -828, -828, -828}},
8210 /* AU.UG..GC */
8211 {{ DEF, -329, -329, -329, -329, -329},
8212 { -609, -888, -888, -888, -888, -888},
8213 { -359, -638, -638, -638, -638, -638},
8214 { -669, -948, -948, -948, -948, -948},
8215 { -549, -828, -828, -828, -828, -828}},
8216 /* AU.UU..GC */
8217 {{ DEF, -329, -329, -329, -329, -329},
8218 { -929,-1208,-1208,-1208,-1208,-1208},
8219 { -439, -718, -718, -718, -718, -718},
8220 { -789,-1068,-1068,-1068,-1068,-1068},
8221 { -619, -898, -898, -898, -898, -898}}}},
8222 /* AU.@@..GU */
8223 {{{ 0, 0, 0, 0, 0, 0},
8224 { DEF, DEF, DEF, DEF, DEF, DEF},
8225 { DEF, DEF, DEF, DEF, DEF, DEF},
8226 { DEF, DEF, DEF, DEF, DEF, DEF},
8227 { DEF, DEF, DEF, DEF, DEF, DEF}},
8228 /* AU.@A..GU */
8229 {{ 0, 0, 0, 0, 0, 0},
8230 { -429, -429, -429, -429, -429, -429},
8231 { -259, -259, -259, -259, -259, -259},
8232 { -339, -339, -339, -339, -339, -339},
8233 { -329, -329, -329, -329, -329, -329}},
8234 /* AU.@C..GU */
8235 {{ 0, 0, 0, 0, 0, 0},
8236 { -599, -599, -599, -599, -599, -599},
8237 { -239, -239, -239, -239, -239, -239},
8238 { -689, -689, -689, -689, -689, -689},
8239 { -329, -329, -329, -329, -329, -329}},
8240 /* AU.@G..GU */
8241 {{ 0, 0, 0, 0, 0, 0},
8242 { -599, -599, -599, -599, -599, -599},
8243 { -239, -239, -239, -239, -239, -239},
8244 { -689, -689, -689, -689, -689, -689},
8245 { -329, -329, -329, -329, -329, -329}},
8246 /* AU.@U..GU */
8247 {{ 0, 0, 0, 0, 0, 0},
8248 { -599, -599, -599, -599, -599, -599},
8249 { -239, -239, -239, -239, -239, -239},
8250 { -689, -689, -689, -689, -689, -689},
8251 { -329, -329, -329, -329, -329, -329}}},
8252 /* AU.A@..GU */
8253 {{{ DEF, -429, -599, -599, -599, -599},
8254 { -100, -479, -649, -649, -649, -649},
8255 { -100, -479, -649, -649, -649, -649},
8256 { -100, -479, -649, -649, -649, -649},
8257 { -100, -479, -649, -649, -649, -649}},
8258 /* AU.AA..GU */
8259 {{ DEF, -429, -599, -599, -599, -599},
8260 { -479, -858,-1028,-1028,-1028,-1028},
8261 { -309, -688, -858, -858, -858, -858},
8262 { -389, -768, -938, -938, -938, -938},
8263 { -379, -758, -928, -928, -928, -928}},
8264 /* AU.AC..GU */
8265 {{ DEF, -429, -599, -599, -599, -599},
8266 { -649,-1028,-1198,-1198,-1198,-1198},
8267 { -289, -668, -838, -838, -838, -838},
8268 { -739,-1118,-1288,-1288,-1288,-1288},
8269 { -379, -758, -928, -928, -928, -928}},
8270 /* AU.AG..GU */
8271 {{ DEF, -429, -599, -599, -599, -599},
8272 { -649,-1028,-1198,-1198,-1198,-1198},
8273 { -289, -668, -838, -838, -838, -838},
8274 { -739,-1118,-1288,-1288,-1288,-1288},
8275 { -379, -758, -928, -928, -928, -928}},
8276 /* AU.AU..GU */
8277 {{ DEF, -429, -599, -599, -599, -599},
8278 { -649,-1028,-1198,-1198,-1198,-1198},
8279 { -289, -668, -838, -838, -838, -838},
8280 { -739,-1118,-1288,-1288,-1288,-1288},
8281 { -379, -758, -928, -928, -928, -928}}},
8282 /* AU.C@..GU */
8283 {{{ DEF, -259, -239, -239, -239}, 
8284 { -100, -309, -289, -289, -289}, 
8285 { -100, -309, -289, -289, -289}, 
8286 { -100, -309, -289, -289, -289}, 
8287 { -100, -309, -289, -289, -289}}, 
8288 /* AU.CA..GU */
8289 {{ DEF, -259, -239, -239, -239}, 
8290 { -479, -688, -668, -668, -668}, 
8291 { -309, -518, -498, -498, -498}}, 

```

```

8292 { -389, -598, -578, -578, -578},  

8293 { -379, -588, -568, -568, -568}},  

8294 /* AU.CC..GU */  

8295 {{ DEF, -259, -239, -239, -239},  

8296 { -649, -858, -838, -838, -838},  

8297 { -289, -498, -478, -478, -478},  

8298 { -739, -948, -928, -928, -928},  

8299 { -379, -588, -568, -568, -568}},  

8300 /* AU.CG..GU */  

8301 {{ DEF, -259, -239, -239, -239},  

8302 { -649, -858, -838, -838, -838},  

8303 { -289, -498, -478, -478, -478},  

8304 { -739, -948, -928, -928, -928},  

8305 { -379, -588, -568, -568, -568}},  

8306 /* AU.CU..GU */  

8307 {{ DEF, -259, -239, -239, -239},  

8308 { -649, -858, -838, -838, -838},  

8309 { -289, -498, -478, -478, -478},  

8310 { -739, -948, -928, -928, -928},  

8311 { -379, -588, -568, -568, -568}}},  

8312 /* AU.G@..GU */  

8313 {{{ DEF, -339, -689, -689, -689},  

8314 { -100, -389, -739, -739, -739},  

8315 { -100, -389, -739, -739, -739},  

8316 { -100, -389, -739, -739, -739},  

8317 { -100, -389, -739, -739, -739}}},  

8318 /* AU.GA..GU */  

8319 {{ DEF, -339, -689, -689, -689},  

8320 { -479, -768,-1118,-1118,-1118},  

8321 { -309, -598, -948, -948, -948},  

8322 { -389, -678,-1028,-1028,-1028},  

8323 { -379, -668,-1018,-1018,-1018}},  

8324 /* AU.GC..GU */  

8325 {{ DEF, -339, -689, -689, -689},  

8326 { -649, -938,-1288,-1288,-1288},  

8327 { -289, -578, -928, -928, -928},  

8328 { -739,-1028,-1378,-1378,-1378},  

8329 { -379, -668,-1018,-1018,-1018}},  

8330 /* AU.GG..GU */  

8331 {{ DEF, -339, -689, -689, -689},  

8332 { -649, -938,-1288,-1288,-1288},  

8333 { -289, -578, -928, -928, -928},  

8334 { -739,-1028,-1378,-1378,-1378},  

8335 { -379, -668,-1018,-1018,-1018}}},  

8336 /* AU.GU..GU */  

8337 {{ DEF, -339, -689, -689, -689},  

8338 { -649, -938,-1288,-1288,-1288},  

8339 { -289, -578, -928, -928, -928},  

8340 { -739,-1028,-1378,-1378,-1378},  

8341 { -379, -668,-1018,-1018,-1018}}},  

8342 /* AU.U@..GU */  

8343 {{{ DEF, -329, -329, -329, -329},  

8344 { -100, -379, -379, -379, -379},  

8345 { -100, -379, -379, -379, -379},  

8346 { -100, -379, -379, -379, -379},  

8347 { -100, -379, -379, -379, -379}}},  

8348 /* AU.UA..GU */  

8349 {{ DEF, -329, -329, -329, -329},  

8350 { -479, -758, -758, -758, -758},  

8351 { -309, -588, -588, -588, -588},  

8352 { -389, -668, -668, -668, -668},  

8353 { -379, -658, -658, -658, -658}},  

8354 /* AU.UC..GU */  

8355 {{ DEF, -329, -329, -329, -329},  

8356 { -649, -928, -928, -928, -928},  

8357 { -289, -568, -568, -568, -568},  

8358 { -739,-1018,-1018,-1018,-1018},  

8359 { -379, -658, -658, -658, -658}}},  

8360 /* AU.UG..GU */  

8361 {{ DEF, -329, -329, -329, -329},  

8362 { -649, -928, -928, -928, -928},  

8363 { -289, -568, -568, -568, -568},  

8364 { -739,-1018,-1018,-1018,-1018},  

8365 { -379, -658, -658, -658, -658}}},  

8366 /* AU.UU..GU */  

8367 {{ DEF, -329, -329, -329, -329},  

8368 { -649, -928, -928, -928, -928},  

8369 { -289, -568, -568, -568, -568},  

8370 { -739,-1018,-1018,-1018,-1018},  

8371 { -379, -658, -658, -658, -658}}}},  

8372 /* AU.@@..UG */  

8373 {{{{ 0, 0, 0, 0, 0},  

8374 { DEF, DEF, DEF, DEF, DEF},  

8375 { DEF, DEF, DEF, DEF, DEF},  

8376 { DEF, DEF, DEF, DEF, DEF},  

8377 { DEF, DEF, DEF, DEF, DEF}}},  

8378 /* AU.@A..UG */

```

```

8379 {{ 0, 0, 0, 0, 0},
8380 { -719, -719, -719, -719, -719},
8381 { -479, -479, -479, -479, -479},
8382 { -659, -659, -659, -659, -659},
8383 { -549, -549, -549, -549, -549}},
8384 /* AU.@C..UG */
8385 {{ 0, 0, 0, 0, 0},
8386 { -789, -789, -789, -789, -789},
8387 { -479, -479, -479, -479, -479},
8388 { -809, -809, -809, -809, -809},
8389 { -439, -439, -439, -439, -439}},
8390 /* AU.@G..UG */
8391 {{ 0, 0, 0, 0, 0},
8392 { -959, -959, -959, -959, -959},
8393 { -359, -359, -359, -359, -359},
8394 { -919, -919, -919, -919, -919},
8395 { -549, -549, -549, -549, -549}},
8396 /* AU.@U..UG */
8397 {{ 0, 0, 0, 0, 0},
8398 { -809, -809, -809, -809, -809},
8399 { -479, -479, -479, -479, -479},
8400 { -809, -809, -809, -809, -809},
8401 { -359, -359, -359, -359, -359}},
8402 /* AU.A@..UG */
8403 {{ DEF, -429, -599, -599, -599},
8404 { -100, -479, -649, -649, -649},
8405 { -100, -479, -649, -649, -649},
8406 { -100, -479, -649, -649, -649},
8407 { -100, -479, -649, -649, -649}},
8408 /* AU.AA..UG */
8409 {{ DEF, -429, -599, -599, -599},
8410 { -769, -1148, -1318, -1318, -1318},
8411 { -529, -908, -1078, -1078, -1078},
8412 { -709, -1088, -1258, -1258, -1258},
8413 { -599, -978, -1148, -1148, -1148}},
8414 /* AU.AC..UG */
8415 {{ DEF, -429, -599, -599, -599},
8416 { -839, -1218, -1388, -1388, -1388},
8417 { -529, -908, -1078, -1078, -1078},
8418 { -859, -1238, -1408, -1408, -1408},
8419 { -489, -868, -1038, -1038, -1038}},
8420 /* AU.AG..UG */
8421 {{ DEF, -429, -599, -599, -599},
8422 { -1009, -1388, -1558, -1558, -1558},
8423 { -409, -788, -958, -958, -958},
8424 { -969, -1348, -1518, -1518, -1518},
8425 { -599, -978, -1148, -1148, -1148}},
8426 /* AU.AU..UG */
8427 {{ DEF, -429, -599, -599, -599},
8428 { -859, -1238, -1408, -1408, -1408},
8429 { -529, -908, -1078, -1078, -1078},
8430 { -859, -1238, -1408, -1408, -1408},
8431 { -409, -788, -958, -958, -958}},
8432 /* AU.C@..UG */
8433 {{ DEF, -259, -239, -239, -239},
8434 { -100, -309, -289, -289, -289},
8435 { -100, -309, -289, -289, -289},
8436 { -100, -309, -289, -289, -289},
8437 { -100, -309, -289, -289, -289}},
8438 /* AU.CA..UG */
8439 {{ DEF, -259, -239, -239, -239},
8440 { -769, -978, -958, -958, -958},
8441 { -529, -738, -718, -718, -718},
8442 { -709, -918, -898, -898, -898},
8443 { -599, -808, -788, -788, -788}},
8444 /* AU.CC..UG */
8445 {{ DEF, -259, -239, -239, -239},
8446 { -839, -1048, -1028, -1028, -1028},
8447 { -529, -738, -718, -718, -718},
8448 { -859, -1068, -1048, -1048, -1048},
8449 { -489, -698, -678, -678, -678}},
8450 /* AU.CG..UG */
8451 {{ DEF, -259, -239, -239, -239},
8452 { -1009, -1218, -1198, -1198, -1198},
8453 { -409, -618, -598, -598, -598},
8454 { -969, -1178, -1158, -1158, -1158},
8455 { -599, -808, -788, -788, -788}},
8456 /* AU.CU..UG */
8457 {{ DEF, -259, -239, -239, -239},
8458 { -859, -1068, -1048, -1048, -1048},
8459 { -529, -738, -718, -718, -718},
8460 { -859, -1068, -1048, -1048, -1048},
8461 { -409, -618, -598, -598, -598}},
8462 /* AU.G@..UG */
8463 {{ DEF, -339, -689, -689, -689},
8464 { -100, -389, -739, -739, -739},
8465 { -100, -389, -739, -739, -739},

```

```

8466 { -100, -389, -739, -739, -739},
8467 { -100, -389, -739, -739, -739}},
8468 /* AU.GA..UG */
8469 {{ DEF, -339, -689, -689, -689},
8470 { -769,-1058,-1408,-1408,-1408},
8471 { -529, -818,-1168,-1168,-1168},
8472 { -709, -998,-1348,-1348,-1348},
8473 { -599, -888,-1238,-1238,-1238}},
8474 /* AU.GC..UG */
8475 {{ DEF, -339, -689, -689, -689},
8476 { -839,-1128,-1478,-1478,-1478},
8477 { -529, -818,-1168,-1168,-1168},
8478 { -859,-1148,-1498,-1498,-1498},
8479 { -489, -778,-1128,-1128,-1128}},
8480 /* AU.GG..UG */
8481 {{ DEF, -339, -689, -689, -689},
8482 {-1009,-1298,-1648,-1648,-1648},
8483 { -409, -698,-1048,-1048,-1048},
8484 { -969,-1258,-1608,-1608,-1608},
8485 { -599, -888,-1238,-1238,-1238}},
8486 /* AU.GU..UG */
8487 {{ DEF, -339, -689, -689, -689},
8488 { -859,-1148,-1498,-1498,-1498},
8489 { -529, -818,-1168,-1168,-1168},
8490 { -859,-1148,-1498,-1498,-1498},
8491 { -409, -698,-1048,-1048,-1048}}),
8492 /* AU.U@..UG */
8493 {{ DEF, -329, -329, -329, -329},
8494 { -100, -379, -379, -379, -379},
8495 { -100, -379, -379, -379, -379},
8496 { -100, -379, -379, -379, -379},
8497 { -100, -379, -379, -379, -379}},
8498 /* AU.UA..UG */
8499 {{ DEF, -329, -329, -329, -329},
8500 { -769,-1048,-1048,-1048,-1048},
8501 { -529, -808, -808, -808, -808},
8502 { -709, -988, -988, -988, -988},
8503 { -599, -878, -878, -878, -878}},
8504 /* AU.UC..UG */
8505 {{ DEF, -329, -329, -329, -329},
8506 { -839,-1118,-1118,-1118,-1118},
8507 { -529, -808, -808, -808, -808},
8508 { -859,-1138,-1138,-1138,-1138},
8509 { -489, -768, -768, -768, -768}},
8510 /* AU.UG..UG */
8511 {{ DEF, -329, -329, -329, -329},
8512 {-1009,-1288,-1288,-1288,-1288},
8513 { -409, -688, -688, -688, -688},
8514 { -969,-1248,-1248,-1248,-1248},
8515 { -599, -878, -878, -878, -878}},
8516 /* AU.UU..UG */
8517 {{ DEF, -329, -329, -329, -329},
8518 { -859,-1138,-1138,-1138,-1138},
8519 { -529, -808, -808, -808, -808},
8520 { -859,-1138,-1138,-1138,-1138},
8521 { -409, -688, -688, -688, -688}}),
8522 /* AU.@@..AU */
8523 {{{ 0, 0, 0, 0, 0},
8524 { DEF, DEF, DEF, DEF, DEF},
8525 { DEF, DEF, DEF, DEF, DEF},
8526 { DEF, DEF, DEF, DEF, DEF},
8527 { DEF, DEF, DEF, DEF, DEF}},
8528 /* AU.@A..AU */
8529 {{ 0, 0, 0, 0, 0},
8530 { -429, -429, -429, -429, -429},
8531 { -259, -259, -259, -259, -259},
8532 { -339, -339, -339, -339, -339},
8533 { -329, -329, -329, -329, -329}},
8534 /* AU.@C..AU */
8535 {{ 0, 0, 0, 0, 0},
8536 { -599, -599, -599, -599, -599},
8537 { -239, -239, -239, -239, -239},
8538 { -689, -689, -689, -689, -689},
8539 { -329, -329, -329, -329, -329}},
8540 /* AU.@G..AU */
8541 {{ 0, 0, 0, 0, 0},
8542 { -599, -599, -599, -599, -599},
8543 { -239, -239, -239, -239, -239},
8544 { -689, -689, -689, -689, -689},
8545 { -329, -329, -329, -329, -329}},
8546 /* AU.@U..AU */
8547 {{ 0, 0, 0, 0, 0},
8548 { -599, -599, -599, -599, -599},
8549 { -239, -239, -239, -239, -239},
8550 { -689, -689, -689, -689, -689},
8551 { -329, -329, -329, -329, -329}}),
8552 /* AU.A@..AU */

```

```

8553 {{ DEF, -429, -599, -599, -599},
8554 { -100, -479, -649, -649, -649},
8555 { -100, -479, -649, -649, -649},
8556 { -100, -479, -649, -649, -649},
8557 { -100, -479, -649, -649, -649}},
8558 /* AU.AA..AU */
8559 {{ DEF, -429, -599, -599, -599},
8560 { -479, -858,-1028,-1028,-1028},
8561 { -309, -688, -858, -858, -858},
8562 { -389, -768, -938, -938, -938},
8563 { -379, -758, -928, -928, -928}},
8564 /* AU.AC..AU */
8565 {{ DEF, -429, -599, -599, -599},
8566 { -649,-1028,-1198,-1198,-1198},
8567 { -289, -668, -838, -838, -838},
8568 { -739,-1118,-1288,-1288,-1288},
8569 { -379, -758, -928, -928, -928}},
8570 /* AU.AG..AU */
8571 {{ DEF, -429, -599, -599, -599},
8572 { -649,-1028,-1198,-1198,-1198},
8573 { -289, -668, -838, -838, -838},
8574 { -739,-1118,-1288,-1288,-1288},
8575 { -379, -758, -928, -928, -928}},
8576 /* AU.AU..AU */
8577 {{ DEF, -429, -599, -599, -599},
8578 { -649,-1028,-1198,-1198,-1198},
8579 { -289, -668, -838, -838, -838},
8580 { -739,-1118,-1288,-1288,-1288},
8581 { -379, -758, -928, -928, -928}},
8582 /* AU.C@..AU */
8583 {{ DEF, -259, -239, -239, -239},
8584 { -100, -309, -289, -289, -289},
8585 { -100, -309, -289, -289, -289},
8586 { -100, -309, -289, -289, -289},
8587 { -100, -309, -289, -289, -289}},
8588 /* AU.CA..AU */
8589 {{ DEF, -259, -239, -239, -239},
8590 { -479, -688, -668, -668, -668},
8591 { -309, -518, -498, -498, -498},
8592 { -389, -598, -578, -578, -578},
8593 { -379, -588, -568, -568, -568}},
8594 /* AU.CC..AU */
8595 {{ DEF, -259, -239, -239, -239},
8596 { -649, -858, -838, -838, -838},
8597 { -289, -498, -478, -478, -478},
8598 { -739, -948, -928, -928, -928},
8599 { -379, -588, -568, -568, -568}},
8600 /* AU.CG..AU */
8601 {{ DEF, -259, -239, -239, -239},
8602 { -649, -858, -838, -838, -838},
8603 { -289, -498, -478, -478, -478},
8604 { -739, -948, -928, -928, -928},
8605 { -379, -588, -568, -568, -568}},
8606 /* AU.CU..AU */
8607 {{ DEF, -259, -239, -239, -239},
8608 { -649, -858, -838, -838, -838},
8609 { -289, -498, -478, -478, -478},
8610 { -739, -948, -928, -928, -928},
8611 { -379, -588, -568, -568, -568}},
8612 /* AU.G@..AU */
8613 {{ DEF, -339, -689, -689, -689},
8614 { -100, -389, -739, -739, -739},
8615 { -100, -389, -739, -739, -739},
8616 { -100, -389, -739, -739, -739},
8617 { -100, -389, -739, -739, -739}},
8618 /* AU.GA..AU */
8619 {{ DEF, -339, -689, -689, -689},
8620 { -479, -768,-1118,-1118,-1118},
8621 { -309, -598, -948, -948, -948},
8622 { -389, -678,-1028,-1028,-1028},
8623 { -379, -668,-1018,-1018,-1018}},
8624 /* AU.GC..AU */
8625 {{ DEF, -339, -689, -689, -689},
8626 { -649, -938,-1288,-1288,-1288},
8627 { -289, -578, -928, -928, -928},
8628 { -739,-1028,-1378,-1378,-1378},
8629 { -379, -668,-1018,-1018,-1018}},
8630 /* AU.GG..AU */
8631 {{ DEF, -339, -689, -689, -689},
8632 { -649, -938,-1288,-1288,-1288},
8633 { -289, -578, -928, -928, -928},
8634 { -739,-1028,-1378,-1378,-1378},
8635 { -379, -668,-1018,-1018,-1018}},
8636 /* AU.GU..AU */
8637 {{ DEF, -339, -689, -689, -689},
8638 { -649, -938,-1288,-1288,-1288},
8639 { -289, -578, -928, -928, -928}},

```

```

8640 { -739, -1028, -1378, -1378, -1378},
8641 { -379, -668, -1018, -1018, -1018} } },
8642 /* AU.U@..AU */
8643 {{ DEF, -329, -329, -329, -329},
8644 { -100, -379, -379, -379, -379},
8645 { -100, -379, -379, -379, -379},
8646 { -100, -379, -379, -379, -379},
8647 { -100, -379, -379, -379, -379} },
8648 /* AU.UA..AU */
8649 {{ DEF, -329, -329, -329, -329},
8650 { -479, -758, -758, -758, -758},
8651 { -309, -588, -588, -588, -588},
8652 { -389, -668, -668, -668, -668},
8653 { -379, -658, -658, -658, -658} },
8654 /* AU.UC..AU */
8655 {{ DEF, -329, -329, -329, -329},
8656 { -649, -928, -928, -928, -928},
8657 { -289, -568, -568, -568, -568},
8658 { -739, -1018, -1018, -1018, -1018},
8659 { -379, -658, -658, -658, -658} },
8660 /* AU.UG..AU */
8661 {{ DEF, -329, -329, -329, -329},
8662 { -649, -928, -928, -928, -928},
8663 { -289, -568, -568, -568, -568},
8664 { -739, -1018, -1018, -1018, -1018},
8665 { -379, -658, -658, -658, -658} },
8666 /* AU.UU..AU */
8667 {{ DEF, -329, -329, -329, -329},
8668 { -649, -928, -928, -928, -928},
8669 { -289, -568, -568, -568, -568},
8670 { -739, -1018, -1018, -1018, -1018},
8671 { -379, -658, -658, -658, -658} } },
8672 /* AU.@@..UA */
8673 {{{ 0, 0, 0, 0, 0},
8674 { DEF, DEF, DEF, DEF, DEF},
8675 { DEF, DEF, DEF, DEF, DEF},
8676 { DEF, DEF, DEF, DEF, DEF},
8677 { DEF, DEF, DEF, DEF, DEF} },
8678 /* AU.@A..UA */
8679 {{ 0, 0, 0, 0, 0},
8680 { -399, -399, -399, -399, -399},
8681 { -429, -429, -429, -429, -429},
8682 { -379, -379, -379, -379, -379},
8683 { -279, -279, -279, -279, -279} },
8684 /* AU.@C..UA */
8685 {{ 0, 0, 0, 0, 0},
8686 { -629, -629, -629, -629, -629},
8687 { -509, -509, -509, -509, -509},
8688 { -679, -679, -679, -679, -679},
8689 { -139, -139, -139, -139, -139} },
8690 /* AU.@G..UA */
8691 {{ 0, 0, 0, 0, 0},
8692 { -889, -889, -889, -889, -889},
8693 { -199, -199, -199, -199, -199},
8694 { -889, -889, -889, -889, -889},
8695 { -279, -279, -279, -279, -279} },
8696 /* AU.@U..UA */
8697 {{ 0, 0, 0, 0, 0},
8698 { -589, -589, -589, -589, -589},
8699 { -179, -179, -179, -179, -179},
8700 { -679, -679, -679, -679, -679},
8701 { -140, -140, -140, -140, -140} } },
8702 /* AU.A@..UA */
8703 {{ DEF, -429, -599, -599, -599},
8704 { -100, -479, -649, -649, -649},
8705 { -100, -479, -649, -649, -649},
8706 { -100, -479, -649, -649, -649},
8707 { -100, -479, -649, -649, -649} },
8708 /* AU.AA..UA */
8709 {{ DEF, -429, -599, -599, -599},
8710 { -449, -828, -998, -998, -998},
8711 { -479, -858, -1028, -1028, -1028},
8712 { -429, -808, -978, -978, -978},
8713 { -329, -708, -878, -878, -878} },
8714 /* AU.AC..UA */
8715 {{ DEF, -429, -599, -599, -599},
8716 { -679, -1058, -1228, -1228, -1228},
8717 { -559, -938, -1108, -1108, -1108},
8718 { -729, -1108, -1278, -1278, -1278},
8719 { -189, -568, -738, -738, -738} },
8720 /* AU.AG..UA */
8721 {{ DEF, -429, -599, -599, -599},
8722 { -939, -1318, -1488, -1488, -1488},
8723 { -249, -628, -798, -798, -798},
8724 { -939, -1318, -1488, -1488, -1488},
8725 { -329, -708, -878, -878, -878} },
8726 /* AU.AU..UA */

```

```

8727 {{ DEF, -429, -599, -599, -599},
8728 { -639,-1018,-1188,-1188,-1188},
8729 { -229, -608, -778, -778, -778},
8730 { -729,-1108,-1278,-1278,-1278},
8731 { -190, -569, -739, -739, -739}},
8732 /* AU.C@..UA */
8733 {{ DEF, -259, -239, -239, -239},
8734 { -100, -309, -289, -289, -289},
8735 { -100, -309, -289, -289, -289},
8736 { -100, -309, -289, -289, -289},
8737 { -100, -309, -289, -289, -289}},
8738 /* AU.CA..UA */
8739 {{ DEF, -259, -239, -239, -239},
8740 { -449, -658, -638, -638, -638},
8741 { -479, -688, -668, -668, -668},
8742 { -429, -638, -618, -618, -618},
8743 { -329, -538, -518, -518, -518}},
8744 /* AU.CC..UA */
8745 {{ DEF, -259, -239, -239, -239},
8746 { -679, -888, -868, -868, -868},
8747 { -559, -768, -748, -748, -748},
8748 { -729, -938, -918, -918, -918},
8749 { -189, -398, -378, -378, -378}},
8750 /* AU.CG..UA */
8751 {{ DEF, -259, -239, -239, -239},
8752 { -939,-1148,-1128,-1128,-1128},
8753 { -249, -458, -438, -438, -438},
8754 { -939,-1148,-1128,-1128,-1128},
8755 { -329, -538, -518, -518, -518}},
8756 /* AU.CU..UA */
8757 {{ DEF, -259, -239, -239, -239},
8758 { -639, -848, -828, -828, -828},
8759 { -229, -438, -418, -418, -418},
8760 { -729, -938, -918, -918, -918},
8761 { -190, -399, -379, -379, -379}},
8762 /* AU.G@..UA */
8763 {{ DEF, -339, -689, -689, -689},
8764 { -100, -389, -739, -739, -739},
8765 { -100, -389, -739, -739, -739},
8766 { -100, -389, -739, -739, -739},
8767 { -100, -389, -739, -739, -739}},
8768 /* AU.GA..UA */
8769 {{ DEF, -339, -689, -689, -689},
8770 { -449, -738,-1088,-1088,-1088},
8771 { -479, -768,-1118,-1118,-1118},
8772 { -429, -718,-1068,-1068,-1068},
8773 { -329, -618, -968, -968, -968}},
8774 /* AU.GC..UA */
8775 {{ DEF, -339, -689, -689, -689},
8776 { -679, -968,-1318,-1318,-1318},
8777 { -559, -848,-1198,-1198,-1198},
8778 { -729,-1018,-1368,-1368,-1368},
8779 { -189, -478, -828, -828, -828}},
8780 /* AU.GG..UA */
8781 {{ DEF, -339, -689, -689, -689},
8782 { -939,-1228,-1578,-1578,-1578},
8783 { -249, -538, -888, -888, -888},
8784 { -939,-1228,-1578,-1578,-1578},
8785 { -329, -618, -968, -968, -968}},
8786 /* AU.GU..UA */
8787 {{ DEF, -339, -689, -689, -689},
8788 { -639, -928,-1278,-1278,-1278},
8789 { -229, -518, -868, -868, -868},
8790 { -729,-1018,-1368,-1368,-1368},
8791 { -190, -479, -829, -829, -829}},
8792 /* AU.U@..UA */
8793 {{ DEF, -329, -329, -329, -329},
8794 { -100, -379, -379, -379, -379},
8795 { -100, -379, -379, -379, -379},
8796 { -100, -379, -379, -379, -379},
8797 { -100, -379, -379, -379, -379}},
8798 /* AU.UA..UA */
8799 {{ DEF, -329, -329, -329, -329},
8800 { -449, -728, -728, -728, -728},
8801 { -479, -758, -758, -758, -758},
8802 { -429, -708, -708, -708, -708},
8803 { -329, -608, -608, -608, -608}},
8804 /* AU.UC..UA */
8805 {{ DEF, -329, -329, -329, -329},
8806 { -679, -958, -958, -958, -958},
8807 { -559, -838, -838, -838, -838},
8808 { -729,-1008,-1008,-1008,-1008},
8809 { -189, -468, -468, -468, -468}},
8810 /* AU.UG..UA */
8811 {{ DEF, -329, -329, -329, -329},
8812 { -939,-1218,-1218,-1218,-1218},
8813 { -249, -528, -528, -528, -528},

```

```
8814 { -939, -1218, -1218, -1218, -1218},  
8815 { -329, -608, -608, -608, -608}},  
8816 /* AU.UU..UA */  
8817 {{ DEF, -329, -329, -329, -329},  
8818 { -639, -918, -918, -918, -918},  
8819 { -229, -508, -508, -508, -508},  
8820 { -729, -1008, -1008, -1008, -1008},  
8821 { -190, -469, -469, -469, -469}}}},  
8822 /* AU.@@.. @ */  
8823 {{{ DEF, DEF, DEF, DEF, DEF},  
8824 { DEF, DEF, DEF, DEF, DEF},  
8825 { DEF, DEF, DEF, DEF, DEF},  
8826 { DEF, DEF, DEF, DEF, DEF},  
8827 { DEF, DEF, DEF, DEF, DEF}}},  
8828 /* AU.@A.. @ */  
8829 {{ DEF, DEF, DEF, DEF, DEF},  
8830 { DEF, DEF, DEF, DEF, DEF},  
8831 { DEF, DEF, DEF, DEF, DEF},  
8832 { DEF, DEF, DEF, DEF, DEF},  
8833 { DEF, DEF, DEF, DEF, DEF}}},  
8834 /* AU.@C.. @ */  
8835 {{ DEF, DEF, DEF, DEF, DEF},  
8836 { DEF, DEF, DEF, DEF, DEF},  
8837 { DEF, DEF, DEF, DEF, DEF},  
8838 { DEF, DEF, DEF, DEF, DEF},  
8839 { DEF, DEF, DEF, DEF, DEF}}},  
8840 /* AU.@G.. @ */  
8841 {{ DEF, DEF, DEF, DEF, DEF},  
8842 { DEF, DEF, DEF, DEF, DEF},  
8843 { DEF, DEF, DEF, DEF, DEF},  
8844 { DEF, DEF, DEF, DEF, DEF},  
8845 { DEF, DEF, DEF, DEF, DEF}}},  
8846 /* AU.@U.. @ */  
8847 {{ DEF, DEF, DEF, DEF, DEF},  
8848 { DEF, DEF, DEF, DEF, DEF},  
8849 { DEF, DEF, DEF, DEF, DEF},  
8850 { DEF, DEF, DEF, DEF, DEF},  
8851 { DEF, DEF, DEF, DEF, DEF}}},  
8852 /* AU.A@.. @ */  
8853 {{ -100, -479, -649, -649, -649},  
8854 { -100, -479, -649, -649, -649},  
8855 { -100, -479, -649, -649, -649},  
8856 { -100, -479, -649, -649, -649},  
8857 { -100, -479, -649, -649, -649}},  
8858 /* AU.AA.. @ */  
8859 {{ -100, -479, -649, -649, -649},  
8860 { -100, -479, -649, -649, -649},  
8861 { -100, -479, -649, -649, -649},  
8862 { -100, -479, -649, -649, -649},  
8863 { -100, -479, -649, -649, -649}},  
8864 /* AU.AC.. @ */  
8865 {{ -100, -479, -649, -649, -649},  
8866 { -100, -479, -649, -649, -649},  
8867 { -100, -479, -649, -649, -649},  
8868 { -100, -479, -649, -649, -649},  
8869 { -100, -479, -649, -649, -649}},  
8870 /* AU.AG.. @ */  
8871 {{ -100, -479, -649, -649, -649},  
8872 { -100, -479, -649, -649, -649},  
8873 { -100, -479, -649, -649, -649},  
8874 { -100, -479, -649, -649, -649},  
8875 { -100, -479, -649, -649, -649}},  
8876 /* AU.AU.. @ */  
8877 {{ -100, -479, -649, -649, -649},  
8878 { -100, -479, -649, -649, -649},  
8879 { -100, -479, -649, -649, -649},  
8880 { -100, -479, -649, -649, -649},  
8881 { -100, -479, -649, -649, -649}}},  
8882 /* AU.C@.. @ */  
8883 {{ -100, -309, -289, -289, -289},  
8884 { -100, -309, -289, -289, -289},  
8885 { -100, -309, -289, -289, -289},  
8886 { -100, -309, -289, -289, -289},  
8887 { -100, -309, -289, -289, -289}},  
8888 /* AU.CA.. @ */  
8889 {{ -100, -309, -289, -289, -289},  
8890 { -100, -309, -289, -289, -289},  
8891 { -100, -309, -289, -289, -289},  
8892 { -100, -309, -289, -289, -289},  
8893 { -100, -309, -289, -289, -289}},  
8894 /* AU.CC.. @ */  
8895 {{ -100, -309, -289, -289, -289},  
8896 { -100, -309, -289, -289, -289},  
8897 { -100, -309, -289, -289, -289},  
8898 { -100, -309, -289, -289, -289},  
8899 { -100, -309, -289, -289, -289}},  
8900 /* AU.CG.. @ */
```

```

8901 {{ -100, -309, -289, -289, -289},
8902 { -100, -309, -289, -289, -289},
8903 { -100, -309, -289, -289, -289},
8904 { -100, -309, -289, -289, -289},
8905 { -100, -309, -289, -289, -289}},
8906 /* AU.CU.. @ */
8907 {{ -100, -309, -289, -289, -289},
8908 { -100, -309, -289, -289, -289},
8909 { -100, -309, -289, -289, -289},
8910 { -100, -309, -289, -289, -289},
8911 { -100, -309, -289, -289, -289}}),
8912 /* AU.G@.. @ */
8913 {{ -100, -389, -739, -739, -739},
8914 { -100, -389, -739, -739, -739},
8915 { -100, -389, -739, -739, -739},
8916 { -100, -389, -739, -739, -739},
8917 { -100, -389, -739, -739, -739}}),
8918 /* AU.GA.. @ */
8919 {{ -100, -389, -739, -739, -739},
8920 { -100, -389, -739, -739, -739},
8921 { -100, -389, -739, -739, -739},
8922 { -100, -389, -739, -739, -739},
8923 { -100, -389, -739, -739, -739}}),
8924 /* AU.GC.. @ */
8925 {{ -100, -389, -739, -739, -739},
8926 { -100, -389, -739, -739, -739},
8927 { -100, -389, -739, -739, -739},
8928 { -100, -389, -739, -739, -739},
8929 { -100, -389, -739, -739, -739}}),
8930 /* AU.GG.. @ */
8931 {{ -100, -389, -739, -739, -739},
8932 { -100, -389, -739, -739, -739},
8933 { -100, -389, -739, -739, -739},
8934 { -100, -389, -739, -739, -739},
8935 { -100, -389, -739, -739, -739}}),
8936 /* AU.GU.. @ */
8937 {{ -100, -389, -739, -739, -739},
8938 { -100, -389, -739, -739, -739},
8939 { -100, -389, -739, -739, -739},
8940 { -100, -389, -739, -739, -739},
8941 { -100, -389, -739, -739, -739}}),
8942 /* AU.U@.. @ */
8943 {{ -100, -379, -379, -379, -379},
8944 { -100, -379, -379, -379, -379},
8945 { -100, -379, -379, -379, -379},
8946 { -100, -379, -379, -379, -379},
8947 { -100, -379, -379, -379, -379}}),
8948 /* AU.UA.. @ */
8949 {{ -100, -379, -379, -379, -379},
8950 { -100, -379, -379, -379, -379},
8951 { -100, -379, -379, -379, -379},
8952 { -100, -379, -379, -379, -379},
8953 { -100, -379, -379, -379, -379}}),
8954 /* AU.UC.. @ */
8955 {{ -100, -379, -379, -379, -379},
8956 { -100, -379, -379, -379, -379},
8957 { -100, -379, -379, -379, -379},
8958 { -100, -379, -379, -379, -379},
8959 { -100, -379, -379, -379, -379}}),
8960 /* AU.UG.. @ */
8961 {{ -100, -379, -379, -379, -379},
8962 { -100, -379, -379, -379, -379},
8963 { -100, -379, -379, -379, -379},
8964 { -100, -379, -379, -379, -379},
8965 { -100, -379, -379, -379, -379}}),
8966 /* AU.UU.. @ */
8967 {{ -100, -379, -379, -379, -379},
8968 { -100, -379, -379, -379, -379},
8969 { -100, -379, -379, -379, -379},
8970 { -100, -379, -379, -379, -379},
8971 { -100, -379, -379, -379, -379}}}),
8972 { /* noPair */ {{{{0}}}}},
8973 /* UA.@@..CG */
8974 {{{{ 0, 0, 0, 0, 0 },
8975 { DEF, DEF, DEF, DEF, DEF },
8976 { DEF, DEF, DEF, DEF, DEF },
8977 { DEF, DEF, DEF, DEF, DEF },
8978 { DEF, DEF, DEF, DEF, DEF }},
8979 /* UA.@A..CG */
8980 {{ 0, 0, 0, 0, 0 },
8981 {-1029,-1029,-1029,-1029,-1029},
8982 { -519, -519, -519, -519, -519 },
8983 { -939, -939, -939, -939, -939 },
8984 { -809, -809, -809, -809, -809 } },
8985 /* UA.@C..CG */
8986 {{ 0, 0, 0, 0, 0 },
8987 { -949, -949, -949, -949, -949 } },

```

```

8988 { -449, -449, -449, -449, -449},
8989 { -939, -939, -939, -939, -939},
8990 { -739, -739, -739, -739, -739}},
8991 /* UA.@G..CG */
8992 {{ 0, 0, 0, 0, 0},
8993 {-1029,-1029,-1029,-1029,-1029},
8994 { -519, -519, -519, -519, -519},
8995 { -939, -939, -939, -939, -939},
8996 { -809, -809, -809, -809, -809}},
8997 /* UA.@U..CG */
8998 {{ 0, 0, 0, 0, 0},
8999 {-1029,-1029,-1029,-1029,-1029},
9000 { -669, -669, -669, -669, -669},
9001 { -939, -939, -939, -939, -939},
9002 { -859, -859, -859, -859, -859}},
9003 /* UA.A@..CG */
9004 {{ DEF, -399, -629, -889, -589},
9005 { -100, -449, -679, -939, -639},
9006 { -100, -449, -679, -939, -639},
9007 { -100, -449, -679, -939, -639},
9008 { -100, -449, -679, -939, -639}},
9009 /* UA.AA..CG */
9010 {{ DEF, -399, -629, -889, -589},
9011 {-1079,-1428,-1658,-1918,-1618},
9012 { -569, -918,-1148,-1408,-1108},
9013 { -989,-1338,-1568,-1828,-1528},
9014 { -859,-1208,-1438,-1698,-1398}},
9015 /* UA.AC..CG */
9016 {{ DEF, -399, -629, -889, -589},
9017 { -999,-1348,-1578,-1838,-1538},
9018 { -499, -848,-1078,-1338,-1038},
9019 { -989,-1338,-1568,-1828,-1528},
9020 { -789,-1138,-1368,-1628,-1328}},
9021 /* UA.AG..CG */
9022 {{ DEF, -399, -629, -889, -589},
9023 {-1079,-1428,-1658,-1918,-1618},
9024 { -569, -918,-1148,-1408,-1108},
9025 { -989,-1338,-1568,-1828,-1528},
9026 { -859,-1208,-1438,-1698,-1398}},
9027 /* UA.AU..CG */
9028 {{ DEF, -399, -629, -889, -589},
9029 {-1079,-1428,-1658,-1918,-1618},
9030 { -719,-1068,-1298,-1558,-1258},
9031 { -989,-1338,-1568,-1828,-1528},
9032 { -909,-1258,-1488,-1748,-1448}},
9033 /* UA.C@..CG */
9034 {{ DEF, -429, -509, -199, -179},
9035 { -100, -479, -559, -249, -229},
9036 { -100, -479, -559, -249, -229},
9037 { -100, -479, -559, -249, -229},
9038 { -100, -479, -559, -249, -229}},
9039 /* UA.CA..CG */
9040 {{ DEF, -429, -509, -199, -179},
9041 {-1079,-1458,-1538,-1228,-1208},
9042 { -569, -948,-1028, -718, -698},
9043 { -989,-1368,-1448,-1138,-1118},
9044 { -859,-1238,-1318,-1008, -988}},
9045 /* UA.CC..CG */
9046 {{ DEF, -429, -509, -199, -179},
9047 { -999,-1378,-1458,-1148,-1128},
9048 { -499, -878, -958, -648, -628},
9049 { -989,-1368,-1448,-1138,-1118},
9050 { -789,-1168,-1248, -938, -918}},
9051 /* UA.CG..CG */
9052 {{ DEF, -429, -509, -199, -179},
9053 {-1079,-1458,-1538,-1228,-1208},
9054 { -569, -948,-1028, -718, -698},
9055 { -989,-1368,-1448,-1138,-1118},
9056 { -859,-1238,-1318,-1008, -988}},
9057 /* UA.CU..CG */
9058 {{ DEF, -429, -509, -199, -179},
9059 {-1079,-1458,-1538,-1228,-1208},
9060 { -719,-1098,-1178, -868, -848},
9061 { -989,-1368,-1448,-1138,-1118},
9062 { -909,-1288,-1368,-1058,-1038}},
9063 /* UA.G@..CG */
9064 {{ DEF, -379, -679, -889, -679},
9065 { -100, -429, -729, -939, -729},
9066 { -100, -429, -729, -939, -729},
9067 { -100, -429, -729, -939, -729},
9068 { -100, -429, -729, -939, -729}},
9069 /* UA.GA..CG */
9070 {{ DEF, -379, -679, -889, -679},
9071 {-1079,-1408,-1708,-1918,-1708},
9072 { -569, -898,-1198,-1408,-1198},
9073 { -989,-1318,-1618,-1828,-1618},
9074 { -859,-1188,-1488,-1698,-1488}},

```

```

9075 /* UA.GC..CG */
9076 {{ DEF, -379, -679, -889, -679},
9077 { -999,-1328,-1628,-1838,-1628},
9078 { -499, -828,-1128,-1338,-1128},
9079 { -989,-1318,-1618,-1828,-1618},
9080 { -789,-1118,-1418,-1628,-1418}},
9081 /* UA.GG..CG */
9082 {{ DEF, -379, -679, -889, -679},
9083 {-1079,-1408,-1708,-1918,-1708},
9084 { -569, -898,-1198,-1408,-1198},
9085 { -989,-1318,-1618,-1828,-1618},
9086 { -859,-1188,-1488,-1698,-1488}},
9087 /* UA.GU..CG */
9088 {{ DEF, -379, -679, -889, -679},
9089 {-1079,-1408,-1708,-1918,-1708},
9090 { -719,-1048,-1348,-1558,-1348},
9091 { -989,-1318,-1618,-1828,-1618},
9092 { -909,-1238,-1538,-1748,-1538}}},
9093 /* UA.U@..CG */
9094 {{{{ DEF, -279, -139, -279, -140},
9095 { -100, -329, -189, -329, -190},
9096 { -100, -329, -189, -329, -190},
9097 { -100, -329, -189, -329, -190},
9098 { -100, -329, -189, -329, -190}},
9099 /* UA.UA..CG */
9100 {{ DEF, -279, -139, -279, -140},
9101 {-1079,-1308,-1168,-1308,-1169},
9102 { -569, -798, -658, -798, -659},
9103 { -989,-1218,-1078,-1218,-1079},
9104 { -859,-1088, -948,-1088, -949}}},
9105 /* UA.UC..CG */
9106 {{ DEF, -279, -139, -279, -140},
9107 { -999,-1228,-1088,-1228,-1089},
9108 { -499, -728, -588, -728, -589},
9109 { -989,-1218,-1078,-1218,-1079},
9110 { -789,-1018, -878,-1018, -879}}},
9111 /* UA.UG..CG */
9112 {{ DEF, -279, -139, -279, -140},
9113 {-1079,-1308,-1168,-1308,-1169},
9114 { -569, -798, -658, -798, -659},
9115 { -989,-1218,-1078,-1218,-1079},
9116 { -859,-1088, -948,-1088, -949}}},
9117 /* UA.UU..CG */
9118 {{ DEF, -279, -139, -279, -140},
9119 {-1079,-1308,-1168,-1308,-1169},
9120 { -719, -948, -808, -948, -809},
9121 { -989,-1218,-1078,-1218,-1079},
9122 { -909,-1138, -998,-1138, -999}}},
9123 /* UA.@@..GC */
9124 {{{{ 0, 0, 0, 0, 0},
9125 { DEF, DEF, DEF, DEF, DEF},
9126 { DEF, DEF, DEF, DEF, DEF},
9127 { DEF, DEF, DEF, DEF, DEF},
9128 { DEF, DEF, DEF, DEF, DEF}}},
9129 /* UA.@A..GC */
9130 {{ 0, 0, 0, 0, 0},
9131 { -519, -519, -519, -519, -519},
9132 { -719, -719, -719, -719, -719},
9133 { -709, -709, -709, -709, -709},
9134 { -499, -499, -499, -499, -499}},
9135 /* UA.@C..GC */
9136 {{ 0, 0, 0, 0, 0},
9137 { -879, -879, -879, -879, -879},
9138 { -309, -309, -309, -309, -309},
9139 { -739, -739, -739, -739, -739},
9140 { -499, -499, -499, -499, -499}},
9141 /* UA.@G..GC */
9142 {{ 0, 0, 0, 0, 0},
9143 { -559, -559, -559, -559, -559},
9144 { -309, -309, -309, -309, -309},
9145 { -619, -619, -619, -619, -619},
9146 { -499, -499, -499, -499, -499}},
9147 /* UA.@U..GC */
9148 {{ 0, 0, 0, 0, 0},
9149 { -879, -879, -879, -879, -879},
9150 { -389, -389, -389, -389, -389},
9151 { -739, -739, -739, -739, -739},
9152 { -569, -569, -569, -569, -569}},
9153 /* UA.A@..GC */
9154 {{{ DEF, -399, -629, -889, -589},
9155 { -100, -449, -679, -939, -639},
9156 { -100, -449, -679, -939, -639},
9157 { -100, -449, -679, -939, -639},
9158 { -100, -449, -679, -939, -639}},
9159 /* UA.AA..GC */
9160 {{ DEF, -399, -629, -889, -589},
9161 { -569, -918,-1148,-1408,-1108},

```

```

9162 { -769, -1118, -1348, -1608, -1308},
9163 { -759, -1108, -1338, -1598, -1298},
9164 { -549, -898, -1128, -1388, -1088}},
9165 /* UA.AC..GC */
9166 {{ DEF, -399, -629, -889, -589},
9167 { -929, -1278, -1508, -1768, -1468},
9168 { -359, -708, -938, -1198, -898},
9169 { -789, -1138, -1368, -1628, -1328},
9170 { -549, -898, -1128, -1388, -1088}},
9171 /* UA.AG..GC */
9172 {{ DEF, -399, -629, -889, -589},
9173 { -609, -958, -1188, -1448, -1148},
9174 { -359, -708, -938, -1198, -898},
9175 { -669, -1018, -1248, -1508, -1208},
9176 { -549, -898, -1128, -1388, -1088}},
9177 /* UA.AU..GC */
9178 {{ DEF, -399, -629, -889, -589},
9179 { -929, -1278, -1508, -1768, -1468},
9180 { -439, -788, -1018, -1278, -978},
9181 { -789, -1138, -1368, -1628, -1328},
9182 { -619, -968, -1198, -1458, -1158}}},
9183 /* UA.C@..GC */
9184 {{ DEF, -429, -509, -199, -179},
9185 { -100, -479, -559, -249, -229},
9186 { -100, -479, -559, -249, -229},
9187 { -100, -479, -559, -249, -229},
9188 { -100, -479, -559, -249, -229}},
9189 /* UA.CA..GC */
9190 {{ DEF, -429, -509, -199, -179},
9191 { -569, -948, -1028, -718, -698},
9192 { -769, -1148, -1228, -918, -898},
9193 { -759, -1138, -1218, -908, -888},
9194 { -549, -928, -1008, -698, -678}},
9195 /* UA.CC..GC */
9196 {{ DEF, -429, -509, -199, -179},
9197 { -929, -1308, -1388, -1078, -1058},
9198 { -359, -738, -818, -508, -488},
9199 { -789, -1168, -1248, -938, -918},
9200 { -549, -928, -1008, -698, -678}},
9201 /* UA.CG..GC */
9202 {{ DEF, -429, -509, -199, -179},
9203 { -609, -988, -1068, -758, -738},
9204 { -359, -738, -818, -508, -488},
9205 { -669, -1048, -1128, -818, -798},
9206 { -549, -928, -1008, -698, -678}},
9207 /* UA.CU..GC */
9208 {{ DEF, -429, -509, -199, -179},
9209 { -929, -1308, -1388, -1078, -1058},
9210 { -439, -818, -898, -588, -568},
9211 { -789, -1168, -1248, -938, -918},
9212 { -619, -998, -1078, -768, -748}}},
9213 /* UA.G@..GC */
9214 {{ DEF, -379, -679, -889, -679},
9215 { -100, -429, -729, -939, -729},
9216 { -100, -429, -729, -939, -729},
9217 { -100, -429, -729, -939, -729},
9218 { -100, -429, -729, -939, -729}},
9219 /* UA.GA..GC */
9220 {{ DEF, -379, -679, -889, -679},
9221 { -569, -898, -1198, -1408, -1198},
9222 { -769, -1098, -1398, -1608, -1398},
9223 { -759, -1088, -1388, -1598, -1388},
9224 { -549, -878, -1178, -1388, -1178}},
9225 /* UA.GC..GC */
9226 {{ DEF, -379, -679, -889, -679},
9227 { -929, -1258, -1558, -1768, -1558},
9228 { -359, -688, -988, -1198, -988},
9229 { -789, -1118, -1418, -1628, -1418},
9230 { -549, -878, -1178, -1388, -1178}},
9231 /* UA.GG..GC */
9232 {{ DEF, -379, -679, -889, -679},
9233 { -609, -938, -1238, -1448, -1238},
9234 { -359, -688, -988, -1198, -988},
9235 { -669, -998, -1298, -1508, -1298},
9236 { -549, -878, -1178, -1388, -1178}},
9237 /* UA.GU..GC */
9238 {{ DEF, -379, -679, -889, -679},
9239 { -929, -1258, -1558, -1768, -1558},
9240 { -439, -768, -1068, -1278, -1068},
9241 { -789, -1118, -1418, -1628, -1418},
9242 { -619, -948, -1248, -1458, -1248}}},
9243 /* UA.U@..GC */
9244 {{ DEF, -279, -139, -279, -140},
9245 { -100, -329, -189, -329, -190},
9246 { -100, -329, -189, -329, -190},
9247 { -100, -329, -189, -329, -190},
9248 { -100, -329, -189, -329, -190}},

```

```

9249 /* UA.UA..GC */
9250 {{ DEF, -279, -139, -279, -140},
9251 { -569, -798, -658, -798, -659},
9252 { -769, -998, -858, -998, -859},
9253 { -759, -988, -848, -988, -849},
9254 { -549, -778, -638, -778, -639}},
9255 /* UA.UC..GC */
9256 {{ DEF, -279, -139, -279, -140},
9257 { -929,-1158,-1018,-1158,-1019},
9258 { -359, -588, -448, -588, -449},
9259 { -789,-1018, -878,-1018, -879},
9260 { -549, -778, -638, -778, -639}},
9261 /* UA.UG..GC */
9262 {{ DEF, -279, -139, -279, -140},
9263 { -609, -838, -698, -838, -699},
9264 { -359, -588, -448, -588, -449},
9265 { -669, -898, -758, -898, -759},
9266 { -549, -778, -638, -778, -639}},
9267 /* UA.UU..GC */
9268 {{ DEF, -279, -139, -279, -140},
9269 { -929,-1158,-1018,-1158,-1019},
9270 { -439, -668, -528, -668, -529},
9271 { -789,-1018, -878,-1018, -879},
9272 { -619, -848, -708, -848, -709}}}},
9273 /* UA.@@..GU */
9274 {{{{ 0, 0, 0, 0, 0},
9275 { DEF, DEF, DEF, DEF, DEF},
9276 { DEF, DEF, DEF, DEF, DEF},
9277 { DEF, DEF, DEF, DEF, DEF},
9278 { DEF, DEF, DEF, DEF, DEF}},
9279 /* UA.@A..GU */
9280 {{ 0, 0, 0, 0, 0},
9281 { -429, -429, -429, -429, -429},
9282 { -259, -259, -259, -259, -259},
9283 { -339, -339, -339, -339, -339},
9284 { -329, -329, -329, -329, -329}},
9285 /* UA.@C..GU */
9286 {{ 0, 0, 0, 0, 0},
9287 { -599, -599, -599, -599, -599},
9288 { -239, -239, -239, -239, -239},
9289 { -689, -689, -689, -689, -689},
9290 { -329, -329, -329, -329, -329}},
9291 /* UA.@G..GU */
9292 {{ 0, 0, 0, 0, 0},
9293 { -599, -599, -599, -599, -599},
9294 { -239, -239, -239, -239, -239},
9295 { -689, -689, -689, -689, -689},
9296 { -329, -329, -329, -329, -329}},
9297 /* UA.@U..GU */
9298 {{ 0, 0, 0, 0, 0},
9299 { -599, -599, -599, -599, -599},
9300 { -239, -239, -239, -239, -239},
9301 { -689, -689, -689, -689, -689},
9302 { -329, -329, -329, -329, -329}}},
9303 /* UA.A@..GU */
9304 {{{{ DEF, -399, -629, -889, -589},
9305 { -100, -449, -679, -939, -639},
9306 { -100, -449, -679, -939, -639},
9307 { -100, -449, -679, -939, -639},
9308 { -100, -449, -679, -939, -639}}},
9309 /* UA.AA..GU */
9310 {{ DEF, -399, -629, -889, -589},
9311 { -479, -828,-1058,-1318,-1018},
9312 { -309, -658, -888,-1148, -848},
9313 { -389, -738, -968,-1228, -928},
9314 { -379, -728, -958,-1218, -918}},
9315 /* UA.AC..GU */
9316 {{ DEF, -399, -629, -889, -589},
9317 { -649, -998,-1228,-1488,-1188},
9318 { -289, -638, -868,-1128, -828},
9319 { -739,-1088,-1318,-1578,-1278},
9320 { -379, -728, -958,-1218, -918}},
9321 /* UA.AG..GU */
9322 {{ DEF, -399, -629, -889, -589},
9323 { -649, -998,-1228,-1488,-1188},
9324 { -289, -638, -868,-1128, -828},
9325 { -739,-1088,-1318,-1578,-1278},
9326 { -379, -728, -958,-1218, -918}},
9327 /* UA.AU..GU */
9328 {{ DEF, -399, -629, -889, -589},
9329 { -649, -998,-1228,-1488,-1188},
9330 { -289, -638, -868,-1128, -828},
9331 { -739,-1088,-1318,-1578,-1278},
9332 { -379, -728, -958,-1218, -918}}},
9333 /* UA.C@..GU */
9334 {{{ DEF, -429, -509, -199, -179},
9335 { -100, -479, -559, -249, -229},

```

```

9336 { -100, -479, -559, -249, -229},
9337 { -100, -479, -559, -249, -229},
9338 { -100, -479, -559, -249, -229}},
9339 /* UA.CA..GU */
9340 {{ DEF, -429, -509, -199, -179},
9341 { -479, -858, -938, -628, -608},
9342 { -309, -688, -768, -458, -438},
9343 { -389, -768, -848, -538, -518},
9344 { -379, -758, -838, -528, -508}},
9345 /* UA.CC..GU */
9346 {{ DEF, -429, -509, -199, -179},
9347 { -649,-1028,-1108, -798, -778},
9348 { -289, -668, -748, -438, -418},
9349 { -739,-1118,-1198, -888, -868},
9350 { -379, -758, -838, -528, -508}},
9351 /* UA.CG..GU */
9352 {{ DEF, -429, -509, -199, -179},
9353 { -649,-1028,-1108, -798, -778},
9354 { -289, -668, -748, -438, -418},
9355 { -739,-1118,-1198, -888, -868},
9356 { -379, -758, -838, -528, -508}},
9357 /* UA.CU..GU */
9358 {{ DEF, -429, -509, -199, -179},
9359 { -649,-1028,-1108, -798, -778},
9360 { -289, -668, -748, -438, -418},
9361 { -739,-1118,-1198, -888, -868},
9362 { -379, -758, -838, -528, -508}},
9363 /* UA.G@..GU */
9364 {{{ DEF, -379, -679, -889, -679},
9365 { -100, -429, -729, -939, -729},
9366 { -100, -429, -729, -939, -729},
9367 { -100, -429, -729, -939, -729},
9368 { -100, -429, -729, -939, -729}},
9369 /* UA.GA..GU */
9370 {{ DEF, -379, -679, -889, -679},
9371 { -479, -808,-1108,-1318,-1108},
9372 { -309, -638, -938,-1148, -938},
9373 { -389, -718,-1018,-1228,-1018},
9374 { -379, -708,-1008,-1218,-1008}},
9375 /* UA.GC..GU */
9376 {{ DEF, -379, -679, -889, -679},
9377 { -649, -978,-1278,-1488,-1278},
9378 { -289, -618, -918,-1128, -918},
9379 { -739,-1068,-1368,-1578,-1368},
9380 { -379, -708,-1008,-1218,-1008}},
9381 /* UA.GG..GU */
9382 {{ DEF, -379, -679, -889, -679},
9383 { -649, -978,-1278,-1488,-1278},
9384 { -289, -618, -918,-1128, -918},
9385 { -739,-1068,-1368,-1578,-1368},
9386 { -379, -708,-1008,-1218,-1008}},
9387 /* UA.GU..GU */
9388 {{ DEF, -379, -679, -889, -679},
9389 { -649, -978,-1278,-1488,-1278},
9390 { -289, -618, -918,-1128, -918},
9391 { -739,-1068,-1368,-1578,-1368},
9392 { -379, -708,-1008,-1218,-1008}},
9393 /* UA.U@..GU */
9394 {{{ DEF, -279, -139, -279, -140},
9395 { -100, -329, -189, -329, -190},
9396 { -100, -329, -189, -329, -190},
9397 { -100, -329, -189, -329, -190},
9398 { -100, -329, -189, -329, -190}},
9399 /* UA.UA..GU */
9400 {{ DEF, -279, -139, -279, -140},
9401 { -479, -708, -568, -708, -569},
9402 { -309, -538, -398, -538, -399},
9403 { -389, -618, -478, -618, -479},
9404 { -379, -608, -468, -608, -469}},
9405 /* UA.UC..GU */
9406 {{ DEF, -279, -139, -279, -140},
9407 { -649, -878, -738, -878, -739},
9408 { -289, -518, -378, -518, -379},
9409 { -739, -968, -828, -968, -829},
9410 { -379, -608, -468, -608, -469}},
9411 /* UA.UG..GU */
9412 {{ DEF, -279, -139, -279, -140},
9413 { -649, -878, -738, -878, -739},
9414 { -289, -518, -378, -518, -379},
9415 { -739, -968, -828, -968, -829},
9416 { -379, -608, -468, -608, -469}},
9417 /* UA.UU..GU */
9418 {{ DEF, -279, -139, -279, -140},
9419 { -649, -878, -738, -878, -739},
9420 { -289, -518, -378, -518, -379},
9421 { -739, -968, -828, -968, -829},
9422 { -379, -608, -468, -608, -469}}}},

```

```

9423 /* UA.@@..UG */
9424 {{{{ 0, 0, 0, 0, 0},
9425 { DEF, DEF, DEF, DEF, DEF},
9426 { DEF, DEF, DEF, DEF, DEF},
9427 { DEF, DEF, DEF, DEF, DEF},
9428 { DEF, DEF, DEF, DEF, DEF}}},
9429 /* UA.@A..UG */
9430 {{ 0, 0, 0, 0, 0},
9431 { -719, -719, -719, -719, -719},
9432 { -479, -479, -479, -479, -479},
9433 { -659, -659, -659, -659, -659},
9434 { -549, -549, -549, -549, -549}},
9435 /* UA.@C..UG */
9436 {{ 0, 0, 0, 0, 0},
9437 { -789, -789, -789, -789, -789},
9438 { -479, -479, -479, -479, -479},
9439 { -809, -809, -809, -809, -809},
9440 { -439, -439, -439, -439, -439}},
9441 /* UA.@G..UG */
9442 {{ 0, 0, 0, 0, 0},
9443 { -959, -959, -959, -959, -959},
9444 { -359, -359, -359, -359, -359},
9445 { -919, -919, -919, -919, -919},
9446 { -549, -549, -549, -549, -549}},
9447 /* UA.@U..UG */
9448 {{ 0, 0, 0, 0, 0},
9449 { -809, -809, -809, -809, -809},
9450 { -479, -479, -479, -479, -479},
9451 { -809, -809, -809, -809, -809},
9452 { -359, -359, -359, -359, -359}},
9453 /* UA.A@..UG */
9454 {{ DEF, -399, -629, -889, -589},
9455 { -100, -449, -679, -939, -639},
9456 { -100, -449, -679, -939, -639},
9457 { -100, -449, -679, -939, -639},
9458 { -100, -449, -679, -939, -639}},
9459 /* UA.AA..UG */
9460 {{ DEF, -399, -629, -889, -589},
9461 { -769, -1118, -1348, -1608, -1308},
9462 { -529, -878, -1108, -1368, -1068},
9463 { -709, -1058, -1288, -1548, -1248},
9464 { -599, -948, -1178, -1438, -1138}},
9465 /* UA.AC..UG */
9466 {{ DEF, -399, -629, -889, -589},
9467 { -839, -1188, -1418, -1678, -1378},
9468 { -529, -878, -1108, -1368, -1068},
9469 { -859, -1208, -1438, -1698, -1398},
9470 { -489, -838, -1068, -1328, -1028}},
9471 /* UA.AG..UG */
9472 {{ DEF, -399, -629, -889, -589},
9473 { -1009, -1358, -1588, -1848, -1548},
9474 { -409, -758, -988, -1248, -948},
9475 { -969, -1318, -1548, -1808, -1508},
9476 { -599, -948, -1178, -1438, -1138}},
9477 /* UA.AU..UG */
9478 {{ DEF, -399, -629, -889, -589},
9479 { -859, -1208, -1438, -1698, -1398},
9480 { -529, -878, -1108, -1368, -1068},
9481 { -859, -1208, -1438, -1698, -1398},
9482 { -409, -758, -988, -1248, -948}},
9483 /* UA.C@..UG */
9484 {{ DEF, -429, -509, -199, -179},
9485 { -100, -479, -559, -249, -229},
9486 { -100, -479, -559, -249, -229},
9487 { -100, -479, -559, -249, -229},
9488 { -100, -479, -559, -249, -229}},
9489 /* UA.CA..UG */
9490 {{ DEF, -429, -509, -199, -179},
9491 { -769, -1148, -1228, -918, -898},
9492 { -529, -908, -988, -678, -658},
9493 { -709, -1088, -1168, -858, -838},
9494 { -599, -978, -1058, -748, -728}},
9495 /* UA.CC..UG */
9496 {{ DEF, -429, -509, -199, -179},
9497 { -839, -1218, -1298, -988, -968},
9498 { -529, -908, -988, -678, -658},
9499 { -859, -1238, -1318, -1008, -988},
9500 { -489, -868, -948, -638, -618}},
9501 /* UA.CG..UG */
9502 {{ DEF, -429, -509, -199, -179},
9503 { -1009, -1388, -1468, -1158, -1138},
9504 { -409, -788, -868, -558, -538},
9505 { -969, -1348, -1428, -1118, -1098},
9506 { -599, -978, -1058, -748, -728}},
9507 /* UA.CU..UG */
9508 {{ DEF, -429, -509, -199, -179},
9509 { -859, -1238, -1318, -1008, -988},

```

```

9510 { -529, -908, -988, -678, -658},
9511 { -859,-1238,-1318,-1008, -988},,
9512 { -409, -788, -868, -558, -538}}},,
9513 /* UA.G@..UG */
9514 {{ DEF, -379, -679, -889, -679},
9515 { -100, -429, -729, -939, -729},,
9516 { -100, -429, -729, -939, -729},,
9517 { -100, -429, -729, -939, -729},,
9518 { -100, -429, -729, -939, -729}},,
9519 /* UA.GA..UG */
9520 {{ DEF, -379, -679, -889, -679},
9521 { -769,-1098,-1398,-1608,-1398},,
9522 { -529, -858,-1158,-1368,-1158},,
9523 { -709,-1038,-1338,-1548,-1338},,
9524 { -599, -928,-1228,-1438,-1228}}},,
9525 /* UA.GC..UG */
9526 {{ DEF, -379, -679, -889, -679},
9527 { -839,-1168,-1468,-1678,-1468},,
9528 { -529, -858,-1158,-1368,-1158},,
9529 { -859,-1188,-1488,-1698,-1488},,
9530 { -489, -818,-1118,-1328,-1118}}},,
9531 /* UA.GG..UG */
9532 {{ DEF, -379, -679, -889, -679},
9533 {-1009,-1338,-1638,-1848,-1638},,
9534 { -409, -738,-1038,-1248,-1038},,
9535 { -969,-1298,-1598,-1808,-1598},,
9536 { -599, -928,-1228,-1438,-1228}}},,
9537 /* UA.GU..UG */
9538 {{ DEF, -379, -679, -889, -679},
9539 { -859,-1188,-1488,-1698,-1488},,
9540 { -529, -858,-1158,-1368,-1158},,
9541 { -859,-1188,-1488,-1698,-1488},,
9542 { -409, -738,-1038,-1248,-1038}}},,
9543 /* UA.U@..UG */
9544 {{ DEF, -279, -139, -279, -140},,
9545 { -100, -329, -189, -329, -190},,
9546 { -100, -329, -189, -329, -190},,
9547 { -100, -329, -189, -329, -190},,
9548 { -100, -329, -189, -329, -190}},,
9549 /* UA.UA..UG */
9550 {{ DEF, -279, -139, -279, -140},,
9551 { -769, -998, -858, -998, -859},,
9552 { -529, -758, -618, -758, -619},,
9553 { -709, -938, -798, -938, -799},,
9554 { -599, -828, -688, -828, -689}}},,
9555 /* UA.UC..UG */
9556 {{ DEF, -279, -139, -279, -140},,
9557 { -839,-1068, -928,-1068, -929},,
9558 { -529, -758, -618, -758, -619},,
9559 { -859,-1088, -948,-1088, -949},,
9560 { -489, -718, -578, -718, -579}}},,
9561 /* UA.UG..UG */
9562 {{ DEF, -279, -139, -279, -140},,
9563 {-1009,-1238,-1098,-1238,-1099},,
9564 { -409, -638, -498, -638, -499},,
9565 { -969,-1198,-1058,-1198,-1059},,
9566 { -599, -828, -688, -828, -689}}},,
9567 /* UA.UU..UG */
9568 {{ DEF, -279, -139, -279, -140},,
9569 { -859,-1088, -948,-1088, -949},,
9570 { -529, -758, -618, -758, -619},,
9571 { -859,-1088, -948,-1088, -949},,
9572 { -409, -638, -498, -638, -499}}},,
9573 /* UA.@@..AU */
9574 {{{ 0, 0, 0, 0, 0},,
9575 { DEF, DEF, DEF, DEF, DEF},,
9576 { DEF, DEF, DEF, DEF, DEF},,
9577 { DEF, DEF, DEF, DEF, DEF},,
9578 { DEF, DEF, DEF, DEF, DEF}}},,
9579 /* UA.@A..AU */
9580 {{ 0, 0, 0, 0, 0},,
9581 { -429, -429, -429, -429, -429},,
9582 { -259, -259, -259, -259, -259},,
9583 { -339, -339, -339, -339, -339},,
9584 { -329, -329, -329, -329, -329}},,
9585 /* UA.@C..AU */
9586 {{ 0, 0, 0, 0, 0},,
9587 { -599, -599, -599, -599, -599},,
9588 { -239, -239, -239, -239, -239},,
9589 { -689, -689, -689, -689, -689},,
9590 { -329, -329, -329, -329, -329}},,
9591 /* UA.@G..AU */
9592 {{ 0, 0, 0, 0, 0},,
9593 { -599, -599, -599, -599, -599},,
9594 { -239, -239, -239, -239, -239},,
9595 { -689, -689, -689, -689, -689},,
9596 { -329, -329, -329, -329, -329}}},,

```

```

9597 /* UA.@U..AU */
9598 {{ 0, 0, 0, 0, 0},
9599 { -599, -599, -599, -599, -599},
9600 { -239, -239, -239, -239, -239},
9601 { -689, -689, -689, -689, -689},
9602 { -329, -329, -329, -329, -329}},
9603 /* UA.A@..AU */
9604 {{ DEF, -399, -629, -889, -589},
9605 { -100, -449, -679, -939, -639},
9606 { -100, -449, -679, -939, -639},
9607 { -100, -449, -679, -939, -639},
9608 { -100, -449, -679, -939, -639}},
9609 /* UA.AA..AU */
9610 {{ DEF, -399, -629, -889, -589},
9611 { -479, -828, -1058, -1318, -1018},
9612 { -309, -658, -888, -1148, -848},
9613 { -389, -738, -968, -1228, -928},
9614 { -379, -728, -958, -1218, -918}},
9615 /* UA.AC..AU */
9616 {{ DEF, -399, -629, -889, -589},
9617 { -649, -998, -1228, -1488, -1188},
9618 { -289, -638, -868, -1128, -828},
9619 { -739, -1088, -1318, -1578, -1278},
9620 { -379, -728, -958, -1218, -918}},
9621 /* UA.AG..AU */
9622 {{ DEF, -399, -629, -889, -589},
9623 { -649, -998, -1228, -1488, -1188},
9624 { -289, -638, -868, -1128, -828},
9625 { -739, -1088, -1318, -1578, -1278},
9626 { -379, -728, -958, -1218, -918}},
9627 /* UA.AU..AU */
9628 {{ DEF, -399, -629, -889, -589},
9629 { -649, -998, -1228, -1488, -1188},
9630 { -289, -638, -868, -1128, -828},
9631 { -739, -1088, -1318, -1578, -1278},
9632 { -379, -728, -958, -1218, -918}},
9633 /* UA.C@..AU */
9634 {{ DEF, -429, -509, -199, -179},
9635 { -100, -479, -559, -249, -229},
9636 { -100, -479, -559, -249, -229},
9637 { -100, -479, -559, -249, -229},
9638 { -100, -479, -559, -249, -229}},
9639 /* UA.CA..AU */
9640 {{ DEF, -429, -509, -199, -179},
9641 { -479, -858, -938, -628, -608},
9642 { -309, -688, -768, -458, -438},
9643 { -389, -768, -848, -538, -518},
9644 { -379, -758, -838, -528, -508}},
9645 /* UA.CC..AU */
9646 {{ DEF, -429, -509, -199, -179},
9647 { -649, -1028, -1108, -798, -778},
9648 { -289, -668, -748, -438, -418},
9649 { -739, -1118, -1198, -888, -868},
9650 { -379, -758, -838, -528, -508}},
9651 /* UA.CG..AU */
9652 {{ DEF, -429, -509, -199, -179},
9653 { -649, -1028, -1108, -798, -778},
9654 { -289, -668, -748, -438, -418},
9655 { -739, -1118, -1198, -888, -868},
9656 { -379, -758, -838, -528, -508}},
9657 /* UA.CU..AU */
9658 {{ DEF, -429, -509, -199, -179},
9659 { -649, -1028, -1108, -798, -778},
9660 { -289, -668, -748, -438, -418},
9661 { -739, -1118, -1198, -888, -868},
9662 { -379, -758, -838, -528, -508}},
9663 /* UA.G@..AU */
9664 {{ DEF, -379, -679, -889, -679},
9665 { -100, -429, -729, -939, -729},
9666 { -100, -429, -729, -939, -729},
9667 { -100, -429, -729, -939, -729},
9668 { -100, -429, -729, -939, -729}},
9669 /* UA.GA..AU */
9670 {{ DEF, -379, -679, -889, -679},
9671 { -479, -808, -1108, -1318, -1108},
9672 { -309, -638, -938, -1148, -938},
9673 { -389, -718, -1018, -1228, -1018},
9674 { -379, -708, -1008, -1218, -1008}},
9675 /* UA.GC..AU */
9676 {{ DEF, -379, -679, -889, -679},
9677 { -649, -978, -1278, -1488, -1278},
9678 { -289, -618, -918, -1128, -918},
9679 { -739, -1068, -1368, -1578, -1368},
9680 { -379, -708, -1008, -1218, -1008}},
9681 /* UA.GG..AU */
9682 {{ DEF, -379, -679, -889, -679},
9683 { -649, -978, -1278, -1488, -1278}},

```

```

9684 { -289, -618, -918,-1128, -918},
9685 { -739,-1068,-1368,-1578,-1368},
9686 { -379, -708,-1008,-1218,-1008}},
9687 /* UA.GU..AU */
9688 {{ DEF, -379, -679, -889, -679},
9689 { -649, -978,-1278,-1488,-1278},
9690 { -289, -618, -918,-1128, -918},
9691 { -739,-1068,-1368,-1578,-1368},
9692 { -379, -708,-1008,-1218,-1008}}},
9693 /* UA.U@..AU */
9694 {{ DEF, -279, -139, -279, -140},
9695 { -100, -329, -189, -329, -190},
9696 { -100, -329, -189, -329, -190},
9697 { -100, -329, -189, -329, -190},
9698 { -100, -329, -189, -329, -190}},
9699 /* UA.UA..AU */
9700 {{ DEF, -279, -139, -279, -140},
9701 { -479, -708, -568, -708, -569},
9702 { -309, -538, -398, -538, -399},
9703 { -389, -618, -478, -618, -479},
9704 { -379, -608, -468, -608, -469}},
9705 /* UA.UC..AU */
9706 {{ DEF, -279, -139, -279, -140},
9707 { -649, -878, -738, -878, -739},
9708 { -289, -518, -378, -518, -379},
9709 { -739, -968, -828, -968, -829},
9710 { -379, -608, -468, -608, -469}},
9711 /* UA.UG..AU */
9712 {{ DEF, -279, -139, -279, -140},
9713 { -649, -878, -738, -878, -739},
9714 { -289, -518, -378, -518, -379},
9715 { -739, -968, -828, -968, -829},
9716 { -379, -608, -468, -608, -469}},
9717 /* UA.UU..AU */
9718 {{ DEF, -279, -139, -279, -140},
9719 { -649, -878, -738, -878, -739},
9720 { -289, -518, -378, -518, -379},
9721 { -739, -968, -828, -968, -829},
9722 { -379, -608, -468, -608, -469}}},
9723 /* UA.@@..UA */
9724 {{{ 0, 0, 0, 0, 0},
9725 { DEF, DEF, DEF, DEF, DEF},
9726 { DEF, DEF, DEF, DEF, DEF},
9727 { DEF, DEF, DEF, DEF, DEF},
9728 { DEF, DEF, DEF, DEF, DEF}},
9729 /* UA.@A..UA */
9730 {{ 0, 0, 0, 0, 0},
9731 { -399, -399, -399, -399, -399},
9732 { -429, -429, -429, -429, -429},
9733 { -379, -379, -379, -379, -379},
9734 { -279, -279, -279, -279, -279}},
9735 /* UA.@C..UA */
9736 {{ 0, 0, 0, 0, 0},
9737 { -629, -629, -629, -629, -629},
9738 { -509, -509, -509, -509, -509},
9739 { -679, -679, -679, -679, -679},
9740 { -139, -139, -139, -139, -139}},
9741 /* UA.@G..UA */
9742 {{ 0, 0, 0, 0, 0},
9743 { -889, -889, -889, -889, -889},
9744 { -199, -199, -199, -199, -199},
9745 { -889, -889, -889, -889, -889},
9746 { -279, -279, -279, -279, -279}},
9747 /* UA.@U..UA */
9748 {{ 0, 0, 0, 0, 0},
9749 { -589, -589, -589, -589, -589},
9750 { -179, -179, -179, -179, -179},
9751 { -679, -679, -679, -679, -679},
9752 { -140, -140, -140, -140, -140}}},
9753 /* UA.A@..UA */
9754 {{ DEF, -399, -629, -889, -589},
9755 { -100, -449, -679, -939, -639},
9756 { -100, -449, -679, -939, -639},
9757 { -100, -449, -679, -939, -639},
9758 { -100, -449, -679, -939, -639}},
9759 /* UA.AA..UA */
9760 {{ DEF, -399, -629, -889, -589},
9761 { -449, -798,-1028,-1288, -988},
9762 { -479, -828,-1058,-1318,-1018},
9763 { -429, -778,-1008,-1268, -968},
9764 { -329, -678, -908,-1168, -868}},
9765 /* UA.AC..UA */
9766 {{ DEF, -399, -629, -889, -589},
9767 { -679,-1028,-1258,-1518,-1218},
9768 { -559, -908,-1138,-1398,-1098},
9769 { -729,-1078,-1308,-1568,-1268},
9770 { -189, -538, -768,-1028, -728}}},

```

```

9771 /* UA.AG..UA */
9772 {{ DEF, -399, -629, -889, -589},
9773 { -939,-1288,-1518,-1778,-1478},
9774 { -249, -598, -828,-1088, -788},
9775 { -939,-1288,-1518,-1778,-1478},
9776 { -329, -678, -908,-1168, -868}},
9777 /* UA.AU..UA */
9778 {{ DEF, -399, -629, -889, -589},
9779 { -639, -988,-1218,-1478,-1178},
9780 { -229, -578, -808,-1068, -768},
9781 { -729,-1078,-1308,-1568,-1268},
9782 { -190, -539, -769,-1029, -729}},
9783 /* UA.C@..UA */
9784 {{{ DEF, -429, -509, -199, -179},
9785 { -100, -479, -559, -249, -229},
9786 { -100, -479, -559, -249, -229},
9787 { -100, -479, -559, -249, -229},
9788 { -100, -479, -559, -249, -229}},
9789 /* UA.CA..UA */
9790 {{ DEF, -429, -509, -199, -179},
9791 { -449, -828, -908, -598, -578},
9792 { -479, -858, -938, -628, -608},
9793 { -429, -808, -888, -578, -558},
9794 { -329, -708, -788, -478, -458}},
9795 /* UA.CC..UA */
9796 {{ DEF, -429, -509, -199, -179},
9797 { -679,-1058,-1138, -828, -808},
9798 { -559, -938,-1018, -708, -688},
9799 { -729,-1108,-1188, -878, -858},
9800 { -189, -568, -648, -338, -318}},
9801 /* UA.CG..UA */
9802 {{ DEF, -429, -509, -199, -179},
9803 { -939,-1318,-1398,-1088,-1068},
9804 { -249, -628, -708, -398, -378},
9805 { -939,-1318,-1398,-1088,-1068},
9806 { -329, -708, -788, -478, -458}},
9807 /* UA.CU..UA */
9808 {{ DEF, -429, -509, -199, -179},
9809 { -639,-1018,-1098, -788, -768},
9810 { -229, -608, -688, -378, -358},
9811 { -729,-1108,-1188, -878, -858},
9812 { -190, -569, -649, -339, -319}},
9813 /* UA.G@..UA */
9814 {{{ DEF, -379, -679, -889, -679},
9815 { -100, -429, -729, -939, -729},
9816 { -100, -429, -729, -939, -729},
9817 { -100, -429, -729, -939, -729},
9818 { -100, -429, -729, -939, -729}},
9819 /* UA.GA..UA */
9820 {{ DEF, -379, -679, -889, -679},
9821 { -449, -778,-1078,-1288,-1078},
9822 { -479, -808,-1108,-1318,-1108},
9823 { -429, -758,-1058,-1268,-1058},
9824 { -329, -658, -958,-1168, -958}},
9825 /* UA.GC..UA */
9826 {{ DEF, -379, -679, -889, -679},
9827 { -679,-1008,-1308,-1518,-1308},
9828 { -559, -888,-1188,-1398,-1188},
9829 { -729,-1058,-1358,-1568,-1358},
9830 { -189, -518, -818,-1028, -818}},
9831 /* UA.GG..UA */
9832 {{ DEF, -379, -679, -889, -679},
9833 { -939,-1268,-1568,-1778,-1568},
9834 { -249, -578, -878,-1088, -878},
9835 { -939,-1268,-1568,-1778,-1568},
9836 { -329, -658, -958,-1168, -958}},
9837 /* UA.GU..UA */
9838 {{ DEF, -379, -679, -889, -679},
9839 { -639, -968,-1268,-1478,-1268},
9840 { -229, -558, -858,-1068, -858},
9841 { -729,-1058,-1358,-1568,-1358},
9842 { -190, -519, -819,-1029, -819}},
9843 /* UA.U@..UA */
9844 {{{ DEF, -279, -139, -279, -140},
9845 { -100, -329, -189, -329, -190},
9846 { -100, -329, -189, -329, -190},
9847 { -100, -329, -189, -329, -190},
9848 { -100, -329, -189, -329, -190}},
9849 /* UA.UA..UA */
9850 {{ DEF, -279, -139, -279, -140},
9851 { -449, -678, -538, -678, -539},
9852 { -479, -708, -568, -708, -569},
9853 { -429, -658, -518, -658, -519},
9854 { -329, -558, -418, -558, -419}},
9855 /* UA.UC..UA */
9856 {{ DEF, -279, -139, -279, -140},
9857 { -679, -908, -768, -908, -769},

```

```
9858 { -559, -788, -648, -788, -649},
9859 { -729, -958, -818, -958, -819},
9860 { -189, -418, -278, -418, -279}},
9861 /* UA.UG..UA */
9862 {{ DEF, -279, -139, -279, -140},
9863 { -939,-1168,-1028,-1168,-1029},
9864 { -249, -478, -338, -478, -339},
9865 { -939,-1168,-1028,-1168,-1029},
9866 { -329, -558, -418, -558, -419}},
9867 /* UA.UU..UA */
9868 {{ DEF, -279, -139, -279, -140},
9869 { -639, -868, -728, -868, -729},
9870 { -229, -458, -318, -458, -319},
9871 { -729, -958, -818, -958, -819},
9872 { -190, -419, -279, -419, -280}}}),
9873 /* UA.@@.. @ */
9874 {{{ DEF, DEF, DEF, DEF, DEF},
9875 { DEF, DEF, DEF, DEF, DEF},
9876 { DEF, DEF, DEF, DEF, DEF},
9877 { DEF, DEF, DEF, DEF, DEF},
9878 { DEF, DEF, DEF, DEF, DEF}},
9879 /* UA.@A.. @ */
9880 {{ DEF, DEF, DEF, DEF, DEF},
9881 { DEF, DEF, DEF, DEF, DEF},
9882 { DEF, DEF, DEF, DEF, DEF},
9883 { DEF, DEF, DEF, DEF, DEF},
9884 { DEF, DEF, DEF, DEF, DEF}},
9885 /* UA.@C.. @ */
9886 {{ DEF, DEF, DEF, DEF, DEF},
9887 { DEF, DEF, DEF, DEF, DEF},
9888 { DEF, DEF, DEF, DEF, DEF},
9889 { DEF, DEF, DEF, DEF, DEF},
9890 { DEF, DEF, DEF, DEF, DEF}},
9891 /* UA.@G.. @ */
9892 {{ DEF, DEF, DEF, DEF, DEF},
9893 { DEF, DEF, DEF, DEF, DEF},
9894 { DEF, DEF, DEF, DEF, DEF},
9895 { DEF, DEF, DEF, DEF, DEF},
9896 { DEF, DEF, DEF, DEF, DEF}},
9897 /* UA.@U.. @ */
9898 {{ DEF, DEF, DEF, DEF, DEF},
9899 { DEF, DEF, DEF, DEF, DEF},
9900 { DEF, DEF, DEF, DEF, DEF},
9901 { DEF, DEF, DEF, DEF, DEF},
9902 { DEF, DEF, DEF, DEF, DEF}}),
9903 /* UA.A@.. @ */
9904 {{ -100, -449, -679, -939, -639},
9905 { -100, -449, -679, -939, -639},
9906 { -100, -449, -679, -939, -639},
9907 { -100, -449, -679, -939, -639},
9908 { -100, -449, -679, -939, -639}},
9909 /* UA.AA.. @ */
9910 {{ -100, -449, -679, -939, -639},
9911 { -100, -449, -679, -939, -639},
9912 { -100, -449, -679, -939, -639},
9913 { -100, -449, -679, -939, -639},
9914 { -100, -449, -679, -939, -639}},
9915 /* UA.AC.. @ */
9916 {{ -100, -449, -679, -939, -639},
9917 { -100, -449, -679, -939, -639},
9918 { -100, -449, -679, -939, -639},
9919 { -100, -449, -679, -939, -639},
9920 { -100, -449, -679, -939, -639}},
9921 /* UA.AG.. @ */
9922 {{ -100, -449, -679, -939, -639},
9923 { -100, -449, -679, -939, -639},
9924 { -100, -449, -679, -939, -639},
9925 { -100, -449, -679, -939, -639},
9926 { -100, -449, -679, -939, -639}},
9927 /* UA.AU.. @ */
9928 {{ -100, -449, -679, -939, -639},
9929 { -100, -449, -679, -939, -639},
9930 { -100, -449, -679, -939, -639},
9931 { -100, -449, -679, -939, -639},
9932 { -100, -449, -679, -939, -639}}),
9933 /* UA.C@.. @ */
9934 {{ -100, -479, -559, -249, -229},
9935 { -100, -479, -559, -249, -229},
9936 { -100, -479, -559, -249, -229},
9937 { -100, -479, -559, -249, -229},
9938 { -100, -479, -559, -249, -229}},
9939 /* UA.CA.. @ */
9940 {{ -100, -479, -559, -249, -229},
9941 { -100, -479, -559, -249, -229},
9942 { -100, -479, -559, -249, -229},
9943 { -100, -479, -559, -249, -229},
9944 { -100, -479, -559, -249, -229}},
```

```

9945 /* UA.CC.. @ */
9946 {{ -100, -479, -559, -249, -229},
9947 { -100, -479, -559, -249, -229},
9948 { -100, -479, -559, -249, -229},
9949 { -100, -479, -559, -249, -229},
9950 { -100, -479, -559, -249, -229}},
9951 /* UA.CG.. @ */
9952 {{ -100, -479, -559, -249, -229},
9953 { -100, -479, -559, -249, -229},
9954 { -100, -479, -559, -249, -229},
9955 { -100, -479, -559, -249, -229},
9956 { -100, -479, -559, -249, -229}},
9957 /* UA.CU.. @ */
9958 {{ -100, -479, -559, -249, -229},
9959 { -100, -479, -559, -249, -229},
9960 { -100, -479, -559, -249, -229},
9961 { -100, -479, -559, -249, -229},
9962 { -100, -479, -559, -249, -229}}}},
9963 /* UA.G@.. @ */
9964 {{ { -100, -429, -729, -939, -729},
9965 { -100, -429, -729, -939, -729},
9966 { -100, -429, -729, -939, -729},
9967 { -100, -429, -729, -939, -729},
9968 { -100, -429, -729, -939, -729}}},
9969 /* UA.GA.. @ */
9970 {{ { -100, -429, -729, -939, -729},
9971 { -100, -429, -729, -939, -729},
9972 { -100, -429, -729, -939, -729},
9973 { -100, -429, -729, -939, -729},
9974 { -100, -429, -729, -939, -729}}},
9975 /* UA.GC.. @ */
9976 {{ { -100, -429, -729, -939, -729},
9977 { -100, -429, -729, -939, -729},
9978 { -100, -429, -729, -939, -729},
9979 { -100, -429, -729, -939, -729},
9980 { -100, -429, -729, -939, -729}}},
9981 /* UA.GG.. @ */
9982 {{ { -100, -429, -729, -939, -729},
9983 { -100, -429, -729, -939, -729},
9984 { -100, -429, -729, -939, -729},
9985 { -100, -429, -729, -939, -729},
9986 { -100, -429, -729, -939, -729}}},
9987 /* UA.GU.. @ */
9988 {{ { -100, -429, -729, -939, -729},
9989 { -100, -429, -729, -939, -729},
9990 { -100, -429, -729, -939, -729},
9991 { -100, -429, -729, -939, -729},
9992 { -100, -429, -729, -939, -729}}}},
9993 /* UA.U@.. @ */
9994 {{ { -100, -329, -189, -329, -190},
9995 { -100, -329, -189, -329, -190},
9996 { -100, -329, -189, -329, -190},
9997 { -100, -329, -189, -329, -190},
9998 { -100, -329, -189, -329, -190}}},
9999 /* UA.UA.. @ */
10000 {{ { -100, -329, -189, -329, -190},
10001 { -100, -329, -189, -329, -190},
10002 { -100, -329, -189, -329, -190},
10003 { -100, -329, -189, -329, -190},
10004 { -100, -329, -189, -329, -190}}},
10005 /* UA.UC.. @ */
10006 {{ { -100, -329, -189, -329, -190},
10007 { -100, -329, -189, -329, -190},
10008 { -100, -329, -189, -329, -190},
10009 { -100, -329, -189, -329, -190},
10010 { -100, -329, -189, -329, -190}}},
10011 /* UA.UG.. @ */
10012 {{ { -100, -329, -189, -329, -190},
10013 { -100, -329, -189, -329, -190},
10014 { -100, -329, -189, -329, -190},
10015 { -100, -329, -189, -329, -190},
10016 { -100, -329, -189, -329, -190}}},
10017 /* UA.UU.. @ */
10018 {{ { -100, -329, -189, -329, -190},
10019 { -100, -329, -189, -329, -190},
10020 { -100, -329, -189, -329, -190},
10021 { -100, -329, -189, -329, -190},
10022 { -100, -329, -189, -329, -190}}}},
10023 /* noPair */ {{{{0}}}},
10024 /* @@@.CG */
10025 {{{ DEF, DEF, DEF, DEF, DEF},
10026 { -100, -100, -100, -100, -100},
10027 { -100, -100, -100, -100, -100},
10028 { -100, -100, -100, -100, -100},
10029 { -100, -100, -100, -100, -100}}},
10030 /* @@@.CG */
10031 {{ DEF, DEF, DEF, DEF},

```

```
10032 {-1079,-1079,-1079,-1079,-1079},  
10033 {-569, -569, -569, -569, -569},  
10034 {-989, -989, -989, -989, -989},  
10035 {-859, -859, -859, -859, -859}},  
10036 /* @.CG */  
10037 {{ DEF, DEF, DEF, DEF, DEF},  
10038 {-999, -999, -999, -999, -999},  
10039 {-499, -499, -499, -499, -499},  
10040 {-989, -989, -989, -989, -989},  
10041 {-789, -789, -789, -789, -789}},  
10042 /* @.CG */  
10043 {{ DEF, DEF, DEF, DEF, DEF},  
10044 {-1079,-1079,-1079,-1079,-1079},  
10045 {-569, -569, -569, -569, -569},  
10046 {-989, -989, -989, -989, -989},  
10047 {-859, -859, -859, -859, -859}},  
10048 /* @.CG */  
10049 {{ DEF, DEF, DEF, DEF, DEF},  
10050 {-1079,-1079,-1079,-1079,-1079},  
10051 {-719, -719, -719, -719, -719},  
10052 {-989, -989, -989, -989, -989},  
10053 {-909, -909, -909, -909, -909}}},  
10054 /* @.AC */  
10055 {{ DEF, DEF, DEF, DEF, DEF},  
10056 {-100, -100, -100, -100, -100},  
10057 {-100, -100, -100, -100, -100},  
10058 {-100, -100, -100, -100, -100},  
10059 {-100, -100, -100, -100, -100}},  
10060 /* @.AA */  
10061 {{ DEF, DEF, DEF, DEF, DEF},  
10062 {-1079,-1079,-1079,-1079,-1079},  
10063 {-569, -569, -569, -569, -569},  
10064 {-989, -989, -989, -989, -989},  
10065 {-859, -859, -859, -859, -859}},  
10066 /* @.AC */  
10067 {{ DEF, DEF, DEF, DEF, DEF},  
10068 {-999, -999, -999, -999, -999},  
10069 {-499, -499, -499, -499, -499},  
10070 {-989, -989, -989, -989, -989},  
10071 {-789, -789, -789, -789, -789}},  
10072 /* @.AG */  
10073 {{ DEF, DEF, DEF, DEF, DEF},  
10074 {-1079,-1079,-1079,-1079,-1079},  
10075 {-569, -569, -569, -569, -569},  
10076 {-989, -989, -989, -989, -989},  
10077 {-859, -859, -859, -859, -859}},  
10078 /* @.AU */  
10079 {{ DEF, DEF, DEF, DEF, DEF},  
10080 {-1079,-1079,-1079,-1079,-1079},  
10081 {-719, -719, -719, -719, -719},  
10082 {-989, -989, -989, -989, -989},  
10083 {-909, -909, -909, -909, -909}}},  
10084 /* @.CA */  
10085 {{ DEF, DEF, DEF, DEF, DEF},  
10086 {-100, -100, -100, -100, -100},  
10087 {-100, -100, -100, -100, -100},  
10088 {-100, -100, -100, -100, -100},  
10089 {-100, -100, -100, -100, -100}},  
10090 /* @.CA */  
10091 {{ DEF, DEF, DEF, DEF, DEF},  
10092 {-1079,-1079,-1079,-1079,-1079},  
10093 {-569, -569, -569, -569, -569},  
10094 {-989, -989, -989, -989, -989},  
10095 {-859, -859, -859, -859, -859}},  
10096 /* @.CC */  
10097 {{ DEF, DEF, DEF, DEF, DEF},  
10098 {-999, -999, -999, -999, -999},  
10099 {-499, -499, -499, -499, -499},  
10100 {-989, -989, -989, -989, -989},  
10101 {-789, -789, -789, -789, -789}},  
10102 /* @.CG */  
10103 {{ DEF, DEF, DEF, DEF, DEF},  
10104 {-1079,-1079,-1079,-1079,-1079},  
10105 {-569, -569, -569, -569, -569},  
10106 {-989, -989, -989, -989, -989},  
10107 {-859, -859, -859, -859, -859}},  
10108 /* @.CU */  
10109 {{ DEF, DEF, DEF, DEF, DEF},  
10110 {-1079,-1079,-1079,-1079,-1079},  
10111 {-719, -719, -719, -719, -719},  
10112 {-989, -989, -989, -989, -989},  
10113 {-909, -909, -909, -909, -909}}},  
10114 /* @.G */  
10115 {{ DEF, DEF, DEF, DEF, DEF},  
10116 {-100, -100, -100, -100, -100},  
10117 {-100, -100, -100, -100, -100},  
10118 {-100, -100, -100, -100, -100},
```

```
10119 { -100, -100, -100, -100, -100}},  
10120 /* @.GA..CG */  
10121 {{ DEF, DEF, DEF, DEF, DEF},  
10122 {-1079, -1079, -1079, -1079, -1079},  
10123 { -569, -569, -569, -569, -569},  
10124 { -989, -989, -989, -989, -989},  
10125 { -859, -859, -859, -859, -859}},  
10126 /* @.GC..CG */  
10127 {{ DEF, DEF, DEF, DEF, DEF},  
10128 { -999, -999, -999, -999, -999},  
10129 { -499, -499, -499, -499, -499},  
10130 { -989, -989, -989, -989, -989},  
10131 { -789, -789, -789, -789, -789}},  
10132 /* @.GG..CG */  
10133 {{ DEF, DEF, DEF, DEF, DEF},  
10134 {-1079, -1079, -1079, -1079, -1079},  
10135 { -569, -569, -569, -569, -569},  
10136 { -989, -989, -989, -989, -989},  
10137 { -859, -859, -859, -859, -859}},  
10138 /* @.GU..CG */  
10139 {{ DEF, DEF, DEF, DEF, DEF},  
10140 {-1079, -1079, -1079, -1079, -1079},  
10141 { -719, -719, -719, -719, -719},  
10142 { -989, -989, -989, -989, -989},  
10143 { -909, -909, -909, -909, -909}},  
10144 /* @.U@..CG */  
10145 {{ DEF, DEF, DEF, DEF, DEF},  
10146 { -100, -100, -100, -100, -100},  
10147 { -100, -100, -100, -100, -100},  
10148 { -100, -100, -100, -100, -100},  
10149 { -100, -100, -100, -100, -100}},  
10150 /* @.UA..CG */  
10151 {{ DEF, DEF, DEF, DEF, DEF},  
10152 {-1079, -1079, -1079, -1079, -1079},  
10153 { -569, -569, -569, -569, -569},  
10154 { -989, -989, -989, -989, -989},  
10155 { -859, -859, -859, -859, -859}},  
10156 /* @.UC..CG */  
10157 {{ DEF, DEF, DEF, DEF, DEF},  
10158 { -999, -999, -999, -999, -999},  
10159 { -499, -499, -499, -499, -499},  
10160 { -989, -989, -989, -989, -989},  
10161 { -789, -789, -789, -789, -789}},  
10162 /* @.UG..CG */  
10163 {{ DEF, DEF, DEF, DEF, DEF},  
10164 {-1079, -1079, -1079, -1079, -1079},  
10165 { -569, -569, -569, -569, -569},  
10166 { -989, -989, -989, -989, -989},  
10167 { -859, -859, -859, -859, -859}},  
10168 /* @.UU..CG */  
10169 {{ DEF, DEF, DEF, DEF, DEF},  
10170 {-1079, -1079, -1079, -1079, -1079},  
10171 { -719, -719, -719, -719, -719},  
10172 { -989, -989, -989, -989, -989},  
10173 { -909, -909, -909, -909, -909}}},  
10174 /* @.@@..GC */  
10175 {{{ DEF, DEF, DEF, DEF, DEF},  
10176 { -100, -100, -100, -100, -100},  
10177 { -100, -100, -100, -100, -100},  
10178 { -100, -100, -100, -100, -100},  
10179 { -100, -100, -100, -100, -100}},  
10180 /* @.@@..GC */  
10181 {{ DEF, DEF, DEF, DEF, DEF},  
10182 { -569, -569, -569, -569, -569},  
10183 { -769, -769, -769, -769, -769},  
10184 { -759, -759, -759, -759, -759},  
10185 { -549, -549, -549, -549, -549}},  
10186 /* @.@@..GC */  
10187 {{ DEF, DEF, DEF, DEF, DEF},  
10188 { -929, -929, -929, -929, -929},  
10189 { -359, -359, -359, -359, -359},  
10190 { -789, -789, -789, -789, -789},  
10191 { -549, -549, -549, -549, -549}},  
10192 /* @.@@..GC */  
10193 {{ DEF, DEF, DEF, DEF, DEF},  
10194 { -609, -609, -609, -609, -609},  
10195 { -359, -359, -359, -359, -359},  
10196 { -669, -669, -669, -669, -669},  
10197 { -549, -549, -549, -549, -549}},  
10198 /* @.@@..GC */  
10199 {{ DEF, DEF, DEF, DEF, DEF},  
10200 { -929, -929, -929, -929, -929},  
10201 { -439, -439, -439, -439, -439},  
10202 { -789, -789, -789, -789, -789},  
10203 { -619, -619, -619, -619, -619}}},  
10204 /* @.@@..GC */  
10205 {{{ DEF, DEF, DEF, DEF, DEF},
```

```
10206 { -100, -100, -100, -100, -100},  
10207 { -100, -100, -100, -100, -100},  
10208 { -100, -100, -100, -100, -100},  
10209 { -100, -100, -100, -100, -100}},  
10210 /* @AA..GC */  
10211 {{ DEF, DEF, DEF, DEF, DEF},  
10212 { -569, -569, -569, -569, -569},  
10213 { -769, -769, -769, -769, -769},  
10214 { -759, -759, -759, -759, -759},  
10215 { -549, -549, -549, -549, -549}},  
10216 /* @AC..GC */  
10217 {{ DEF, DEF, DEF, DEF, DEF},  
10218 { -929, -929, -929, -929, -929},  
10219 { -359, -359, -359, -359, -359},  
10220 { -789, -789, -789, -789, -789},  
10221 { -549, -549, -549, -549, -549}},  
10222 /* @AG..GC */  
10223 {{ DEF, DEF, DEF, DEF, DEF},  
10224 { -609, -609, -609, -609, -609},  
10225 { -359, -359, -359, -359, -359},  
10226 { -669, -669, -669, -669, -669},  
10227 { -549, -549, -549, -549, -549}},  
10228 /* @AU..GC */  
10229 {{ DEF, DEF, DEF, DEF, DEF},  
10230 { -929, -929, -929, -929, -929},  
10231 { -439, -439, -439, -439, -439},  
10232 { -789, -789, -789, -789, -789},  
10233 { -619, -619, -619, -619, -619}}},  
10234 /* @C@..GC */  
10235 {{ DEF, DEF, DEF, DEF, DEF},  
10236 { -100, -100, -100, -100, -100},  
10237 { -100, -100, -100, -100, -100},  
10238 { -100, -100, -100, -100, -100},  
10239 { -100, -100, -100, -100, -100}},  
10240 /* @CA..GC */  
10241 {{ DEF, DEF, DEF, DEF, DEF},  
10242 { -569, -569, -569, -569, -569},  
10243 { -769, -769, -769, -769, -769},  
10244 { -759, -759, -759, -759, -759},  
10245 { -549, -549, -549, -549, -549}},  
10246 /* @CC..GC */  
10247 {{ DEF, DEF, DEF, DEF, DEF},  
10248 { -929, -929, -929, -929, -929},  
10249 { -359, -359, -359, -359, -359},  
10250 { -789, -789, -789, -789, -789},  
10251 { -549, -549, -549, -549, -549}},  
10252 /* @CG..GC */  
10253 {{ DEF, DEF, DEF, DEF, DEF},  
10254 { -609, -609, -609, -609, -609},  
10255 { -359, -359, -359, -359, -359},  
10256 { -669, -669, -669, -669, -669},  
10257 { -549, -549, -549, -549, -549}},  
10258 /* @CU..GC */  
10259 {{ DEF, DEF, DEF, DEF, DEF},  
10260 { -929, -929, -929, -929, -929},  
10261 { -439, -439, -439, -439, -439},  
10262 { -789, -789, -789, -789, -789},  
10263 { -619, -619, -619, -619, -619}}},  
10264 /* @G@..GC */  
10265 {{ DEF, DEF, DEF, DEF, DEF},  
10266 { -100, -100, -100, -100, -100},  
10267 { -100, -100, -100, -100, -100},  
10268 { -100, -100, -100, -100, -100},  
10269 { -100, -100, -100, -100, -100}},  
10270 /* @GA..GC */  
10271 {{ DEF, DEF, DEF, DEF, DEF},  
10272 { -569, -569, -569, -569, -569},  
10273 { -769, -769, -769, -769, -769},  
10274 { -759, -759, -759, -759, -759},  
10275 { -549, -549, -549, -549, -549}},  
10276 /* @GC..GC */  
10277 {{ DEF, DEF, DEF, DEF, DEF},  
10278 { -929, -929, -929, -929, -929},  
10279 { -359, -359, -359, -359, -359},  
10280 { -789, -789, -789, -789, -789},  
10281 { -549, -549, -549, -549, -549}},  
10282 /* @GG..GC */  
10283 {{ DEF, DEF, DEF, DEF, DEF},  
10284 { -609, -609, -609, -609, -609},  
10285 { -359, -359, -359, -359, -359},  
10286 { -669, -669, -669, -669, -669},  
10287 { -549, -549, -549, -549, -549}},  
10288 /* @GU..GC */  
10289 {{ DEF, DEF, DEF, DEF, DEF},  
10290 { -929, -929, -929, -929, -929},  
10291 { -439, -439, -439, -439, -439},  
10292 { -789, -789, -789, -789, -789}},
```

```

10293 { -619, -619, -619, -619, -619}}},  

10294 /* @.U@..GC */  

10295 {{ DEF, DEF, DEF, DEF, DEF},  

10296 { -100, -100, -100, -100, -100},  

10297 { -100, -100, -100, -100, -100},  

10298 { -100, -100, -100, -100, -100},  

10299 { -100, -100, -100, -100, -100}},  

10300 /* @.UA..GC */  

10301 {{ DEF, DEF, DEF, DEF, DEF},  

10302 { -569, -569, -569, -569, -569},  

10303 { -769, -769, -769, -769, -769},  

10304 { -759, -759, -759, -759, -759},  

10305 { -549, -549, -549, -549, -549}},  

10306 /* @.UC..GC */  

10307 {{ DEF, DEF, DEF, DEF, DEF},  

10308 { -929, -929, -929, -929, -929},  

10309 { -359, -359, -359, -359, -359},  

10310 { -789, -789, -789, -789, -789},  

10311 { -549, -549, -549, -549, -549}},  

10312 /* @.UG..GC */  

10313 {{ DEF, DEF, DEF, DEF, DEF},  

10314 { -609, -609, -609, -609, -609},  

10315 { -359, -359, -359, -359, -359},  

10316 { -669, -669, -669, -669, -669},  

10317 { -549, -549, -549, -549, -549}},  

10318 /* @.UU..GC */  

10319 {{ DEF, DEF, DEF, DEF, DEF},  

10320 { -929, -929, -929, -929, -929},  

10321 { -439, -439, -439, -439, -439},  

10322 { -789, -789, -789, -789, -789},  

10323 { -619, -619, -619, -619, -619}}},  

10324 /* @.@@..GU */  

10325 {{{ DEF, DEF, DEF, DEF, DEF},  

10326 { -100, -100, -100, -100, -100},  

10327 { -100, -100, -100, -100, -100},  

10328 { -100, -100, -100, -100, -100},  

10329 { -100, -100, -100, -100, -100}},  

10330 /* @.@A..GU */  

10331 {{ DEF, DEF, DEF, DEF, DEF},  

10332 { -479, -479, -479, -479, -479},  

10333 { -309, -309, -309, -309, -309},  

10334 { -389, -389, -389, -389, -389},  

10335 { -379, -379, -379, -379, -379}},  

10336 /* @.@C..GU */  

10337 {{ DEF, DEF, DEF, DEF, DEF},  

10338 { -649, -649, -649, -649, -649},  

10339 { -289, -289, -289, -289, -289},  

10340 { -739, -739, -739, -739, -739},  

10341 { -379, -379, -379, -379, -379}},  

10342 /* @.@G..GU */  

10343 {{ DEF, DEF, DEF, DEF, DEF},  

10344 { -649, -649, -649, -649, -649},  

10345 { -289, -289, -289, -289, -289},  

10346 { -739, -739, -739, -739, -739},  

10347 { -379, -379, -379, -379, -379}},  

10348 /* @.@U..GU */  

10349 {{ DEF, DEF, DEF, DEF, DEF},  

10350 { -649, -649, -649, -649, -649},  

10351 { -289, -289, -289, -289, -289},  

10352 { -739, -739, -739, -739, -739},  

10353 { -379, -379, -379, -379, -379}}},  

10354 /* @.A@..GU */  

10355 {{ DEF, DEF, DEF, DEF, DEF},  

10356 { -100, -100, -100, -100, -100},  

10357 { -100, -100, -100, -100, -100},  

10358 { -100, -100, -100, -100, -100},  

10359 { -100, -100, -100, -100, -100}},  

10360 /* @.AA..GU */  

10361 {{ DEF, DEF, DEF, DEF, DEF},  

10362 { -479, -479, -479, -479, -479},  

10363 { -309, -309, -309, -309, -309},  

10364 { -389, -389, -389, -389, -389},  

10365 { -379, -379, -379, -379, -379}},  

10366 /* @.AC..GU */  

10367 {{ DEF, DEF, DEF, DEF, DEF},  

10368 { -649, -649, -649, -649, -649},  

10369 { -289, -289, -289, -289, -289},  

10370 { -739, -739, -739, -739, -739},  

10371 { -379, -379, -379, -379, -379}},  

10372 /* @.AG..GU */  

10373 {{ DEF, DEF, DEF, DEF, DEF},  

10374 { -649, -649, -649, -649, -649},  

10375 { -289, -289, -289, -289, -289},  

10376 { -739, -739, -739, -739, -739},  

10377 { -379, -379, -379, -379, -379}},  

10378 /* @.AU..GU */  

10379 {{ DEF, DEF, DEF, DEF},  


```

```
10380 { -649, -649, -649, -649, -649},  
10381 { -289, -289, -289, -289, -289},  
10382 { -739, -739, -739, -739, -739},  
10383 { -379, -379, -379, -379, -379}}},  
10384 /* @.C@..GU */  
10385 {{ DEF, DEF, DEF, DEF, DEF},  
10386 { -100, -100, -100, -100, -100},  
10387 { -100, -100, -100, -100, -100},  
10388 { -100, -100, -100, -100, -100},  
10389 { -100, -100, -100, -100, -100}}},  
10390 /* @.CA..GU */  
10391 {{ DEF, DEF, DEF, DEF, DEF},  
10392 { -479, -479, -479, -479, -479},  
10393 { -309, -309, -309, -309, -309},  
10394 { -389, -389, -389, -389, -389},  
10395 { -379, -379, -379, -379, -379}}},  
10396 /* @.CC..GU */  
10397 {{ DEF, DEF, DEF, DEF, DEF},  
10398 { -649, -649, -649, -649, -649},  
10399 { -289, -289, -289, -289, -289},  
10400 { -739, -739, -739, -739, -739},  
10401 { -379, -379, -379, -379, -379}}},  
10402 /* @.CG..GU */  
10403 {{ DEF, DEF, DEF, DEF, DEF},  
10404 { -649, -649, -649, -649, -649},  
10405 { -289, -289, -289, -289, -289},  
10406 { -739, -739, -739, -739, -739},  
10407 { -379, -379, -379, -379, -379}}},  
10408 /* @.CU..GU */  
10409 {{ DEF, DEF, DEF, DEF, DEF},  
10410 { -649, -649, -649, -649, -649},  
10411 { -289, -289, -289, -289, -289},  
10412 { -739, -739, -739, -739, -739},  
10413 { -379, -379, -379, -379, -379}}},  
10414 /* @.G@..GU */  
10415 {{ DEF, DEF, DEF, DEF, DEF},  
10416 { -100, -100, -100, -100, -100},  
10417 { -100, -100, -100, -100, -100},  
10418 { -100, -100, -100, -100, -100},  
10419 { -100, -100, -100, -100, -100}}},  
10420 /* @.GA..GU */  
10421 {{ DEF, DEF, DEF, DEF, DEF},  
10422 { -479, -479, -479, -479, -479},  
10423 { -309, -309, -309, -309, -309},  
10424 { -389, -389, -389, -389, -389},  
10425 { -379, -379, -379, -379, -379}}},  
10426 /* @.GC..GU */  
10427 {{ DEF, DEF, DEF, DEF, DEF},  
10428 { -649, -649, -649, -649, -649},  
10429 { -289, -289, -289, -289, -289},  
10430 { -739, -739, -739, -739, -739},  
10431 { -379, -379, -379, -379, -379}}},  
10432 /* @.GG..GU */  
10433 {{ DEF, DEF, DEF, DEF, DEF},  
10434 { -649, -649, -649, -649, -649},  
10435 { -289, -289, -289, -289, -289},  
10436 { -739, -739, -739, -739, -739},  
10437 { -379, -379, -379, -379, -379}}},  
10438 /* @.GU..GU */  
10439 {{ DEF, DEF, DEF, DEF, DEF},  
10440 { -649, -649, -649, -649, -649},  
10441 { -289, -289, -289, -289, -289},  
10442 { -739, -739, -739, -739, -739},  
10443 { -379, -379, -379, -379, -379}}},  
10444 /* @.U@..GU */  
10445 {{ DEF, DEF, DEF, DEF, DEF},  
10446 { -100, -100, -100, -100, -100},  
10447 { -100, -100, -100, -100, -100},  
10448 { -100, -100, -100, -100, -100},  
10449 { -100, -100, -100, -100, -100}}},  
10450 /* @.UA..GU */  
10451 {{ DEF, DEF, DEF, DEF, DEF},  
10452 { -479, -479, -479, -479, -479},  
10453 { -309, -309, -309, -309, -309},  
10454 { -389, -389, -389, -389, -389},  
10455 { -379, -379, -379, -379, -379}}},  
10456 /* @.UC..GU */  
10457 {{ DEF, DEF, DEF, DEF, DEF},  
10458 { -649, -649, -649, -649, -649},  
10459 { -289, -289, -289, -289, -289},  
10460 { -739, -739, -739, -739, -739},  
10461 { -379, -379, -379, -379, -379}}},  
10462 /* @.UG..GU */  
10463 {{ DEF, DEF, DEF, DEF, DEF},  
10464 { -649, -649, -649, -649, -649},  
10465 { -289, -289, -289, -289, -289},  
10466 { -739, -739, -739, -739, -739}},
```

```

10467 { -379, -379, -379, -379, -379},  

10468 /* @.UU..GU */  

10469 {{ DEF, DEF, DEF, DEF, DEF},  

10470 { -649, -649, -649, -649, -649},  

10471 { -289, -289, -289, -289, -289},  

10472 { -739, -739, -739, -739, -739},  

10473 { -379, -379, -379, -379, -379}}},  

10474 /* @.@@..UG */  

10475 {{{ DEF, DEF, DEF, DEF, DEF},  

10476 { -100, -100, -100, -100, -100},  

10477 { -100, -100, -100, -100, -100},  

10478 { -100, -100, -100, -100, -100},  

10479 { -100, -100, -100, -100, -100}}},  

10480 /* @.CA..UG */  

10481 {{ DEF, DEF, DEF, DEF, DEF},  

10482 { -769, -769, -769, -769, -769},  

10483 { -529, -529, -529, -529, -529},  

10484 { -709, -709, -709, -709, -709},  

10485 { -599, -599, -599, -599, -599}}},  

10486 /* @.C@..UG */  

10487 {{ DEF, DEF, DEF, DEF, DEF},  

10488 { -839, -839, -839, -839, -839},  

10489 { -529, -529, -529, -529, -529},  

10490 { -859, -859, -859, -859, -859},  

10491 { -489, -489, -489, -489, -489}}},  

10492 /* @.G@..UG */  

10493 {{ DEF, DEF, DEF, DEF, DEF},  

10494 {-1009, -1009, -1009, -1009, -1009},  

10495 { -409, -409, -409, -409, -409},  

10496 { -969, -969, -969, -969, -969},  

10497 { -599, -599, -599, -599, -599}}},  

10498 /* @.EU..UG */  

10499 {{ DEF, DEF, DEF, DEF, DEF},  

10500 { -859, -859, -859, -859, -859},  

10501 { -529, -529, -529, -529, -529},  

10502 { -859, -859, -859, -859, -859},  

10503 { -409, -409, -409, -409, -409}}},  

10504 /* @.A@..UG */  

10505 {{ DEF, DEF, DEF, DEF, DEF},  

10506 { -100, -100, -100, -100, -100},  

10507 { -100, -100, -100, -100, -100},  

10508 { -100, -100, -100, -100, -100},  

10509 { -100, -100, -100, -100, -100}}},  

10510 /* @.AA..UG */  

10511 {{ DEF, DEF, DEF, DEF, DEF},  

10512 { -769, -769, -769, -769, -769},  

10513 { -529, -529, -529, -529, -529},  

10514 { -709, -709, -709, -709, -709},  

10515 { -599, -599, -599, -599, -599}}},  

10516 /* @.AC..UG */  

10517 {{ DEF, DEF, DEF, DEF, DEF},  

10518 { -839, -839, -839, -839, -839},  

10519 { -529, -529, -529, -529, -529},  

10520 { -859, -859, -859, -859, -859},  

10521 { -489, -489, -489, -489, -489}}},  

10522 /* @.AG..UG */  

10523 {{ DEF, DEF, DEF, DEF, DEF},  

10524 {-1009, -1009, -1009, -1009, -1009},  

10525 { -409, -409, -409, -409, -409},  

10526 { -969, -969, -969, -969, -969},  

10527 { -599, -599, -599, -599, -599}}},  

10528 /* @.AU..UG */  

10529 {{ DEF, DEF, DEF, DEF, DEF},  

10530 { -859, -859, -859, -859, -859},  

10531 { -529, -529, -529, -529, -529},  

10532 { -859, -859, -859, -859, -859},  

10533 { -409, -409, -409, -409, -409}}},  

10534 /* @.C@..UG */  

10535 {{ DEF, DEF, DEF, DEF, DEF},  

10536 { -100, -100, -100, -100, -100},  

10537 { -100, -100, -100, -100, -100},  

10538 { -100, -100, -100, -100, -100},  

10539 { -100, -100, -100, -100, -100}}},  

10540 /* @.CA..UG */  

10541 {{ DEF, DEF, DEF, DEF, DEF},  

10542 { -769, -769, -769, -769, -769},  

10543 { -529, -529, -529, -529, -529},  

10544 { -709, -709, -709, -709, -709},  

10545 { -599, -599, -599, -599, -599}}},  

10546 /* @.CC..UG */  

10547 {{ DEF, DEF, DEF, DEF, DEF},  

10548 { -839, -839, -839, -839, -839},  

10549 { -529, -529, -529, -529, -529},  

10550 { -859, -859, -859, -859, -859},  

10551 { -489, -489, -489, -489, -489}}},  

10552 /* @.CG..UG */  

10553 {{ DEF, DEF, DEF, DEF},

```

```
10554 {-1009,-1009,-1009,-1009,-1009},  
10555 {-409,-409,-409,-409,-409},  
10556 {-969,-969,-969,-969,-969},  
10557 {-599,-599,-599,-599,-599}},  
10558 /* @CU..UG */  
10559 {{ DEF, DEF, DEF, DEF, DEF},  
10560 {-859,-859,-859,-859,-859},  
10561 {-529,-529,-529,-529,-529},  
10562 {-859,-859,-859,-859,-859},  
10563 {-409,-409,-409,-409,-409}}},  
10564 /* @G@..UG */  
10565 {{ DEF, DEF, DEF, DEF, DEF},  
10566 {-100,-100,-100,-100,-100},  
10567 {-100,-100,-100,-100,-100},  
10568 {-100,-100,-100,-100,-100},  
10569 {-100,-100,-100,-100,-100}},  
10570 /* @GA..UG */  
10571 {{ DEF, DEF, DEF, DEF, DEF},  
10572 {-769,-769,-769,-769,-769},  
10573 {-529,-529,-529,-529,-529},  
10574 {-709,-709,-709,-709,-709},  
10575 {-599,-599,-599,-599,-599}},  
10576 /* @GC..UG */  
10577 {{ DEF, DEF, DEF, DEF, DEF},  
10578 {-839,-839,-839,-839,-839},  
10579 {-529,-529,-529,-529,-529},  
10580 {-859,-859,-859,-859,-859},  
10581 {-489,-489,-489,-489,-489}},  
10582 /* @GG..UG */  
10583 {{ DEF, DEF, DEF, DEF, DEF},  
10584 {-1009,-1009,-1009,-1009,-1009},  
10585 {-409,-409,-409,-409,-409},  
10586 {-969,-969,-969,-969,-969},  
10587 {-599,-599,-599,-599,-599}},  
10588 /* @GU..UG */  
10589 {{ DEF, DEF, DEF, DEF, DEF},  
10590 {-859,-859,-859,-859,-859},  
10591 {-529,-529,-529,-529,-529},  
10592 {-859,-859,-859,-859,-859},  
10593 {-409,-409,-409,-409,-409}}},  
10594 /* @U@..UG */  
10595 {{ DEF, DEF, DEF, DEF, DEF},  
10596 {-100,-100,-100,-100,-100},  
10597 {-100,-100,-100,-100,-100},  
10598 {-100,-100,-100,-100,-100},  
10599 {-100,-100,-100,-100,-100}},  
10600 /* @UA..UG */  
10601 {{ DEF, DEF, DEF, DEF, DEF},  
10602 {-769,-769,-769,-769,-769},  
10603 {-529,-529,-529,-529,-529},  
10604 {-709,-709,-709,-709,-709},  
10605 {-599,-599,-599,-599,-599}},  
10606 /* @UC..UG */  
10607 {{ DEF, DEF, DEF, DEF, DEF},  
10608 {-839,-839,-839,-839,-839},  
10609 {-529,-529,-529,-529,-529},  
10610 {-859,-859,-859,-859,-859},  
10611 {-489,-489,-489,-489,-489}},  
10612 /* @UG..UG */  
10613 {{ DEF, DEF, DEF, DEF, DEF},  
10614 {-1009,-1009,-1009,-1009,-1009},  
10615 {-409,-409,-409,-409,-409},  
10616 {-969,-969,-969,-969,-969},  
10617 {-599,-599,-599,-599,-599}},  
10618 /* @UU..UG */  
10619 {{ DEF, DEF, DEF, DEF, DEF},  
10620 {-859,-859,-859,-859,-859},  
10621 {-529,-529,-529,-529,-529},  
10622 {-859,-859,-859,-859,-859},  
10623 {-409,-409,-409,-409,-409}}},  
10624 /* @@@@..AU */  
10625 {{{ DEF, DEF, DEF, DEF, DEF},  
10626 {-100,-100,-100,-100,-100},  
10627 {-100,-100,-100,-100,-100},  
10628 {-100,-100,-100,-100,-100},  
10629 {-100,-100,-100,-100,-100}},  
10630 /* @@@A..AU */  
10631 {{ DEF, DEF, DEF, DEF, DEF},  
10632 {-479,-479,-479,-479,-479},  
10633 {-309,-309,-309,-309,-309},  
10634 {-389,-389,-389,-389,-389},  
10635 {-379,-379,-379,-379,-379}},  
10636 /* @@@C..AU */  
10637 {{ DEF, DEF, DEF, DEF, DEF},  
10638 {-649,-649,-649,-649,-649},  
10639 {-289,-289,-289,-289,-289},  
10640 {-739,-739,-739,-739,-739},
```

```

10641 { -379, -379, -379, -379, -379}},  

10642 /* @.G..AU */  

10643 {{ DEF, DEF, DEF, DEF},  

10644 { -649, -649, -649, -649, -649},  

10645 { -289, -289, -289, -289, -289},  

10646 { -739, -739, -739, -739, -739},  

10647 { -379, -379, -379, -379, -379}},  

10648 /* @.U..AU */  

10649 {{ DEF, DEF, DEF, DEF, DEF},  

10650 { -649, -649, -649, -649, -649},  

10651 { -289, -289, -289, -289, -289},  

10652 { -739, -739, -739, -739, -739},  

10653 { -379, -379, -379, -379, -379}}},  

10654 /* @.A@..AU */  

10655 {{ { DEF, DEF, DEF, DEF, DEF},  

10656 { -100, -100, -100, -100, -100},  

10657 { -100, -100, -100, -100, -100},  

10658 { -100, -100, -100, -100, -100},  

10659 { -100, -100, -100, -100, -100}}},  

10660 /* @.AA..AU */  

10661 {{ DEF, DEF, DEF, DEF, DEF},  

10662 { -479, -479, -479, -479, -479},  

10663 { -309, -309, -309, -309, -309},  

10664 { -389, -389, -389, -389, -389},  

10665 { -379, -379, -379, -379, -379}},  

10666 /* @.AC..AU */  

10667 {{ DEF, DEF, DEF, DEF, DEF},  

10668 { -649, -649, -649, -649, -649},  

10669 { -289, -289, -289, -289, -289},  

10670 { -739, -739, -739, -739, -739},  

10671 { -379, -379, -379, -379, -379}},  

10672 /* @.AG..AU */  

10673 {{ DEF, DEF, DEF, DEF, DEF},  

10674 { -649, -649, -649, -649, -649},  

10675 { -289, -289, -289, -289, -289},  

10676 { -739, -739, -739, -739, -739},  

10677 { -379, -379, -379, -379, -379}},  

10678 /* @.AU..AU */  

10679 {{ DEF, DEF, DEF, DEF, DEF},  

10680 { -649, -649, -649, -649, -649},  

10681 { -289, -289, -289, -289, -289},  

10682 { -739, -739, -739, -739, -739},  

10683 { -379, -379, -379, -379, -379}}},  

10684 /* @.CG..AU */  

10685 {{ { DEF, DEF, DEF, DEF, DEF},  

10686 { -100, -100, -100, -100, -100},  

10687 { -100, -100, -100, -100, -100},  

10688 { -100, -100, -100, -100, -100},  

10689 { -100, -100, -100, -100, -100}}},  

10690 /* @.CA..AU */  

10691 {{ DEF, DEF, DEF, DEF, DEF},  

10692 { -479, -479, -479, -479, -479},  

10693 { -309, -309, -309, -309, -309},  

10694 { -389, -389, -389, -389, -389},  

10695 { -379, -379, -379, -379, -379}},  

10696 /* @.CC..AU */  

10697 {{ DEF, DEF, DEF, DEF, DEF},  

10698 { -649, -649, -649, -649, -649},  

10699 { -289, -289, -289, -289, -289},  

10700 { -739, -739, -739, -739, -739},  

10701 { -379, -379, -379, -379, -379}},  

10702 /* @.CG..AU */  

10703 {{ DEF, DEF, DEF, DEF, DEF},  

10704 { -649, -649, -649, -649, -649},  

10705 { -289, -289, -289, -289, -289},  

10706 { -739, -739, -739, -739, -739},  

10707 { -379, -379, -379, -379, -379}},  

10708 /* @.CU..AU */  

10709 {{ DEF, DEF, DEF, DEF, DEF},  

10710 { -649, -649, -649, -649, -649},  

10711 { -289, -289, -289, -289, -289},  

10712 { -739, -739, -739, -739, -739},  

10713 { -379, -379, -379, -379, -379}}},  

10714 /* @.G@..AU */  

10715 {{ { DEF, DEF, DEF, DEF, DEF},  

10716 { -100, -100, -100, -100, -100},  

10717 { -100, -100, -100, -100, -100},  

10718 { -100, -100, -100, -100, -100},  

10719 { -100, -100, -100, -100, -100}}},  

10720 /* @.GA..AU */  

10721 {{ DEF, DEF, DEF, DEF, DEF},  

10722 { -479, -479, -479, -479, -479},  

10723 { -309, -309, -309, -309, -309},  

10724 { -389, -389, -389, -389, -389},  

10725 { -379, -379, -379, -379, -379}}},  

10726 /* @.GC..AU */  

10727 {{ DEF, DEF, DEF, DEF},

```

```
10728 { -649, -649, -649, -649, -649},  
10729 { -289, -289, -289, -289, -289},  
10730 { -739, -739, -739, -739, -739},  
10731 { -379, -379, -379, -379, -379}},  
10732 /* @.GG..AU */  
10733 {{ DEF, DEF, DEF, DEF, DEF},  
10734 { -649, -649, -649, -649, -649},  
10735 { -289, -289, -289, -289, -289},  
10736 { -739, -739, -739, -739, -739},  
10737 { -379, -379, -379, -379, -379}},  
10738 /* @.GU..AU */  
10739 {{ DEF, DEF, DEF, DEF, DEF},  
10740 { -649, -649, -649, -649, -649},  
10741 { -289, -289, -289, -289, -289},  
10742 { -739, -739, -739, -739, -739},  
10743 { -379, -379, -379, -379, -379}}},  
10744 /* @.U@..AU */  
10745 {{ DEF, DEF, DEF, DEF, DEF},  
10746 { -100, -100, -100, -100, -100},  
10747 { -100, -100, -100, -100, -100},  
10748 { -100, -100, -100, -100, -100},  
10749 { -100, -100, -100, -100, -100}}},  
10750 /* @.UA..AU */  
10751 {{ DEF, DEF, DEF, DEF, DEF},  
10752 { -479, -479, -479, -479, -479},  
10753 { -309, -309, -309, -309, -309},  
10754 { -389, -389, -389, -389, -389},  
10755 { -379, -379, -379, -379, -379}},  
10756 /* @.UC..AU */  
10757 {{ DEF, DEF, DEF, DEF, DEF},  
10758 { -649, -649, -649, -649, -649},  
10759 { -289, -289, -289, -289, -289},  
10760 { -739, -739, -739, -739, -739},  
10761 { -379, -379, -379, -379, -379}},  
10762 /* @.UG..AU */  
10763 {{ DEF, DEF, DEF, DEF, DEF},  
10764 { -649, -649, -649, -649, -649},  
10765 { -289, -289, -289, -289, -289},  
10766 { -739, -739, -739, -739, -739},  
10767 { -379, -379, -379, -379, -379}},  
10768 /* @.UU..AU */  
10769 {{ DEF, DEF, DEF, DEF, DEF},  
10770 { -649, -649, -649, -649, -649},  
10771 { -289, -289, -289, -289, -289},  
10772 { -739, -739, -739, -739, -739},  
10773 { -379, -379, -379, -379, -379}}},  
10774 /* @.@@..UA */  
10775 {{{ DEF, DEF, DEF, DEF, DEF},  
10776 { -100, -100, -100, -100, -100},  
10777 { -100, -100, -100, -100, -100},  
10778 { -100, -100, -100, -100, -100},  
10779 { -100, -100, -100, -100, -100}}},  
10780 /* @.@@..UA */  
10781 {{ DEF, DEF, DEF, DEF, DEF},  
10782 { -449, -449, -449, -449, -449},  
10783 { -479, -479, -479, -479, -479},  
10784 { -429, -429, -429, -429, -429},  
10785 { -329, -329, -329, -329, -329}},  
10786 /* @.@@..UA */  
10787 {{ DEF, DEF, DEF, DEF, DEF},  
10788 { -679, -679, -679, -679, -679},  
10789 { -559, -559, -559, -559, -559},  
10790 { -729, -729, -729, -729, -729},  
10791 { -189, -189, -189, -189, -189}},  
10792 /* @.@@..UA */  
10793 {{ DEF, DEF, DEF, DEF, DEF},  
10794 { -939, -939, -939, -939, -939},  
10795 { -249, -249, -249, -249, -249},  
10796 { -939, -939, -939, -939, -939},  
10797 { -329, -329, -329, -329, -329}},  
10798 /* @.@@..UA */  
10799 {{ DEF, DEF, DEF, DEF, DEF},  
10800 { -639, -639, -639, -639, -639},  
10801 { -229, -229, -229, -229, -229},  
10802 { -729, -729, -729, -729, -729},  
10803 { -190, -190, -190, -190, -190}}},  
10804 /* @.@@..UA */  
10805 {{{ DEF, DEF, DEF, DEF, DEF},  
10806 { -100, -100, -100, -100, -100},  
10807 { -100, -100, -100, -100, -100},  
10808 { -100, -100, -100, -100, -100},  
10809 { -100, -100, -100, -100, -100}}},  
10810 /* @.AA..UA */  
10811 {{ DEF, DEF, DEF, DEF, DEF},  
10812 { -449, -449, -449, -449, -449},  
10813 { -479, -479, -479, -479, -479},  
10814 { -429, -429, -429, -429, -429},
```

```
10815 { -329, -329, -329, -329, -329}},  
10816 /* @.AC..UA */  
10817 {{ DEF, DEF, DEF, DEF},  
10818 { -679, -679, -679, -679, -679},  
10819 { -559, -559, -559, -559, -559},  
10820 { -729, -729, -729, -729, -729},  
10821 { -189, -189, -189, -189, -189}},  
10822 /* @.AG..UA */  
10823 {{ DEF, DEF, DEF, DEF},  
10824 { -939, -939, -939, -939, -939},  
10825 { -249, -249, -249, -249, -249},  
10826 { -939, -939, -939, -939, -939},  
10827 { -329, -329, -329, -329, -329}},  
10828 /* @.AU..UA */  
10829 {{ DEF, DEF, DEF, DEF},  
10830 { -639, -639, -639, -639, -639},  
10831 { -229, -229, -229, -229, -229},  
10832 { -729, -729, -729, -729, -729},  
10833 { -190, -190, -190, -190, -190}}},  
10834 /* @.C@..UA */  
10835 {{{{ DEF, DEF, DEF, DEF, DEF},  
10836 { -100, -100, -100, -100, -100},  
10837 { -100, -100, -100, -100, -100},  
10838 { -100, -100, -100, -100, -100},  
10839 { -100, -100, -100, -100, -100}},  
10840 /* @.CA..UA */  
10841 {{ DEF, DEF, DEF, DEF},  
10842 { -449, -449, -449, -449, -449},  
10843 { -479, -479, -479, -479, -479},  
10844 { -429, -429, -429, -429, -429},  
10845 { -329, -329, -329, -329, -329}},  
10846 /* @.CC..UA */  
10847 {{ DEF, DEF, DEF, DEF},  
10848 { -679, -679, -679, -679, -679},  
10849 { -559, -559, -559, -559, -559},  
10850 { -729, -729, -729, -729, -729},  
10851 { -189, -189, -189, -189, -189}},  
10852 /* @.CG..UA */  
10853 {{ DEF, DEF, DEF, DEF},  
10854 { -939, -939, -939, -939, -939},  
10855 { -249, -249, -249, -249, -249},  
10856 { -939, -939, -939, -939, -939},  
10857 { -329, -329, -329, -329, -329}},  
10858 /* @.CU..UA */  
10859 {{ DEF, DEF, DEF, DEF},  
10860 { -639, -639, -639, -639, -639},  
10861 { -229, -229, -229, -229, -229},  
10862 { -729, -729, -729, -729, -729},  
10863 { -190, -190, -190, -190, -190}}},  
10864 /* @.G@..UA */  
10865 {{{ DEF, DEF, DEF, DEF, DEF},  
10866 { -100, -100, -100, -100, -100},  
10867 { -100, -100, -100, -100, -100},  
10868 { -100, -100, -100, -100, -100},  
10869 { -100, -100, -100, -100, -100}},  
10870 /* @.GA..UA */  
10871 {{ DEF, DEF, DEF, DEF},  
10872 { -449, -449, -449, -449, -449},  
10873 { -479, -479, -479, -479, -479},  
10874 { -429, -429, -429, -429, -429},  
10875 { -329, -329, -329, -329, -329}},  
10876 /* @.GC..UA */  
10877 {{ DEF, DEF, DEF, DEF},  
10878 { -679, -679, -679, -679, -679},  
10879 { -559, -559, -559, -559, -559},  
10880 { -729, -729, -729, -729, -729},  
10881 { -189, -189, -189, -189, -189}},  
10882 /* @.GG..UA */  
10883 {{ DEF, DEF, DEF, DEF},  
10884 { -939, -939, -939, -939, -939},  
10885 { -249, -249, -249, -249, -249},  
10886 { -939, -939, -939, -939, -939},  
10887 { -329, -329, -329, -329, -329}},  
10888 /* @.GU..UA */  
10889 {{ DEF, DEF, DEF, DEF},  
10890 { -639, -639, -639, -639, -639},  
10891 { -229, -229, -229, -229, -229},  
10892 { -729, -729, -729, -729, -729},  
10893 { -190, -190, -190, -190, -190}}},  
10894 /* @.U@..UA */  
10895 {{{ DEF, DEF, DEF, DEF},  
10896 { -100, -100, -100, -100, -100},  
10897 { -100, -100, -100, -100, -100},  
10898 { -100, -100, -100, -100, -100},  
10899 { -100, -100, -100, -100, -100}},  
10900 /* @.UA..UA */  
10901 {{ DEF, DEF, DEF, DEF},
```

```
10902 { -449, -449, -449, -449, -449},  
10903 { -479, -479, -479, -479, -479},  
10904 { -429, -429, -429, -429, -429},  
10905 { -329, -329, -329, -329, -329}},  
10906 /* @.UC..UA */  
10907 {{ DEF, DEF, DEF, DEF, DEF},  
10908 { -679, -679, -679, -679, -679},  
10909 { -559, -559, -559, -559, -559},  
10910 { -729, -729, -729, -729, -729},  
10911 { -189, -189, -189, -189, -189}},  
10912 /* @.UG..UA */  
10913 {{ DEF, DEF, DEF, DEF, DEF},  
10914 { -939, -939, -939, -939, -939},  
10915 { -249, -249, -249, -249, -249},  
10916 { -939, -939, -939, -939, -939},  
10917 { -329, -329, -329, -329, -329}},  
10918 /* @.UU..UA */  
10919 {{ DEF, DEF, DEF, DEF, DEF},  
10920 { -639, -639, -639, -639, -639},  
10921 { -229, -229, -229, -229, -229},  
10922 { -729, -729, -729, -729, -729},  
10923 { -190, -190, -190, -190, -190}}},  
10924 /* @.@@.. @ */  
10925 {{{ -100, -100, -100, -100, -100},  
10926 { -100, -100, -100, -100, -100},  
10927 { -100, -100, -100, -100, -100},  
10928 { -100, -100, -100, -100, -100},  
10929 { -100, -100, -100, -100, -100}},  
10930 /* @.@A.. @ */  
10931 {{ -100, -100, -100, -100, -100},  
10932 { -100, -100, -100, -100, -100},  
10933 { -100, -100, -100, -100, -100},  
10934 { -100, -100, -100, -100, -100},  
10935 { -100, -100, -100, -100, -100}},  
10936 /* @.C.. @ */  
10937 {{ -100, -100, -100, -100, -100},  
10938 { -100, -100, -100, -100, -100},  
10939 { -100, -100, -100, -100, -100},  
10940 { -100, -100, -100, -100, -100},  
10941 { -100, -100, -100, -100, -100}},  
10942 /* @.G.. @ */  
10943 {{ -100, -100, -100, -100, -100},  
10944 { -100, -100, -100, -100, -100},  
10945 { -100, -100, -100, -100, -100},  
10946 { -100, -100, -100, -100, -100},  
10947 { -100, -100, -100, -100, -100}},  
10948 /* @.GU.. @ */  
10949 {{ -100, -100, -100, -100, -100},  
10950 { -100, -100, -100, -100, -100},  
10951 { -100, -100, -100, -100, -100},  
10952 { -100, -100, -100, -100, -100},  
10953 { -100, -100, -100, -100, -100}}},  
10954 /* @.A@.. @ */  
10955 {{{ -100, -100, -100, -100, -100},  
10956 { -100, -100, -100, -100, -100},  
10957 { -100, -100, -100, -100, -100},  
10958 { -100, -100, -100, -100, -100},  
10959 { -100, -100, -100, -100, -100}},  
10960 /* @.AA.. @ */  
10961 {{ -100, -100, -100, -100, -100},  
10962 { -100, -100, -100, -100, -100},  
10963 { -100, -100, -100, -100, -100},  
10964 { -100, -100, -100, -100, -100},  
10965 { -100, -100, -100, -100, -100}},  
10966 /* @.AC.. @ */  
10967 {{ -100, -100, -100, -100, -100},  
10968 { -100, -100, -100, -100, -100},  
10969 { -100, -100, -100, -100, -100},  
10970 { -100, -100, -100, -100, -100},  
10971 { -100, -100, -100, -100, -100}},  
10972 /* @.AG.. @ */  
10973 {{ -100, -100, -100, -100, -100},  
10974 { -100, -100, -100, -100, -100},  
10975 { -100, -100, -100, -100, -100},  
10976 { -100, -100, -100, -100, -100},  
10977 { -100, -100, -100, -100, -100}},  
10978 /* @.AU.. @ */  
10979 {{ -100, -100, -100, -100, -100},  
10980 { -100, -100, -100, -100, -100},  
10981 { -100, -100, -100, -100, -100},  
10982 { -100, -100, -100, -100, -100},  
10983 { -100, -100, -100, -100, -100}}},  
10984 /* @.C@.. @ */  
10985 {{ -100, -100, -100, -100, -100},  
10986 { -100, -100, -100, -100, -100},  
10987 { -100, -100, -100, -100, -100},  
10988 { -100, -100, -100, -100, -100},
```

```
10989 { -100, -100, -100, -100, -100}},  
10990 /* @.CA.. @ */  
10991 {{ -100, -100, -100, -100, -100},  
10992 { -100, -100, -100, -100, -100},  
10993 { -100, -100, -100, -100, -100},  
10994 { -100, -100, -100, -100, -100},  
10995 { -100, -100, -100, -100, -100}},  
10996 /* @.CC.. @ */  
10997 {{ -100, -100, -100, -100, -100},  
10998 { -100, -100, -100, -100, -100},  
10999 { -100, -100, -100, -100, -100},  
11000 { -100, -100, -100, -100, -100},  
11001 { -100, -100, -100, -100, -100}},  
11002 /* @.CG.. @ */  
11003 {{ -100, -100, -100, -100, -100},  
11004 { -100, -100, -100, -100, -100},  
11005 { -100, -100, -100, -100, -100},  
11006 { -100, -100, -100, -100, -100},  
11007 { -100, -100, -100, -100, -100}},  
11008 /* @.CU.. @ */  
11009 {{ -100, -100, -100, -100, -100},  
11010 { -100, -100, -100, -100, -100},  
11011 { -100, -100, -100, -100, -100},  
11012 { -100, -100, -100, -100, -100},  
11013 { -100, -100, -100, -100, -100}}},  
11014 /* @.G@.. @ */  
11015 {{ { -100, -100, -100, -100, -100},  
11016 { -100, -100, -100, -100, -100},  
11017 { -100, -100, -100, -100, -100},  
11018 { -100, -100, -100, -100, -100},  
11019 { -100, -100, -100, -100, -100}},  
11020 /* @.GA.. @ */  
11021 {{ { -100, -100, -100, -100, -100},  
11022 { -100, -100, -100, -100, -100},  
11023 { -100, -100, -100, -100, -100},  
11024 { -100, -100, -100, -100, -100},  
11025 { -100, -100, -100, -100, -100}},  
11026 /* @.GC.. @ */  
11027 {{ { -100, -100, -100, -100, -100},  
11028 { -100, -100, -100, -100, -100},  
11029 { -100, -100, -100, -100, -100},  
11030 { -100, -100, -100, -100, -100},  
11031 { -100, -100, -100, -100, -100}},  
11032 /* @.GG.. @ */  
11033 {{ { -100, -100, -100, -100, -100},  
11034 { -100, -100, -100, -100, -100},  
11035 { -100, -100, -100, -100, -100},  
11036 { -100, -100, -100, -100, -100},  
11037 { -100, -100, -100, -100, -100}},  
11038 /* @.GU.. @ */  
11039 {{ { -100, -100, -100, -100, -100},  
11040 { -100, -100, -100, -100, -100},  
11041 { -100, -100, -100, -100, -100},  
11042 { -100, -100, -100, -100, -100},  
11043 { -100, -100, -100, -100, -100}}},  
11044 /* @.U@.. @ */  
11045 {{ { -100, -100, -100, -100, -100},  
11046 { -100, -100, -100, -100, -100},  
11047 { -100, -100, -100, -100, -100},  
11048 { -100, -100, -100, -100, -100},  
11049 { -100, -100, -100, -100, -100}},  
11050 /* @.UA.. @ */  
11051 {{ { -100, -100, -100, -100, -100},  
11052 { -100, -100, -100, -100, -100},  
11053 { -100, -100, -100, -100, -100},  
11054 { -100, -100, -100, -100, -100},  
11055 { -100, -100, -100, -100, -100}},  
11056 /* @.UC.. @ */  
11057 {{ { -100, -100, -100, -100, -100},  
11058 { -100, -100, -100, -100, -100},  
11059 { -100, -100, -100, -100, -100},  
11060 { -100, -100, -100, -100, -100},  
11061 { -100, -100, -100, -100, -100}},  
11062 /* @.UG.. @ */  
11063 {{ { -100, -100, -100, -100, -100},  
11064 { -100, -100, -100, -100, -100},  
11065 { -100, -100, -100, -100, -100},  
11066 { -100, -100, -100, -100, -100},  
11067 { -100, -100, -100, -100, -100}},  
11068 /* @.UU.. @ */  
11069 {{ { -100, -100, -100, -100, -100},  
11070 { -100, -100, -100, -100, -100},  
11071 { -100, -100, -100, -100, -100},  
11072 { -100, -100, -100, -100, -100},  
11073 { -100, -100, -100, -100, -100}}}}};  
11074  
11075
```

## 18.151 ViennaRNA/constraints/basic.h File Reference

Functions and data structures for constraining secondary structure predictions and evaluation.

Include dependency graph for basic.h: This graph shows which files directly or indirectly include this file:

### Macros

- `#define VRNA_CONSTRAINT_FILE 0`  
*Flag for `vrna_constraints_add()` to indicate that constraints are present in a text file.*
- `#define VRNA_CONSTRAINT_SOFT_MFE 0`  
*Indicate generation of constraints for MFE folding.*
- `#define VRNA_CONSTRAINT_SOFT_PF VRNA_OPTION_PF`  
*Indicate generation of constraints for partition function computation.*
- `#define VRNA_DECOMP_PAIR_HP (unsigned char)1`  
*Flag passed to generic softt constraints callback to indicate hairpin loop decomposition step.*
- `#define VRNA_DECOMP_PAIR_IL (unsigned char)2`  
*Indicator for interior loop decomposition step.*
- `#define VRNA_DECOMP_PAIR_DL (unsigned char)3`  
*Indicator for multibranch loop decomposition step.*
- `#define VRNA_DECOMP_DL_DL_DL_DL (unsigned char)5`  
*Indicator for decomposition of multibranch loop part.*
- `#define VRNA_DECOMP_DL_STEM (unsigned char)6`  
*Indicator for decomposition of multibranch loop part.*
- `#define VRNA_DECOMP_DL_DL (unsigned char)7`  
*Indicator for decomposition of multibranch loop part.*
- `#define VRNA_DECOMP_DL_UP (unsigned char)8`  
*Indicator for decomposition of multibranch loop part.*
- `#define VRNA_DECOMP_DL_DL_STEM (unsigned char)9`  
*Indicator for decomposition of multibranch loop part.*
- `#define VRNA_DECOMP_DL_COAXIAL (unsigned char)10`  
*Indicator for decomposition of multibranch loop part.*
- `#define VRNA_DECOMP_DL_COAXIAL_ENC (unsigned char)11`  
*Indicator for decomposition of multibranch loop part.*
- `#define VRNA_DECOMP_EXT_EXT (unsigned char)12`  
*Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_UP (unsigned char)13`  
*Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_STEM (unsigned char)14`  
*Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_EXT_EXT (unsigned char)15`  
*Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_STEM_EXT (unsigned char)16`  
*Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_STEM_OUTSIDE (unsigned char)17`  
*Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_EXT_STEM (unsigned char)18`  
*Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_EXT_STEM1 (unsigned char)19`  
*Indicator for decomposition of exterior loop part.*

## Functions

- void `vrna_constraints_add` (`vrna_fold_compound_t` \*vc, const char \*constraint, unsigned int options)  
*Add constraints to a `vrna_fold_compound_t` data structure.*

### 18.151.1 Detailed Description

Functions and data structures for constraining secondary structure predictions and evaluation.

## 18.152 basic.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_H
2 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_H
3
4 #include <ViennaRNA/fold_compound.h>
5
99 #define VRNA_CONSTRAINT_FILE      0
100
107 #define VRNA_CONSTRAINT_SOFT_MFE  0
108
115 #define VRNA_CONSTRAINT_SOFT_PF   VRNA_OPTION_PF
116
129 #define VRNA_DECOMP_PAIR_HP      (unsigned char)1
130
144 #define VRNA_DECOMP_PAIR_IL      (unsigned char)2
145
159 #define VRNA_DECOMP_PAIR_ML      (unsigned char)3
160 #define VRNA_DECOMP_PAIR_ML_EXT  (unsigned char)23
161
162 #define VRNA_DECOMP_PAIR_ML_OUTSIDE (unsigned char)4
176 #define VRNA_DECOMP_ML_ML_ML    (unsigned char)5
177
191 #define VRNA_DECOMP_ML_STEM     (unsigned char)6
192
206 #define VRNA_DECOMP_ML_ML      (unsigned char)7
207
222 #define VRNA_DECOMP_ML_UP      (unsigned char)8
223
238 #define VRNA_DECOMP_ML_ML_STEM (unsigned char)9
239
254 #define VRNA_DECOMP_ML_COAXIAL (unsigned char)10
255
270 #define VRNA_DECOMP_ML_COAXIAL_ENC (unsigned char)11
271
286 #define VRNA_DECOMP_EXT_EXT    (unsigned char)12
287
302 #define VRNA_DECOMP_EXT_UP     (unsigned char)13
303
317 #define VRNA_DECOMP_EXT_STEM   (unsigned char)14
318
332 #define VRNA_DECOMP_EXT_EXT_EXT (unsigned char)15
333
348 #define VRNA_DECOMP_EXT_STEM_EXT (unsigned char)16
349
356 #define VRNA_DECOMP_EXT_STEM_OUTSIDE (unsigned char)17
357
372 #define VRNA_DECOMP_EXT_EXT_STEM (unsigned char)18
373
389 #define VRNA_DECOMP_EXT_EXT_STEM1 (unsigned char)19
390
391 #define VRNA_DECOMP_EXT_STEM_EXT1 (unsigned char)20
392
393 #define VRNA_DECOMP_EXT_L       (unsigned char)21
394 #define VRNA_DECOMP_EXT_EXT_L  (unsigned char)22
395
439 void vrna_constraints_add(vrna_fold_compound_t *vc,
440                           const char *constraint,
441                           unsigned int options);
442
443
444 #endif

```

## 18.153 ViennaRNA/datastructures/basic.h File Reference

Various data structures and pre-processor macros.

Include dependency graph for basic.h: This graph shows which files directly or indirectly include this file:

## Data Structures

- struct [vrna\\_basepair\\_s](#)  
*Base pair data structure used in subopt.c.* [More...](#)
- struct [vrna\\_cpair\\_s](#)  
*this datastructure is used as input parameter in functions of PS\_dot.c* [More...](#)
- struct [vrna\\_color\\_s](#)
- struct [vrna\\_data\\_linear\\_s](#)
- struct [vrna\\_sect\\_s](#)  
*Stack of partial structures for backtracking.* [More...](#)
- struct [vrna\\_bp\\_stack\\_s](#)  
*Base pair stack element.* [More...](#)
- struct [pu\\_contrib](#)  
*contributions to p\_u* [More...](#)
- struct [interact](#)  
*interaction data structure for RNAup* [More...](#)
- struct [pu\\_out](#)  
*Collection of all free\_energy of beeing unpaired values for output.* [More...](#)
- struct [constrain](#)  
*constraints for cofolding* [More...](#)
- struct [duplexT](#)  
*Data structure for RNAduplex.* [More...](#)
- struct [node](#)  
*Data structure for RNAsnoo (fold energy list)* [More...](#)
- struct [snoopT](#)  
*Data structure for RNAsnoo.* [More...](#)
- struct [dupVar](#)  
*Data structure used in RNApkplex.* [More...](#)

## Typedefs

- typedef struct [vrna\\_basepair\\_s](#) **vrna\_basepair\_t**  
*Typename for the base pair repesenting data structure vrna\_basepair\_s.*
- typedef struct [vrna\\_elem\\_prob\\_s](#) **vrna\_plist\_t**  
*Typename for the base pair list repesenting data structure vrna\_elem\_prob\_s.*
- typedef struct [vrna\\_bp\\_stack\\_s](#) **vrna\_bp\_stack\_t**  
*Typename for the base pair stack repesenting data structure vrna\_bp\_stack\_s.*
- typedef struct [vrna\\_cpair\\_s](#) **vrna\_cpair\_t**  
*Typename for data structure vrna\_cpair\_s.*
- typedef struct [vrna\\_sect\\_s](#) **vrna\_sect\_t**  
*Typename for stack of partial structures vrna\_sect\_s.*
- typedef double **FLT\_OR\_DBL**  
*Typename for floating point number in partition function computations.*
- typedef struct [vrna\\_basepair\\_s](#) **PAIR**  
*Old typename of vrna\_basepair\_s.*
- typedef struct [vrna\\_elem\\_prob\\_s](#) **plist**  
*Old typename of vrna\_elem\_prob\_s.*
- typedef struct [vrna\\_cpair\\_s](#) **cpair**  
*Old typename of vrna\_cpair\_s.*
- typedef struct [vrna\\_sect\\_s](#) **sect**  
*Old typename of vrna\_sect\_s.*
- typedef struct [vrna\\_bp\\_stack\\_s](#) **bondT**

- **Old typename of `vrna_bp_stack_s`.**
- **typedef struct `pu_contrib` `pu_contrib`**  
*contributions to p\_u*
- **typedef struct `interact` `interact`**  
*interaction data structure for RNAup*
- **typedef struct `pu_out` `pu_out`**  
*Collection of all free\_energy of beeing unpaired values for output.*
- **typedef struct `constraint` `constraint`**  
*constraints for cofolding*
- **typedef struct `node` `folden`**  
*Data structure for RNAsnoop (fold energy list)*
- **typedef struct `dupVar` `dupVar`**  
*Data structure used in RNApkplex.*

## Functions

- void `vrna_C11_features` (void)  
*Dummy symbol to check whether the library was build using C11/C++11 features.*

### 18.153.1 Detailed Description

Various data structures and pre-processor macros.

## 18.154 basic.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_DATA_STRUCTURES_H
2 #define VIENNA_RNA_PACKAGE_DATA_STRUCTURES_H
3
4 /* below are several convenience typedef's we use throughout the ViennaRNA library */
5
6 typedef struct vrna_basepair_s vrna_basepair_t;
7
8 typedef struct vrna_elem_prob_s vrna plist_t;
9
10 typedef struct vrna_bp_stack_s vrna_bp_stack_t;
11
12 typedef struct vrna_cpair_s vrna_cpair_t;
13
14 typedef struct vrna_sect_s vrna_sect_t;
15
16 typedef struct vrna_data_linear_s vrna_data_lin_t;
17
18 typedef struct vrna_color_s vrna_color_t;
19
20 #ifdef USE_FLOAT_PF
21     typedef float FLT_OR_DBL;
22 #else
23     typedef double FLT_OR_DBL;
24 #endif
25
26
27 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
28
29 /* the following typedefs are for backward compatibility only */
30
31 typedef struct vrna_basepair_s PAIR;
32
33 typedef struct vrna_elem_prob_s plist;
34
35 typedef struct vrna_cpair_s cpair;
36
37 typedef struct vrna_sect_s sect;
38
39 typedef struct vrna_bp_stack_s bondT;
40
41 #endif
42
43 #include <ViennaRNA/params/constants.h>
44 #include <ViennaRNA/fold_compound.h>
45 #include <ViennaRNA/model.h>
```

```
85 #include <ViennaRNA/params/basic.h>
86 #include <ViennaRNA/dp_matrices.h>
87 #include <ViennaRNA/constraints/hard.h>
88 #include <ViennaRNA/constraints/soft.h>
89 #include <ViennaRNA/grammar.h>
90 #include "ViennaRNA/structured_domains.h"
91 #include "ViennaRNA/unstructured_domains.h"
92 #include "ViennaRNA/utils/structures.h"
93 */
94 /**
95 * ##### Here are the type definitions of various datastructures
96 * shared among the Vienna RNA Package
97 * #####
98 */
99 */
100
104 struct vrna_basepair_s {
105     int i;
106     int j;
107 };
108
112 struct vrna_cpair_s {
113     int i, j, mfe;
114     float p, hue, sat;
115     int type;
116 };
117
118 struct vrna_color_s {
119     float hue;
120     float sat;
121     float bri;
122 };
123
124 struct vrna_data_linear_s {
125     unsigned int position;
126     float value;
127     vrna_color_t color;
128 };
129
130
134 struct vrna_sect_s {
135     int i;
136     int j;
137     int ml;
138 };
139
143 struct vrna_bp_stack_s {
144     unsigned int i;
145     unsigned int j;
146 };
147
148
149 /**
150 * ##### RNAup data structures
151 * #####
152 */
153 */
154
158 typedef struct pu_contrib {
159     double **H;
160     double **I;
161     double **M;
162     double **E;
163     int length;
164     int w;
165 } pu_contrib;
166
170 typedef struct interact {
171     double *Pi;
172     double *Gi;
173     double Gikjl;
175     double Gikjl_wo;
176     int i;
177     int k;
178     int j;
179     int l;
180     int length;
181 } interact;
182
186 typedef struct pu_out {
187     int len;
188     int u_vals;
189     int contribs;
190     char **header;
191     double **u_values;
192 } pu_out;
193
```

```
197 typedef struct constrain {
198     int *index;
199     char *ptype;
200 } constraint;
201
202 /*
203 * ##### RNAduplex data structures
204 * #####
205 */
206
207
211 typedef struct {
212     int i;
213     int j;
214     int end;
215     char *structure;
216     double energy;
217     double energy_backtrack;
218     double opening_backtrack_x;
219     double opening_backtrack_y;
220     int offset;
221     double dG1;
222     double dG2;
223     double ddG;
224     int tb;
225     int te;
226     int qb;
227     int qe;
228 } duplexT;
229
230 /*
231 * ##### RNAsnoop data structures
232 * #####
233 */
234
235
239 typedef struct node {
240     int k;
241     int energy;
242     struct node *next;
243 } folden;
244
248 typedef struct {
249     int i;
250     int j;
251     int u;
252     char *structure;
253     float energy;
254     float Duplex_El;
255     float Duplex_Er;
256     float Loop_E;
257     float Loop_D;
258     float pscd;
259     float psct;
260     float pscg;
261     float Duplex_Ol;
262     float Duplex_Or;
263     float Duplex_Ot;
264     float fullStemEnergy;
265 } snoopT;
266
267
268 /*
269 * ##### PKplex data structures
270 * #####
271 */
272
273
277 typedef struct dupVar {
278     int i;
279     int j;
280     int end;
281     char *pk_helix;
282     char *structure;
283     double energy;
284     int offset;
285     double dG1;
286     double dG2;
287     double ddG;
288     int tb;
289     int te;
290     int qb;
291     int qe;
292     int inactive;
293     int processed;
294 } dupVar;
295
```

```

326 #ifndef VRNA_DISABLE_C11_FEATURES
327 void vrna_C11_features(void);
328
329
330 #endif
331
337 #endif

```

## 18.155 ViennaRNA/params/basic.h File Reference

Functions to deal with sets of energy parameters.

Include dependency graph for basic.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [vrna\\_param\\_s](#)  
*The datastructure that contains temperature scaled energy parameters. [More...](#)*
- struct [vrna\\_exp\\_param\\_s](#)  
*The data structure that contains temperature scaled Boltzmann weights of the energy parameters. [More...](#)*

### Typedefs

- [typedef struct vrna\\_param\\_s vrna\\_param\\_t](#)  
*Typename for the free energy parameter data structure [vrna\\_params](#).*
- [typedef struct vrna\\_exp\\_param\\_s vrna\\_exp\\_param\\_t](#)  
*Typename for the Boltzmann factor data structure [vrna\\_exp\\_params](#).*
- [typedef struct vrna\\_param\\_s paramT](#)  
*Old typename of [vrna\\_param\\_s](#).*
- [typedef struct vrna\\_exp\\_param\\_s pf\\_paramT](#)  
*Old typename of [vrna\\_exp\\_param\\_s](#).*

### Functions

- [vrna\\_param\\_t \\* vrna\\_params \(vrna\\_md\\_t \\*md\)](#)  
*Get a data structure containing prescaled free energy parameters.*
- [vrna\\_param\\_t \\* vrna\\_params\\_copy \(vrna\\_param\\_t \\*par\)](#)  
*Get a copy of the provided free energy parameters.*
- [vrna\\_exp\\_param\\_t \\* vrna\\_exp\\_params \(vrna\\_md\\_t \\*md\)](#)  
*Get a data structure containing prescaled free energy parameters already transformed to Boltzmann factors.*
- [vrna\\_exp\\_param\\_t \\* vrna\\_exp\\_params\\_comparative \(unsigned int n\\_seq, vrna\\_md\\_t \\*md\)](#)  
*Get a data structure containing prescaled free energy parameters already transformed to Boltzmann factors (alifold version)*
- [vrna\\_exp\\_param\\_t \\* vrna\\_exp\\_params\\_copy \(vrna\\_exp\\_param\\_t \\*par\)](#)  
*Get a copy of the provided free energy parameters (provided as Boltzmann factors)*
- [void vrna\\_params\\_subst \(vrna\\_fold\\_compound\\_t \\*vc, vrna\\_param\\_t \\*par\)](#)  
*Update/Reset energy parameters data structure within a [vrna\\_fold\\_compound\\_t](#).*
- [void vrna\\_exp\\_params\\_subst \(vrna\\_fold\\_compound\\_t \\*vc, vrna\\_exp\\_param\\_t \\*params\)](#)  
*Update the energy parameters for subsequent partition function computations.*
- [void vrna\\_exp\\_params\\_rescale \(vrna\\_fold\\_compound\\_t \\*vc, double \\*mfe\)](#)  
*Rescale Boltzmann factors for partition function computations.*
- [void vrna\\_params\\_reset \(vrna\\_fold\\_compound\\_t \\*vc, vrna\\_md\\_t \\*md\\_p\)](#)  
*Reset free energy parameters within a [vrna\\_fold\\_compound\\_t](#) according to provided, or default model details.*
- [void vrna\\_exp\\_params\\_reset \(vrna\\_fold\\_compound\\_t \\*vc, vrna\\_md\\_t \\*md\\_p\)](#)  
*Reset Boltzmann factors for partition function computations within a [vrna\\_fold\\_compound\\_t](#) according to provided, or default model details.*

- `vrna_exp_param_t * get_scaled_pf_parameters (void)`
- `vrna_exp_param_t * get_boltzmann_factors (double temperature, double betaScale, vrna_md_t md, double pf_scale)`  
*Get precomputed Boltzmann factors of the loop type dependent energy contributions with independent thermodynamic temperature.*
- `vrna_exp_param_t * get_boltzmann_factor_copy (vrna_exp_param_t *parameters)`  
*Get a copy of already precomputed Boltzmann factors.*
- `vrna_exp_param_t * get_scaled_alipf_parameters (unsigned int n_seq)`  
*Get precomputed Boltzmann factors of the loop type dependent energy contributions (alifold variant)*
- `vrna_exp_param_t * get_boltzmann_factors_ali (unsigned int n_seq, double temperature, double betaScale, vrna_md_t md, double pf_scale)`  
*Get precomputed Boltzmann factors of the loop type dependent energy contributions (alifold variant) with independent thermodynamic temperature.*
- `vrna_param_t * scale_parameters (void)`  
*Get precomputed energy contributions for all the known loop types.*
- `vrna_param_t * get_scaled_parameters (double temperature, vrna_md_t md)`  
*Get precomputed energy contributions for all the known loop types.*

### 18.155.1 Detailed Description

Functions to deal with sets of energy parameters.

## 18.156 basic.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PARAMS_H
2 #define VIENNA_RNA_PACKAGE_PARAMS_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(_clang_)
6 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 #  define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
36 typedef struct vrna_param_s vrna_param_t;
38 typedef struct vrna_exp_param_s vrna_exp_param_t;
39
40 #include <ViennaRNA/params/constants.h>
41 #include <ViennaRNA/datastructures/basic.h>
42 #include <ViennaRNA/fold_compound.h>
43 #include <ViennaRNA/model.h>
44
45 #define VRNA_GQUAD_MAX_STACK_SIZE    7
46 #define VRNA_GQUAD_MIN_STACK_SIZE    2
47 #define VRNA_GQUAD_MAX_LINKER_LENGTH 15
48 #define VRNA_GQUAD_MIN_LINKER_LENGTH 1
49 #define VRNA_GQUAD_MIN_BOX_SIZE      (((4 * VRNA_GQUAD_MIN_STACK_SIZE) + \
50                                         (3 * VRNA_GQUAD_MIN_LINKER_LENGTH)) \
51 #define VRNA_GQUAD_MAX_BOX_SIZE      (((4 * VRNA_GQUAD_MAX_STACK_SIZE) + \
52                                         (3 * VRNA_GQUAD_MAX_LINKER_LENGTH)) \
53
57 struct vrna_param_s {
58     int         id;
59     int         stack[NBPAIRS + 1][NBPAIRS + 1];
60     int         hairpin[31];
61     int         bulge[MAXLOOP + 1];
62     int         internal_loop[MAXLOOP + 1];
63     int         mismatchExt[NBPAIRS + 1][5][5];
64     int         mismatchI[NBPAIRS + 1][5][5];
65     int         mismatchInI[NBPAIRS + 1][5][5];
66     int         mismatch23I[NBPAIRS + 1][5][5];
67     int         mismatchH[NBPAIRS + 1][5][5];
68     int         mismatchM[NBPAIRS + 1][5][5];
69     int         dangle5[NBPAIRS + 1][5];
70     int         dangle3[NBPAIRS + 1][5];

```

```

71 int      int11[NBPAIRS + 1][NBPAIRS + 1][5][5];
72 int      int21[NBPAIRS + 1][NBPAIRS + 1][5][5][5];
73 int      int22[NBPAIRS + 1][NBPAIRS + 1][5][5][5][5];
74 int      ninio[5];
75 double   lxc;
76 int      MLbase;
77 int      MLintern[NBPAIRS + 1];
78 int      MLclosing;
79 int      TerminalAU;
80 int      DuplexInit;
81 int      Tetraloop_E[200];
82 char     Tetraloops[1401];
83 int      Triloop_E[40];
84 char     Triloops[241];
85 int      Hexaloop_E[40];
86 char     Hexaloops[1801];
87 int      TripleC;
88 int      MultipleCA;
89 int      MultipleCB;
90 int      gquad[VRNA_GQUAD_MAX_STACK_SIZE + 1][3 * VRNA_GQUAD_MAX_LINKER_LENGTH + 1];
91 int      gquadLayerMismatch;
92 int      gquadLayerMismatchMax;
93
94 double   temperature;
95 vrna_md_t model_details;
96 char     param_file[256];
97 };
98
103 struct vrna_exp_param_s {
104     int      id;
105    double   expstack[NBPAIRS + 1][NBPAIRS + 1];
106    double   expairpin[31];
107    double   expbulge[MAXLOOP + 1];
108    double   expinternal[MAXLOOP + 1];
109    double   expmismatchExt[NBPAIRS + 1][5][5];
110    double   expmismatchI[NBPAIRS + 1][5][5];
111    double   expmismatch23I[NBPAIRS + 1][5][5];
112    double   expmismatchInI[NBPAIRS + 1][5][5];
113    double   expmismatchH[NBPAIRS + 1][5][5];
114    double   expmismatchM[NBPAIRS + 1][5][5];
115    double   expdangle5[NBPAIRS + 1][5];
116    double   expdangle3[NBPAIRS + 1][5];
117    double   expint11[NBPAIRS + 1][NBPAIRS + 1][5][5];
118    double   expint21[NBPAIRS + 1][NBPAIRS + 1][5][5][5];
119    double   expint22[NBPAIRS + 1][NBPAIRS + 1][5][5][5][5];
120    double   expnino[5][MAXLOOP + 1];
121    double   lxc;
122    double   expMLbase;
123    double   expMLintern[NBPAIRS + 1];
124    double   expMLclosing;
125    double   expTermAU;
126    double   expDuplexInit;
127    double   exptetra[40];
128    double   exptri[40];
129    double   exphex[40];
130    char     Tetraloops[1401];
131    double   expTriloop[40];
132    char     Triloops[241];
133    char     Hexaloops[1801];
134    double   expTripleC;
135    double   expMultipleCA;
136    double   expMultipleCB;
137    double   expgquad[VRNA_GQUAD_MAX_STACK_SIZE + 1][3 * VRNA_GQUAD_MAX_LINKER_LENGTH + 1];
138    double   expgquadLayerMismatch;
139    int      gquadLayerMismatchMax;
140
141    double   kT;
142    double   pf_scale;
143    double   temperature;
144    double   alpha;
145    vrna_md_t model_details;
146    char     param_file[256];
147 };
148
149 vrna_param_t *
150 vrna_params(vrna_md_t *md);
151
152
153
154 vrna_param_t *
155 vrna_params_copy(vrna_param_t *par);
156
157
158
159 vrna_param_t *
160 vrna_params_copy(vrna_param_t *par);
161
162 vrna_exp_param_t *
163 vrna_exp_params(vrna_md_t *md);
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214

```

```
228 vrna_exp_param_t *
229 vrna_exp_params_comparative(unsigned int n_seq,
230                               vrna_md_t *md);
231
232
233 vrna_exp_param_t *
234 vrna_exp_params_copy(vrna_exp_param_t *par);
235
236 void
237 vrna_params_subst(vrna_fold_compound_t *vc,
238                     vrna_param_t *par);
239
240
241 void
242 vrna_exp_params_subst(vrna_fold_compound_t *vc,
243                        vrna_exp_param_t *params);
244
245
246 void
247 vrna_exp_params_rescale(vrna_fold_compound_t *vc,
248                          double *mfe);
249
250
251 void
252 vrna_params_reset(vrna_fold_compound_t *vc,
253                     vrna_md_t *md_p);
254
255
256 void
257 vrna_exp_params_reset(vrna_fold_compound_t *vc,
258                        vrna_md_t *md_p);
259
260
261 void
262 vrna_params_prepare(vrna_fold_compound_t *vc,
263                      unsigned int options);
264
265
266 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
267
268 typedef struct vrna_param_s paramT;
269
270 typedef struct vrna_exp_param_s pf_paramT;
271
272 DEPRECATED(vrna_param_t *get_parameter_copy(vrna_param_t *par),
273            "Use vrna_params_copy() instead");
274
275 DEPRECATED(vrna_exp_param_t *get_scaled_pf_parameters(void),
276            "Use vrna_params() instead");
277
278 DEPRECATED(vrna_exp_param_t *get_boltzmann_factors(double temperature,
279                                                    double betaScale,
280                                                    vrna_md_t md,
281                                                    double pf_scale),
282            "Use vrna_exp_params() instead");
283
284 DEPRECATED(vrna_exp_param_t *get_boltzmann_factor_copy(vrna_exp_param_t *parameters),
285            "Use vrna_exp_params_copy() instead");
286
287 DEPRECATED(vrna_exp_param_t *get_scaled_alipf_parameters(unsigned int n_seq),
288            "Use vrna_exp_params_comparative() instead");
289
290 DEPRECATED(vrna_exp_param_t *get_boltzmann_factors_ali(unsigned int n_seq,
291                                                    double temperature,
292                                                    double betaScale,
293                                                    vrna_md_t md,
294                                                    double pf_scale),
295            "Use vrna_exp_params_comparative() instead");
296
297 DEPRECATED(vrna_param_t *scale_parameters(void),
298            "Use vrna_params() instead");
299
300 DEPRECATED(vrna_param_t *get_scaled_parameters(double temperature,
301                                                vrna_md_t md),
302            "Use vrna_params() instead");
303
304 DEPRECATED(vrna_param_t *copy_parameters(void), "Use vrna_params_copy() instead");
305 DEPRECATED(vrna_param_t *set_parameters(vrna_param_t *dest), "Use vrna_params_copy() instead");
306 DEPRECATED(vrna_exp_param_t *scale_pf_parameters(void), "Use vrna_exp_params() instead");
307 DEPRECATED(vrna_exp_param_t *copy_pf_param(void), "Use vrna_exp_params_copy() instead");
308 DEPRECATED(vrna_exp_param_t *set_pf_param(vrna_param_t *dest),
309            "Use vrna_exp_params_copy() instead");
310
311 #endif
312
313 #endif
```

## 18.157 ViennaRNA/utils/basic.h File Reference

General utility- and helper-functions used throughout the *ViennaRNA Package*.

Include dependency graph for basic.h: This graph shows which files directly or indirectly include this file:

### Macros

- `#define VRNA_INPUT_ERROR 1U`  
*Output flag of `get_input_line()`: "An ERROR has occurred, maybe EOF".*
- `#define VRNA_INPUT_QUIT 2U`  
*Output flag of `get_input_line()`: "the user requested quitting the program".*
- `#define VRNA_INPUT_MISC 4U`  
*Output flag of `get_input_line()`: "something was read".*
- `#define VRNA_INPUT_FASTA_HEADER 8U`  
*Input/Output flag of `get_input_line()`:  
if used as input option this tells `get_input_line()` that the data to be read should comply with the FASTA format.*
- `#define VRNA_INPUT_CONSTRAINT 32U`  
*Input flag for `get_input_line()`:  
Tell `get_input_line()` that we assume to read a structure constraint.*
- `#define VRNA_INPUT_NO_TRUNCATION 256U`  
*Input switch for `get_input_line()`: "do not truncate the line by eliminating white spaces at end of line".*
- `#define VRNA_INPUT_NO_REST 512U`  
*Input switch for `vrna_file_fasta_read_record()`: "do fill rest array".*
- `#define VRNA_INPUT_NO_SPAN 1024U`  
*Input switch for `vrna_file_fasta_read_record()`: "never allow data to span more than one line".*
- `#define VRNA_INPUT_NOSKIP_BLANK_LINES 2048U`  
*Input switch for `vrna_file_fasta_read_record()`: "do not skip empty lines".*
- `#define VRNA_INPUT_BLANK_LINE 4096U`  
*Output flag for `vrna_file_fasta_read_record()`: "read an empty line".*
- `#define VRNA_INPUT_NOSKIP_COMMENTS 128U`  
*Input switch for `get_input_line()`: "do not skip comment lines".*
- `#define VRNA_INPUT_COMMENT 8192U`  
*Output flag for `vrna_file_fasta_read_record()`: "read a comment".*
- `#define MIN2(A, B) ((A) < (B) ? (A) : (B))`  
*Get the minimum of two comparable values.*
- `#define MAX2(A, B) ((A) > (B) ? (A) : (B))`  
*Get the maximum of two comparable values.*
- `#define MIN3(A, B, C) (MIN2((MIN2((A), (B))), (C)))`  
*Get the minimum of three comparable values.*
- `#define MAX3(A, B, C) (MAX2((MAX2((A), (B))), (C)))`  
*Get the maximum of three comparable values.*

### Functions

- `void * vrna_alloc (unsigned size)`  
*Allocate space safely.*
- `void * vrna_realloc (void *p, unsigned size)`  
*Reallocate space safely.*
- `void vrna_init_rand (void)`  
*Initialize seed for random number generator.*
- `void vrna_init_rand_seed (unsigned int seed)`  
*Initialize the random number generator with a pre-defined seed.*

- double **vrna\_urn** (void)  
*get a random number from [0..1]*
- int **vrna\_int\_urn** (int from, int to)  
*Generates a pseudo random integer in a specified range.*
- char \* **vrna\_time\_stamp** (void)  
*Get a timestamp.*
- unsigned int **get\_input\_line** (char \*\*string, unsigned int options)
- int \* **vrna\_idx\_row\_wise** (unsigned int length)  
*Get an index mapper array (iindx) for accessing the energy matrices, e.g. in partition function related functions.*
- int \* **vrna\_idx\_col\_wise** (unsigned int length)  
*Get an index mapper array (indx) for accessing the energy matrices, e.g. in MFE related functions.*
- void **vrna\_message\_error** (const char \*format,...)  
*Print an error message and die.*
- void **vrna\_message\_error** (const char \*format, va\_list args)  
*Print an error message and die.*
- void **vrna\_message\_warning** (const char \*format,...)  
*Print a warning message.*
- void **vrna\_message\_vwarning** (const char \*format, va\_list args)  
*Print a warning message.*
- void **vrna\_message\_info** (FILE \*fp, const char \*format,...)  
*Print an info message.*
- void **vrna\_message\_vinfo** (FILE \*fp, const char \*format, va\_list args)  
*Print an info message.*
- void **vrna\_message\_input\_seq\_simple** (void)  
*Print a line to stdout that asks for an input sequence.*
- void **vrna\_message\_input\_seq** (const char \*s)  
*Print a line with a user defined string and a ruler to stdout.*
- char \* **get\_line** (FILE \*fp)  
*Read a line of arbitrary length from a stream.*
- void **print\_tty\_input\_seq** (void)  
*Print a line to stdout that asks for an input sequence.*
- void **print\_tty\_input\_seq\_str** (const char \*s)  
*Print a line with a user defined string and a ruler to stdout.*
- void **warn\_user** (const char message[ ])  
*Print a warning message.*
- void **nrerror** (const char message[ ])  
*Die with an error message.*
- void \* **space** (unsigned size)  
*Allocate space safely.*
- void \* **xrealloc** (void \*p, unsigned size)  
*Reallocate space safely.*
- void **init\_rand** (void)  
*Make random number seeds.*
- double **urn** (void)  
*get a random number from [0..1]*
- int **int\_urn** (int from, int to)  
*Generates a pseudo random integer in a specified range.*
- void **filecopy** (FILE \*from, FILE \*to)  
*Inefficient cp*
- char \* **time\_stamp** (void)  
*Get a timestamp.*

## Variables

- unsigned short `xsubi` [3]  
*Current 48 bit random number.*

### 18.157.1 Detailed Description

General utility- and helper-functions used throughout the *ViennaRNA Package*.

### 18.157.2 Function Documentation

#### 18.157.2.1 `get_line()`

```
char * get_line (
    FILE * fp )
```

Read a line of arbitrary length from a stream.

Returns a pointer to the resulting string. The necessary memory is allocated and should be released using `free()` when the string is no longer needed.

**Deprecated** Use [vrna\\_read\\_line\(\)](#) as a substitute!

#### Parameters

|                 |  |
|-----------------|--|
| <code>fp</code> | A file pointer to the stream where the function should read from |
|-----------------|--|

#### Returns

A pointer to the resulting string

#### 18.157.2.2 `print_tty_input_seq()`

```
void print_tty_input_seq (
    void )
```

Print a line to `stdout` that asks for an input sequence.

There will also be a ruler (scale line) printed that helps orientation of the sequence positions

**Deprecated** Use [vrna\\_message\\_input\\_seq\\_simple\(\)](#) instead!

#### 18.157.2.3 `print_tty_input_seq_str()`

```
void print_tty_input_seq_str (
    const char * s )
```

Print a line with a user defined string and a ruler to `stdout`.

(usually this is used to ask for user input) There will also be a ruler (scale line) printed that helps orientation of the sequence positions

**Deprecated** Use [vrna\\_message\\_input\\_seq\(\)](#) instead!

#### 18.157.2.4 `warn_user()`

```
void warn_user (
    const char message[] )
```

Print a warning message.

Print a warning message to *stderr*

**Deprecated** Use [vrna\\_message\\_warning\(\)](#) instead!

#### 18.157.2.5 nrerror()

```
void nrerror (
    const char message[ ] )
```

Die with an error message.

**Deprecated** Use [vrna\\_message\\_error\(\)](#) instead!

#### 18.157.2.6 space()

```
void * space (
    unsigned size )
```

Allocate space safely.

**Deprecated** Use [vrna\\_alloc\(\)](#) instead!

#### 18.157.2.7 xrealloc()

```
void * xrealloc (
    void * p,
    unsigned size )
```

Reallocate space safely.

**Deprecated** Use [vrna\\_realloc\(\)](#) instead!

#### 18.157.2.8 init\_rand()

```
void init_rand (
    void )
```

Make random number seeds.

**Deprecated** Use [vrna\\_init\\_rand\(\)](#) instead!

#### 18.157.2.9 urn()

```
double urn (
    void )
```

get a random number from [0..1]

**Deprecated** Use [vrna\\_urn\(\)](#) instead!

### 18.157.2.10 int\_urn()

```
int int_urn (
    int from,
    int to )
```

Generates a pseudo random integer in a specified range.

**Deprecated** Use [vrna\\_int\\_urn\(\)](#) instead!

### 18.157.2.11 filecopy()

```
void filecopy (
    FILE * from,
    FILE * to )
```

Inefficient cp

**Deprecated** Use [vrna\\_file\\_copy\(\)](#) instead!

### 18.157.2.12 time\_stamp()

```
char * time_stamp (
    void )
```

Get a timestamp.

**Deprecated** Use [vrna\\_time\\_stamp\(\)](#) instead!

## 18.158 basic.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_UTILS_H
2 #define VIENNA_RNA_PACKAGE_UTILS_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 #  define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
27 /* two helper macros to indicate whether a function should be exported in
28 * the library or stays hidden */
29 #define PUBLIC
30 #define PRIVATE static
31
35 #define VRNA_INPUT_ERROR          1U
39 #define VRNA_INPUT_QUIT           2U
43 #define VRNA_INPUT_MISC           4U
44
52 #define VRNA_INPUT_FASTA_HEADER   8U
53
54 /*
55 * @brief Input flag for get_input_line():
56 * Tell get_input_line() that we assume to read a nucleotide sequence
57 *
58 */
59 #define VRNA_INPUT_SEQUENCE        16U
60
65 #define VRNA_INPUT_CONSTRAINT      32U
66
71 #define VRNA_INPUT_NO_TRUNCATION   256U
72
76 #define VRNA_INPUT_NO_REST         512U
77
81 #define VRNA_INPUT_NO_SPAN        1024U
```

```
82
83 #define VRNA_INPUT_NOSKIP_BLANK_LINES      2048U
84
85 #define VRNA_INPUT_BLANK_LINE              4096U
86
87 #define VRNA_INPUT_NOSKIP_COMMENTS        128U
88
89 #define VRNA_INPUT_COMMENT                8192U
90
91
92 #define MIN2(A, B)          ((A) < (B) ? (A) : (B))
93
94 #define MAX2(A, B)          ((A) > (B) ? (A) : (B))
95
96 #define MIN3(A, B, C)        (MIN2((A), (B))), (C)))
97
98 #define MAX3(A, B, C)        (MAX2((A), (B))), (C)))
99
100
101 #include <stdio.h>
102 #include <stdarg.h>
103
104 #include <ViennaRNA/datastructures/basic.h>
105
106
107 #ifdef WITH_DMALLOC
108 /* use dmalloc library to check for memory management bugs */
109 #include "dmalloc.h"
110 #define vrna_malloc(S)           calloc(1, (S))
111 #define vrna_realloc(p, S)      xrealloc(p, S)
112 #else
113
114 void *
115 vrna_malloc(unsigned size);
116
117
118 void *
119 vrna_realloc(void *p,
120             unsigned size);
121
122
123 #endif
124
125
126 void
127 vrna_init_rand(void);
128
129
130 void
131 vrna_init_rand_seed(unsigned int seed);
132
133
134 extern unsigned short xsubi[3];
135
136
137 double
138 vrna_urn(void);
139
140
141 int
142 vrna_int_urn(int from,
143               int to);
144
145
146 char *
147 vrna_time_stamp(void);
148
149
150
151 unsigned int
152 get_input_line(char **string,
153                 unsigned int options);
154
155
156
157 int *
158 vrna_idx_row_wise(unsigned int length);
159
160
161 int *
162 vrna_idx_col_wise(unsigned int length);
163
164
165 void
166 vrna_message_error(const char *format,
167                      ...);
168
169
170 void
171 vrna_message_verror(const char *format,
172                      va_list args);
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
```

```

341 void
342 vrna_message_warning(const char *format,
343 ...);
344
345
357 void
358 vrna_message_vwarning(const char *format,
359 va_list args);
360
361
373 void
374 vrna_message_info(FILE *fp,
375 const char *format,
376 ...);
377
378
390 void
391 vrna_message_vinfo(FILE *fp,
392 const char *format,
393 va_list args);
394
395
401 void
402 vrna_message_input_seq_simple(void);
403
404
413 void
414 vrna_message_input_seq(const char *s);
415
416
417 void
418 vrna_message_input_msa(const char *s);
419
420
425 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
426
427 DEPRECATED(int *get_idx(unsigned int length), "Use vrna_idx_col_wise() instead");
428
429 DEPRECATED(int *get_iidx(unsigned int length), "Use vrna_idx_row_wise() instead");
430
443 DEPRECATED(char *get_line(FILE *fp), "Use vrna_read_line() instead");
444
451 DEPRECATED(void print_tty_input_seq(void), "Use vrna_message_input_seq_simple() instead");
452
461 DEPRECATED(void print_tty_input_seq_str(const char *s), "Use vrna_message_input_seq() instead");
462
470 DEPRECATED(void warn_user(const char message[]), "Use vrna_message_warning() instead");
471
477 DEPRECATED(void nrerror(const char message[]), "Use vrna_message_error() instead()");
478
484 DEPRECATED(void *space(unsigned size), "Use vrna_alloc() instead");
485
491 DEPRECATED(void *xrealloc(void *p,
492 unsigned size), "Use vrna_realloc() instead");
493
498 DEPRECATED(void init_rand(void), "Use vrna_init_rand() instead");
499
505 DEPRECATED(double urn(void), "Use vrna_urn() instead");
506
512 DEPRECATED(int int_urn(int from,
513 int to), "Use vrna_int_urn() instead()");
514
520 DEPRECATED(void filecopy(FILE *from,
521 FILE *to), "Use vrna_file_copy() instead");
522
528 DEPRECATED(char *time_stamp(void), "Use vrna_time_stamp() instead");
529
530 #endif
531
532 #endif

```

## 18.159 ViennaRNA/params/constants.h File Reference

Energy parameter constants.

Include dependency graph for constants.h: This graph shows which files directly or indirectly include this file:

### Macros

- #define GASCONST 1.98717 /\* in [cal/K] \*/
- #define K0 273.15
- #define INF 10000000 /\* (INT\_MAX/10) \*/

- #define FORBIDDEN 9999
- #define BONUS 10000
- #define NBPAIRS 7
- #define TURN 3
- #define MAXLOOP 30

## 18.159.1 Detailed Description

Energy parameter constants.

## 18.159.2 Macro Definition Documentation

### 18.159.2.1 GASCONST

```
#define GASCONST 1.98717 /* in [cal/K] */  
The gas constant
```

### 18.159.2.2 K0

```
#define K0 273.15  
0 deg Celsius in Kelvin
```

### 18.159.2.3 INF

```
#define INF 10000000 /* (INT_MAX/10) */  
Infinity as used in minimization routines
```

### 18.159.2.4 FORBIDDEN

```
#define FORBIDDEN 9999  
forbidden
```

### 18.159.2.5 BONUS

```
#define BONUS 10000  
bonus contribution
```

### 18.159.2.6 NBPAIRS

```
#define NBPAIRS 7  
The number of distinguishable base pairs
```

### 18.159.2.7 TURN

```
#define TURN 3  
The minimum loop length
```

### 18.159.2.8 MAXLOOP

```
#define MAXLOOP 30  
The maximum loop length
```

## 18.160 constants.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PARAMS_CONSTANTS_H
2 #define VIENNA_RNA_PACKAGE_PARAMS_CONSTANTS_H
3
4 #include <limits.h>
5
13 #define GASCONST 1.98717 /* in [cal/K] */
15 #define K0 273.15
17 #define INF 10000000 /* (INT_MAX/10) */
18
19 #define EMAX (INF/10)
21 #define FORBIDDEN 9999
23 #define BONUS 10000
25 #define NBPAIRS 7
27 #define TURN 3
29 #define MAXLOOP 30
30
31 #define UNIT 100
32
33 #define MINPSCORE -2 * UNIT
34
35 #endif

```

## 18.161 ViennaRNA/params/convert.h File Reference

Functions and definitions for energy parameter file format conversion.

This graph shows which files directly or indirectly include this file:

### Macros

- `#define VRNA_CONVERT_OUTPUT_ALL 1U`
- `#define VRNA_CONVERT_OUTPUT_HP 2U`
- `#define VRNA_CONVERT_OUTPUT_STACK 4U`
- `#define VRNA_CONVERT_OUTPUT_MM_HP 8U`
- `#define VRNA_CONVERT_OUTPUT_MM_INT 16U`
- `#define VRNA_CONVERT_OUTPUT_MM_INT_1N 32U`
- `#define VRNA_CONVERT_OUTPUT_MM_INT_23 64U`
- `#define VRNA_CONVERT_OUTPUT_MM_MULTI 128U`
- `#define VRNA_CONVERT_OUTPUT_MM_EXT 256U`
- `#define VRNA_CONVERT_OUTPUT_DANGLE5 512U`
- `#define VRNA_CONVERT_OUTPUT_DANGLE3 1024U`
- `#define VRNA_CONVERT_OUTPUT_INT_11 2048U`
- `#define VRNA_CONVERT_OUTPUT_INT_21 4096U`
- `#define VRNA_CONVERT_OUTPUT_INT_22 8192U`
- `#define VRNA_CONVERT_OUTPUT_BULGE 16384U`
- `#define VRNA_CONVERT_OUTPUT_INT 32768U`
- `#define VRNA_CONVERT_OUTPUT_ML 65536U`
- `#define VRNA_CONVERT_OUTPUT_MISC 131072U`
- `#define VRNA_CONVERT_OUTPUT_SPECIAL_HP 262144U`
- `#define VRNA_CONVERT_OUTPUT_VANILLA 524288U`
- `#define VRNA_CONVERT_OUTPUT_NINIO 1048576U`
- `#define VRNA_CONVERT_OUTPUT_DUMP 2097152U`

### Functions

- `void convert_parameter_file (const char *iname, const char *oname, unsigned int options)`

#### 18.161.1 Detailed Description

Functions and definitions for energy parameter file format conversion.

## 18.162 convert.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PARAMS_CONVERT_H
2 #define VIENNA_RNA_PACKAGE_PARAMS_CONVERT_H
3
4 #define VRNA_CONVERT_OUTPUT_ALL          1U
5 #define VRNA_CONVERT_OUTPUT_HP          2U
6 #define VRNA_CONVERT_OUTPUT_STACK        4U
7 #define VRNA_CONVERT_OUTPUT_MM_HP        8U
8 #define VRNA_CONVERT_OUTPUT_MM_INT      16U
9 #define VRNA_CONVERT_OUTPUT_MM_INT_1N    32U
10 #define VRNA_CONVERT_OUTPUT_MM_INT_23   64U
11 #define VRNA_CONVERT_OUTPUT_MM_MULTI    128U
12 #define VRNA_CONVERT_OUTPUT_MM_EXT     256U
13 #define VRNA_CONVERT_OUTPUT_DANGLE5    512U
14 #define VRNA_CONVERT_OUTPUT_DANGLE3    1024U
15 #define VRNA_CONVERT_OUTPUT_INT_11     2048U
16 #define VRNA_CONVERT_OUTPUT_INT_21     4096U
17 #define VRNA_CONVERT_OUTPUT_INT_22     8192U
18 #define VRNA_CONVERT_OUTPUT_BULGE      16384U
19 #define VRNA_CONVERT_OUTPUT_INT        32768U
20 #define VRNA_CONVERT_OUTPUT_ML         65536U
21 #define VRNA_CONVERT_OUTPUT_MISC       131072U
22 #define VRNA_CONVERT_OUTPUT_SPECIAL_HP 262144U
23 #define VRNA_CONVERT_OUTPUT_VANILLA    524288U
24 #define VRNA_CONVERT_OUTPUT_NINIO      1048576U
25 #define VRNA_CONVERT_OUTPUT_DUMP       2097152U
26
27 void convert_parameter_file(const char *iname,
28                             const char *oname,
29                             unsigned int options);
30
31
32 #endif

```

## 18.163 default.h

```

1 /*
2  prototypes for energy_par.c
3 */
4
5 #ifndef VIENNA_RNA_PACKAGE_PARAMS_DEFAULT_H
6 #define VIENNA_RNA_PACKAGE_PARAMS_DEFAULT_H
7
8 #include <ViennaRNA/params/constants.h>
9
10 #define PUBLIC
11
12
13 extern double lxc37; /* parameter for logarithmic loop
14           energy extrapolation */
15
16 extern int stack37[NBPAIRS+1][NBPAIRS+1];
17 extern int stackdH[NBPAIRS+1][NBPAIRS+1]; /* stack enthalpies */
18
19 extern int hairpin37[31];
20 extern int hairpindH[31];
21 extern int bulge37[31];
22 extern int bulgedH[31];
23 extern int internal_loop37[31];
24 extern int internal_looppH[31];
25 extern int mismatchI37[NBPAIRS+1][5][5]; /* interior loop mismatches */
26 extern int mismatchIdH[NBPAIRS+1][5][5]; /* interior loop mismatches */
27 extern int mismatchInI37[NBPAIRS+1][5][5]; /* interior loop mismatches */
28 extern int mismatch23I37[NBPAIRS+1][5][5]; /* interior loop mismatches */
29 extern int mismatchInIdH[NBPAIRS+1][5][5]; /* interior loop mismatches */
30 extern int mismatch23IdH[NBPAIRS+1][5][5]; /* interior loop mismatches */
31 extern int mismatchH37[NBPAIRS+1][5][5]; /* same for hairpins */
32 extern int mismatchM37[NBPAIRS+1][5][5]; /* same for multiloops */
33 extern int mismatchHdH[NBPAIRS+1][5][5]; /* same for hairpins */
34 extern int mismatchMdH[NBPAIRS+1][5][5]; /* same for multiloops */
35 extern int mismatchExt37[NBPAIRS+1][5][5];
36 extern int mismatchExtdH[NBPAIRS+1][5][5];
37
38 extern int dangle5_37[NBPAIRS+1][5]; /* 5' dangle exterior of pair */
39 extern int dangle3_37[NBPAIRS+1][5]; /* 3' dangle */
40 extern int dangle3_dH[NBPAIRS+1][5]; /* corresponding enthalpies */
41 extern int dangle5_dH[NBPAIRS+1][5];
42
43 extern int int11_37[NBPAIRS+1][NBPAIRS+1][5][5]; /* 1x1 interior loops */
44 extern int int11_dH[NBPAIRS+1][NBPAIRS+1][5][5];
45
46 extern int int21_37[NBPAIRS+1][NBPAIRS+1][5][5][5]; /* 2x1 interior loops */

```

```

47 extern int int21_dH[NBPAIRS+1][NBPAIRS+1][5][5][5];
48
49 extern int int22_37[NBPAIRS+1][NBPAIRS+1][5][5][5][5]; /* 2x2 interior loops */
50 extern int int22_dH[NBPAIRS+1][NBPAIRS+1][5][5][5][5];
51
52 /* constants for linearly destabilizing contributions for multi-loops
53   F = ML_closing + ML_intern*(k-1) + ML_BASE*u */
54 extern int ML_BASE37;
55 extern int ML_BASEdH;
56 extern int ML_closing37;
57 extern int ML_closingdH;
58 extern int ML_intern37;
59 extern int ML_interndH;
60
61 extern int TripleC37;
62 extern int TripleCdH;
63 extern int MultipleCA37;
64 extern int MultipleCAdH;
65 extern int MultipleCB37;
66 extern int MultipleCbDh;
67
68 /* Ninio-correction for asymmetric internal loops with branches n1 and n2 */
69 /*   ninio_energy = min{max_ninio, |n1-n2|*F_ninio[min{4.0, n1, n2}]} */
70 extern int MAX_NINIO;                                /* maximum correction */
71 extern int ninio37;
72 extern int niniodH;
73 /* penalty for helices terminated by AU (actually not GC) */
74 extern int TerminalAU37;
75 extern int TerminalAUdH;
76 /* penalty for forming bi-molecular duplex */
77 extern int DuplexInit37;
78 extern int DuplexInitdH;
79 /* stabilizing contribution due to special hairpins of size 4 (tetraloops) */
80 extern char Tetraloops[281]; /* string containing the special tetraloops */
81 extern int Tetraloop37[40]; /* Bonus energy for special tetraloops */
82 extern int TetraloopdH[40];
83 extern char Triloops[241]; /* string containing the special triloops */
84 extern int Triloop37[40]; /* Bonus energy for special Triloops */
85 extern int TriloopdH[40]; /* Bonus energy for special Triloops */
86 extern char Hexaloops[361]; /* string containing the special hexaloops */
87 extern int Hexaloop37[40]; /* Bonus energy for special Hexaloops */
88 extern int HexaloopdH[40]; /* Bonus energy for special Hexaloops */
89
90 extern int GQuadAlpha37;
91 extern int GQuadAlphadH;
92 extern int GQuadBeta37;
93 extern int GQuadBetadH;
94 extern int GQuadLayerMismatch37; /* penalty per incompatible gquad layer in a sub-alignment (applied
twice for inner layers) */
95 extern int GQuadLayerMismatchH;
96 extern int GQuadLayerMismatchMax; /* maximum number of mismatching sequences in the alignment when gquad
should be formed */
97
98 extern double Tmeasure; /* temperature of param measurements */
99
100 #endiff

```

## 18.164 intl11.h

```

1 PUBLIC int intl1_37[NBPAIRS+1][NBPAIRS+1][5][5] =
2 {{{{ INF, INF, INF, INF, INF }
3 , { INF, INF, INF, INF, INF }
4 , { INF, INF, INF, INF, INF }
5 , { INF, INF, INF, INF, INF }
6 , { INF, INF, INF, INF, INF }
7 }
8 , { { INF, INF, INF, INF, INF }
9 , { INF, INF, INF, INF, INF }
10 , { INF, INF, INF, INF, INF }
11 , { INF, INF, INF, INF, INF }
12 , { INF, INF, INF, INF, INF }
13 }
14 , { { INF, INF, INF, INF, INF }
15 , { INF, INF, INF, INF, INF }
16 , { INF, INF, INF, INF, INF }
17 , { INF, INF, INF, INF, INF }
18 , { INF, INF, INF, INF, INF }
19 }
20 , { { INF, INF, INF, INF, INF }
21 , { INF, INF, INF, INF, INF }
22 , { INF, INF, INF, INF, INF }
23 , { INF, INF, INF, INF, INF }
24 , { INF, INF, INF, INF, INF }
25 }
26 , { { INF, INF, INF, INF, INF }

```

```
27 , { INF, INF, INF, INF, INF }
28 , { INF, INF, INF, INF, INF }
29 , { INF, INF, INF, INF, INF }
30 , { INF, INF, INF, INF, INF }
31 }
32 , { { INF, INF, INF, INF, INF }
33 , { INF, INF, INF, INF, INF }
34 , { INF, INF, INF, INF, INF }
35 , { INF, INF, INF, INF, INF }
36 , { INF, INF, INF, INF, INF }
37 }
38 , { { INF, INF, INF, INF, INF }
39 , { INF, INF, INF, INF, INF }
40 , { INF, INF, INF, INF, INF }
41 , { INF, INF, INF, INF, INF }
42 , { INF, INF, INF, INF, INF }
43 }
44 , { { INF, INF, INF, INF, INF }
45 , { INF, INF, INF, INF, INF }
46 , { INF, INF, INF, INF, INF }
47 , { INF, INF, INF, INF, INF }
48 , { INF, INF, INF, INF, INF }
49 }
50 }
51 , { { { INF, INF, INF, INF, INF }
52 , { INF, INF, INF, INF, INF }
53 , { INF, INF, INF, INF, INF }
54 , { INF, INF, INF, INF, INF }
55 , { INF, INF, INF, INF, INF }
56 }
57 , { { 90, 90, 50, 50, 50 }
58 , { 90, 90, 50, 50, 50 }
59 , { 50, 50, 50, 50, 50 }
60 , { 50, 50, 50, -140, 50 }
61 , { 50, 50, 50, 50, 40 }
62 }
63 , { { 90, 90, 50, 50, 60 }
64 , { 90, 90, -40, 50, 50 }
65 , { 60, 30, 50, 50, 60 }
66 , { 50, -10, 50, -220, 50 }
67 , { 50, 50, 0, 50, -10 }
68 }
69 , { { 120, 120, 120, 120, 120 }
70 , { 120, 60, 50, 120, 120 }
71 , { 120, 120, 120, 120, 120 }
72 , { 120, -20, 120, -140, 120 }
73 , { 120, 120, 100, 120, 110 }
74 }
75 , { { 220, 220, 170, 120, 120 }
76 , { 220, 220, 130, 120, 120 }
77 , { 170, 120, 170, 120, 120 }
78 , { 120, 120, 120, -140, 120 }
79 , { 120, 120, 120, 120, 110 }
80 }
81 , { { 120, 120, 120, 120, 120 }
82 , { 120, 120, 120, 120, 120 }
83 , { 120, 120, 120, 120, 120 }
84 , { 120, 120, 120, -140, 120 }
85 , { 120, 120, 120, 120, 80 }
86 }
87 , { { 120, 120, 120, 120, 120 }
88 , { 120, 120, 120, 120, 120 }
89 , { 120, 120, 120, 120, 120 }
90 , { 120, 120, 120, -140, 120 }
91 , { 120, 120, 120, 120, 120 }
92 }
93 , { { 220, 220, 170, 120, 120 }
94 , { 220, 220, 130, 120, 120 }
95 , { 170, 120, 170, 120, 120 }
96 , { 120, 120, 120, -140, 120 }
97 , { 120, 120, 120, 120, 120 }
98 }
99 }
100 , { { { INF, INF, INF, INF, INF }
101 , { INF, INF, INF, INF, INF }
102 , { INF, INF, INF, INF, INF }
103 , { INF, INF, INF, INF, INF }
104 , { INF, INF, INF, INF, INF }
105 }
106 , { { 90, 90, 60, 50, 50 }
107 , { 90, 90, 30, -10, 50 }
108 , { 50, -40, 50, 50, 0 }
109 , { 50, 50, 50, -220, 50 }
110 , { 60, 50, 60, 50, -10 }
111 }
112 , { { 80, 80, 50, 50, 50 }
113 , { 80, 80, 50, 50, 50 }
```

```
114 ,{ 50, 50, 50, 50, 50}
115 ,{ 50, 50, 50, -230, 50}
116 ,{ 50, 50, 50, 50, -60}
117 }
118 ,{{ 190, 190, 120, 150, 150}
119 ,{ 190, 190, 120, 150, 120}
120 ,{ 120, 120, 120, 120, 120}
121 ,{ 120, 120, 120, -140, 120}
122 ,{ 150, 120, 120, 120, 150}
123 }
124 ,{{ 160, 160, 120, 120, 120}
125 ,{ 160, 160, 120, 100, 120}
126 ,{ 120, 120, 120, 120, 120}
127 ,{ 120, 120, 120, -140, 120}
128 ,{ 120, 120, 120, 120, 70}
129 }
130 ,{{ 120, 120, 120, 120, 120}
131 ,{ 120, 120, 120, 120, 120}
132 ,{ 120, 120, 120, 120, 120}
133 ,{ 120, 120, 120, -140, 120}
134 ,{ 120, 120, 120, 120, 80}
135 }
136 ,{{ 120, 120, 120, 120, 120}
137 ,{ 120, 120, 120, 120, 120}
138 ,{ 120, 120, 120, 120, 120}
139 ,{ 120, 120, 120, -140, 120}
140 ,{ 120, 120, 120, 120, 120}
141 }
142 ,{{ 190, 190, 120, 150, 150}
143 ,{ 190, 190, 120, 150, 120}
144 ,{ 120, 120, 120, 120, 120}
145 ,{ 120, 120, 120, -140, 120}
146 ,{ 150, 120, 120, 120, 150}
147 }
148 }
149 ,{{ { INF, INF, INF, INF, INF}
150 ,{ INF, INF, INF, INF, INF}
151 ,{ INF, INF, INF, INF, INF}
152 ,{ INF, INF, INF, INF, INF}
153 ,{ INF, INF, INF, INF, INF}
154 }
155 ,{{ 120, 120, 120, 120, 120}
156 ,{ 120, 60, 120, -20, 120}
157 ,{ 120, 50, 120, 120, 100}
158 ,{ 120, 120, 120, -140, 120}
159 ,{ 120, 120, 120, 120, 110}
160 }
161 ,{{ 190, 190, 120, 120, 150}
162 ,{ 190, 190, 120, 120, 120}
163 ,{ 120, 120, 120, 120, 120}
164 ,{ 150, 150, 120, -140, 120}
165 ,{ 150, 120, 120, 120, 150}
166 }
167 ,{{ 190, 190, 190, 190, 190}
168 ,{ 190, 190, 190, 190, 190}
169 ,{ 190, 190, 190, 190, 190}
170 ,{ 190, 190, 190, -70, 190}
171 ,{ 190, 190, 190, 190, 120}
172 }
173 ,{{ 190, 190, 190, 190, 190}
174 ,{ 190, 190, 190, 190, 190}
175 ,{ 190, 190, 190, 190, 190}
176 ,{ 190, 190, 190, -70, 190}
177 ,{ 190, 190, 190, 190, 160}
178 }
179 ,{{ 190, 190, 190, 190, 190}
180 ,{ 190, 190, 190, 190, 190}
181 ,{ 190, 190, 190, 190, 190}
182 ,{ 190, 190, 190, -70, 190}
183 ,{ 190, 190, 190, 190, 120}
184 }
185 ,{{ 190, 190, 190, 190, 190}
186 ,{ 190, 190, 190, 190, 190}
187 ,{ 190, 190, 190, 190, 190}
188 ,{ 190, 190, 190, -70, 190}
189 ,{ 190, 190, 190, 190, 160}
190 }
191 ,{{ 190, 190, 190, 190, 190}
192 ,{ 190, 190, 190, 190, 190}
193 ,{ 190, 190, 190, 190, 190}
194 ,{ 190, 190, 190, -70, 190}
195 ,{ 190, 190, 190, 190, 160}
196 }
197 }
198 ,{{ { INF, INF, INF, INF, INF}
199 ,{ INF, INF, INF, INF, INF}
200 ,{ INF, INF, INF, INF, INF}
```

```
201 , {   INF,   INF,   INF,   INF,   INF}
202 , {   INF,   INF,   INF,   INF,   INF}
203 }
204 , { { 220,  220,  170,  120,  120}
205 , { 220,  220,  120,  120,  120}
206 , { 170,  130,  170,  120,  120}
207 , { 120,  120,  120, -140,  120}
208 , { 120,  120,  120,  120,  110}
209 }
210 , { { 160,  160,  120,  120,  120}
211 , { 160,  160,  120,  120,  120}
212 , { 120,  120,  120,  120,  120}
213 , { 120,  100,  120, -140,  120}
214 , { 120,  120,  120,  120,   70}
215 }
216 , { { 190,  190,  190,  190,  190}
217 , { 190,  190,  190,  190,  190}
218 , { 190,  190,  190,  190,  190}
219 , { 190,  190,  190, -70,  190}
220 , { 190,  190,  190,  190,  160}
221 }
222 , { { 190,  190,  190,  190,  190}
223 , { 190,  190,  190,  190,  190}
224 , { 190,  190,  190,  190,  190}
225 , { 190,  190,  190, -70,  190}
226 , { 190,  190,  190,  190,  190}
227 }
228 , { { 190,  190,  190,  190,  190}
229 , { 190,  190,  190,  190,  190}
230 , { 190,  190,  190,  190,  190}
231 , { 190,  190,  190, -70,  190}
232 , { 190,  190,  190,  190,  160}
233 }
234 , { { 190,  190,  190,  190,  190}
235 , { 190,  190,  190,  190,  190}
236 , { 190,  190,  190,  190,  190}
237 , { 190,  190,  190, -70,  190}
238 , { 190,  190,  190,  190,  190}
239 }
240 , { { 220,  220,  190,  190,  190}
241 , { 220,  220,  190,  190,  190}
242 , { 190,  190,  190,  190,  190}
243 , { 190,  190,  190, -70,  190}
244 , { 190,  190,  190,  190,  190}
245 }
246 }
247 , { { {   INF,   INF,   INF,   INF,   INF}
248 , {   INF,   INF,   INF,   INF,   INF}
249 , {   INF,   INF,   INF,   INF,   INF}
250 , {   INF,   INF,   INF,   INF,   INF}
251 , {   INF,   INF,   INF,   INF,   INF}
252 }
253 , { { 120,  120,  120,  120,  120}
254 , { 120,  120,  120,  120,  120}
255 , { 120,  120,  120,  120,  120}
256 , { 120,  120,  120, -140,  120}
257 , { 120,  120,  120,  120,   80}
258 }
259 , { { 120,  120,  120,  120,  120}
260 , { 120,  120,  120,  120,  120}
261 , { 120,  120,  120,  120,  120}
262 , { 120,  120,  120, -140,  120}
263 , { 120,  120,  120,  120,   80}
264 }
265 , { { 190,  190,  190,  190,  190}
266 , { 190,  190,  190,  190,  190}
267 , { 190,  190,  190,  190,  190}
268 , { 190,  190,  190, -70,  190}
269 , { 190,  190,  190,  190,  120}
270 }
271 , { { 190,  190,  190,  190,  190}
272 , { 190,  190,  190,  190,  190}
273 , { 190,  190,  190,  190,  190}
274 , { 190,  190,  190, -70,  190}
275 , { 190,  190,  190,  190,  160}
276 }
277 , { { 190,  190,  190,  190,  190}
278 , { 190,  190,  190,  190,  190}
279 , { 190,  190,  190,  190,  190}
280 , { 190,  190,  190, -70,  190}
281 , { 190,  190,  190,  190,  120}
282 }
283 , { { 190,  190,  190,  190,  190}
284 , { 190,  190,  190,  190,  190}
285 , { 190,  190,  190,  190,  190}
286 , { 190,  190,  190, -70,  190}
287 , { 190,  190,  190,  190,  150}
```

```
288    }
289 ,{{ 190,   190,   190,   190,   190}
290 ,{ 190,   190,   190,   190,   190}
291 ,{ 190,   190,   190,   190,   190}
292 ,{ 190,   190,   190,   -70,   190}
293 ,{ 190,   190,   190,   190,   160}
294 }
295 }
296 ,{{ INF,   INF,   INF,   INF,   INF}
297 ,{ INF,   INF,   INF,   INF,   INF}
298 ,{ INF,   INF,   INF,   INF,   INF}
299 ,{ INF,   INF,   INF,   INF,   INF}
300 ,{ INF,   INF,   INF,   INF,   INF}
301 }
302 ,{{ 120,   120,   120,   120,   120}
303 ,{ 120,   120,   120,   120,   120}
304 ,{ 120,   120,   120,   120,   120}
305 ,{ 120,   120,   120,   -140,   120}
306 ,{ 120,   120,   120,   120,   120}
307 }
308 ,{{ 120,   120,   120,   120,   120}
309 ,{ 120,   120,   120,   120,   120}
310 ,{ 120,   120,   120,   120,   120}
311 ,{ 120,   120,   120,   -140,   120}
312 ,{ 120,   120,   120,   120,   120}
313 }
314 ,{{ 190,   190,   190,   190,   190}
315 ,{ 190,   190,   190,   190,   190}
316 ,{ 190,   190,   190,   190,   190}
317 ,{ 190,   190,   190,   -70,   190}
318 ,{ 190,   190,   190,   190,   160}
319 }
320 ,{{ 190,   190,   190,   190,   190}
321 ,{ 190,   190,   190,   190,   190}
322 ,{ 190,   190,   190,   190,   190}
323 ,{ 190,   190,   190,   -70,   190}
324 ,{ 190,   190,   190,   190,   190}
325 }
326 ,{{ 190,   190,   190,   190,   190}
327 ,{ 190,   190,   190,   190,   190}
328 ,{ 190,   190,   190,   190,   190}
329 ,{ 190,   190,   190,   -70,   190}
330 ,{ 190,   190,   190,   190,   150}
331 }
332 ,{{ 190,   190,   190,   190,   190}
333 ,{ 190,   190,   190,   190,   190}
334 ,{ 190,   190,   190,   190,   190}
335 ,{ 190,   190,   190,   -70,   190}
336 ,{ 190,   190,   190,   190,   170}
337 }
338 ,{{ 190,   190,   190,   190,   190}
339 ,{ 190,   190,   190,   190,   190}
340 ,{ 190,   190,   190,   190,   190}
341 ,{ 190,   190,   190,   -70,   190}
342 ,{ 190,   190,   190,   190,   190}
343 }
344 }
345 ,{{ INF,   INF,   INF,   INF,   INF}
346 ,{ INF,   INF,   INF,   INF,   INF}
347 ,{ INF,   INF,   INF,   INF,   INF}
348 ,{ INF,   INF,   INF,   INF,   INF}
349 ,{ INF,   INF,   INF,   INF,   INF}
350 }
351 ,{{ 220,   220,   170,   120,   120}
352 ,{ 220,   220,   120,   120,   120}
353 ,{ 170,   130,   170,   120,   120}
354 ,{ 120,   120,   120,   -140,   120}
355 ,{ 120,   120,   120,   120,   120}
356 }
357 ,{{ 190,   190,   120,   120,   150}
358 ,{ 190,   190,   120,   120,   120}
359 ,{ 120,   120,   120,   120,   120}
360 ,{ 150,   150,   120,   -140,   120}
361 ,{ 150,   120,   120,   120,   150}
362 }
363 ,{{ 190,   190,   190,   190,   190}
364 ,{ 190,   190,   190,   190,   190}
365 ,{ 190,   190,   190,   190,   190}
366 ,{ 190,   190,   190,   -70,   190}
367 ,{ 190,   190,   190,   190,   160}
368 }
369 ,{{ 220,   220,   190,   190,   190}
370 ,{ 220,   220,   190,   190,   190}
371 ,{ 190,   190,   190,   190,   190}
372 ,{ 190,   190,   190,   -70,   190}
373 ,{ 190,   190,   190,   190,   190}
374 }
```

```

375 ,{{ 190, 190, 190, 190, 190}
376 ,{ 190, 190, 190, 190, 190}
377 ,{ 190, 190, 190, 190, 190}
378 ,{ 190, 190, 190, -70, 190}
379 ,{ 190, 190, 190, 190, 160}
380 }
381 ,{{ 190, 190, 190, 190, 190}
382 ,{ 190, 190, 190, 190, 190}
383 ,{ 190, 190, 190, 190, 190}
384 ,{ 190, 190, 190, -70, 190}
385 ,{ 190, 190, 190, 190, 190}
386 }
387 ,{{ 220, 220, 190, 190, 190}
388 ,{ 220, 220, 190, 190, 190}
389 ,{ 190, 190, 190, 190, 190}
390 ,{ 190, 190, 190, -70, 190}
391 ,{ 190, 190, 190, 190, 190}
392 }
393 }};

```

## 18.165 intl11dH.h

```

1 PUBLIC int intl1_dH[NBPAIRS+1][NBPAIRS+1][5][5] =
2 {{{{ INF, INF, INF, INF, INF}
3 ,{{ INF, INF, INF, INF, INF}
4 ,{{ INF, INF, INF, INF, INF}
5 ,{{ INF, INF, INF, INF, INF}
6 ,{{ INF, INF, INF, INF, INF}
7 }
8 ,{{ INF, INF, INF, INF, INF}
9 ,{{ INF, INF, INF, INF, INF}
10 ,{{ INF, INF, INF, INF, INF}
11 ,{{ INF, INF, INF, INF, INF}
12 ,{{ INF, INF, INF, INF, INF}
13 }
14 ,{{ INF, INF, INF, INF, INF}
15 ,{{ INF, INF, INF, INF, INF}
16 ,{{ INF, INF, INF, INF, INF}
17 ,{{ INF, INF, INF, INF, INF}
18 ,{{ INF, INF, INF, INF, INF}
19 }
20 ,{{ INF, INF, INF, INF, INF}
21 ,{{ INF, INF, INF, INF, INF}
22 ,{{ INF, INF, INF, INF, INF}
23 ,{{ INF, INF, INF, INF, INF}
24 ,{{ INF, INF, INF, INF, INF}
25 }
26 ,{{ INF, INF, INF, INF, INF}
27 ,{{ INF, INF, INF, INF, INF}
28 ,{{ INF, INF, INF, INF, INF}
29 ,{{ INF, INF, INF, INF, INF}
30 ,{{ INF, INF, INF, INF, INF}
31 }
32 ,{{ INF, INF, INF, INF, INF}
33 ,{{ INF, INF, INF, INF, INF}
34 ,{{ INF, INF, INF, INF, INF}
35 ,{{ INF, INF, INF, INF, INF}
36 ,{{ INF, INF, INF, INF, INF}
37 }
38 ,{{ INF, INF, INF, INF, INF}
39 ,{{ INF, INF, INF, INF, INF}
40 ,{{ INF, INF, INF, INF, INF}
41 ,{{ INF, INF, INF, INF, INF}
42 ,{{ INF, INF, INF, INF, INF}
43 }
44 ,{{ INF, INF, INF, INF, INF}
45 ,{{ INF, INF, INF, INF, INF}
46 ,{{ INF, INF, INF, INF, INF}
47 ,{{ INF, INF, INF, INF, INF}
48 ,{{ INF, INF, INF, INF, INF}
49 }
50 }
51 ,{{{{ INF, INF, INF, INF, INF}
52 ,{{{{ INF, INF, INF, INF, INF}
53 ,{{{{ INF, INF, INF, INF, INF}
54 ,{{{{ INF, INF, INF, INF, INF}
55 ,{{{{ INF, INF, INF, INF, INF}
56 }
57 ,{{{ -1050, -1050, -1050, -1050, -1050}
58 ,{{{ -1050, -1050, -1050, -1050, -1050}
59 ,{{{ -1050, -1050, -1050, -1050, -1050}
60 ,{{{ -1050, -1050, -1050, -1840, -1050}
61 ,{{{ -1050, -1050, -1050, -1050, -1050}
62 }
63 ,{{{ -1050, -1050, -1050, -1050, -1050}

```

```
64 , { -1050, -1050, -1050, -1050, -1050}
65 , { -1050, -1050, -1050, -1050, -1050}
66 , { -1050, -1050, -1050, -1840, -1050}
67 , { -1050, -1050, -1050, -1050, -1390}
68 }
69 , {{ -550, -550, -550, -550, -550}
70 , { -550, -550, -550, -550, -550}
71 , { -550, -550, -550, -550, -550}
72 , { -550, -550, -550, -1340, -550}
73 , { -550, -550, -550, -550, -890}
74 }
75 , {{ -550, -550, -550, -550, -550}
76 , { -550, -550, -550, -550, -550}
77 , { -550, -550, -550, -550, -550}
78 , { -550, -550, -550, -1340, -550}
79 , { -550, -550, -550, -550, -550}
80 }
81 , {{ -550, -550, -550, -550, -550}
82 , { -550, -550, -550, -550, -550}
83 , { -550, -550, -550, -550, -550}
84 , { -550, -550, -550, -1340, -550}
85 , { -550, -550, -550, -550, -890}
86 }
87 , {{ -550, -550, -550, -550, -550}
88 , { -550, -550, -550, -550, -550}
89 , { -550, -550, -550, -550, -550}
90 , { -550, -550, -550, -1340, -550}
91 , { -550, -550, -550, -550, -550}
92 }
93 , {{ -550, -550, -550, -550, -550}
94 , { -550, -550, -550, -550, -550}
95 , { -550, -550, -550, -550, -550}
96 , { -550, -550, -550, -1340, -550}
97 , { -550, -550, -550, -550, -550}
98 }
99 }
100 , {{{ INF, INF, INF, INF, INF}
101 , { INF, INF, INF, INF, INF}
102 , { INF, INF, INF, INF, INF}
103 , { INF, INF, INF, INF, INF}
104 , { INF, INF, INF, INF, INF}
105 }
106 , {{ -1050, -1050, -1050, -1050, -1050}
107 , { -1050, -1050, -1050, -1050, -1050}
108 , { -1050, -1050, -1050, -1050, -1050}
109 , { -1050, -1050, -1050, -1840, -1050}
110 , { -1050, -1050, -1050, -1050, -1390}
111 }
112 , {{ -1050, -1050, -1050, -1050, -1050}
113 , { -1050, -1050, -1050, -1050, -1050}
114 , { -1050, -1050, -1050, -1050, -1050}
115 , { -1050, -1050, -1050, -1840, -1050}
116 , { -1050, -1050, -1050, -1050, -1730}
117 }
118 , {{ -550, -550, -550, -550, -550}
119 , { -550, -550, -550, -550, -550}
120 , { -550, -550, -550, -550, -550}
121 , { -550, -550, -550, -1340, -550}
122 , { -550, -550, -550, -550, -1230}
123 }
124 , {{ -550, -550, -550, -550, -550}
125 , { -550, -550, -550, -550, -550}
126 , { -550, -550, -550, -550, -550}
127 , { -550, -550, -550, -1340, -550}
128 , { -550, -550, -550, -550, -890}
129 }
130 , {{ -550, -550, -550, -550, -550}
131 , { -550, -550, -550, -550, -550}
132 , { -550, -550, -550, -550, -550}
133 , { -550, -550, -550, -1340, -550}
134 , { -550, -550, -550, -550, -1230}
135 }
136 , {{ -550, -550, -550, -550, -550}
137 , { -550, -550, -550, -550, -550}
138 , { -550, -550, -550, -550, -550}
139 , { -550, -550, -550, -1340, -550}
140 , { -550, -550, -550, -550, -890}
141 }
142 , {{ -550, -550, -550, -550, -550}
143 , { -550, -550, -550, -550, -550}
144 , { -550, -550, -550, -550, -550}
145 , { -550, -550, -550, -1340, -550}
146 , { -550, -550, -550, -550, -890}
147 }
148 }
149 , {{{ INF, INF, INF, INF, INF}
150 , { INF, INF, INF, INF, INF}
```

```
151 , {   INF,   INF,   INF,   INF,   INF}
152 , {   INF,   INF,   INF,   INF,   INF}
153 , {   INF,   INF,   INF,   INF,   INF}
154 }
155 , { { -550, -550, -550, -550, -550}
156 , { -550, -550, -550, -550, -550}
157 , { -550, -550, -550, -550, -550}
158 , { -550, -550, -550, -1340, -550}
159 , { -550, -550, -550, -550, -890}
160 }
161 , { { -550, -550, -550, -550, -550}
162 , { -550, -550, -550, -550, -550}
163 , { -550, -550, -550, -550, -550}
164 , { -550, -550, -550, -1340, -550}
165 , { -550, -550, -550, -550, -1230}
166 }
167 , { { -50, -50, -50, -50, -50}
168 , { -50, -50, -50, -50, -50}
169 , { -50, -50, -50, -50, -50}
170 , { -50, -50, -50, -830, -50}
171 , { -50, -50, -50, -50, -730}
172 }
173 , { { -50, -50, -50, -50, -50}
174 , { -50, -50, -50, -50, -50}
175 , { -50, -50, -50, -50, -50}
176 , { -50, -50, -50, -830, -50}
177 , { -50, -50, -50, -50, -390}
178 }
179 , { { -50, -50, -50, -50, -50}
180 , { -50, -50, -50, -50, -50}
181 , { -50, -50, -50, -50, -50}
182 , { -50, -50, -50, -830, -50}
183 , { -50, -50, -50, -50, -730}
184 }
185 , { { -50, -50, -50, -50, -50}
186 , { -50, -50, -50, -50, -50}
187 , { -50, -50, -50, -50, -50}
188 , { -50, -50, -50, -830, -50}
189 , { -50, -50, -50, -50, -390}
190 }
191 , { { -50, -50, -50, -50, -50}
192 , { -50, -50, -50, -50, -50}
193 , { -50, -50, -50, -50, -50}
194 , { -50, -50, -50, -830, -50}
195 , { -50, -50, -50, -50, -390}
196 }
197 }
198 , { {   INF,   INF,   INF,   INF,   INF}
199 , {   INF,   INF,   INF,   INF,   INF}
200 , {   INF,   INF,   INF,   INF,   INF}
201 , {   INF,   INF,   INF,   INF,   INF}
202 , {   INF,   INF,   INF,   INF,   INF}
203 }
204 , { { -550, -550, -550, -550, -550}
205 , { -550, -550, -550, -550, -550}
206 , { -550, -550, -550, -550, -550}
207 , { -550, -550, -550, -1340, -550}
208 , { -550, -550, -550, -550, -550}
209 }
210 , { { -550, -550, -550, -550, -550}
211 , { -550, -550, -550, -550, -550}
212 , { -550, -550, -550, -550, -550}
213 , { -550, -550, -550, -1340, -550}
214 , { -550, -550, -550, -550, -890}
215 }
216 , { { -50, -50, -50, -50, -50}
217 , { -50, -50, -50, -50, -50}
218 , { -50, -50, -50, -50, -50}
219 , { -50, -50, -50, -830, -50}
220 , { -50, -50, -50, -50, -390}
221 }
222 , { { -50, -50, -50, -50, -50}
223 , { -50, -50, -50, -50, -50}
224 , { -50, -50, -50, -50, -50}
225 , { -50, -50, -50, -830, -50}
226 , { -50, -50, -50, -50, -50}
227 }
228 , { { -50, -50, -50, -50, -50}
229 , { -50, -50, -50, -50, -50}
230 , { -50, -50, -50, -50, -50}
231 , { -50, -50, -50, -830, -50}
232 , { -50, -50, -50, -50, -390}
233 }
234 , { { -50, -50, -50, -50, -50}
235 , { -50, -50, -50, -50, -50}
236 , { -50, -50, -50, -50, -50}
237 , { -50, -50, -50, -830, -50}
```

```
238 , { -50, -50, -50, -50, -50}
239 }
240 , { { -50, -50, -50, -50, -50}
241 , { -50, -50, -50, -50, -50}
242 , { -50, -50, -50, -50, -50}
243 , { -50, -50, -50, -830, -50}
244 , { -50, -50, -50, -50, -50}
245 }
246 }
247 , { { { INF, INF, INF, INF, INF}
248 , { INF, INF, INF, INF, INF}
249 , { INF, INF, INF, INF, INF}
250 , { INF, INF, INF, INF, INF}
251 , { INF, INF, INF, INF, INF}
252 }
253 , { { -550, -550, -550, -550, -550}
254 , { -550, -550, -550, -550, -550}
255 , { -550, -550, -550, -550, -550}
256 , { -550, -550, -550, -1340, -550}
257 , { -550, -550, -550, -550, -890}
258 }
259 , { { -550, -550, -550, -550, -550}
260 , { -550, -550, -550, -550, -550}
261 , { -550, -550, -550, -550, -550}
262 , { -550, -550, -550, -1340, -550}
263 , { -550, -550, -550, -550, -1230}
264 }
265 , { { -50, -50, -50, -50, -50}
266 , { -50, -50, -50, -50, -50}
267 , { -50, -50, -50, -50, -50}
268 , { -50, -50, -50, -830, -50}
269 , { -50, -50, -50, -50, -730}
270 }
271 , { { -50, -50, -50, -50, -50}
272 , { -50, -50, -50, -50, -50}
273 , { -50, -50, -50, -50, -50}
274 , { -50, -50, -50, -830, -50}
275 , { -50, -50, -50, -50, -390}
276 }
277 , { { -50, -50, -50, -50, -50}
278 , { -50, -50, -50, -50, -50}
279 , { -50, -50, -50, -50, -50}
280 , { -50, -50, -50, -830, -50}
281 , { -50, -50, -50, -50, -730}
282 }
283 , { { -50, -50, -50, -50, -50}
284 , { -50, -50, -50, -50, -50}
285 , { -50, -50, -50, -50, -50}
286 , { -50, -50, -50, -830, -50}
287 , { -50, -50, -50, -50, -390}
288 }
289 , { { -50, -50, -50, -50, -50}
290 , { -50, -50, -50, -50, -50}
291 , { -50, -50, -50, -50, -50}
292 , { -50, -50, -50, -830, -50}
293 , { -50, -50, -50, -50, -390}
294 }
295 }
296 , { { { INF, INF, INF, INF, INF}
297 , { INF, INF, INF, INF, INF}
298 , { INF, INF, INF, INF, INF}
299 , { INF, INF, INF, INF, INF}
300 , { INF, INF, INF, INF, INF}
301 }
302 , { { -550, -550, -550, -550, -550}
303 , { -550, -550, -550, -550, -550}
304 , { -550, -550, -550, -550, -550}
305 , { -550, -550, -550, -1340, -550}
306 , { -550, -550, -550, -550, -550}
307 }
308 , { { -550, -550, -550, -550, -550}
309 , { -550, -550, -550, -550, -550}
310 , { -550, -550, -550, -550, -550}
311 , { -550, -550, -550, -1340, -550}
312 , { -550, -550, -550, -550, -890}
313 }
314 , { { -50, -50, -50, -50, -50}
315 , { -50, -50, -50, -50, -50}
316 , { -50, -50, -50, -50, -50}
317 , { -50, -50, -50, -830, -50}
318 , { -50, -50, -50, -50, -390}
319 }
320 , { { -50, -50, -50, -50, -50}
321 , { -50, -50, -50, -50, -50}
322 , { -50, -50, -50, -50, -50}
323 , { -50, -50, -50, -830, -50}
324 , { -50, -50, -50, -50, -50}
```

```

325 }
326 ,{ { -50, -50, -50, -50, -50}
327 ,{ -50, -50, -50, -50, -50}
328 ,{ -50, -50, -50, -50, -50}
329 ,{ -50, -50, -50, -830, -50}
330 ,{ -50, -50, -50, -50, -390}
331 }
332 ,{ { -50, -50, -50, -50, -50}
333 ,{ -50, -50, -50, -50, -50}
334 ,{ -50, -50, -50, -50, -50}
335 ,{ -50, -50, -50, -830, -50}
336 ,{ -50, -50, -50, -50, -50}
337 }
338 ,{ { -50, -50, -50, -50, -50}
339 ,{ -50, -50, -50, -50, -50}
340 ,{ -50, -50, -50, -50, -50}
341 ,{ -50, -50, -50, -830, -50}
342 ,{ -50, -50, -50, -50, -50}
343 }
344 }
345 ,{ { INF, INF, INF, INF, INF}
346 ,{ INF, INF, INF, INF, INF}
347 ,{ INF, INF, INF, INF, INF}
348 ,{ INF, INF, INF, INF, INF}
349 ,{ INF, INF, INF, INF, INF}
350 }
351 ,{ { -550, -550, -550, -550, -550}
352 ,{ -550, -550, -550, -550, -550}
353 ,{ -550, -550, -550, -550, -550}
354 ,{ -550, -550, -550, -1340, -550}
355 ,{ -550, -550, -550, -550, -550}
356 }
357 ,{ { -550, -550, -550, -550, -550}
358 ,{ -550, -550, -550, -550, -550}
359 ,{ -550, -550, -550, -550, -550}
360 ,{ -550, -550, -550, -1340, -550}
361 ,{ -550, -550, -550, -550, -890}
362 }
363 ,{ { -50, -50, -50, -50, -50}
364 ,{ -50, -50, -50, -50, -50}
365 ,{ -50, -50, -50, -50, -50}
366 ,{ -50, -50, -50, -830, -50}
367 ,{ -50, -50, -50, -50, -390}
368 }
369 ,{ { -50, -50, -50, -50, -50}
370 ,{ -50, -50, -50, -50, -50}
371 ,{ -50, -50, -50, -50, -50}
372 ,{ -50, -50, -50, -830, -50}
373 ,{ -50, -50, -50, -50, -50}
374 }
375 ,{ { -50, -50, -50, -50, -50}
376 ,{ -50, -50, -50, -50, -50}
377 ,{ -50, -50, -50, -50, -50}
378 ,{ -50, -50, -50, -830, -50}
379 ,{ -50, -50, -50, -50, -390}
380 }
381 ,{ { -50, -50, -50, -50, -50}
382 ,{ -50, -50, -50, -50, -50}
383 ,{ -50, -50, -50, -50, -50}
384 ,{ -50, -50, -50, -830, -50}
385 ,{ -50, -50, -50, -50, -50}
386 }
387 ,{ { -50, -50, -50, -50, -50}
388 ,{ -50, -50, -50, -50, -50}
389 ,{ -50, -50, -50, -50, -50}
390 ,{ -50, -50, -50, -830, -50}
391 ,{ -50, -50, -50, -50, -50}
392 }
393 }};


```

## 18.166 intl21.h

```

1 PUBLIC int int21_37[NBPAIRS+1][NBPAIRS+1][5][5][5] =
2 {{{{ {INF, INF, INF, INF, INF}
3 ,{ {INF, INF, INF, INF, INF}
4 ,{ {INF, INF, INF, INF, INF}
5 ,{ {INF, INF, INF, INF, INF}
6 ,{ {INF, INF, INF, INF, INF}
7 }
8 ,{ {INF, INF, INF, INF, INF}
9 ,{ {INF, INF, INF, INF, INF}
10 ,{ {INF, INF, INF, INF, INF}
11 ,{ {INF, INF, INF, INF, INF}
12 ,{ {INF, INF, INF, INF, INF}
13 }


```

```
14 , {{ INF, INF, INF, INF, INF}
15 , { INF, INF, INF, INF, INF}
16 , { INF, INF, INF, INF, INF}
17 , { INF, INF, INF, INF, INF}
18 , { INF, INF, INF, INF, INF}
19 }
20 , {{ INF, INF, INF, INF, INF}
21 , { INF, INF, INF, INF, INF}
22 , { INF, INF, INF, INF, INF}
23 , { INF, INF, INF, INF, INF}
24 , { INF, INF, INF, INF, INF}
25 }
26 , {{ INF, INF, INF, INF, INF}
27 , { INF, INF, INF, INF, INF}
28 , { INF, INF, INF, INF, INF}
29 , { INF, INF, INF, INF, INF}
30 , { INF, INF, INF, INF, INF}
31 }
32 }
33 , {{{ INF, INF, INF, INF, INF}
34 , { INF, INF, INF, INF, INF}
35 , { INF, INF, INF, INF, INF}
36 , { INF, INF, INF, INF, INF}
37 , { INF, INF, INF, INF, INF}
38 }
39 , {{ INF, INF, INF, INF, INF}
40 , { INF, INF, INF, INF, INF}
41 , { INF, INF, INF, INF, INF}
42 , { INF, INF, INF, INF, INF}
43 , { INF, INF, INF, INF, INF}
44 }
45 , {{ INF, INF, INF, INF, INF}
46 , { INF, INF, INF, INF, INF}
47 , { INF, INF, INF, INF, INF}
48 , { INF, INF, INF, INF, INF}
49 , { INF, INF, INF, INF, INF}
50 }
51 , {{ INF, INF, INF, INF, INF}
52 , { INF, INF, INF, INF, INF}
53 , { INF, INF, INF, INF, INF}
54 , { INF, INF, INF, INF, INF}
55 , { INF, INF, INF, INF, INF}
56 }
57 , {{ INF, INF, INF, INF, INF}
58 , { INF, INF, INF, INF, INF}
59 , { INF, INF, INF, INF, INF}
60 , { INF, INF, INF, INF, INF}
61 , { INF, INF, INF, INF, INF}
62 }
63 }
64 , {{{ INF, INF, INF, INF, INF}
65 , { INF, INF, INF, INF, INF}
66 , { INF, INF, INF, INF, INF}
67 , { INF, INF, INF, INF, INF}
68 , { INF, INF, INF, INF, INF}
69 }
70 , {{ INF, INF, INF, INF, INF}
71 , { INF, INF, INF, INF, INF}
72 , { INF, INF, INF, INF, INF}
73 , { INF, INF, INF, INF, INF}
74 , { INF, INF, INF, INF, INF}
75 }
76 , {{ INF, INF, INF, INF, INF}
77 , { INF, INF, INF, INF, INF}
78 , { INF, INF, INF, INF, INF}
79 , { INF, INF, INF, INF, INF}
80 , { INF, INF, INF, INF, INF}
81 }
82 , {{ INF, INF, INF, INF, INF}
83 , { INF, INF, INF, INF, INF}
84 , { INF, INF, INF, INF, INF}
85 , { INF, INF, INF, INF, INF}
86 , { INF, INF, INF, INF, INF}
87 }
88 , {{ INF, INF, INF, INF, INF}
89 , { INF, INF, INF, INF, INF}
90 , { INF, INF, INF, INF, INF}
91 , { INF, INF, INF, INF, INF}
92 , { INF, INF, INF, INF, INF}
93 }
94 }
95 , {{{ INF, INF, INF, INF, INF}
96 , { INF, INF, INF, INF, INF}
97 , { INF, INF, INF, INF, INF}
98 , { INF, INF, INF, INF, INF}
99 , { INF, INF, INF, INF, INF}
100 }}
```

```
101 , {{ INF, INF, INF, INF, INF }
102 , { INF, INF, INF, INF, INF }
103 , { INF, INF, INF, INF, INF }
104 , { INF, INF, INF, INF, INF }
105 , { INF, INF, INF, INF, INF }
106 }
107 , {{ INF, INF, INF, INF, INF }
108 , { INF, INF, INF, INF, INF }
109 , { INF, INF, INF, INF, INF }
110 , { INF, INF, INF, INF, INF }
111 , { INF, INF, INF, INF, INF }
112 }
113 , {{ INF, INF, INF, INF, INF }
114 , { INF, INF, INF, INF, INF }
115 , { INF, INF, INF, INF, INF }
116 , { INF, INF, INF, INF, INF }
117 , { INF, INF, INF, INF, INF }
118 }
119 , {{ INF, INF, INF, INF, INF }
120 , { INF, INF, INF, INF, INF }
121 , { INF, INF, INF, INF, INF }
122 , { INF, INF, INF, INF, INF }
123 , { INF, INF, INF, INF, INF }
124 }
125 }
126 , {{{ INF, INF, INF, INF, INF }
127 , { INF, INF, INF, INF, INF }
128 , { INF, INF, INF, INF, INF }
129 , { INF, INF, INF, INF, INF }
130 , { INF, INF, INF, INF, INF }
131 }
132 , {{ INF, INF, INF, INF, INF }
133 , { INF, INF, INF, INF, INF }
134 , { INF, INF, INF, INF, INF }
135 , { INF, INF, INF, INF, INF }
136 , { INF, INF, INF, INF, INF }
137 }
138 , {{ INF, INF, INF, INF, INF }
139 , { INF, INF, INF, INF, INF }
140 , { INF, INF, INF, INF, INF }
141 , { INF, INF, INF, INF, INF }
142 , { INF, INF, INF, INF, INF }
143 }
144 , {{ INF, INF, INF, INF, INF }
145 , { INF, INF, INF, INF, INF }
146 , { INF, INF, INF, INF, INF }
147 , { INF, INF, INF, INF, INF }
148 , { INF, INF, INF, INF, INF }
149 }
150 , {{ INF, INF, INF, INF, INF }
151 , { INF, INF, INF, INF, INF }
152 , { INF, INF, INF, INF, INF }
153 , { INF, INF, INF, INF, INF }
154 , { INF, INF, INF, INF, INF }
155 }
156 }
157 , {{{ INF, INF, INF, INF, INF }
158 , { INF, INF, INF, INF, INF }
159 , { INF, INF, INF, INF, INF }
160 , { INF, INF, INF, INF, INF }
161 , { INF, INF, INF, INF, INF }
162 }
163 , {{ INF, INF, INF, INF, INF }
164 , { INF, INF, INF, INF, INF }
165 , { INF, INF, INF, INF, INF }
166 , { INF, INF, INF, INF, INF }
167 , { INF, INF, INF, INF, INF }
168 }
169 , {{ INF, INF, INF, INF, INF }
170 , { INF, INF, INF, INF, INF }
171 , { INF, INF, INF, INF, INF }
172 , { INF, INF, INF, INF, INF }
173 , { INF, INF, INF, INF, INF }
174 }
175 , {{ INF, INF, INF, INF, INF }
176 , { INF, INF, INF, INF, INF }
177 , { INF, INF, INF, INF, INF }
178 , { INF, INF, INF, INF, INF }
179 , { INF, INF, INF, INF, INF }
180 }
181 , {{ INF, INF, INF, INF, INF }
182 , { INF, INF, INF, INF, INF }
183 , { INF, INF, INF, INF, INF }
184 , { INF, INF, INF, INF, INF }
185 , { INF, INF, INF, INF, INF }
186 }
187 }
```

```
188 ,{{{{ INF, INF, INF, INF, INF}
189 ,{ INF, INF, INF, INF, INF}
190 ,{ INF, INF, INF, INF, INF}
191 ,{ INF, INF, INF, INF, INF}
192 ,{ INF, INF, INF, INF, INF}
193 }
194 ,{{ INF, INF, INF, INF, INF}
195 ,{ INF, INF, INF, INF, INF}
196 ,{ INF, INF, INF, INF, INF}
197 ,{ INF, INF, INF, INF, INF}
198 ,{ INF, INF, INF, INF, INF}
199 }
200 ,{{ INF, INF, INF, INF, INF}
201 ,{ INF, INF, INF, INF, INF}
202 ,{ INF, INF, INF, INF, INF}
203 ,{ INF, INF, INF, INF, INF}
204 ,{ INF, INF, INF, INF, INF}
205 }
206 ,{{ INF, INF, INF, INF, INF}
207 ,{ INF, INF, INF, INF, INF}
208 ,{ INF, INF, INF, INF, INF}
209 ,{ INF, INF, INF, INF, INF}
210 ,{ INF, INF, INF, INF, INF}
211 }
212 ,{{ INF, INF, INF, INF, INF}
213 ,{ INF, INF, INF, INF, INF}
214 ,{ INF, INF, INF, INF, INF}
215 ,{ INF, INF, INF, INF, INF}
216 ,{ INF, INF, INF, INF, INF}
217 }
218 }
219 ,{{{{ INF, INF, INF, INF, INF}
220 ,{ INF, INF, INF, INF, INF}
221 ,{ INF, INF, INF, INF, INF}
222 ,{ INF, INF, INF, INF, INF}
223 ,{ INF, INF, INF, INF, INF}
224 }
225 ,{{ INF, INF, INF, INF, INF}
226 ,{ INF, INF, INF, INF, INF}
227 ,{ INF, INF, INF, INF, INF}
228 ,{ INF, INF, INF, INF, INF}
229 ,{ INF, INF, INF, INF, INF}
230 }
231 ,{{ INF, INF, INF, INF, INF}
232 ,{ INF, INF, INF, INF, INF}
233 ,{ INF, INF, INF, INF, INF}
234 ,{ INF, INF, INF, INF, INF}
235 ,{ INF, INF, INF, INF, INF}
236 }
237 ,{{ INF, INF, INF, INF, INF}
238 ,{ INF, INF, INF, INF, INF}
239 ,{ INF, INF, INF, INF, INF}
240 ,{ INF, INF, INF, INF, INF}
241 ,{ INF, INF, INF, INF, INF}
242 }
243 ,{{ INF, INF, INF, INF, INF}
244 ,{ INF, INF, INF, INF, INF}
245 ,{ INF, INF, INF, INF, INF}
246 ,{ INF, INF, INF, INF, INF}
247 ,{ INF, INF, INF, INF, INF}
248 }
249 }
250 }
251 ,{{{{ INF, INF, INF, INF, INF}
252 ,{ INF, INF, INF, INF, INF}
253 ,{ INF, INF, INF, INF, INF}
254 ,{ INF, INF, INF, INF, INF}
255 ,{ INF, INF, INF, INF, INF}
256 }
257 ,{{ INF, INF, INF, INF, INF}
258 ,{ INF, INF, INF, INF, INF}
259 ,{ INF, INF, INF, INF, INF}
260 ,{ INF, INF, INF, INF, INF}
261 ,{ INF, INF, INF, INF, INF}
262 }
263 ,{{ INF, INF, INF, INF, INF}
264 ,{ INF, INF, INF, INF, INF}
265 ,{ INF, INF, INF, INF, INF}
266 ,{ INF, INF, INF, INF, INF}
267 ,{ INF, INF, INF, INF, INF}
268 }
269 ,{{ INF, INF, INF, INF, INF}
270 ,{ INF, INF, INF, INF, INF}
271 ,{ INF, INF, INF, INF, INF}
272 ,{ INF, INF, INF, INF, INF}
273 ,{ INF, INF, INF, INF, INF}
274 }
```

```
275 ,{{ INF, INF, INF, INF, INF}
276 ,{ INF, INF, INF, INF, INF}
277 ,{ INF, INF, INF, INF, INF}
278 ,{ INF, INF, INF, INF, INF}
279 ,{ INF, INF, INF, INF, INF}
280 }
281 }
282 ,{{ 230, 230, 230, 230, 230}
283 ,{ 230, 230, 230, 230, 230}
284 ,{ 230, 230, 230, 230, 230}
285 ,{ 230, 230, 230, 230, 230}
286 ,{ 230, 230, 230, 230, 230}
287 }
288 ,{{ 230, 230, 230, 110, 230}
289 ,{ 230, 230, 230, 110, 230}
290 ,{ 230, 230, 230, 110, 230}
291 ,{ 110, 110, 110, 110, 110}
292 ,{ 230, 230, 230, 110, 230}
293 }
294 ,{{ 230, 230, 230, 230, 230}
295 ,{ 230, 230, 230, 230, 230}
296 ,{ 230, 230, 230, 230, 230}
297 ,{ 230, 230, 230, 230, 230}
298 ,{ 230, 230, 230, 230, 230}
299 }
300 ,{{ 230, 110, 230, 110, 230}
301 ,{ 110, 110, 110, 110, 110}
302 ,{ 230, 110, 230, 110, 230}
303 ,{ 110, 110, 110, 110, 110}
304 ,{ 230, 110, 230, 110, 230}
305 }
306 ,{{ 230, 230, 230, 230, 150}
307 ,{ 230, 230, 230, 230, 150}
308 ,{ 230, 230, 230, 230, 150}
309 ,{ 230, 150, 150, 150, 150}
310 ,{ 150, 150, 150, 150, 150}
311 }
312 }
313 ,{{ 250, 250, 250, 230, 230}
314 ,{ 250, 250, 230, 230, 230}
315 ,{ 250, 230, 250, 230, 230}
316 ,{ 230, 230, 230, 230, 230}
317 ,{ 250, 250, 230, 230, 230}
318 }
319 ,{{ 250, 250, 230, 110, 230}
320 ,{ 250, 250, 230, 110, 230}
321 ,{ 230, 230, 170, 110, 230}
322 ,{ 110, 80, 110, 110, 110}
323 ,{ 230, 230, 230, 110, 230}
324 }
325 ,{{ 250, 250, 250, 230, 230}
326 ,{ 230, 230, 230, 230, 230}
327 ,{ 250, 230, 250, 230, 230}
328 ,{ 230, 230, 230, 230, 230}
329 ,{ 250, 250, 230, 230, 230}
330 }
331 ,{{ 230, 170, 230, 110, 230}
332 ,{ 230, 170, 230, 80, 230}
333 ,{ 230, 110, 230, 110, 230}
334 ,{ 120, 120, 110, 110, 110}
335 ,{ 230, 110, 230, 110, 230}
336 }
337 ,{{ 230, 230, 230, 230, 150}
338 ,{ 230, 230, 230, 230, 150}
339 ,{ 230, 230, 220, 230, 150}
340 ,{ 230, 230, 230, 230, 150}
341 ,{ 170, 150, 170, 150, 140}
342 }
343 }
344 ,{{ 300, 300, 300, 300, 300}
345 ,{ 300, 300, 300, 300, 300}
346 ,{ 300, 300, 300, 300, 300}
347 ,{ 300, 300, 300, 300, 300}
348 ,{ 300, 300, 300, 300, 300}
349 }
350 ,{{ 300, 300, 300, 190, 300}
351 ,{ 300, 300, 300, 190, 300}
352 ,{ 300, 300, 300, 190, 300}
353 ,{ 190, 190, 190, 190, 190}
354 ,{ 300, 300, 300, 190, 300}
355 }
356 ,{{ 300, 300, 300, 300, 300}
357 ,{ 300, 300, 300, 300, 300}
358 ,{ 300, 300, 300, 300, 300}
359 ,{ 300, 300, 300, 300, 300}
360 ,{ 300, 300, 300, 300, 300}
361 }
```

```
362 ,{{ 300, 190, 300, 190, 300}
363 ,{ 300, 190, 300, 190, 300}
364 ,{ 300, 190, 300, 190, 300}
365 ,{ 190, 190, 190, 190, 190}
366 ,{ 300, 190, 300, 190, 300}
367 }
368 ,{{ 300, 300, 300, 300, 220}
369 ,{ 300, 300, 300, 300, 220}
370 ,{ 300, 300, 300, 300, 220}
371 ,{ 300, 300, 300, 300, 220}
372 ,{ 220, 220, 220, 220, 220}
373 }
374 }
375 ,{{{{ 300, 300, 300, 300, 300}
376 ,{ 300, 300, 300, 300, 300}
377 ,{ 300, 300, 300, 300, 300}
378 ,{ 300, 300, 300, 300, 300}
379 ,{ 300, 300, 300, 300, 300}
380 }
381 ,{{{{ 300, 300, 300, 190, 300}
382 ,{ 300, 300, 300, 190, 300}
383 ,{ 300, 300, 300, 190, 300}
384 ,{ 190, 190, 190, 190, 190}
385 ,{ 300, 300, 300, 190, 300}
386 }
387 ,{{{{ 300, 300, 300, 300, 300}
388 ,{ 300, 300, 300, 300, 300}
389 ,{ 300, 300, 300, 300, 300}
390 ,{ 300, 300, 300, 300, 300}
391 ,{ 300, 300, 300, 300, 300}
392 }
393 ,{{{{ 300, 190, 300, 190, 300}
394 ,{ 190, 190, 190, 190, 190}
395 ,{ 300, 190, 300, 190, 300}
396 ,{ 190, 190, 190, 190, 190}
397 ,{ 300, 190, 300, 190, 300}
398 }
399 ,{{{{ 300, 300, 300, 300, 220}
400 ,{ 300, 300, 300, 300, 220}
401 ,{ 300, 300, 300, 300, 220}
402 ,{ 300, 300, 300, 300, 220}
403 ,{ 220, 220, 220, 220, 220}
404 }
405 }
406 ,{{{{ 300, 300, 300, 300, 300}
407 ,{ 300, 300, 300, 300, 300}
408 ,{ 300, 300, 300, 300, 300}
409 ,{ 300, 300, 300, 300, 300}
410 ,{ 300, 300, 300, 300, 300}
411 }
412 ,{{{{ 300, 300, 300, 190, 300}
413 ,{ 300, 300, 300, 190, 300}
414 ,{ 300, 300, 300, 190, 300}
415 ,{ 190, 190, 190, 190, 190}
416 ,{ 300, 300, 300, 190, 300}
417 }
418 ,{{{{ 300, 300, 300, 300, 300}
419 ,{ 300, 300, 300, 300, 300}
420 ,{ 300, 300, 300, 300, 300}
421 ,{ 300, 300, 300, 300, 300}
422 ,{ 300, 300, 300, 300, 300}
423 }
424 ,{{{{ 300, 190, 300, 190, 300}
425 ,{ 300, 190, 300, 190, 300}
426 ,{ 300, 190, 300, 190, 300}
427 ,{ 190, 190, 190, 190, 190}
428 ,{ 300, 190, 300, 190, 300}
429 }
430 ,{{{{ 300, 300, 300, 300, 220}
431 ,{ 300, 300, 300, 300, 220}
432 ,{ 300, 300, 300, 300, 220}
433 ,{ 300, 300, 300, 300, 220}
434 ,{ 220, 220, 220, 220, 220}
435 }
436 }
437 ,{{{{ 300, 300, 300, 300, 300}
438 ,{ 300, 300, 300, 300, 300}
439 ,{ 300, 300, 300, 300, 300}
440 ,{ 300, 300, 300, 300, 300}
441 ,{ 300, 300, 300, 300, 300}
442 }
443 ,{{{{ 300, 300, 300, 190, 300}
444 ,{ 300, 300, 300, 190, 300}
445 ,{ 300, 300, 300, 190, 300}
446 ,{ 190, 190, 190, 190, 190}
447 ,{ 300, 300, 300, 190, 300}
448 }
```

```
449 ,{{ 300, 300, 300, 300, 300}
450 ,{{ 300, 300, 300, 300, 300}
451 ,{{ 300, 300, 300, 300, 300}
452 ,{{ 300, 300, 300, 300, 300}
453 ,{{ 300, 300, 300, 300, 300}
454 }
455 ,{{ 300, 190, 300, 190, 300}
456 ,{{ 190, 190, 190, 190, 190}
457 ,{{ 300, 190, 300, 190, 300}
458 ,{{ 190, 190, 190, 190, 190}
459 ,{{ 300, 190, 300, 190, 300}
460 }
461 ,{{ 300, 300, 300, 300, 220}
462 ,{{ 300, 300, 300, 300, 220}
463 ,{{ 300, 300, 300, 300, 220}
464 ,{{ 300, 300, 300, 300, 220}
465 ,{{ 220, 220, 220, 220, 220}
466 }
467 }
468 ,{{{ 300, 300, 300, 300, 300}
469 ,{{ 300, 300, 300, 300, 300}
470 ,{{ 300, 300, 300, 300, 300}
471 ,{{ 300, 300, 300, 300, 300}
472 ,{{ 300, 300, 300, 300, 300}
473 }
474 ,{{ 300, 300, 300, 190, 300}
475 ,{{ 300, 300, 300, 190, 300}
476 ,{{ 300, 300, 300, 190, 300}
477 ,{{ 190, 190, 190, 190, 190}
478 ,{{ 300, 300, 300, 190, 300}
479 }
480 ,{{ 300, 300, 300, 300, 300}
481 ,{{ 300, 300, 300, 300, 300}
482 ,{{ 300, 300, 300, 300, 300}
483 ,{{ 300, 300, 300, 300, 300}
484 ,{{ 300, 300, 300, 300, 300}
485 }
486 ,{{ 300, 190, 300, 190, 300}
487 ,{{ 300, 190, 300, 190, 300}
488 ,{{ 300, 190, 300, 190, 300}
489 ,{{ 190, 190, 190, 190, 190}
490 ,{{ 300, 190, 300, 190, 300}
491 }
492 ,{{ 300, 300, 300, 300, 220}
493 ,{{ 300, 300, 300, 300, 220}
494 ,{{ 300, 300, 300, 300, 220}
495 ,{{ 300, 300, 300, 300, 220}
496 ,{{ 220, 220, 220, 220, 220}
497 }
498 }
499 }
500 ,{{{ {{ INF, INF, INF, INF, INF}
501 ,{{ {{ INF, INF, INF, INF, INF}
502 ,{{ {{ INF, INF, INF, INF, INF}
503 ,{{ {{ INF, INF, INF, INF, INF}
504 ,{{ {{ INF, INF, INF, INF, INF}
505 }
506 ,{{ {{ INF, INF, INF, INF, INF}
507 ,{{ {{ INF, INF, INF, INF, INF}
508 ,{{ {{ INF, INF, INF, INF, INF}
509 ,{{ {{ INF, INF, INF, INF, INF}
510 ,{{ {{ INF, INF, INF, INF, INF}
511 }
512 ,{{ {{ INF, INF, INF, INF, INF}
513 ,{{ {{ INF, INF, INF, INF, INF}
514 ,{{ {{ INF, INF, INF, INF, INF}
515 ,{{ {{ INF, INF, INF, INF, INF}
516 ,{{ {{ INF, INF, INF, INF, INF}
517 }
518 ,{{ {{ INF, INF, INF, INF, INF}
519 ,{{ {{ INF, INF, INF, INF, INF}
520 ,{{ {{ INF, INF, INF, INF, INF}
521 ,{{ {{ INF, INF, INF, INF, INF}
522 ,{{ {{ INF, INF, INF, INF, INF}
523 }
524 ,{{ {{ INF, INF, INF, INF, INF}
525 ,{{ {{ INF, INF, INF, INF, INF}
526 ,{{ {{ INF, INF, INF, INF, INF}
527 ,{{ {{ INF, INF, INF, INF, INF}
528 ,{{ {{ INF, INF, INF, INF, INF}
529 }
530 }
531 ,{{{ 250, 250, 230, 230, 230}
532 ,{{ 250, 250, 230, 230, 230}
533 ,{{ 230, 230, 230, 230, 230}
534 ,{{ 230, 230, 230, 230, 230}
535 ,{{ 230, 230, 230, 230, 230}
```

```
536    }
537 ,{{ 250, 250, 230, 230, 230}
538 ,{ 250, 250, 230, 210, 230}
539 ,{ 230, 230, 230, 230, 230}
540 ,{ 120, 120, 110, 110, 110}
541 ,{ 230, 230, 230, 230, 230}
542 }
543 ,{{ 230, 230, 230, 230, 230}
544 ,{ 230, 230, 230, 230, 230}
545 ,{ 230, 230, 230, 230, 230}
546 ,{ 230, 230, 230, 230, 230}
547 ,{ 230, 230, 190, 230, 230}
548 }
549 ,{{ 230, 110, 230, 110, 230}
550 ,{ 110, 110, 110, 110, 110}
551 ,{ 230, 110, 230, 110, 230}
552 ,{ 110, 110, 110, 110, 110}
553 ,{ 230, 110, 230, 110, 230}
554 }
555 ,{{ 230, 230, 230, 230, 150}
556 ,{ 230, 230, 230, 230, 150}
557 ,{ 230, 230, 230, 230, 150}
558 ,{ 230, 230, 230, 230, 150}
559 ,{ 150, 150, 150, 150, 150}
560 }
561 }
562 ,{{{{ 230, 230, 230, 230, 230}
563 ,{ 230, 230, 230, 230, 230}
564 ,{ 230, 230, 230, 230, 230}
565 ,{ 230, 230, 230, 230, 230}
566 ,{ 230, 230, 230, 230, 230}
567 }
568 ,{{ 230, 230, 230, 230, 230}
569 ,{ 230, 230, 230, 230, 230}
570 ,{ 230, 230, 230, 230, 230}
571 ,{ 110, 110, 110, 110, 110}
572 ,{ 230, 230, 230, 230, 230}
573 }
574 ,{{ 230, 230, 230, 230, 230}
575 ,{ 230, 230, 230, 230, 230}
576 ,{ 230, 230, 230, 230, 230}
577 ,{ 230, 230, 230, 230, 230}
578 ,{ 230, 230, 230, 230, 230}
579 }
580 ,{{ 230, 110, 230, 110, 230}
581 ,{ 230, 110, 230, 110, 230}
582 ,{ 230, 110, 230, 110, 230}
583 ,{ 110, 110, 110, 110, 110}
584 ,{ 230, 110, 230, 110, 230}
585 }
586 ,{{ 230, 230, 230, 230, 150}
587 ,{ 230, 230, 230, 230, 150}
588 ,{ 230, 230, 230, 230, 150}
589 ,{ 230, 230, 230, 230, 150}
590 ,{ 150, 150, 150, 150, 150}
591 }
592 }
593 ,{{{ 300, 300, 300, 300, 300}
594 ,{ 300, 300, 300, 300, 300}
595 ,{ 300, 300, 300, 300, 300}
596 ,{ 300, 300, 300, 300, 300}
597 ,{ 300, 300, 300, 300, 300}
598 }
599 ,{{ 300, 300, 300, 300, 300}
600 ,{ 300, 300, 300, 300, 300}
601 ,{ 300, 300, 300, 300, 300}
602 ,{ 190, 190, 190, 190, 190}
603 ,{ 300, 300, 300, 300, 300}
604 }
605 ,{{ 300, 300, 300, 300, 300}
606 ,{ 300, 300, 300, 300, 300}
607 ,{ 300, 300, 300, 300, 300}
608 ,{ 300, 300, 300, 300, 300}
609 ,{ 300, 300, 300, 300, 300}
610 }
611 ,{{ 300, 190, 300, 190, 300}
612 ,{ 300, 190, 300, 190, 300}
613 ,{ 300, 190, 300, 190, 300}
614 ,{ 190, 190, 190, 190, 190}
615 ,{ 300, 190, 300, 190, 300}
616 }
617 ,{{ 300, 300, 300, 300, 220}
618 ,{ 300, 300, 300, 300, 220}
619 ,{ 300, 300, 300, 300, 220}
620 ,{ 300, 300, 300, 300, 220}
621 ,{ 220, 220, 220, 220, 220}
622 }
```

```
623  }
624 ,{{ { 300, 300, 300, 300, 300}
625 ,{ 300, 300, 300, 300, 300}
626 ,{ 300, 300, 300, 300, 300}
627 ,{ 300, 300, 300, 300, 300}
628 ,{ 300, 300, 300, 300, 300}
629 }
630 ,{ { 300, 300, 300, 300, 300}
631 ,{ 300, 250, 300, 210, 300}
632 ,{ 300, 300, 300, 300, 300}
633 ,{ 190, 120, 190, 190, 190}
634 ,{ 300, 300, 300, 300, 300}
635 }
636 ,{ { 300, 300, 300, 300, 300}
637 ,{ 300, 300, 300, 300, 300}
638 ,{ 300, 300, 300, 300, 300}
639 ,{ 300, 300, 300, 300, 300}
640 ,{ 300, 300, 190, 300, 300}
641 }
642 ,{ { 300, 190, 300, 190, 300}
643 ,{ 190, 190, 190, 190, 190}
644 ,{ 300, 190, 300, 190, 300}
645 ,{ 190, 190, 190, 190, 190}
646 ,{ 300, 190, 300, 190, 300}
647 }
648 ,{ { 300, 300, 300, 300, 220}
649 ,{ 300, 300, 300, 300, 220}
650 ,{ 300, 300, 300, 300, 220}
651 ,{ 300, 300, 300, 300, 220}
652 ,{ 220, 220, 220, 220, 220}
653 }
654 }
655 ,{{ { 300, 300, 300, 300, 300}
656 ,{ 300, 300, 300, 300, 300}
657 ,{ 300, 300, 300, 300, 300}
658 ,{ 300, 300, 300, 300, 300}
659 ,{ 300, 300, 300, 300, 300}
660 }
661 ,{ { 300, 300, 300, 300, 300}
662 ,{ 300, 300, 300, 300, 300}
663 ,{ 300, 300, 300, 300, 300}
664 ,{ 190, 190, 190, 190, 190}
665 ,{ 300, 300, 300, 300, 300}
666 }
667 ,{ { 300, 300, 300, 300, 300}
668 ,{ 300, 300, 300, 300, 300}
669 ,{ 300, 300, 300, 300, 300}
670 ,{ 300, 300, 300, 300, 300}
671 ,{ 300, 300, 300, 300, 300}
672 }
673 ,{ { 300, 190, 300, 190, 300}
674 ,{ 300, 190, 300, 190, 300}
675 ,{ 300, 190, 300, 190, 300}
676 ,{ 190, 190, 190, 190, 190}
677 ,{ 300, 190, 300, 190, 300}
678 }
679 ,{ { 300, 300, 300, 300, 220}
680 ,{ 300, 300, 300, 300, 220}
681 ,{ 300, 300, 300, 300, 220}
682 ,{ 300, 300, 300, 300, 220}
683 ,{ 220, 220, 220, 220, 220}
684 }
685 }
686 ,{{ { 300, 300, 300, 300, 300}
687 ,{ 300, 300, 300, 300, 300}
688 ,{ 300, 300, 300, 300, 300}
689 ,{ 300, 300, 300, 300, 300}
690 ,{ 300, 300, 300, 300, 300}
691 }
692 ,{ { 300, 300, 300, 300, 300}
693 ,{ 300, 300, 300, 300, 300}
694 ,{ 300, 300, 300, 300, 300}
695 ,{ 190, 190, 190, 190, 190}
696 ,{ 300, 300, 300, 300, 300}
697 }
698 ,{ { 300, 300, 300, 300, 300}
699 ,{ 300, 300, 300, 300, 300}
700 ,{ 300, 300, 300, 300, 300}
701 ,{ 300, 300, 300, 300, 300}
702 ,{ 300, 300, 300, 300, 300}
703 }
704 ,{ { 300, 190, 300, 190, 300}
705 ,{ 190, 190, 190, 190, 190}
706 ,{ 300, 190, 300, 190, 300}
707 ,{ 190, 190, 190, 190, 190}
708 ,{ 300, 190, 300, 190, 300}
709 }
```

```
710 ,{{ 300, 300, 300, 300, 220}
711 ,{{ 300, 300, 300, 300, 220}
712 ,{{ 300, 300, 300, 300, 220}
713 ,{{ 300, 300, 300, 300, 220}
714 ,{{ 220, 220, 220, 220, 220}
715 }
716 }
717 ,{{{{ INF, INF, INF, INF, INF}
718 ,{{{{ INF, INF, INF, INF, INF}
719 ,{{{{ INF, INF, INF, INF, INF}
720 ,{{{{ INF, INF, INF, INF, INF}
721 ,{{{{ INF, INF, INF, INF, INF}
722 }
723 ,{{{{ INF, INF, INF, INF, INF}
724 ,{{{{ INF, INF, INF, INF, INF}
725 ,{{{{ INF, INF, INF, INF, INF}
726 ,{{{{ INF, INF, INF, INF, INF}
727 ,{{{{ INF, INF, INF, INF, INF}
728 }
729 ,{{{{ INF, INF, INF, INF, INF}
730 ,{{{{ INF, INF, INF, INF, INF}
731 ,{{{{ INF, INF, INF, INF, INF}
732 ,{{{{ INF, INF, INF, INF, INF}
733 ,{{{{ INF, INF, INF, INF, INF}
734 }
735 ,{{{{ INF, INF, INF, INF, INF}
736 ,{{{{ INF, INF, INF, INF, INF}
737 ,{{{{ INF, INF, INF, INF, INF}
738 ,{{{{ INF, INF, INF, INF, INF}
739 ,{{{{ INF, INF, INF, INF, INF}
740 }
741 ,{{{{ INF, INF, INF, INF, INF}
742 ,{{{{ INF, INF, INF, INF, INF}
743 ,{{{{ INF, INF, INF, INF, INF}
744 ,{{{{ INF, INF, INF, INF, INF}
745 ,{{{{ 220, 220, 220, 220, 220}
746 }
747 }
748 }
749 ,{{{{{{ INF, INF, INF, INF, INF}
750 ,{{{{{{ INF, INF, INF, INF, INF}
751 ,{{{{{{ INF, INF, INF, INF, INF}
752 ,{{{{{{ INF, INF, INF, INF, INF}
753 ,{{{{{{ INF, INF, INF, INF, INF}
754 }
755 ,{{{{{{ INF, INF, INF, INF, INF}
756 ,{{{{{{ INF, INF, INF, INF, INF}
757 ,{{{{{{ INF, INF, INF, INF, INF}
758 ,{{{{{{ INF, INF, INF, INF, INF}
759 ,{{{{{{ INF, INF, INF, INF, INF}
760 }
761 ,{{{{{{ INF, INF, INF, INF, INF}
762 ,{{{{{{ INF, INF, INF, INF, INF}
763 ,{{{{{{ INF, INF, INF, INF, INF}
764 ,{{{{{{ INF, INF, INF, INF, INF}
765 ,{{{{{{ INF, INF, INF, INF, INF}
766 }
767 ,{{{{{{ INF, INF, INF, INF, INF}
768 ,{{{{{{ INF, INF, INF, INF, INF}
769 ,{{{{{{ INF, INF, INF, INF, INF}
770 ,{{{{{{ INF, INF, INF, INF, INF}
771 ,{{{{{{ INF, INF, INF, INF, INF}
772 }
773 ,{{{{{{ INF, INF, INF, INF, INF}
774 ,{{{{{{ INF, INF, INF, INF, INF}
775 ,{{{{{{ INF, INF, INF, INF, INF}
776 ,{{{{{{ INF, INF, INF, INF, INF}
777 ,{{{{{{ INF, INF, INF, INF, INF}
778 }
779 }
780 ,{{{{{{ 300, 300, 300, 300, 300}
781 ,{{{{{{ 300, 300, 300, 300, 300}
782 ,{{{{{{ 300, 300, 300, 300, 300}
783 ,{{{{{{ 300, 300, 300, 300, 300}
784 ,{{{{{{ 300, 300, 300, 300, 300}
785 }
786 ,{{{{{{ 300, 300, 300, 300, 300}
787 ,{{{{{{ 300, 250, 300, 210, 300}
788 ,{{{{{{ 300, 300, 300, 300, 300}
789 ,{{{{{{ 190, 120, 190, 190, 190}
790 ,{{{{{{ 300, 300, 300, 300, 300}
791 }
792 ,{{{{{{ 300, 300, 300, 300, 300}
793 ,{{{{{{ 300, 300, 300, 300, 300}
794 ,{{{{{{ 300, 300, 300, 300, 300}
795 ,{{{{{{ 300, 300, 300, 300, 300}
796 ,{{{{{{ 300, 300, 190, 300, 300}}
```

```
797    }
798 ,{{ 300,   190,   300,   190,   300}
799 ,{ 190,   190,   190,   190,   190}
800 ,{ 300,   190,   300,   190,   300}
801 ,{ 190,   190,   190,   190,   190}
802 ,{ 300,   190,   300,   190,   300}
803 }
804 ,{{ 300,   300,   300,   300,   220}
805 ,{ 300,   300,   300,   300,   220}
806 ,{ 300,   300,   300,   300,   220}
807 ,{ 300,   300,   300,   300,   220}
808 ,{ 220,   220,   220,   220,   220}
809 }
810 }
811 ,{{{{ 300,   300,   300,   300,   300}
812 ,{ 300,   300,   300,   300,   300}
813 ,{ 300,   300,   300,   300,   300}
814 ,{ 300,   300,   300,   300,   300}
815 ,{ 300,   300,   300,   300,   300}
816 }
817 ,{{{{ 300,   300,   300,   300,   300}
818 ,{ 300,   300,   300,   300,   300}
819 ,{ 300,   300,   300,   300,   300}
820 ,{ 190,   190,   190,   190,   190}
821 ,{ 300,   300,   300,   300,   300}
822 }
823 ,{{{{ 300,   300,   300,   300,   300}
824 ,{ 300,   300,   300,   300,   300}
825 ,{ 300,   300,   300,   300,   300}
826 ,{ 300,   300,   300,   300,   300}
827 ,{ 300,   300,   300,   300,   300}
828 }
829 ,{{{{ 300,   190,   300,   190,   300}
830 ,{ 300,   190,   300,   190,   300}
831 ,{ 300,   190,   300,   190,   300}
832 ,{ 190,   190,   190,   190,   190}
833 ,{ 300,   190,   300,   190,   300}
834 }
835 ,{{{{ 300,   300,   300,   300,   220}
836 ,{ 300,   300,   300,   300,   220}
837 ,{ 300,   300,   300,   300,   220}
838 ,{ 300,   300,   300,   300,   220}
839 ,{ 220,   220,   220,   220,   220}
840 }
841 }
842 ,{{{{ 370,   370,   370,   370,   370}
843 ,{ 370,   370,   370,   370,   370}
844 ,{ 370,   370,   370,   370,   370}
845 ,{ 370,   370,   370,   370,   370}
846 ,{ 370,   370,   370,   370,   370}
847 }
848 ,{{{{ 370,   370,   370,   370,   370}
849 ,{ 370,   370,   370,   370,   370}
850 ,{ 370,   370,   370,   370,   370}
851 ,{ 260,   260,   260,   260,   260}
852 ,{ 370,   370,   370,   370,   370}
853 }
854 ,{{{{ 370,   370,   370,   370,   370}
855 ,{ 370,   370,   370,   370,   370}
856 ,{ 370,   370,   370,   370,   370}
857 ,{ 370,   370,   370,   370,   370}
858 ,{ 370,   370,   370,   370,   370}
859 }
860 ,{{{{ 370,   260,   370,   260,   370}
861 ,{ 370,   260,   370,   260,   370}
862 ,{ 370,   260,   370,   260,   370}
863 ,{ 260,   260,   260,   260,   260}
864 ,{ 370,   260,   370,   260,   370}
865 }
866 ,{{{{ 370,   370,   370,   370,   300}
867 ,{ 370,   370,   370,   370,   300}
868 ,{ 370,   370,   370,   370,   300}
869 ,{ 370,   370,   370,   370,   300}
870 ,{ 300,   300,   300,   300,   300}
871 }
872 }
873 ,{{{{ 370,   370,   370,   370,   370}
874 ,{ 370,   370,   370,   370,   370}
875 ,{ 370,   370,   370,   370,   370}
876 ,{ 370,   370,   370,   370,   370}
877 ,{ 370,   370,   370,   370,   370}
878 }
879 ,{{{{ 370,   370,   370,   370,   370}
880 ,{ 370,   250,   370,   210,   370}
881 ,{ 370,   370,   370,   370,   370}
882 ,{ 260,   120,   260,   260,   260}
883 ,{ 370,   370,   370,   370,   370}
```

```
884    }
885 ,{{ 370, 370, 370, 370, 370}
886 ,{ 370, 370, 370, 370, 370}
887 ,{ 370, 370, 370, 370, 370}
888 ,{ 370, 370, 370, 370, 370}
889 ,{ 370, 370, 190, 370, 370}
890 }
891 ,{{ 370, 260, 370, 260, 370}
892 ,{ 260, 260, 260, 260, 260}
893 ,{ 370, 260, 370, 260, 370}
894 ,{ 260, 260, 260, 260, 260}
895 ,{ 370, 260, 370, 260, 370}
896 }
897 ,{{ 370, 370, 370, 370, 300}
898 ,{ 370, 370, 370, 370, 300}
899 ,{ 370, 370, 370, 370, 300}
900 ,{ 370, 370, 370, 370, 300}
901 ,{ 300, 300, 300, 300, 300}
902 }
903 }
904 ,{{{{ 370, 370, 370, 370, 370}
905 ,{ 370, 370, 370, 370, 370}
906 ,{ 370, 370, 370, 370, 370}
907 ,{ 370, 370, 370, 370, 370}
908 ,{ 370, 370, 370, 370, 370}
909 }
910 ,{{{{ 370, 370, 370, 370, 370}
911 ,{ 370, 370, 370, 370, 370}
912 ,{ 370, 370, 370, 370, 370}
913 ,{ 260, 260, 260, 260, 260}
914 ,{ 370, 370, 370, 370, 370}
915 }
916 ,{{{{ 370, 370, 370, 370, 370}
917 ,{ 370, 370, 370, 370, 370}
918 ,{ 370, 370, 370, 370, 370}
919 ,{ 370, 370, 370, 370, 370}
920 ,{ 370, 370, 370, 370, 370}
921 }
922 ,{{{{ 370, 260, 370, 260, 370}
923 ,{ 370, 260, 370, 260, 370}
924 ,{ 370, 260, 370, 260, 370}
925 ,{ 260, 260, 260, 260, 260}
926 ,{ 370, 260, 370, 260, 370}
927 }
928 ,{{{{ 370, 370, 370, 370, 300}
929 ,{ 370, 370, 370, 370, 300}
930 ,{ 370, 370, 370, 370, 300}
931 ,{ 370, 370, 370, 370, 300}
932 ,{ 300, 300, 300, 300, 300}
933 }
934 }
935 ,{{{{ 370, 370, 370, 370, 370}
936 ,{ 370, 370, 370, 370, 370}
937 ,{ 370, 370, 370, 370, 370}
938 ,{ 370, 370, 370, 370, 370}
939 ,{ 370, 370, 370, 370, 370}
940 }
941 ,{{{{ 370, 370, 370, 370, 370}
942 ,{ 370, 370, 370, 370, 370}
943 ,{ 370, 370, 370, 370, 370}
944 ,{ 260, 260, 260, 260, 260}
945 ,{ 370, 370, 370, 370, 370}
946 }
947 ,{{{{ 370, 370, 370, 370, 370}
948 ,{ 370, 370, 370, 370, 370}
949 ,{ 370, 370, 370, 370, 370}
950 ,{ 370, 370, 370, 370, 370}
951 ,{ 370, 370, 370, 370, 370}
952 }
953 ,{{{{ 370, 260, 370, 260, 370}
954 ,{ 260, 260, 260, 260, 260}
955 ,{ 370, 260, 370, 260, 370}
956 ,{ 260, 260, 260, 260, 260}
957 ,{ 370, 260, 370, 260, 370}
958 }
959 ,{{{{ 370, 370, 370, 370, 300}
960 ,{ 370, 370, 370, 370, 300}
961 ,{ 370, 370, 370, 370, 300}
962 ,{ 370, 370, 370, 370, 300}
963 ,{ 300, 300, 300, 300, 300}
964 }
965 }
966 ,{{{{ 370, 370, 370, 370, 370}
967 ,{ 370, 370, 370, 370, 370}
968 ,{ 370, 370, 370, 370, 370}
969 ,{ 370, 370, 370, 370, 370}
970 ,{ 370, 370, 370, 370, 370}
```

```
971    }
972 ,{{ 370,   370,   370,   370,   370}
973 ,{ 370,   370,   370,   370,   370}
974 ,{ 370,   370,   370,   370,   370}
975 ,{ 260,   260,   260,   260,   260}
976 ,{ 370,   370,   370,   370,   370}
977 }
978 ,{{ 370,   370,   370,   370,   370}
979 ,{ 370,   370,   370,   370,   370}
980 ,{ 370,   370,   370,   370,   370}
981 ,{ 370,   370,   370,   370,   370}
982 ,{ 370,   370,   370,   370,   370}
983 }
984 ,{{ 370,   260,   370,   260,   370}
985 ,{ 370,   260,   370,   260,   370}
986 ,{ 370,   260,   370,   260,   370}
987 ,{ 260,   260,   260,   260,   260}
988 ,{ 370,   260,   370,   260,   370}
989 }
990 ,{{ 370,   370,   370,   370,   300}
991 ,{ 370,   370,   370,   370,   300}
992 ,{ 370,   370,   370,   370,   300}
993 ,{ 370,   370,   370,   370,   300}
994 ,{ 300,   300,   300,   300,   300}
995 }
996 }
997 }
998 ,{{{ INF,   INF,   INF,   INF,   INF}
999 ,{ INF,   INF,   INF,   INF,   INF}
1000 ,{ INF,   INF,   INF,   INF,   INF}
1001 ,{ INF,   INF,   INF,   INF,   INF}
1002 ,{ INF,   INF,   INF,   INF,   INF}
1003 }
1004 ,{{ INF,   INF,   INF,   INF,   INF}
1005 ,{ INF,   INF,   INF,   INF,   INF}
1006 ,{ INF,   INF,   INF,   INF,   INF}
1007 ,{ INF,   INF,   INF,   INF,   INF}
1008 ,{ INF,   INF,   INF,   INF,   INF}
1009 }
1010 ,{{ INF,   INF,   INF,   INF,   INF}
1011 ,{ INF,   INF,   INF,   INF,   INF}
1012 ,{ INF,   INF,   INF,   INF,   INF}
1013 ,{ INF,   INF,   INF,   INF,   INF}
1014 ,{ INF,   INF,   INF,   INF,   INF}
1015 }
1016 ,{{ INF,   INF,   INF,   INF,   INF}
1017 ,{ INF,   INF,   INF,   INF,   INF}
1018 ,{ INF,   INF,   INF,   INF,   INF}
1019 ,{ INF,   INF,   INF,   INF,   INF}
1020 ,{ INF,   INF,   INF,   INF,   INF}
1021 }
1022 ,{{ INF,   INF,   INF,   INF,   INF}
1023 ,{ INF,   INF,   INF,   INF,   INF}
1024 ,{ INF,   INF,   INF,   INF,   INF}
1025 ,{ INF,   INF,   INF,   INF,   INF}
1026 ,{ INF,   INF,   INF,   INF,   INF}
1027 }
1028 }
1029 ,{{{ 300,   300,   300,   300,   300}
1030 ,{ 300,   300,   300,   300,   300}
1031 ,{ 300,   300,   300,   300,   300}
1032 ,{ 300,   300,   300,   300,   300}
1033 ,{ 300,   300,   300,   300,   300}
1034 }
1035 ,{{ 300,   300,   300,   190,   300}
1036 ,{ 300,   300,   300,   190,   300}
1037 ,{ 300,   300,   300,   190,   300}
1038 ,{ 190,   190,   190,   190,   190}
1039 ,{ 300,   300,   300,   190,   300}
1040 }
1041 ,{{ 300,   300,   300,   300,   300}
1042 ,{ 300,   300,   300,   300,   300}
1043 ,{ 300,   300,   300,   300,   300}
1044 ,{ 300,   300,   300,   300,   300}
1045 ,{ 300,   300,   300,   300,   300}
1046 }
1047 ,{{ 300,   190,   300,   190,   300}
1048 ,{ 190,   190,   190,   190,   190}
1049 ,{ 300,   190,   300,   190,   300}
1050 ,{ 190,   190,   190,   190,   190}
1051 ,{ 300,   190,   300,   190,   300}
1052 }
1053 ,{{ 300,   300,   300,   300,   220}
1054 ,{ 300,   300,   300,   300,   220}
1055 ,{ 300,   300,   300,   300,   220}
1056 ,{ 300,   300,   300,   300,   220}
1057 ,{ 220,   220,   220,   220,   220}
```

```
1058     }
1059   }
1060   ,{{ 300, 300, 300, 300, 300}
1061   ,{ 300, 300, 300, 300, 300}
1062   ,{ 300, 300, 300, 300, 300}
1063   ,{ 300, 300, 300, 300, 300}
1064   ,{ 300, 300, 300, 300, 300}
1065   }
1066   ,{{ 300, 300, 300, 190, 300}
1067   ,{ 300, 300, 300, 190, 300}
1068   ,{ 300, 300, 300, 190, 300}
1069   ,{ 190, 190, 190, 190, 190}
1070   ,{ 300, 300, 300, 190, 300}
1071   }
1072   ,{{ 300, 300, 300, 300, 300}
1073   ,{ 300, 300, 300, 300, 300}
1074   ,{ 300, 300, 300, 300, 300}
1075   ,{ 300, 300, 300, 300, 300}
1076   ,{ 300, 300, 300, 300, 300}
1077   }
1078   ,{{ 300, 190, 300, 190, 300}
1079   ,{ 300, 190, 300, 190, 300}
1080   ,{ 300, 190, 300, 190, 300}
1081   ,{ 190, 190, 190, 190, 190}
1082   ,{ 300, 190, 300, 190, 300}
1083   }
1084   ,{{ 300, 300, 300, 300, 220}
1085   ,{ 300, 300, 300, 300, 220}
1086   ,{ 300, 300, 300, 300, 220}
1087   ,{ 300, 300, 300, 300, 220}
1088   ,{ 220, 220, 220, 220, 220}
1089   }
1090   }
1091   ,{{ 370, 370, 370, 370, 370}
1092   ,{ 370, 370, 370, 370, 370}
1093   ,{ 370, 370, 370, 370, 370}
1094   ,{ 370, 370, 370, 370, 370}
1095   ,{ 370, 370, 370, 370, 370}
1096   }
1097   ,{{ 370, 370, 370, 260, 370}
1098   ,{ 370, 370, 370, 260, 370}
1099   ,{ 370, 370, 370, 260, 370}
1100   ,{ 260, 260, 260, 260, 260}
1101   ,{ 370, 370, 370, 260, 370}
1102   }
1103   ,{{ 370, 370, 370, 370, 370}
1104   ,{ 370, 370, 370, 370, 370}
1105   ,{ 370, 370, 370, 370, 370}
1106   ,{ 370, 370, 370, 370, 370}
1107   ,{ 370, 370, 370, 370, 370}
1108   }
1109   ,{{ 370, 260, 370, 260, 370}
1110   ,{ 370, 260, 370, 260, 370}
1111   ,{ 370, 260, 370, 260, 370}
1112   ,{ 260, 260, 260, 260, 260}
1113   ,{ 370, 260, 370, 260, 370}
1114   }
1115   ,{{ 370, 370, 370, 370, 300}
1116   ,{ 370, 370, 370, 370, 300}
1117   ,{ 370, 370, 370, 370, 300}
1118   ,{ 370, 370, 370, 370, 300}
1119   ,{ 300, 300, 300, 300, 300}
1120   }
1121   }
1122   ,{{ 370, 370, 370, 370, 370}
1123   ,{ 370, 370, 370, 370, 370}
1124   ,{ 370, 370, 370, 370, 370}
1125   ,{ 370, 370, 370, 370, 370}
1126   ,{ 370, 370, 370, 370, 370}
1127   }
1128   ,{{ 370, 370, 370, 260, 370}
1129   ,{ 370, 370, 370, 260, 370}
1130   ,{ 370, 370, 370, 260, 370}
1131   ,{ 260, 260, 260, 260, 260}
1132   ,{ 370, 370, 370, 260, 370}
1133   }
1134   ,{{ 370, 370, 370, 370, 370}
1135   ,{ 370, 370, 370, 370, 370}
1136   ,{ 370, 370, 370, 370, 370}
1137   ,{ 370, 370, 370, 370, 370}
1138   ,{ 370, 370, 370, 370, 370}
1139   }
1140   ,{{ 370, 260, 370, 260, 370}
1141   ,{ 260, 260, 260, 260, 260}
1142   ,{ 370, 260, 370, 260, 370}
1143   ,{ 260, 260, 260, 260, 260}
1144   ,{ 370, 260, 370, 260, 370}
```

```
1145      }
1146 ,{{ 370, 370, 370, 370, 300}
1147 ,{ 370, 370, 370, 370, 300}
1148 ,{ 370, 370, 370, 370, 300}
1149 ,{ 370, 370, 370, 370, 300}
1150 ,{ 300, 300, 300, 300, 300}
1151 }
1152 }
1153 ,{{{ 370, 370, 370, 370, 370}
1154 ,{ 370, 370, 370, 370, 370}
1155 ,{ 370, 370, 370, 370, 370}
1156 ,{ 370, 370, 370, 370, 370}
1157 ,{ 370, 370, 370, 370, 370}
1158 }
1159 ,{{ 370, 370, 370, 260, 370}
1160 ,{ 370, 370, 370, 260, 370}
1161 ,{ 370, 370, 370, 260, 370}
1162 ,{ 260, 260, 260, 260, 260}
1163 ,{ 370, 370, 370, 260, 370}
1164 }
1165 ,{{ 370, 370, 370, 370, 370}
1166 ,{ 370, 370, 370, 370, 370}
1167 ,{ 370, 370, 370, 370, 370}
1168 ,{ 370, 370, 370, 370, 370}
1169 ,{ 370, 370, 370, 370, 370}
1170 }
1171 ,{{ 370, 260, 370, 260, 370}
1172 ,{ 370, 260, 370, 260, 370}
1173 ,{ 370, 260, 370, 260, 370}
1174 ,{ 260, 260, 260, 260, 260}
1175 ,{ 370, 260, 370, 260, 370}
1176 }
1177 ,{{ 370, 370, 370, 370, 300}
1178 ,{ 370, 370, 370, 370, 300}
1179 ,{ 370, 370, 370, 370, 300}
1180 ,{ 370, 370, 370, 370, 300}
1181 ,{ 300, 300, 300, 300, 300}
1182 }
1183 }
1184 ,{{{ 370, 370, 370, 370, 370}
1185 ,{ 370, 370, 370, 370, 370}
1186 ,{ 370, 370, 370, 370, 370}
1187 ,{ 370, 370, 370, 370, 370}
1188 ,{ 370, 370, 370, 370, 370}
1189 }
1190 ,{{{ 370, 370, 370, 260, 370}
1191 ,{ 370, 370, 370, 260, 370}
1192 ,{ 370, 370, 370, 260, 370}
1193 ,{ 260, 260, 260, 260, 260}
1194 ,{ 370, 370, 370, 260, 370}
1195 }
1196 ,{{{ 370, 370, 370, 370, 370}
1197 ,{ 370, 370, 370, 370, 370}
1198 ,{ 370, 370, 370, 370, 370}
1199 ,{ 370, 370, 370, 370, 370}
1200 ,{ 370, 370, 370, 370, 370}
1201 }
1202 ,{{{ 370, 260, 370, 260, 370}
1203 ,{ 260, 260, 260, 260, 260}
1204 ,{ 370, 260, 370, 260, 370}
1205 ,{ 260, 260, 260, 260, 260}
1206 ,{ 370, 260, 370, 260, 370}
1207 }
1208 ,{{{ 370, 370, 370, 370, 300}
1209 ,{ 370, 370, 370, 370, 300}
1210 ,{ 370, 370, 370, 370, 300}
1211 ,{ 370, 370, 370, 370, 300}
1212 ,{ 300, 300, 300, 300, 300}
1213 }
1214 }
1215 ,{{{ 370, 370, 370, 370, 370}
1216 ,{ 370, 370, 370, 370, 370}
1217 ,{ 370, 370, 370, 370, 370}
1218 ,{ 370, 370, 370, 370, 370}
1219 ,{ 370, 370, 370, 370, 370}
1220 }
1221 ,{{{ 370, 370, 370, 260, 370}
1222 ,{ 370, 370, 370, 260, 370}
1223 ,{ 370, 370, 370, 260, 370}
1224 ,{ 260, 260, 260, 260, 260}
1225 ,{ 370, 370, 370, 260, 370}
1226 }
1227 ,{{{ 370, 370, 370, 370, 370}
1228 ,{ 370, 370, 370, 370, 370}
1229 ,{ 370, 370, 370, 370, 370}
1230 ,{ 370, 370, 370, 370, 370}
1231 ,{ 370, 370, 370, 370, 370}
```

```
1232      }
1233 ,{{ 370, 260, 370, 260, 370}
1234 ,{ 370, 260, 370, 260, 370}
1235 ,{ 370, 260, 370, 260, 370}
1236 ,{ 260, 260, 260, 260, 260}
1237 ,{ 370, 260, 370, 260, 370}
1238 }
1239 ,{{ 370, 370, 370, 370, 300}
1240 ,{ 370, 370, 370, 370, 300}
1241 ,{ 370, 370, 370, 370, 300}
1242 ,{ 370, 370, 370, 370, 300}
1243 ,{ 300, 300, 300, 300, 300}
1244 }
1245 }
1246 }
1247 ,{{{ INF, INF, INF, INF, INF}
1248 ,{ INF, INF, INF, INF, INF}
1249 ,{ INF, INF, INF, INF, INF}
1250 ,{ INF, INF, INF, INF, INF}
1251 ,{ INF, INF, INF, INF, INF}
1252 }
1253 ,{{ INF, INF, INF, INF, INF}
1254 ,{ INF, INF, INF, INF, INF}
1255 ,{ INF, INF, INF, INF, INF}
1256 ,{ INF, INF, INF, INF, INF}
1257 ,{ INF, INF, INF, INF, INF}
1258 }
1259 ,{{ INF, INF, INF, INF, INF}
1260 ,{ INF, INF, INF, INF, INF}
1261 ,{ INF, INF, INF, INF, INF}
1262 ,{ INF, INF, INF, INF, INF}
1263 ,{ INF, INF, INF, INF, INF}
1264 }
1265 ,{{ INF, INF, INF, INF, INF}
1266 ,{ INF, INF, INF, INF, INF}
1267 ,{ INF, INF, INF, INF, INF}
1268 ,{ INF, INF, INF, INF, INF}
1269 ,{ INF, INF, INF, INF, INF}
1270 }
1271 ,{{ INF, INF, INF, INF, INF}
1272 ,{ INF, INF, INF, INF, INF}
1273 ,{ INF, INF, INF, INF, INF}
1274 ,{ INF, INF, INF, INF, INF}
1275 ,{ INF, INF, INF, INF, INF}
1276 }
1277 }
1278 ,{{{{ 300, 300, 300, 300, 300}
1279 ,{ 300, 300, 300, 300, 300}
1280 ,{ 300, 300, 300, 300, 300}
1281 ,{ 300, 300, 300, 300, 300}
1282 ,{ 300, 300, 300, 300, 300}
1283 }
1284 ,{{ 300, 300, 300, 300, 300}
1285 ,{ 300, 300, 300, 300, 300}
1286 ,{ 300, 300, 300, 300, 300}
1287 ,{ 190, 190, 190, 190, 190}
1288 ,{ 300, 300, 300, 300, 300}
1289 }
1290 ,{{ 300, 300, 300, 300, 300}
1291 ,{ 300, 300, 300, 300, 300}
1292 ,{ 300, 300, 300, 300, 300}
1293 ,{ 300, 300, 300, 300, 300}
1294 ,{ 300, 300, 300, 300, 300}
1295 }
1296 ,{{ 300, 190, 300, 190, 300}
1297 ,{ 190, 190, 190, 190, 190}
1298 ,{ 300, 190, 300, 190, 300}
1299 ,{ 190, 190, 190, 190, 190}
1300 ,{ 300, 190, 300, 190, 300}
1301 }
1302 ,{{ 300, 300, 300, 300, 220}
1303 ,{ 300, 300, 300, 300, 220}
1304 ,{ 300, 300, 300, 300, 220}
1305 ,{ 300, 300, 300, 300, 220}
1306 ,{ 220, 220, 220, 220, 220}
1307 }
1308 }
1309 ,{{{{ 300, 300, 300, 300, 300}
1310 ,{ 300, 300, 300, 300, 300}
1311 ,{ 300, 300, 300, 300, 300}
1312 ,{ 300, 300, 300, 300, 300}
1313 ,{ 300, 300, 300, 300, 300}
1314 }
1315 ,{{ 300, 300, 300, 300, 300}
1316 ,{ 300, 300, 300, 300, 300}
1317 ,{ 300, 300, 300, 300, 300}
1318 ,{ 190, 190, 190, 190, 190}}
```

```
1319 , { 300, 300, 300, 300, 300}
1320 }
1321 , {{ 300, 300, 300, 300, 300}
1322 , { 300, 300, 300, 300, 300}
1323 , { 300, 300, 300, 300, 300}
1324 , { 300, 300, 300, 300, 300}
1325 , { 300, 300, 300, 300, 300}
1326 }
1327 , {{ 300, 190, 300, 190, 300}
1328 , { 300, 190, 300, 190, 300}
1329 , { 300, 190, 300, 190, 300}
1330 , { 190, 190, 190, 190, 190}
1331 , { 300, 190, 300, 190, 300}
1332 }
1333 , {{ 300, 300, 300, 300, 220}
1334 , { 300, 300, 300, 300, 220}
1335 , { 300, 300, 300, 300, 220}
1336 , { 300, 300, 300, 300, 220}
1337 , { 220, 220, 220, 220, 220}
1338 }
1339 }
1340 , {{{ 370, 370, 370, 370, 370}
1341 , { 370, 370, 370, 370, 370}
1342 , { 370, 370, 370, 370, 370}
1343 , { 370, 370, 370, 370, 370}
1344 , { 370, 370, 370, 370, 370}
1345 }
1346 , {{ 370, 370, 370, 370, 370}
1347 , { 370, 370, 370, 370, 370}
1348 , { 370, 370, 370, 370, 370}
1349 , { 260, 260, 260, 260, 260}
1350 , { 370, 370, 370, 370, 370}
1351 }
1352 , {{ 370, 370, 370, 370, 370}
1353 , { 370, 370, 370, 370, 370}
1354 , { 370, 370, 370, 370, 370}
1355 , { 370, 370, 370, 370, 370}
1356 , { 370, 370, 370, 370, 370}
1357 }
1358 , {{ 370, 260, 370, 260, 370}
1359 , { 370, 260, 370, 260, 370}
1360 , { 370, 260, 370, 260, 370}
1361 , { 260, 260, 260, 260, 260}
1362 , { 370, 260, 370, 260, 370}
1363 }
1364 , {{ 370, 370, 370, 370, 300}
1365 , { 370, 370, 370, 370, 300}
1366 , { 370, 370, 370, 370, 300}
1367 , { 370, 370, 370, 370, 300}
1368 , { 300, 300, 300, 300, 300}
1369 }
1370 }
1371 , {{{ 370, 370, 370, 370, 370}
1372 , { 370, 370, 370, 370, 370}
1373 , { 370, 370, 370, 370, 370}
1374 , { 370, 370, 370, 370, 370}
1375 , { 370, 370, 370, 370, 370}
1376 }
1377 , {{ 370, 370, 370, 370, 370}
1378 , { 370, 370, 370, 370, 370}
1379 , { 370, 370, 370, 370, 370}
1380 , { 260, 260, 260, 260, 260}
1381 , { 370, 370, 370, 370, 370}
1382 }
1383 , {{ 370, 370, 370, 370, 370}
1384 , { 370, 370, 370, 370, 370}
1385 , { 370, 370, 370, 370, 370}
1386 , { 370, 370, 370, 370, 370}
1387 , { 370, 370, 370, 370, 370}
1388 }
1389 , {{ 370, 260, 370, 260, 370}
1390 , { 260, 260, 260, 260, 260}
1391 , { 370, 260, 370, 260, 370}
1392 , { 260, 260, 260, 260, 260}
1393 , { 370, 260, 370, 260, 370}
1394 }
1395 , {{ 370, 370, 370, 370, 300}
1396 , { 370, 370, 370, 370, 300}
1397 , { 370, 370, 370, 370, 300}
1398 , { 370, 370, 370, 370, 300}
1399 , { 300, 300, 300, 300, 300}
1400 }
1401 }
1402 , {{{ 370, 370, 370, 370, 370}
1403 , { 370, 370, 370, 370, 370}
1404 , { 370, 370, 370, 370, 370}
1405 , { 370, 370, 370, 370, 370}
```

```
1406 , { 370, 370, 370, 370, 370}
1407 }
1408 , {{ 370, 370, 370, 370, 370}
1409 , { 370, 370, 370, 370, 370}
1410 , { 370, 370, 370, 370, 370}
1411 , { 260, 260, 260, 260, 260}
1412 , { 370, 370, 370, 370, 370}
1413 }
1414 , {{ 370, 370, 370, 370, 370}
1415 , { 370, 370, 370, 370, 370}
1416 , { 370, 370, 370, 370, 370}
1417 , { 370, 370, 370, 370, 370}
1418 , { 370, 370, 370, 370, 370}
1419 }
1420 , {{ 370, 260, 370, 260, 370}
1421 , { 370, 260, 370, 260, 370}
1422 , { 370, 260, 370, 260, 370}
1423 , { 260, 260, 260, 260, 260}
1424 , { 370, 260, 370, 260, 370}
1425 }
1426 , {{ 370, 370, 370, 370, 300}
1427 , { 370, 370, 370, 370, 300}
1428 , { 370, 370, 370, 370, 300}
1429 , { 370, 370, 370, 370, 300}
1430 , { 300, 300, 300, 300, 300}
1431 }
1432 }
1433 , {{ { 370, 370, 370, 370, 370}
1434 , { 370, 370, 370, 370, 370}
1435 , { 370, 370, 370, 370, 370}
1436 , { 370, 370, 370, 370, 370}
1437 , { 370, 370, 370, 370, 370}
1438 }
1439 , {{ 370, 370, 370, 370, 370}
1440 , { 370, 370, 370, 370, 370}
1441 , { 370, 370, 370, 370, 370}
1442 , { 260, 260, 260, 260, 260}
1443 , { 370, 370, 370, 370, 370}
1444 }
1445 , {{ 370, 370, 370, 370, 370}
1446 , { 370, 370, 370, 370, 370}
1447 , { 370, 370, 370, 370, 370}
1448 , { 370, 370, 370, 370, 370}
1449 , { 370, 370, 370, 370, 370}
1450 }
1451 , {{ { 370, 260, 370, 260, 370}
1452 , { 260, 260, 260, 260, 260}
1453 , { 370, 260, 370, 260, 370}
1454 , { 260, 260, 260, 260, 260}
1455 , { 370, 260, 370, 260, 370}
1456 }
1457 , {{ { 370, 370, 370, 370, 300}
1458 , { 370, 370, 370, 370, 300}
1459 , { 370, 370, 370, 370, 300}
1460 , { 370, 370, 370, 370, 300}
1461 , { 300, 300, 300, 300, 300}
1462 }
1463 }
1464 , {{ { { 370, 370, 370, 370, 370}
1465 , { 370, 370, 370, 370, 370}
1466 , { 370, 370, 370, 370, 370}
1467 , { 370, 370, 370, 370, 370}
1468 , { 370, 370, 370, 370, 370}
1469 }
1470 , {{ { 370, 370, 370, 370, 370}
1471 , { 370, 370, 370, 370, 370}
1472 , { 370, 370, 370, 370, 370}
1473 , { 260, 260, 260, 260, 260}
1474 , { 370, 370, 370, 370, 370}
1475 }
1476 , {{ { 370, 370, 370, 370, 370}
1477 , { 370, 370, 370, 370, 370}
1478 , { 370, 370, 370, 370, 370}
1479 , { 370, 370, 370, 370, 370}
1480 , { 370, 370, 370, 370, 370}
1481 }
1482 , {{ { { 370, 260, 370, 260, 370}
1483 , { 370, 260, 370, 260, 370}
1484 , { 370, 260, 370, 260, 370}
1485 , { 260, 260, 260, 260, 260}
1486 , { 370, 260, 370, 260, 370}
1487 }
1488 , {{ { { 370, 370, 370, 370, 300}
1489 , { 370, 370, 370, 370, 300}
1490 , { 370, 370, 370, 370, 300}
1491 , { 370, 370, 370, 370, 300}
1492 , { 300, 300, 300, 300, 300}
```

```
1493     }
1494   }
1495 }
1496 ,{{{{ INF, INF, INF, INF, INF}
1497 ,{ INF, INF, INF, INF, INF}
1498 ,{ INF, INF, INF, INF, INF}
1499 ,{ INF, INF, INF, INF, INF}
1500 ,{ INF, INF, INF, INF, INF}
1501 }
1502 ,{ { INF, INF, INF, INF, INF}
1503 ,{ INF, INF, INF, INF, INF}
1504 ,{ INF, INF, INF, INF, INF}
1505 ,{ INF, INF, INF, INF, INF}
1506 ,{ INF, INF, INF, INF, INF}
1507 }
1508 ,{ { INF, INF, INF, INF, INF}
1509 ,{ INF, INF, INF, INF, INF}
1510 ,{ INF, INF, INF, INF, INF}
1511 ,{ INF, INF, INF, INF, INF}
1512 ,{ INF, INF, INF, INF, INF}
1513 }
1514 ,{ { INF, INF, INF, INF, INF}
1515 ,{ INF, INF, INF, INF, INF}
1516 ,{ INF, INF, INF, INF, INF}
1517 ,{ INF, INF, INF, INF, INF}
1518 ,{ INF, INF, INF, INF, INF}
1519 }
1520 ,{ { INF, INF, INF, INF, INF}
1521 ,{ INF, INF, INF, INF, INF}
1522 ,{ INF, INF, INF, INF, INF}
1523 ,{ INF, INF, INF, INF, INF}
1524 ,{ INF, INF, INF, INF, INF}
1525 }
1526 }
1527 ,{{{ 300, 300, 300, 300, 300}
1528 ,{ 300, 300, 300, 300, 300}
1529 ,{ 300, 300, 300, 300, 300}
1530 ,{ 300, 300, 300, 300, 300}
1531 ,{ 300, 300, 300, 300, 300}
1532 }
1533 ,{ { 300, 300, 300, 190, 300}
1534 ,{ 300, 300, 300, 190, 300}
1535 ,{ 300, 300, 300, 190, 300}
1536 ,{ 190, 190, 190, 190, 190}
1537 ,{ 300, 300, 300, 190, 300}
1538 }
1539 ,{ { 300, 300, 300, 300, 300}
1540 ,{ 300, 300, 300, 300, 300}
1541 ,{ 300, 300, 300, 300, 300}
1542 ,{ 300, 300, 300, 300, 300}
1543 ,{ 300, 300, 300, 300, 300}
1544 }
1545 ,{ { 300, 190, 300, 190, 300}
1546 ,{ 190, 190, 190, 190, 190}
1547 ,{ 300, 190, 300, 190, 300}
1548 ,{ 190, 190, 190, 190, 190}
1549 ,{ 300, 190, 300, 190, 300}
1550 }
1551 ,{ { 300, 300, 300, 300, 220}
1552 ,{ 300, 300, 300, 300, 220}
1553 ,{ 300, 300, 300, 300, 220}
1554 ,{ 300, 300, 300, 300, 220}
1555 ,{ 220, 220, 220, 220, 220}
1556 }
1557 }
1558 ,{{{ 300, 300, 300, 300, 300}
1559 ,{ 300, 300, 300, 300, 300}
1560 ,{ 300, 300, 300, 300, 300}
1561 ,{ 300, 300, 300, 300, 300}
1562 ,{ 300, 300, 300, 300, 300}
1563 }
1564 ,{ { 300, 300, 300, 190, 300}
1565 ,{ 300, 300, 300, 190, 300}
1566 ,{ 300, 300, 300, 190, 300}
1567 ,{ 190, 190, 190, 190, 190}
1568 ,{ 300, 300, 300, 190, 300}
1569 }
1570 ,{ { 300, 300, 300, 300, 300}
1571 ,{ 300, 300, 300, 300, 300}
1572 ,{ 300, 300, 300, 300, 300}
1573 ,{ 300, 300, 300, 300, 300}
1574 ,{ 300, 300, 300, 300, 300}
1575 }
1576 ,{ { 300, 190, 300, 190, 300}
1577 ,{ 300, 190, 300, 190, 300}
1578 ,{ 300, 190, 300, 190, 300}
1579 ,{ 190, 190, 190, 190, 190}
```

```
1580 , { 300, 190, 300, 190, 300}
1581 }
1582 , {{ 300, 300, 300, 300, 220}
1583 , { 300, 300, 300, 300, 220}
1584 , { 300, 300, 300, 300, 220}
1585 , { 300, 300, 300, 300, 220}
1586 , { 220, 220, 220, 220, 220}
1587 }
1588 }
1589 , {{{ 370, 370, 370, 370, 370}
1590 , { 370, 370, 370, 370, 370}
1591 , { 370, 370, 370, 370, 370}
1592 , { 370, 370, 370, 370, 370}
1593 , { 370, 370, 370, 370, 370}
1594 }
1595 , {{ 370, 370, 370, 260, 370}
1596 , { 370, 370, 370, 260, 370}
1597 , { 370, 370, 370, 260, 370}
1598 , { 260, 260, 260, 260, 260}
1599 , { 370, 370, 370, 260, 370}
1600 }
1601 , {{ 370, 370, 370, 370, 370}
1602 , { 370, 370, 370, 370, 370}
1603 , { 370, 370, 370, 370, 370}
1604 , { 370, 370, 370, 370, 370}
1605 , { 370, 370, 370, 370, 370}
1606 }
1607 , {{ 370, 260, 370, 260, 370}
1608 , { 370, 260, 370, 260, 370}
1609 , { 370, 260, 370, 260, 370}
1610 , { 260, 260, 260, 260, 260}
1611 , { 370, 260, 370, 260, 370}
1612 }
1613 , {{ 370, 370, 370, 370, 300}
1614 , { 370, 370, 370, 370, 300}
1615 , { 370, 370, 370, 370, 300}
1616 , { 370, 370, 370, 370, 300}
1617 , { 300, 300, 300, 300, 300}
1618 }
1619 }
1620 , {{{ 370, 370, 370, 370, 370}
1621 , { 370, 370, 370, 370, 370}
1622 , { 370, 370, 370, 370, 370}
1623 , { 370, 370, 370, 370, 370}
1624 , { 370, 370, 370, 370, 370}
1625 }
1626 , {{ 370, 370, 370, 260, 370}
1627 , { 370, 370, 370, 260, 370}
1628 , { 370, 370, 370, 260, 370}
1629 , { 260, 260, 260, 260, 260}
1630 , { 370, 370, 370, 260, 370}
1631 }
1632 , {{ 370, 370, 370, 370, 370}
1633 , { 370, 370, 370, 370, 370}
1634 , { 370, 370, 370, 370, 370}
1635 , { 370, 370, 370, 370, 370}
1636 , { 370, 370, 370, 370, 370}
1637 }
1638 , {{ 370, 260, 370, 260, 370}
1639 , { 260, 260, 260, 260, 260}
1640 , { 370, 260, 370, 260, 370}
1641 , { 260, 260, 260, 260, 260}
1642 , { 370, 260, 370, 260, 370}
1643 }
1644 , {{ 370, 370, 370, 370, 300}
1645 , { 370, 370, 370, 370, 300}
1646 , { 370, 370, 370, 370, 300}
1647 , { 370, 370, 370, 370, 300}
1648 , { 300, 300, 300, 300, 300}
1649 }
1650 }
1651 , {{{ 370, 370, 370, 370, 370}
1652 , { 370, 370, 370, 370, 370}
1653 , { 370, 370, 370, 370, 370}
1654 , { 370, 370, 370, 370, 370}
1655 , { 370, 370, 370, 370, 370}
1656 }
1657 , {{ 370, 370, 370, 260, 370}
1658 , { 370, 370, 370, 260, 370}
1659 , { 370, 370, 370, 260, 370}
1660 , { 260, 260, 260, 260, 260}
1661 , { 370, 370, 370, 260, 370}
1662 }
1663 , {{ 370, 370, 370, 370, 370}
1664 , { 370, 370, 370, 370, 370}
1665 , { 370, 370, 370, 370, 370}
1666 , { 370, 370, 370, 370, 370}
```

```
1667 , { 370, 370, 370, 370, 370}
1668 }
1669 , {{ 370, 260, 370, 260, 370}
1670 , { 370, 260, 370, 260, 370}
1671 , { 370, 260, 370, 260, 370}
1672 , { 260, 260, 260, 260, 260}
1673 , { 370, 260, 370, 260, 370}
1674 }
1675 , {{ 370, 370, 370, 370, 300}
1676 , { 370, 370, 370, 370, 300}
1677 , { 370, 370, 370, 370, 300}
1678 , { 370, 370, 370, 370, 300}
1679 , { 300, 300, 300, 300, 300}
1680 }
1681 }
1682 , {{{ 370, 370, 370, 370, 370}
1683 , { 370, 370, 370, 370, 370}
1684 , { 370, 370, 370, 370, 370}
1685 , { 370, 370, 370, 370, 370}
1686 , { 370, 370, 370, 370, 370}
1687 }
1688 , {{ 370, 370, 370, 260, 370}
1689 , { 370, 370, 370, 260, 370}
1690 , { 370, 370, 370, 260, 370}
1691 , { 260, 260, 260, 260, 260}
1692 , { 370, 370, 370, 260, 370}
1693 }
1694 , {{ 370, 370, 370, 370, 370}
1695 , { 370, 370, 370, 370, 370}
1696 , { 370, 370, 370, 370, 370}
1697 , { 370, 370, 370, 370, 370}
1698 , { 370, 370, 370, 370, 370}
1699 }
1700 , {{ 370, 260, 370, 260, 370}
1701 , { 260, 260, 260, 260, 260}
1702 , { 370, 260, 370, 260, 370}
1703 , { 260, 260, 260, 260, 260}
1704 , { 370, 260, 370, 260, 370}
1705 }
1706 , {{ 370, 370, 370, 370, 300}
1707 , { 370, 370, 370, 370, 300}
1708 , { 370, 370, 370, 370, 300}
1709 , { 370, 370, 370, 370, 300}
1710 , { 300, 300, 300, 300, 300}
1711 }
1712 }
1713 , {{{ 370, 370, 370, 370, 370}
1714 , { 370, 370, 370, 370, 370}
1715 , { 370, 370, 370, 370, 370}
1716 , { 370, 370, 370, 370, 370}
1717 , { 370, 370, 370, 370, 370}
1718 }
1719 , {{ 370, 370, 370, 260, 370}
1720 , { 370, 370, 370, 260, 370}
1721 , { 370, 370, 370, 260, 370}
1722 , { 260, 260, 260, 260, 260}
1723 , { 370, 370, 370, 260, 370}
1724 }
1725 , {{ 370, 370, 370, 370, 370}
1726 , { 370, 370, 370, 370, 370}
1727 , { 370, 370, 370, 370, 370}
1728 , { 370, 370, 370, 370, 370}
1729 , { 370, 370, 370, 370, 370}
1730 }
1731 , {{ 370, 260, 370, 260, 370}
1732 , { 370, 260, 370, 260, 370}
1733 , { 370, 260, 370, 260, 370}
1734 , { 260, 260, 260, 260, 260}
1735 , { 370, 260, 370, 260, 370}
1736 }
1737 , {{ 370, 370, 370, 370, 300}
1738 , { 370, 370, 370, 370, 300}
1739 , { 370, 370, 370, 370, 300}
1740 , { 370, 370, 370, 370, 300}
1741 , { 300, 300, 300, 300, 300}
1742 }
1743 }
1744 }
1745 , {{{{ INF, INF, INF, INF, INF}
1746 , { INF, INF, INF, INF, INF}
1747 , { INF, INF, INF, INF, INF}
1748 , { INF, INF, INF, INF, INF}
1749 , { INF, INF, INF, INF, INF}
1750 }
1751 , {{ INF, INF, INF, INF, INF}
1752 , { INF, INF, INF, INF, INF}
1753 , { INF, INF, INF, INF, INF}}
```

```
1754 , { INF, INF, INF, INF, INF }
1755 , { INF, INF, INF, INF, INF }
1756 }
1757 , { { INF, INF, INF, INF, INF }
1758 , { INF, INF, INF, INF, INF }
1759 , { INF, INF, INF, INF, INF }
1760 , { INF, INF, INF, INF, INF }
1761 , { INF, INF, INF, INF, INF }
1762 }
1763 , { { INF, INF, INF, INF, INF }
1764 , { INF, INF, INF, INF, INF }
1765 , { INF, INF, INF, INF, INF }
1766 , { INF, INF, INF, INF, INF }
1767 , { INF, INF, INF, INF, INF }
1768 }
1769 , { { INF, INF, INF, INF, INF }
1770 , { INF, INF, INF, INF, INF }
1771 , { INF, INF, INF, INF, INF }
1772 , { INF, INF, INF, INF, INF }
1773 , { INF, INF, INF, INF, INF }
1774 }
1775 }
1776 , { { { 300, 300, 300, 300, 300 }
1777 , { 300, 300, 300, 300, 300 }
1778 , { 300, 300, 300, 300, 300 }
1779 , { 300, 300, 300, 300, 300 }
1780 , { 300, 300, 300, 300, 300 }
1781 }
1782 , { { 300, 300, 300, 300, 300 }
1783 , { 300, 300, 300, 300, 300 }
1784 , { 300, 300, 300, 300, 300 }
1785 , { 190, 190, 190, 190, 190 }
1786 , { 300, 300, 300, 300, 300 }
1787 }
1788 , { { 300, 300, 300, 300, 300 }
1789 , { 300, 300, 300, 300, 300 }
1790 , { 300, 300, 300, 300, 300 }
1791 , { 300, 300, 300, 300, 300 }
1792 , { 300, 300, 300, 300, 300 }
1793 }
1794 , { { 300, 190, 300, 190, 300 }
1795 , { 190, 190, 190, 190, 190 }
1796 , { 300, 190, 300, 190, 300 }
1797 , { 190, 190, 190, 190, 190 }
1798 , { 300, 190, 300, 190, 300 }
1799 }
1800 , { { 300, 300, 300, 300, 220 }
1801 , { 300, 300, 300, 300, 220 }
1802 , { 300, 300, 300, 300, 220 }
1803 , { 300, 300, 300, 300, 220 }
1804 , { 220, 220, 220, 220, 220 }
1805 }
1806 }
1807 , { { { 300, 300, 300, 300, 300 }
1808 , { 300, 300, 300, 300, 300 }
1809 , { 300, 300, 300, 300, 300 }
1810 , { 300, 300, 300, 300, 300 }
1811 , { 300, 300, 300, 300, 300 }
1812 }
1813 , { { 300, 300, 300, 300, 300 }
1814 , { 300, 300, 300, 300, 300 }
1815 , { 300, 300, 300, 300, 300 }
1816 , { 190, 190, 190, 190, 190 }
1817 , { 300, 300, 300, 300, 300 }
1818 }
1819 , { { 300, 300, 300, 300, 300 }
1820 , { 300, 300, 300, 300, 300 }
1821 , { 300, 300, 300, 300, 300 }
1822 , { 300, 300, 300, 300, 300 }
1823 , { 300, 300, 300, 300, 300 }
1824 }
1825 , { { 300, 190, 300, 190, 300 }
1826 , { 300, 190, 300, 190, 300 }
1827 , { 300, 190, 300, 190, 300 }
1828 , { 190, 190, 190, 190, 190 }
1829 , { 300, 190, 300, 190, 300 }
1830 }
1831 , { { 300, 300, 300, 300, 220 }
1832 , { 300, 300, 300, 300, 220 }
1833 , { 300, 300, 300, 300, 220 }
1834 , { 300, 300, 300, 300, 220 }
1835 , { 220, 220, 220, 220, 220 }
1836 }
1837 }
1838 , { { { 370, 370, 370, 370, 370 }
1839 , { 370, 370, 370, 370, 370 }
1840 , { 370, 370, 370, 370, 370 }
```

```
1841 , { 370, 370, 370, 370, 370}
1842 , { 370, 370, 370, 370, 370}
1843 }
1844 , {{ 370, 370, 370, 370, 370}
1845 , { 370, 370, 370, 370, 370}
1846 , { 370, 370, 370, 370, 370}
1847 , { 260, 260, 260, 260, 260}
1848 , { 370, 370, 370, 370, 370}
1849 }
1850 , {{ 370, 370, 370, 370, 370}
1851 , { 370, 370, 370, 370, 370}
1852 , { 370, 370, 370, 370, 370}
1853 , { 370, 370, 370, 370, 370}
1854 , { 370, 370, 370, 370, 370}
1855 }
1856 , {{ 370, 260, 370, 260, 370}
1857 , { 370, 260, 370, 260, 370}
1858 , { 370, 260, 370, 260, 370}
1859 , { 260, 260, 260, 260, 260}
1860 , { 370, 260, 370, 260, 370}
1861 }
1862 , {{ 370, 370, 370, 370, 300}
1863 , { 370, 370, 370, 370, 300}
1864 , { 370, 370, 370, 370, 300}
1865 , { 370, 370, 370, 370, 300}
1866 , { 300, 300, 300, 300, 300}
1867 }
1868 }
1869 , {{ { 370, 370, 370, 370, 370}
1870 , { 370, 370, 370, 370, 370}
1871 , { 370, 370, 370, 370, 370}
1872 , { 370, 370, 370, 370, 370}
1873 , { 370, 370, 370, 370, 370}
1874 }
1875 , {{ { 370, 370, 370, 370, 370}
1876 , { 370, 370, 370, 370, 370}
1877 , { 370, 370, 370, 370, 370}
1878 , { 260, 260, 260, 260, 260}
1879 , { 370, 370, 370, 370, 370}
1880 }
1881 , {{ { 370, 370, 370, 370, 370}
1882 , { 370, 370, 370, 370, 370}
1883 , { 370, 370, 370, 370, 370}
1884 , { 370, 370, 370, 370, 370}
1885 , { 370, 370, 370, 370, 370}
1886 }
1887 , {{ { 370, 260, 370, 260, 370}
1888 , { 260, 260, 260, 260, 260}
1889 , { 370, 260, 370, 260, 370}
1890 , { 260, 260, 260, 260, 260}
1891 , { 370, 260, 370, 260, 370}
1892 }
1893 , {{ { 370, 370, 370, 370, 300}
1894 , { 370, 370, 370, 370, 300}
1895 , { 370, 370, 370, 370, 300}
1896 , { 370, 370, 370, 370, 300}
1897 , { 300, 300, 300, 300, 300}
1898 }
1899 }
1900 , {{ {{ 370, 370, 370, 370, 370}
1901 , { 370, 370, 370, 370, 370}
1902 , { 370, 370, 370, 370, 370}
1903 , { 370, 370, 370, 370, 370}
1904 , { 370, 370, 370, 370, 370}
1905 }
1906 , {{ { 370, 370, 370, 370, 370}
1907 , { 370, 370, 370, 370, 370}
1908 , { 370, 370, 370, 370, 370}
1909 , { 260, 260, 260, 260, 260}
1910 , { 370, 370, 370, 370, 370}
1911 }
1912 , {{ { 370, 370, 370, 370, 370}
1913 , { 370, 370, 370, 370, 370}
1914 , { 370, 370, 370, 370, 370}
1915 , { 370, 370, 370, 370, 370}
1916 , { 370, 370, 370, 370, 370}
1917 }
1918 , {{ { 370, 260, 370, 260, 370}
1919 , { 370, 260, 370, 260, 370}
1920 , { 370, 260, 370, 260, 370}
1921 , { 260, 260, 260, 260, 260}
1922 , { 370, 260, 370, 260, 370}
1923 }
1924 , {{ { 370, 370, 370, 370, 300}
1925 , { 370, 370, 370, 370, 300}
1926 , { 370, 370, 370, 370, 300}
1927 , { 370, 370, 370, 370, 300}
```

```

1928 , { 300, 300, 300, 300, 300}
1929 }
1930 }
1931 ,{{ 370, 370, 370, 370, 370}
1932 ,{ 370, 370, 370, 370, 370}
1933 ,{ 370, 370, 370, 370, 370}
1934 ,{ 370, 370, 370, 370, 370}
1935 ,{ 370, 370, 370, 370, 370}
1936 }
1937 ,{{ 370, 370, 370, 370, 370}
1938 ,{ 370, 370, 370, 370, 370}
1939 ,{ 370, 370, 370, 370, 370}
1940 ,{ 260, 260, 260, 260, 260}
1941 ,{ 370, 370, 370, 370, 370}
1942 }
1943 ,{{ 370, 370, 370, 370, 370}
1944 ,{ 370, 370, 370, 370, 370}
1945 ,{ 370, 370, 370, 370, 370}
1946 ,{ 370, 370, 370, 370, 370}
1947 ,{ 370, 370, 370, 370, 370}
1948 }
1949 ,{{ 370, 260, 370, 260, 370}
1950 ,{ 260, 260, 260, 260, 260}
1951 ,{ 370, 260, 370, 260, 370}
1952 ,{ 260, 260, 260, 260, 260}
1953 ,{ 370, 260, 370, 260, 370}
1954 }
1955 ,{{ 370, 370, 370, 370, 300}
1956 ,{ 370, 370, 370, 370, 300}
1957 ,{ 370, 370, 370, 370, 300}
1958 ,{ 370, 370, 370, 370, 300}
1959 ,{ 300, 300, 300, 300, 300}
1960 }
1961 }
1962 ,{{ 370, 370, 370, 370, 370}
1963 ,{ 370, 370, 370, 370, 370}
1964 ,{ 370, 370, 370, 370, 370}
1965 ,{ 370, 370, 370, 370, 370}
1966 ,{ 370, 370, 370, 370, 370}
1967 }
1968 ,{{ 370, 370, 370, 370, 370}
1969 ,{ 370, 370, 370, 370, 370}
1970 ,{ 370, 370, 370, 370, 370}
1971 ,{ 260, 260, 260, 260, 260}
1972 ,{ 370, 370, 370, 370, 370}
1973 }
1974 ,{{ 370, 370, 370, 370, 370}
1975 ,{ 370, 370, 370, 370, 370}
1976 ,{ 370, 370, 370, 370, 370}
1977 ,{ 370, 370, 370, 370, 370}
1978 ,{ 370, 370, 370, 370, 370}
1979 }
1980 ,{{ 370, 260, 370, 260, 370}
1981 ,{ 370, 260, 370, 260, 370}
1982 ,{ 370, 260, 370, 260, 370}
1983 ,{ 260, 260, 260, 260, 260}
1984 ,{ 370, 260, 370, 260, 370}
1985 }
1986 ,{{ 370, 370, 370, 370, 300}
1987 ,{ 370, 370, 370, 370, 300}
1988 ,{ 370, 370, 370, 370, 300}
1989 ,{ 370, 370, 370, 370, 300}
1990 ,{ 300, 300, 300, 300, 300}
1991 }
1992 }
1993 }};

```

## 18.167 intl21dH.h

```

1 PUBLIC int int21_dH[NBPAIRS+1][NBPAIRS+1][5][5][5] =
2 {{{{{ INF, INF, INF, INF, INF
3 ,{ INF, INF, INF, INF, INF
4 ,{ INF, INF, INF, INF, INF
5 ,{ INF, INF, INF, INF, INF
6 ,{ INF, INF, INF, INF, INF
7 }
8 ,{ { INF, INF, INF, INF, INF
9 ,{ { INF, INF, INF, INF, INF
10 ,{ { INF, INF, INF, INF, INF
11 ,{ { INF, INF, INF, INF, INF
12 ,{ { INF, INF, INF, INF, INF
13 }
14 ,{ { INF, INF, INF, INF, INF
15 ,{ { INF, INF, INF, INF, INF
16 ,{ { INF, INF, INF, INF, INF

```

```
17 , { INF, INF, INF, INF, INF}
18 , { INF, INF, INF, INF, INF}
19 }
20 , { { INF, INF, INF, INF, INF}
21 , { INF, INF, INF, INF, INF}
22 , { INF, INF, INF, INF, INF}
23 , { INF, INF, INF, INF, INF}
24 , { INF, INF, INF, INF, INF}
25 }
26 , { { INF, INF, INF, INF, INF}
27 , { INF, INF, INF, INF, INF}
28 , { INF, INF, INF, INF, INF}
29 , { INF, INF, INF, INF, INF}
30 , { INF, INF, INF, INF, INF}
31 }
32 }
33 , { { { INF, INF, INF, INF, INF}
34 , { INF, INF, INF, INF, INF}
35 , { INF, INF, INF, INF, INF}
36 , { INF, INF, INF, INF, INF}
37 , { INF, INF, INF, INF, INF}
38 }
39 , { { INF, INF, INF, INF, INF}
40 , { INF, INF, INF, INF, INF}
41 , { INF, INF, INF, INF, INF}
42 , { INF, INF, INF, INF, INF}
43 , { INF, INF, INF, INF, INF}
44 }
45 , { { INF, INF, INF, INF, INF}
46 , { INF, INF, INF, INF, INF}
47 , { INF, INF, INF, INF, INF}
48 , { INF, INF, INF, INF, INF}
49 , { INF, INF, INF, INF, INF}
50 }
51 , { { INF, INF, INF, INF, INF}
52 , { INF, INF, INF, INF, INF}
53 , { INF, INF, INF, INF, INF}
54 , { INF, INF, INF, INF, INF}
55 , { INF, INF, INF, INF, INF}
56 }
57 , { { INF, INF, INF, INF, INF}
58 , { INF, INF, INF, INF, INF}
59 , { INF, INF, INF, INF, INF}
60 , { INF, INF, INF, INF, INF}
61 , { INF, INF, INF, INF, INF}
62 }
63 }
64 , { { { INF, INF, INF, INF, INF}
65 , { INF, INF, INF, INF, INF}
66 , { INF, INF, INF, INF, INF}
67 , { INF, INF, INF, INF, INF}
68 , { INF, INF, INF, INF, INF}
69 }
70 , { { INF, INF, INF, INF, INF}
71 , { INF, INF, INF, INF, INF}
72 , { INF, INF, INF, INF, INF}
73 , { INF, INF, INF, INF, INF}
74 , { INF, INF, INF, INF, INF}
75 }
76 , { { INF, INF, INF, INF, INF}
77 , { INF, INF, INF, INF, INF}
78 , { INF, INF, INF, INF, INF}
79 , { INF, INF, INF, INF, INF}
80 , { INF, INF, INF, INF, INF}
81 }
82 , { { INF, INF, INF, INF, INF}
83 , { INF, INF, INF, INF, INF}
84 , { INF, INF, INF, INF, INF}
85 , { INF, INF, INF, INF, INF}
86 , { INF, INF, INF, INF, INF}
87 }
88 , { { INF, INF, INF, INF, INF}
89 , { INF, INF, INF, INF, INF}
90 , { INF, INF, INF, INF, INF}
91 , { INF, INF, INF, INF, INF}
92 , { INF, INF, INF, INF, INF}
93 }
94 }
95 , { { { INF, INF, INF, INF, INF}
96 , { INF, INF, INF, INF, INF}
97 , { INF, INF, INF, INF, INF}
98 , { INF, INF, INF, INF, INF}
99 , { INF, INF, INF, INF, INF}
100 }
101 , { { { INF, INF, INF, INF, INF}
102 , { INF, INF, INF, INF, INF}
103 , { INF, INF, INF, INF, INF}
```

```
104 , { INF, INF, INF, INF, INF}
105 , { INF, INF, INF, INF, INF}
106 }
107 , {{ INF, INF, INF, INF, INF}
108 , { INF, INF, INF, INF, INF}
109 , { INF, INF, INF, INF, INF}
110 , { INF, INF, INF, INF, INF}
111 , { INF, INF, INF, INF, INF}
112 }
113 , {{ INF, INF, INF, INF, INF}
114 , { INF, INF, INF, INF, INF}
115 , { INF, INF, INF, INF, INF}
116 , { INF, INF, INF, INF, INF}
117 , { INF, INF, INF, INF, INF}
118 }
119 , {{ INF, INF, INF, INF, INF}
120 , { INF, INF, INF, INF, INF}
121 , { INF, INF, INF, INF, INF}
122 , { INF, INF, INF, INF, INF}
123 , { INF, INF, INF, INF, INF}
124 }
125 }
126 , {{{ INF, INF, INF, INF, INF}
127 , { INF, INF, INF, INF, INF}
128 , { INF, INF, INF, INF, INF}
129 , { INF, INF, INF, INF, INF}
130 , { INF, INF, INF, INF, INF}
131 }
132 , {{ INF, INF, INF, INF, INF}
133 , { INF, INF, INF, INF, INF}
134 , { INF, INF, INF, INF, INF}
135 , { INF, INF, INF, INF, INF}
136 , { INF, INF, INF, INF, INF}
137 }
138 , {{ INF, INF, INF, INF, INF}
139 , { INF, INF, INF, INF, INF}
140 , { INF, INF, INF, INF, INF}
141 , { INF, INF, INF, INF, INF}
142 , { INF, INF, INF, INF, INF}
143 }
144 , {{ INF, INF, INF, INF, INF}
145 , { INF, INF, INF, INF, INF}
146 , { INF, INF, INF, INF, INF}
147 , { INF, INF, INF, INF, INF}
148 , { INF, INF, INF, INF, INF}
149 }
150 , {{ INF, INF, INF, INF, INF}
151 , { INF, INF, INF, INF, INF}
152 , { INF, INF, INF, INF, INF}
153 , { INF, INF, INF, INF, INF}
154 , { INF, INF, INF, INF, INF}
155 }
156 }
157 , {{{ INF, INF, INF, INF, INF}
158 , { INF, INF, INF, INF, INF}
159 , { INF, INF, INF, INF, INF}
160 , { INF, INF, INF, INF, INF}
161 , { INF, INF, INF, INF, INF}
162 }
163 , {{ INF, INF, INF, INF, INF}
164 , { INF, INF, INF, INF, INF}
165 , { INF, INF, INF, INF, INF}
166 , { INF, INF, INF, INF, INF}
167 , { INF, INF, INF, INF, INF}
168 }
169 , {{ INF, INF, INF, INF, INF}
170 , { INF, INF, INF, INF, INF}
171 , { INF, INF, INF, INF, INF}
172 , { INF, INF, INF, INF, INF}
173 , { INF, INF, INF, INF, INF}
174 }
175 , {{ INF, INF, INF, INF, INF}
176 , { INF, INF, INF, INF, INF}
177 , { INF, INF, INF, INF, INF}
178 , { INF, INF, INF, INF, INF}
179 , { INF, INF, INF, INF, INF}
180 }
181 , {{ INF, INF, INF, INF, INF}
182 , { INF, INF, INF, INF, INF}
183 , { INF, INF, INF, INF, INF}
184 , { INF, INF, INF, INF, INF}
185 , { INF, INF, INF, INF, INF}
186 }
187 }
188 , {{{ INF, INF, INF, INF, INF}
189 , { INF, INF, INF, INF, INF}
190 , { INF, INF, INF, INF, INF}
```

```
191 , { INF, INF, INF, INF, INF }
192 , { INF, INF, INF, INF, INF }
193 }
194 , { { INF, INF, INF, INF, INF }
195 , { INF, INF, INF, INF, INF }
196 , { INF, INF, INF, INF, INF }
197 , { INF, INF, INF, INF, INF }
198 , { INF, INF, INF, INF, INF }
199 }
200 , { { INF, INF, INF, INF, INF }
201 , { INF, INF, INF, INF, INF }
202 , { INF, INF, INF, INF, INF }
203 , { INF, INF, INF, INF, INF }
204 , { INF, INF, INF, INF, INF }
205 }
206 , { { INF, INF, INF, INF, INF }
207 , { INF, INF, INF, INF, INF }
208 , { INF, INF, INF, INF, INF }
209 , { INF, INF, INF, INF, INF }
210 , { INF, INF, INF, INF, INF }
211 }
212 , { { INF, INF, INF, INF, INF }
213 , { INF, INF, INF, INF, INF }
214 , { INF, INF, INF, INF, INF }
215 , { INF, INF, INF, INF, INF }
216 , { INF, INF, INF, INF, INF }
217 }
218 }
219 , { { { INF, INF, INF, INF, INF }
220 , { INF, INF, INF, INF, INF }
221 , { INF, INF, INF, INF, INF }
222 , { INF, INF, INF, INF, INF }
223 , { INF, INF, INF, INF, INF }
224 }
225 , { { { INF, INF, INF, INF, INF }
226 , { INF, INF, INF, INF, INF }
227 , { INF, INF, INF, INF, INF }
228 , { INF, INF, INF, INF, INF }
229 , { INF, INF, INF, INF, INF }
230 }
231 , { { { INF, INF, INF, INF, INF }
232 , { INF, INF, INF, INF, INF }
233 , { INF, INF, INF, INF, INF }
234 , { INF, INF, INF, INF, INF }
235 , { INF, INF, INF, INF, INF }
236 }
237 , { { { INF, INF, INF, INF, INF }
238 , { INF, INF, INF, INF, INF }
239 , { INF, INF, INF, INF, INF }
240 , { INF, INF, INF, INF, INF }
241 , { INF, INF, INF, INF, INF }
242 }
243 , { { { INF, INF, INF, INF, INF }
244 , { INF, INF, INF, INF, INF }
245 , { INF, INF, INF, INF, INF }
246 , { INF, INF, INF, INF, INF }
247 , { INF, INF, INF, INF, INF }
248 }
249 }
250 }
251 , { { { { INF, INF, INF, INF, INF }
252 , { INF, INF, INF, INF, INF }
253 , { INF, INF, INF, INF, INF }
254 , { INF, INF, INF, INF, INF }
255 , { INF, INF, INF, INF, INF }
256 }
257 , { { { { INF, INF, INF, INF, INF }
258 , { INF, INF, INF, INF, INF }
259 , { INF, INF, INF, INF, INF }
260 , { INF, INF, INF, INF, INF }
261 , { INF, INF, INF, INF, INF }
262 }
263 , { { { { INF, INF, INF, INF, INF }
264 , { INF, INF, INF, INF, INF }
265 , { INF, INF, INF, INF, INF }
266 , { INF, INF, INF, INF, INF }
267 , { INF, INF, INF, INF, INF }
268 }
269 , { { { { INF, INF, INF, INF, INF }
270 , { INF, INF, INF, INF, INF }
271 , { INF, INF, INF, INF, INF }
272 , { INF, INF, INF, INF, INF }
273 , { INF, INF, INF, INF, INF }
274 }
275 , { { { { INF, INF, INF, INF, INF }
276 , { INF, INF, INF, INF, INF }
277 , { INF, INF, INF, INF, INF }
```

```
278 , {   INF,   INF,   INF,   INF,   INF}
279 , {   INF,   INF,   INF,   INF,   INF}
280 }
281 }
282 , {{ { 350,   350,   350,   350,   350}
283 , { 350,   350,   350,   350,   350}
284 , { 350,   350,   350,   350,   350}
285 , { 350,   350,   350,   350,   350}
286 , { 350,   350,   350,   350,   350}
287 }
288 , {{ { 350,   350,   350, -230,   350}
289 , { 350,   350,   350, -230,   350}
290 , { 350,   350,   350, -230,   350}
291 , { -230, -230, -230, -230, -230}
292 , { 350,   350,   350, -230,   350}
293 }
294 , {{ { 350,   350,   350,   350,   350}
295 , { 350,   350,   350,   350,   350}
296 , { 350,   350,   350,   350,   350}
297 , { 350,   350,   350,   350,   350}
298 , { 350,   350,   350,   350,   350}
299 }
300 , {{ { 350, -230,   350, -230,   350}
301 , { -230, -230, -230, -230, -230}
302 , { 350, -230,   350, -230,   350}
303 , { -230, -230, -230, -230, -230}
304 , { 350, -230,   350, -230,   350}
305 }
306 , {{ { 350,   350,   350,   350, -670}
307 , { 350,   350,   350,   350, -670}
308 , { 350,   350,   350,   350, -670}
309 , { 350,   350,   350,   350, -670}
310 , { -670, -670, -670, -670, -670}
311 }
312 }
313 , {{ { 780,   640,   780,   350,   350}
314 , { 350,   350,   350,   350,   350}
315 , { 780,   350,   780,   350,   350}
316 , { 350,   350,   350,   350,   350}
317 , { 640,   640,   350,   350,   350}
318 }
319 , {{ { 350,   350,   350,   250,   350}
320 , { 350,   260,   350,   250,   350}
321 , { 350,   350, -250, -230,   350}
322 , { -230, -230, -230, -230, -230}
323 , { 350,   350,   350, -230,   350}
324 }
325 , {{ { 780,   640,   780,   350,   350}
326 , { 350,   160,   350,   350,   350}
327 , { 780,   350,   780,   350,   350}
328 , { 350,   350,   350,   350,   350}
329 , { 640,   640,   350,   350,   350}
330 }
331 , {{ { 350, -160,   350, -230,   350}
332 , { 350, -160,   350, -410,   350}
333 , { 350, -230,   350, -230,   350}
334 , { -230, -310, -230, -230, -230}
335 , { 350, -230,   350, -230,   350}
336 }
337 , {{ { 580,   350,   580,   350, -580}
338 , { 350,   350,   350,   350, -670}
339 , { 580,   350,   580,   350, -580}
340 , { 350,   350,   350,   350, -670}
341 , { -670, -670, -690, -670, -700}
342 }
343 }
344 , {{ { 850,   850,   850,   850,   850}
345 , { 850,   850,   850,   850,   850}
346 , { 850,   850,   850,   850,   850}
347 , { 850,   850,   850,   850,   850}
348 , { 850,   850,   850,   850,   850}
349 }
350 , {{ { 850,   850,   850,   280,   850}
351 , { 850,   850,   850,   280,   850}
352 , { 850,   850,   850,   280,   850}
353 , { 280,   280,   280,   280,   280}
354 , { 850,   850,   850,   280,   850}
355 }
356 , {{ { 850,   850,   850,   850,   850}
357 , { 850,   850,   850,   850,   850}
358 , { 850,   850,   850,   850,   850}
359 , { 850,   850,   850,   850,   850}
360 , { 850,   850,   850,   850,   850}
361 }
362 , {{ { 850,   280,   850,   280,   850}
363 , { 850,   280,   850,   280,   850}
364 , { 850,   280,   850,   280,   850}
```

```
365 , { 280, 280, 280, 280, 280}
366 , { 850, 280, 850, 280, 850}
367 }
368 , {{ 850, 850, 850, 850, -160}
369 , { 850, 850, 850, 850, -160}
370 , { 850, 850, 850, 850, -160}
371 , { 850, 850, 850, 850, -160}
372 , { -160, -160, -160, -160, -160}
373 }
374 }
375 , {{ { 850, 850, 850, 850, 850}
376 , { 850, 850, 850, 850, 850}
377 , { 850, 850, 850, 850, 850}
378 , { 850, 850, 850, 850, 850}
379 , { 850, 850, 850, 850, 850}
380 }
381 , {{ { 850, 850, 850, 280, 850}
382 , { 850, 850, 850, 280, 850}
383 , { 850, 850, 850, 280, 850}
384 , { 280, 280, 280, 280, 280}
385 , { 850, 850, 850, 280, 850}
386 }
387 , {{ { 850, 850, 850, 850, 850}
388 , { 850, 850, 850, 850, 850}
389 , { 850, 850, 850, 850, 850}
390 , { 850, 850, 850, 850, 850}
391 , { 850, 850, 850, 850, 850}
392 }
393 , {{ { 850, 280, 850, 280, 850}
394 , { 280, 280, 280, 280, 280}
395 , { 850, 280, 850, 280, 850}
396 , { 280, 280, 280, 280, 280}
397 , { 850, 280, 850, 280, 850}
398 }
399 , {{ { 850, 850, 850, 850, -160}
400 , { 850, 850, 850, 850, -160}
401 , { 850, 850, 850, 850, -160}
402 , { 850, 850, 850, 850, -160}
403 , { -160, -160, -160, -160, -160}
404 }
405 }
406 , {{ { 850, 850, 850, 850, 850}
407 , { 850, 850, 850, 850, 850}
408 , { 850, 850, 850, 850, 850}
409 , { 850, 850, 850, 850, 850}
410 , { 850, 850, 850, 850, 850}
411 }
412 , {{ { 850, 850, 850, 280, 850}
413 , { 850, 850, 850, 280, 850}
414 , { 850, 850, 850, 280, 850}
415 , { 280, 280, 280, 280, 280}
416 , { 850, 850, 850, 280, 850}
417 }
418 , {{ { 850, 850, 850, 850, 850}
419 , { 850, 850, 850, 850, 850}
420 , { 850, 850, 850, 850, 850}
421 , { 850, 850, 850, 850, 850}
422 , { 850, 850, 850, 850, 850}
423 }
424 , {{ { 850, 280, 850, 280, 850}
425 , { 850, 280, 850, 280, 850}
426 , { 850, 280, 850, 280, 850}
427 , { 280, 280, 280, 280, 280}
428 , { 850, 280, 850, 280, 850}
429 }
430 , {{ { 850, 850, 850, 850, -160}
431 , { 850, 850, 850, 850, -160}
432 , { 850, 850, 850, 850, -160}
433 , { 850, 850, 850, 850, -160}
434 , { -160, -160, -160, -160, -160}
435 }
436 }
437 , {{ { 850, 850, 850, 850, 850}
438 , { 850, 850, 850, 850, 850}
439 , { 850, 850, 850, 850, 850}
440 , { 850, 850, 850, 850, 850}
441 , { 850, 850, 850, 850, 850}
442 }
443 , {{ { 850, 850, 850, 280, 850}
444 , { 850, 850, 850, 280, 850}
445 , { 850, 850, 850, 280, 850}
446 , { 280, 280, 280, 280, 280}
447 , { 850, 850, 850, 280, 850}
448 }
449 , {{ { 850, 850, 850, 850, 850}
450 , { 850, 850, 850, 850, 850}
451 , { 850, 850, 850, 850, 850}
```

```
452 ,{ 850, 850, 850, 850, 850}
453 ,{ 850, 850, 850, 850, 850}
454 }
455 ,{{ 850, 280, 850, 280, 850}
456 ,{ 280, 280, 280, 280, 280}
457 ,{ 850, 280, 850, 280, 850}
458 ,{ 280, 280, 280, 280, 280}
459 ,{ 850, 280, 850, 280, 850}
460 }
461 ,{{ 850, 850, 850, 850, -160}
462 ,{ 850, 850, 850, 850, -160}
463 ,{ 850, 850, 850, 850, -160}
464 ,{ 850, 850, 850, 850, -160}
465 ,{ -160, -160, -160, -160, -160}
466 }
467 }
468 ,{{{ 850, 850, 850, 850, 850}
469 ,{ 850, 850, 850, 850, 850}
470 ,{ 850, 850, 850, 850, 850}
471 ,{ 850, 850, 850, 850, 850}
472 ,{ 850, 850, 850, 850, 850}
473 }
474 ,{{ 850, 850, 850, 280, 850}
475 ,{ 850, 850, 850, 280, 850}
476 ,{ 850, 850, 850, 280, 850}
477 ,{ 280, 280, 280, 280, 280}
478 ,{ 850, 850, 850, 280, 850}
479 }
480 ,{{ 850, 850, 850, 850, 850}
481 ,{ 850, 850, 850, 850, 850}
482 ,{ 850, 850, 850, 850, 850}
483 ,{ 850, 850, 850, 850, 850}
484 ,{ 850, 850, 850, 850, 850}
485 }
486 ,{{ 850, 280, 850, 280, 850}
487 ,{ 850, 280, 850, 280, 850}
488 ,{ 850, 280, 850, 280, 850}
489 ,{ 280, 280, 280, 280, 280}
490 ,{ 850, 280, 850, 280, 850}
491 }
492 ,{{ 850, 850, 850, 850, -160}
493 ,{ 850, 850, 850, 850, -160}
494 ,{ 850, 850, 850, 850, -160}
495 ,{ 850, 850, 850, 850, -160}
496 ,{ -160, -160, -160, -160, -160}
497 }
498 }
499 }
500 ,{{{ {{ INF, INF, INF, INF, INF}
501 ,{ INF, INF, INF, INF, INF}
502 ,{ INF, INF, INF, INF, INF}
503 ,{ INF, INF, INF, INF, INF}
504 ,{ INF, INF, INF, INF, INF}
505 }
506 ,{{ INF, INF, INF, INF, INF}
507 ,{ INF, INF, INF, INF, INF}
508 ,{ INF, INF, INF, INF, INF}
509 ,{ INF, INF, INF, INF, INF}
510 ,{ INF, INF, INF, INF, INF}
511 }
512 ,{{ INF, INF, INF, INF, INF}
513 ,{ INF, INF, INF, INF, INF}
514 ,{ INF, INF, INF, INF, INF}
515 ,{ INF, INF, INF, INF, INF}
516 ,{ INF, INF, INF, INF, INF}
517 }
518 ,{{ INF, INF, INF, INF, INF}
519 ,{ INF, INF, INF, INF, INF}
520 ,{ INF, INF, INF, INF, INF}
521 ,{ INF, INF, INF, INF, INF}
522 ,{ INF, INF, INF, INF, INF}
523 }
524 ,{{ INF, INF, INF, INF, INF}
525 ,{ INF, INF, INF, INF, INF}
526 ,{ INF, INF, INF, INF, INF}
527 ,{ INF, INF, INF, INF, INF}
528 ,{ INF, INF, INF, INF, INF}
529 }
530 }
531 ,{{{ 690, 690, 350, 350, 350}
532 ,{ 690, 690, 350, 350, 350}
533 ,{ 350, 350, 350, 350, 350}
534 ,{ 350, 350, 350, 350, 350}
535 ,{ 350, 350, 350, 350, 350}
536 }
537 ,{{{ 690, 690, 350, 350, 350}
538 ,{ 690, 690, 350, 240, 350}
```

```
539 , { 350, 350, 350, 350, 350}
540 , { -230, -500, -230, -230, -230}
541 , { 350, 350, 350, 350, 350}
542 }
543 , { { 350, 350, 350, 350, 350}
544 , { 350, 350, 350, 350, 350}
545 , { 350, 350, 350, 350, 350}
546 , { 350, 350, 350, 350, 350}
547 , { 350, 350, 130, 350, 350}
548 }
549 , { { 350, -230, 350, -230, 350}
550 , { -230, -230, -230, -230, -230}
551 , { 350, -230, 350, -230, 350}
552 , { -230, -230, -230, -230, -230}
553 , { 350, -230, 350, -230, 350}
554 }
555 , { { 350, 350, 350, 350, -670}
556 , { 350, 350, 350, 350, -670}
557 , { 350, 350, 350, 350, -670}
558 , { 350, 350, 350, 350, -670}
559 , { -670, -670, -670, -670, -670}
560 }
561 }
562 , { { { 350, 350, 350, 350, 350}
563 , { 350, 350, 350, 350, 350}
564 , { 350, 350, 350, 350, 350}
565 , { 350, 350, 350, 350, 350}
566 , { 350, 350, 350, 350, 350}
567 }
568 , { { 350, 350, 350, 350, 350}
569 , { 350, 350, 350, 350, 350}
570 , { 350, 350, 350, 350, 350}
571 , { -230, -230, -230, -230, -230}
572 , { 350, 350, 350, 350, 350}
573 }
574 , { { 350, 350, 350, 350, 350}
575 , { 350, 350, 350, 350, 350}
576 , { 350, 350, 350, 350, 350}
577 , { 350, 350, 350, 350, 350}
578 , { 350, 350, 350, 350, 350}
579 }
580 , { { 350, -230, 350, -230, 350}
581 , { 350, -230, 350, -230, 350}
582 , { 350, -230, 350, -230, 350}
583 , { -230, -230, -230, -230, -230}
584 , { 350, -230, 350, -230, 350}
585 }
586 , { { { 350, 350, 350, 350, -670}
587 , { 350, 350, 350, 350, -670}
588 , { 350, 350, 350, 350, -670}
589 , { 350, 350, 350, 350, -670}
590 , { -670, -670, -670, -670, -670}
591 }
592 }
593 , { { { 850, 850, 850, 850, 850}
594 , { 850, 850, 850, 850, 850}
595 , { 850, 850, 850, 850, 850}
596 , { 850, 850, 850, 850, 850}
597 , { 850, 850, 850, 850, 850}
598 }
599 , { { 850, 850, 850, 850, 850}
600 , { 850, 850, 850, 850, 850}
601 , { 850, 850, 850, 850, 850}
602 , { 280, 280, 280, 280, 280}
603 , { 850, 850, 850, 850, 850}
604 }
605 , { { 850, 850, 850, 850, 850}
606 , { 850, 850, 850, 850, 850}
607 , { 850, 850, 850, 850, 850}
608 , { 850, 850, 850, 850, 850}
609 , { 850, 850, 850, 850, 850}
610 }
611 , { { 850, 280, 850, 280, 850}
612 , { 850, 280, 850, 280, 850}
613 , { 850, 280, 850, 280, 850}
614 , { 280, 280, 280, 280, 280}
615 , { 850, 280, 850, 280, 850}
616 }
617 , { { 850, 850, 850, 850, -160}
618 , { 850, 850, 850, 850, -160}
619 , { 850, 850, 850, 850, -160}
620 , { 850, 850, 850, 850, -160}
621 , { -160, -160, -160, -160, -160}
622 }
623 }
624 , { { { 850, 850, 850, 850, 850}
625 , { 850, 850, 850, 850, 850}
```

```
626 ,{ 850, 850, 850, 850, 850}
627 ,{ 850, 850, 850, 850, 850}
628 ,{ 850, 850, 850, 850, 850}
629 }
630 ,{{ 850, 850, 850, 850, 850}
631 ,{ 850, 690, 850, 240, 850}
632 ,{ 850, 850, 850, 850, 850}
633 ,{ 280, -500, 280, 280, 280}
634 ,{ 850, 850, 850, 850, 850}
635 }
636 ,{{ 850, 850, 850, 850, 850}
637 ,{ 850, 850, 850, 850, 850}
638 ,{ 850, 850, 850, 850, 850}
639 ,{ 850, 850, 850, 850, 850}
640 ,{ 850, 850, 130, 850, 850}
641 }
642 ,{{ 850, 280, 850, 280, 850}
643 ,{ 280, 280, 280, 280, 280}
644 ,{ 850, 280, 850, 280, 850}
645 ,{ 280, 280, 280, 280, 280}
646 ,{ 850, 280, 850, 280, 850}
647 }
648 ,{{ 850, 850, 850, 850, -160}
649 ,{ 850, 850, 850, 850, -160}
650 ,{ 850, 850, 850, 850, -160}
651 ,{ 850, 850, 850, 850, -160}
652 ,{ -160, -160, -160, -160, -160}
653 }
654 }
655 ,{{{{ 850, 850, 850, 850, 850}
656 ,{ 850, 850, 850, 850, 850}
657 ,{ 850, 850, 850, 850, 850}
658 ,{ 850, 850, 850, 850, 850}
659 ,{ 850, 850, 850, 850, 850}
660 }
661 ,{{{ 850, 850, 850, 850, 850}
662 ,{ 850, 850, 850, 850, 850}
663 ,{ 850, 850, 850, 850, 850}
664 ,{ 280, 280, 280, 280, 280}
665 ,{ 850, 850, 850, 850, 850}
666 }
667 ,{{{{ 850, 850, 850, 850, 850}
668 ,{ 850, 850, 850, 850, 850}
669 ,{ 850, 850, 850, 850, 850}
670 ,{ 850, 850, 850, 850, 850}
671 ,{ 850, 850, 850, 850, 850}
672 }
673 ,{{{{ 850, 280, 850, 280, 850}
674 ,{ 850, 280, 850, 280, 850}
675 ,{ 850, 280, 850, 280, 850}
676 ,{ 280, 280, 280, 280, 280}
677 ,{ 850, 280, 850, 280, 850}
678 }
679 ,{{{{ 850, 850, 850, 850, -160}
680 ,{ 850, 850, 850, 850, -160}
681 ,{ 850, 850, 850, 850, -160}
682 ,{ 850, 850, 850, 850, -160}
683 ,{ -160, -160, -160, -160, -160}
684 }
685 }
686 ,{{{{ 850, 850, 850, 850, 850}
687 ,{ 850, 850, 850, 850, 850}
688 ,{ 850, 850, 850, 850, 850}
689 ,{ 850, 850, 850, 850, 850}
690 ,{ 850, 850, 850, 850, 850}
691 }
692 ,{{{{ 850, 850, 850, 850, 850}
693 ,{ 850, 850, 850, 850, 850}
694 ,{ 850, 850, 850, 850, 850}
695 ,{ 280, 280, 280, 280, 280}
696 ,{ 850, 850, 850, 850, 850}
697 }
698 ,{{{{ 850, 850, 850, 850, 850}
699 ,{ 850, 850, 850, 850, 850}
700 ,{ 850, 850, 850, 850, 850}
701 ,{ 850, 850, 850, 850, 850}
702 ,{ 850, 850, 850, 850, 850}
703 }
704 ,{{{{ 850, 280, 850, 280, 850}
705 ,{ 280, 280, 280, 280, 280}
706 ,{ 850, 280, 850, 280, 850}
707 ,{ 280, 280, 280, 280, 280}
708 ,{ 850, 280, 850, 280, 850}
709 }
710 ,{{{{ 850, 850, 850, 850, -160}
711 ,{ 850, 850, 850, 850, -160}
712 ,{ 850, 850, 850, 850, -160}
```

```
713 , { 850, 850, 850, 850, -160}
714 , { -160, -160, -160, -160, -160}
715 }
716 }
717 , {{ { 850, 850, 850, 850, 850}
718 , { 850, 850, 850, 850, 850}
719 , { 850, 850, 850, 850, 850}
720 , { 850, 850, 850, 850, 850}
721 , { 850, 850, 850, 850, 850}
722 }
723 , {{ { 850, 850, 850, 850, 850}
724 , { 850, 850, 850, 850, 850}
725 , { 850, 850, 850, 850, 850}
726 , { 280, 280, 280, 280, 280}
727 , { 850, 850, 850, 850, 850}
728 }
729 , {{ { 850, 850, 850, 850, 850}
730 , { 850, 850, 850, 850, 850}
731 , { 850, 850, 850, 850, 850}
732 , { 850, 850, 850, 850, 850}
733 , { 850, 850, 850, 850, 850}
734 }
735 , {{ { 850, 280, 850, 280, 850}
736 , { 850, 280, 850, 280, 850}
737 , { 850, 280, 850, 280, 850}
738 , { 280, 280, 280, 280, 280}
739 , { 850, 280, 850, 280, 850}
740 }
741 , {{ { 850, 850, 850, 850, -160}
742 , { 850, 850, 850, 850, -160}
743 , { 850, 850, 850, 850, -160}
744 , { 850, 850, 850, 850, -160}
745 , { -160, -160, -160, -160, -160}
746 }
747 }
748 }
749 , {{{ { INF, INF, INF, INF, INF}
750 , { INF, INF, INF, INF, INF}
751 , { INF, INF, INF, INF, INF}
752 , { INF, INF, INF, INF, INF}
753 , { INF, INF, INF, INF, INF}
754 }
755 , {{ { INF, INF, INF, INF, INF}
756 , { INF, INF, INF, INF, INF}
757 , { INF, INF, INF, INF, INF}
758 , { INF, INF, INF, INF, INF}
759 , { INF, INF, INF, INF, INF}
760 }
761 , {{ { INF, INF, INF, INF, INF}
762 , { INF, INF, INF, INF, INF}
763 , { INF, INF, INF, INF, INF}
764 , { INF, INF, INF, INF, INF}
765 , { INF, INF, INF, INF, INF}
766 }
767 , {{ { INF, INF, INF, INF, INF}
768 , { INF, INF, INF, INF, INF}
769 , { INF, INF, INF, INF, INF}
770 , { INF, INF, INF, INF, INF}
771 , { INF, INF, INF, INF, INF}
772 }
773 , {{ { INF, INF, INF, INF, INF}
774 , { INF, INF, INF, INF, INF}
775 , { INF, INF, INF, INF, INF}
776 , { INF, INF, INF, INF, INF}
777 , { INF, INF, INF, INF, INF}
778 }
779 }
780 , {{ { 850, 850, 850, 850, 850}
781 , { 850, 850, 850, 850, 850}
782 , { 850, 850, 850, 850, 850}
783 , { 850, 850, 850, 850, 850}
784 , { 850, 850, 850, 850, 850}
785 }
786 , {{ { 850, 850, 850, 850, 850}
787 , { 850, 690, 850, 240, 850}
788 , { 850, 850, 850, 850, 850}
789 , { 280, -500, 280, 280, 280}
790 , { 850, 850, 850, 850, 850}
791 }
792 , {{ { 850, 850, 850, 850, 850}
793 , { 850, 850, 850, 850, 850}
794 , { 850, 850, 850, 850, 850}
795 , { 850, 850, 850, 850, 850}
796 , { 850, 850, 130, 850, 850}
797 }
798 , {{ { 850, 280, 850, 280, 850}
799 , { 280, 280, 280, 280, 280}
```

```
800 ,{ 850, 280, 850, 280, 850}
801 ,{ 280, 280, 280, 280, 280}
802 ,{ 850, 280, 850, 280, 850}
803 }
804 ,{{ 850, 850, 850, 850, -160}
805 ,{ 850, 850, 850, 850, -160}
806 ,{ 850, 850, 850, 850, -160}
807 ,{ 850, 850, 850, 850, -160}
808 ,{ -160, -160, -160, -160, -160}
809 }
810 }
811 ,{{{{ 850, 850, 850, 850, 850}
812 ,{ 850, 850, 850, 850, 850}
813 ,{ 850, 850, 850, 850, 850}
814 ,{ 850, 850, 850, 850, 850}
815 ,{ 850, 850, 850, 850, 850}
816 }
817 ,{{ 850, 850, 850, 850, 850}
818 ,{ 850, 850, 850, 850, 850}
819 ,{ 850, 850, 850, 850, 850}
820 ,{ 280, 280, 280, 280, 280}
821 ,{ 850, 850, 850, 850, 850}
822 }
823 ,{{ 850, 850, 850, 850, 850}
824 ,{ 850, 850, 850, 850, 850}
825 ,{ 850, 850, 850, 850, 850}
826 ,{ 850, 850, 850, 850, 850}
827 ,{ 850, 850, 850, 850, 850}
828 }
829 ,{{ 850, 280, 850, 280, 850}
830 ,{ 850, 280, 850, 280, 850}
831 ,{ 850, 280, 850, 280, 850}
832 ,{ 280, 280, 280, 280, 280}
833 ,{ 850, 280, 850, 280, 850}
834 }
835 ,{{ 850, 850, 850, 850, -160}
836 ,{ 850, 850, 850, 850, -160}
837 ,{ 850, 850, 850, 850, -160}
838 ,{ 850, 850, 850, 850, -160}
839 ,{ -160, -160, -160, -160, -160}
840 }
841 }
842 ,{{{{ 1350, 1350, 1350, 1350, 1350}
843 ,{ 1350, 1350, 1350, 1350, 1350}
844 ,{ 1350, 1350, 1350, 1350, 1350}
845 ,{ 1350, 1350, 1350, 1350, 1350}
846 ,{ 1350, 1350, 1350, 1350, 1350}
847 }
848 ,{{ 1350, 1350, 1350, 1350, 1350}
849 ,{ 1350, 1350, 1350, 1350, 1350}
850 ,{ 1350, 1350, 1350, 1350, 1350}
851 ,{ 780, 780, 780, 780, 780}
852 ,{ 1350, 1350, 1350, 1350, 1350}
853 }
854 ,{{ 1350, 1350, 1350, 1350, 1350}
855 ,{ 1350, 1350, 1350, 1350, 1350}
856 ,{ 1350, 1350, 1350, 1350, 1350}
857 ,{ 1350, 1350, 1350, 1350, 1350}
858 ,{ 1350, 1350, 1350, 1350, 1350}
859 }
860 ,{{ 1350, 780, 1350, 780, 1350}
861 ,{ 1350, 780, 1350, 780, 1350}
862 ,{ 1350, 780, 1350, 780, 1350}
863 ,{ 780, 780, 780, 780, 780}
864 ,{ 1350, 780, 1350, 780, 1350}
865 }
866 ,{{ 1350, 1350, 1350, 1350, 340}
867 ,{ 1350, 1350, 1350, 1350, 340}
868 ,{ 1350, 1350, 1350, 1350, 340}
869 ,{ 1350, 1350, 1350, 1350, 340}
870 ,{ 340, 340, 340, 340, 340}
871 }
872 }
873 ,{{{{ 1350, 1350, 1350, 1350, 1350}
874 ,{ 1350, 1350, 1350, 1350, 1350}
875 ,{ 1350, 1350, 1350, 1350, 1350}
876 ,{ 1350, 1350, 1350, 1350, 1350}
877 ,{ 1350, 1350, 1350, 1350, 1350}
878 }
879 ,{{ 1350, 1350, 1350, 1350, 1350}
880 ,{ 1350, 690, 1350, 240, 1350}
881 ,{ 1350, 1350, 1350, 1350, 1350}
882 ,{ 780, -500, 780, 780, 780}
883 ,{ 1350, 1350, 1350, 1350, 1350}
884 }
885 ,{{ 1350, 1350, 1350, 1350, 1350}
886 ,{ 1350, 1350, 1350, 1350, 1350}
```

```
887 ,{ 1350, 1350, 1350, 1350, 1350}
888 ,{ 1350, 1350, 1350, 1350, 1350}
889 ,{ 1350, 1350, 130, 1350, 1350}
890 }
891 ,{{ 1350, 780, 1350, 780, 1350}
892 ,{ 780, 780, 780, 780, 780}
893 ,{ 1350, 780, 1350, 780, 1350}
894 ,{ 780, 780, 780, 780, 780}
895 ,{ 1350, 780, 1350, 780, 1350}
896 }
897 ,{{ 1350, 1350, 1350, 1350, 340}
898 ,{ 1350, 1350, 1350, 1350, 340}
899 ,{ 1350, 1350, 1350, 1350, 340}
900 ,{ 1350, 1350, 1350, 1350, 340}
901 ,{ 340, 340, 340, 340, 340}
902 }
903 }
904 ,{{{ 1350, 1350, 1350, 1350, 1350}
905 ,{ 1350, 1350, 1350, 1350, 1350}
906 ,{ 1350, 1350, 1350, 1350, 1350}
907 ,{ 1350, 1350, 1350, 1350, 1350}
908 ,{ 1350, 1350, 1350, 1350, 1350}
909 }
910 ,{{ 1350, 1350, 1350, 1350, 1350}
911 ,{ 1350, 1350, 1350, 1350, 1350}
912 ,{ 1350, 1350, 1350, 1350, 1350}
913 ,{ 780, 780, 780, 780, 780}
914 ,{ 1350, 1350, 1350, 1350, 1350}
915 }
916 ,{{ 1350, 1350, 1350, 1350, 1350}
917 ,{ 1350, 1350, 1350, 1350, 1350}
918 ,{ 1350, 1350, 1350, 1350, 1350}
919 ,{ 1350, 1350, 1350, 1350, 1350}
920 ,{ 1350, 1350, 1350, 1350, 1350}
921 }
922 ,{{ 1350, 780, 1350, 780, 1350}
923 ,{ 1350, 780, 1350, 780, 1350}
924 ,{ 1350, 780, 1350, 780, 1350}
925 ,{ 780, 780, 780, 780, 780}
926 ,{ 1350, 780, 1350, 780, 1350}
927 }
928 ,{{ 1350, 1350, 1350, 1350, 340}
929 ,{ 1350, 1350, 1350, 1350, 340}
930 ,{ 1350, 1350, 1350, 1350, 340}
931 ,{ 1350, 1350, 1350, 1350, 340}
932 ,{ 340, 340, 340, 340, 340}
933 }
934 }
935 ,{{{ 1350, 1350, 1350, 1350, 1350}
936 ,{ 1350, 1350, 1350, 1350, 1350}
937 ,{ 1350, 1350, 1350, 1350, 1350}
938 ,{ 1350, 1350, 1350, 1350, 1350}
939 ,{ 1350, 1350, 1350, 1350, 1350}
940 }
941 ,{{ 1350, 1350, 1350, 1350, 1350}
942 ,{ 1350, 1350, 1350, 1350, 1350}
943 ,{ 1350, 1350, 1350, 1350, 1350}
944 ,{ 780, 780, 780, 780, 780}
945 ,{ 1350, 1350, 1350, 1350, 1350}
946 }
947 ,{{ 1350, 1350, 1350, 1350, 1350}
948 ,{ 1350, 1350, 1350, 1350, 1350}
949 ,{ 1350, 1350, 1350, 1350, 1350}
950 ,{ 1350, 1350, 1350, 1350, 1350}
951 ,{ 1350, 1350, 1350, 1350, 1350}
952 }
953 ,{{ 1350, 780, 1350, 780, 1350}
954 ,{ 780, 780, 780, 780, 780}
955 ,{ 1350, 780, 1350, 780, 1350}
956 ,{ 780, 780, 780, 780, 780}
957 ,{ 1350, 780, 1350, 780, 1350}
958 }
959 ,{{ 1350, 1350, 1350, 1350, 340}
960 ,{ 1350, 1350, 1350, 1350, 340}
961 ,{ 1350, 1350, 1350, 1350, 340}
962 ,{ 1350, 1350, 1350, 1350, 340}
963 ,{ 340, 340, 340, 340, 340}
964 }
965 }
966 ,{{{ 1350, 1350, 1350, 1350, 1350}
967 ,{ 1350, 1350, 1350, 1350, 1350}
968 ,{ 1350, 1350, 1350, 1350, 1350}
969 ,{ 1350, 1350, 1350, 1350, 1350}
970 ,{ 1350, 1350, 1350, 1350, 1350}
971 }
972 ,{{ 1350, 1350, 1350, 1350, 1350}
973 ,{ 1350, 1350, 1350, 1350, 1350}
```

```
974 ,{ 1350, 1350, 1350, 1350, 1350}
975 ,{ 780, 780, 780, 780, 780}
976 ,{ 1350, 1350, 1350, 1350, 1350}
977 }
978 ,{{ 1350, 1350, 1350, 1350, 1350}
979 ,{ 1350, 1350, 1350, 1350, 1350}
980 ,{ 1350, 1350, 1350, 1350, 1350}
981 ,{ 1350, 1350, 1350, 1350, 1350}
982 ,{ 1350, 1350, 1350, 1350, 1350}
983 }
984 ,{{ 1350, 780, 1350, 780, 1350}
985 ,{ 1350, 780, 1350, 780, 1350}
986 ,{ 1350, 780, 1350, 780, 1350}
987 ,{ 780, 780, 780, 780, 780}
988 ,{ 1350, 780, 1350, 780, 1350}
989 }
990 ,{{ 1350, 1350, 1350, 1350, 340}
991 ,{ 1350, 1350, 1350, 1350, 340}
992 ,{ 1350, 1350, 1350, 1350, 340}
993 ,{ 1350, 1350, 1350, 1350, 340}
994 ,{ 340, 340, 340, 340, 340}
995 }
996 }
997 }
998 ,{{{ INF, INF, INF, INF, INF}
999 ,{ INF, INF, INF, INF, INF}
1000 ,{ INF, INF, INF, INF, INF}
1001 ,{ INF, INF, INF, INF, INF}
1002 ,{ INF, INF, INF, INF, INF}
1003 }
1004 ,{{ INF, INF, INF, INF, INF}
1005 ,{ INF, INF, INF, INF, INF}
1006 ,{ INF, INF, INF, INF, INF}
1007 ,{ INF, INF, INF, INF, INF}
1008 ,{ INF, INF, INF, INF, INF}
1009 }
1010 ,{{ INF, INF, INF, INF, INF}
1011 ,{ INF, INF, INF, INF, INF}
1012 ,{ INF, INF, INF, INF, INF}
1013 ,{ INF, INF, INF, INF, INF}
1014 ,{ INF, INF, INF, INF, INF}
1015 }
1016 ,{{ INF, INF, INF, INF, INF}
1017 ,{ INF, INF, INF, INF, INF}
1018 ,{ INF, INF, INF, INF, INF}
1019 ,{ INF, INF, INF, INF, INF}
1020 ,{ INF, INF, INF, INF, INF}
1021 }
1022 ,{{ INF, INF, INF, INF, INF}
1023 ,{ INF, INF, INF, INF, INF}
1024 ,{ INF, INF, INF, INF, INF}
1025 ,{ INF, INF, INF, INF, INF}
1026 ,{ INF, INF, INF, INF, INF}
1027 }
1028 }
1029 ,{{{ 850, 850, 850, 850, 850}
1030 ,{ 850, 850, 850, 850, 850}
1031 ,{ 850, 850, 850, 850, 850}
1032 ,{ 850, 850, 850, 850, 850}
1033 ,{ 850, 850, 850, 850, 850}
1034 }
1035 ,{{ 850, 850, 850, 280, 850}
1036 ,{ 850, 850, 850, 280, 850}
1037 ,{ 850, 850, 850, 280, 850}
1038 ,{ 280, 280, 280, 280, 280}
1039 ,{ 850, 850, 850, 280, 850}
1040 }
1041 ,{{ 850, 850, 850, 850, 850}
1042 ,{ 850, 850, 850, 850, 850}
1043 ,{ 850, 850, 850, 850, 850}
1044 ,{ 850, 850, 850, 850, 850}
1045 ,{ 850, 850, 850, 850, 850}
1046 }
1047 ,{{ 850, 280, 850, 280, 850}
1048 ,{ 280, 280, 280, 280, 280}
1049 ,{ 850, 280, 850, 280, 850}
1050 ,{ 280, 280, 280, 280, 280}
1051 ,{ 850, 280, 850, 280, 850}
1052 }
1053 ,{{ 850, 850, 850, 850, -160}
1054 ,{ 850, 850, 850, 850, -160}
1055 ,{ 850, 850, 850, 850, -160}
1056 ,{ 850, 850, 850, 850, -160}
1057 ,{ -160, -160, -160, -160, -160}
1058 }
1059 }
1060 ,{{{ 850, 850, 850, 850, 850}
```

```
1061 , { 850, 850, 850, 850, 850}
1062 , { 850, 850, 850, 850, 850}
1063 , { 850, 850, 850, 850, 850}
1064 , { 850, 850, 850, 850, 850}
1065 }
1066 , {{ 850, 850, 850, 280, 850}
1067 , { 850, 850, 850, 280, 850}
1068 , { 850, 850, 850, 280, 850}
1069 , { 280, 280, 280, 280, 280}
1070 , { 850, 850, 850, 280, 850}
1071 }
1072 , {{ 850, 850, 850, 850, 850}
1073 , { 850, 850, 850, 850, 850}
1074 , { 850, 850, 850, 850, 850}
1075 , { 850, 850, 850, 850, 850}
1076 , { 850, 850, 850, 850, 850}
1077 }
1078 , {{ 850, 280, 850, 280, 850}
1079 , { 850, 280, 850, 280, 850}
1080 , { 850, 280, 850, 280, 850}
1081 , { 280, 280, 280, 280, 280}
1082 , { 850, 280, 850, 280, 850}
1083 }
1084 , {{ 850, 850, 850, 850, -160}
1085 , { 850, 850, 850, 850, -160}
1086 , { 850, 850, 850, 850, -160}
1087 , { 850, 850, 850, 850, -160}
1088 , { -160, -160, -160, -160, -160}
1089 }
1090 }
1091 , {{{ 1350, 1350, 1350, 1350, 1350}
1092 , { 1350, 1350, 1350, 1350, 1350}
1093 , { 1350, 1350, 1350, 1350, 1350}
1094 , { 1350, 1350, 1350, 1350, 1350}
1095 , { 1350, 1350, 1350, 1350, 1350}
1096 }
1097 , {{ 1350, 1350, 1350, 780, 1350}
1098 , { 1350, 1350, 1350, 780, 1350}
1099 , { 1350, 1350, 1350, 780, 1350}
1100 , { 780, 780, 780, 780, 780}
1101 , { 1350, 1350, 1350, 780, 1350}
1102 }
1103 , {{ 1350, 1350, 1350, 1350, 1350}
1104 , { 1350, 1350, 1350, 1350, 1350}
1105 , { 1350, 1350, 1350, 1350, 1350}
1106 , { 1350, 1350, 1350, 1350, 1350}
1107 , { 1350, 1350, 1350, 1350, 1350}
1108 }
1109 , {{ 1350, 780, 1350, 780, 1350}
1110 , { 1350, 780, 1350, 780, 1350}
1111 , { 1350, 780, 1350, 780, 1350}
1112 , { 780, 780, 780, 780, 780}
1113 , { 1350, 780, 1350, 780, 1350}
1114 }
1115 , {{ 1350, 1350, 1350, 1350, 340}
1116 , { 1350, 1350, 1350, 1350, 340}
1117 , { 1350, 1350, 1350, 1350, 340}
1118 , { 1350, 1350, 1350, 1350, 340}
1119 , { 340, 340, 340, 340, 340}
1120 }
1121 }
1122 , {{{ 1350, 1350, 1350, 1350, 1350}
1123 , { 1350, 1350, 1350, 1350, 1350}
1124 , { 1350, 1350, 1350, 1350, 1350}
1125 , { 1350, 1350, 1350, 1350, 1350}
1126 , { 1350, 1350, 1350, 1350, 1350}
1127 }
1128 , {{ 1350, 1350, 1350, 780, 1350}
1129 , { 1350, 1350, 1350, 780, 1350}
1130 , { 1350, 1350, 1350, 780, 1350}
1131 , { 780, 780, 780, 780, 780}
1132 , { 1350, 1350, 1350, 780, 1350}
1133 }
1134 , {{ 1350, 1350, 1350, 1350, 1350}
1135 , { 1350, 1350, 1350, 1350, 1350}
1136 , { 1350, 1350, 1350, 1350, 1350}
1137 , { 1350, 1350, 1350, 1350, 1350}
1138 , { 1350, 1350, 1350, 1350, 1350}
1139 }
1140 , {{ 1350, 780, 1350, 780, 1350}
1141 , { 780, 780, 780, 780, 780}
1142 , { 1350, 780, 1350, 780, 1350}
1143 , { 780, 780, 780, 780, 780}
1144 , { 1350, 780, 1350, 780, 1350}
1145 }
1146 , {{ 1350, 1350, 1350, 1350, 340}
1147 , { 1350, 1350, 1350, 1350, 340}
```

```
1148 ,{ 1350, 1350, 1350, 1350, 340}
1149 ,{ 1350, 1350, 1350, 1350, 340}
1150 ,{ 340, 340, 340, 340, 340}
1151 }
1152 }
1153 ,{{ 1350, 1350, 1350, 1350, 1350}
1154 ,{ 1350, 1350, 1350, 1350, 1350}
1155 ,{ 1350, 1350, 1350, 1350, 1350}
1156 ,{ 1350, 1350, 1350, 1350, 1350}
1157 ,{ 1350, 1350, 1350, 1350, 1350}
1158 }
1159 ,{{ 1350, 1350, 1350, 780, 1350}
1160 ,{ 1350, 1350, 1350, 780, 1350}
1161 ,{ 1350, 1350, 1350, 780, 1350}
1162 ,{ 780, 780, 780, 780, 780}
1163 ,{ 1350, 1350, 1350, 780, 1350}
1164 }
1165 ,{{ 1350, 1350, 1350, 1350, 1350}
1166 ,{ 1350, 1350, 1350, 1350, 1350}
1167 ,{ 1350, 1350, 1350, 1350, 1350}
1168 ,{ 1350, 1350, 1350, 1350, 1350}
1169 ,{ 1350, 1350, 1350, 1350, 1350}
1170 }
1171 ,{{ 1350, 780, 1350, 780, 1350}
1172 ,{ 1350, 780, 1350, 780, 1350}
1173 ,{ 1350, 780, 1350, 780, 1350}
1174 ,{ 780, 780, 780, 780, 780}
1175 ,{ 1350, 780, 1350, 780, 1350}
1176 }
1177 ,{{ 1350, 1350, 1350, 1350, 340}
1178 ,{ 1350, 1350, 1350, 1350, 340}
1179 ,{ 1350, 1350, 1350, 1350, 340}
1180 ,{ 1350, 1350, 1350, 1350, 340}
1181 ,{ 340, 340, 340, 340, 340}
1182 }
1183 }
1184 ,{{ 1350, 1350, 1350, 1350, 1350}
1185 ,{ 1350, 1350, 1350, 1350, 1350}
1186 ,{ 1350, 1350, 1350, 1350, 1350}
1187 ,{ 1350, 1350, 1350, 1350, 1350}
1188 ,{ 1350, 1350, 1350, 1350, 1350}
1189 }
1190 ,{{ 1350, 1350, 1350, 780, 1350}
1191 ,{ 1350, 1350, 1350, 780, 1350}
1192 ,{ 1350, 1350, 1350, 780, 1350}
1193 ,{ 780, 780, 780, 780, 780}
1194 ,{ 1350, 1350, 1350, 780, 1350}
1195 }
1196 ,{{ 1350, 1350, 1350, 1350, 1350}
1197 ,{ 1350, 1350, 1350, 1350, 1350}
1198 ,{ 1350, 1350, 1350, 1350, 1350}
1199 ,{ 1350, 1350, 1350, 1350, 1350}
1200 ,{ 1350, 1350, 1350, 1350, 1350}
1201 }
1202 ,{{ 1350, 780, 1350, 780, 1350}
1203 ,{ 780, 780, 780, 780, 780}
1204 ,{ 1350, 780, 1350, 780, 1350}
1205 ,{ 780, 780, 780, 780, 780}
1206 ,{ 1350, 780, 1350, 780, 1350}
1207 }
1208 ,{{ 1350, 1350, 1350, 1350, 340}
1209 ,{ 1350, 1350, 1350, 1350, 340}
1210 ,{ 1350, 1350, 1350, 1350, 340}
1211 ,{ 1350, 1350, 1350, 1350, 340}
1212 ,{ 340, 340, 340, 340, 340}
1213 }
1214 }
1215 ,{{ 1350, 1350, 1350, 1350, 1350}
1216 ,{ 1350, 1350, 1350, 1350, 1350}
1217 ,{ 1350, 1350, 1350, 1350, 1350}
1218 ,{ 1350, 1350, 1350, 1350, 1350}
1219 ,{ 1350, 1350, 1350, 1350, 1350}
1220 }
1221 ,{{ 1350, 1350, 1350, 780, 1350}
1222 ,{ 1350, 1350, 1350, 780, 1350}
1223 ,{ 1350, 1350, 1350, 780, 1350}
1224 ,{ 780, 780, 780, 780, 780}
1225 ,{ 1350, 1350, 1350, 780, 1350}
1226 }
1227 ,{{ 1350, 1350, 1350, 1350, 1350}
1228 ,{ 1350, 1350, 1350, 1350, 1350}
1229 ,{ 1350, 1350, 1350, 1350, 1350}
1230 ,{ 1350, 1350, 1350, 1350, 1350}
1231 ,{ 1350, 1350, 1350, 1350, 1350}
1232 }
1233 ,{{ 1350, 780, 1350, 780, 1350}
1234 ,{ 1350, 780, 1350, 780, 1350}
```

```
1235 ,{ 1350,    780,   1350,    780,   1350}
1236 ,{    780,    780,    780,    780,    780}
1237 ,{ 1350,    780,   1350,    780,   1350}
1238 }
1239 ,{{ 1350,   1350,   1350,   1350,   340}
1240 ,{ 1350,   1350,   1350,   1350,   340}
1241 ,{ 1350,   1350,   1350,   1350,   340}
1242 ,{ 1350,   1350,   1350,   1350,   340}
1243 ,{    340,    340,    340,    340,    340}
1244 }
1245 }
1246 }
1247 ,{{{  INF,    INF,    INF,    INF,    INF}
1248 ,{    INF,    INF,    INF,    INF,    INF}
1249 ,{    INF,    INF,    INF,    INF,    INF}
1250 ,{    INF,    INF,    INF,    INF,    INF}
1251 ,{    INF,    INF,    INF,    INF,    INF}
1252 }
1253 ,{{  INF,    INF,    INF,    INF,    INF}
1254 ,{    INF,    INF,    INF,    INF,    INF}
1255 ,{    INF,    INF,    INF,    INF,    INF}
1256 ,{    INF,    INF,    INF,    INF,    INF}
1257 ,{    INF,    INF,    INF,    INF,    INF}
1258 }
1259 ,{{  INF,    INF,    INF,    INF,    INF}
1260 ,{    INF,    INF,    INF,    INF,    INF}
1261 ,{    INF,    INF,    INF,    INF,    INF}
1262 ,{    INF,    INF,    INF,    INF,    INF}
1263 ,{    INF,    INF,    INF,    INF,    INF}
1264 }
1265 ,{{  INF,    INF,    INF,    INF,    INF}
1266 ,{    INF,    INF,    INF,    INF,    INF}
1267 ,{    INF,    INF,    INF,    INF,    INF}
1268 ,{    INF,    INF,    INF,    INF,    INF}
1269 ,{    INF,    INF,    INF,    INF,    INF}
1270 }
1271 ,{{  INF,    INF,    INF,    INF,    INF}
1272 ,{    INF,    INF,    INF,    INF,    INF}
1273 ,{    INF,    INF,    INF,    INF,    INF}
1274 ,{    INF,    INF,    INF,    INF,    INF}
1275 ,{    INF,    INF,    INF,    INF,    INF}
1276 }
1277 }
1278 ,{{{  850,    850,    850,    850,    850}
1279 ,{    850,    850,    850,    850,    850}
1280 ,{    850,    850,    850,    850,    850}
1281 ,{    850,    850,    850,    850,    850}
1282 ,{    850,    850,    850,    850,    850}
1283 }
1284 ,{{  850,    850,    850,    850,    850}
1285 ,{    850,    850,    850,    850,    850}
1286 ,{    850,    850,    850,    850,    850}
1287 ,{    280,    280,    280,    280,    280}
1288 ,{    850,    850,    850,    850,    850}
1289 }
1290 ,{{  850,    850,    850,    850,    850}
1291 ,{    850,    850,    850,    850,    850}
1292 ,{    850,    850,    850,    850,    850}
1293 ,{    850,    850,    850,    850,    850}
1294 ,{    850,    850,    850,    850,    850}
1295 }
1296 ,{{  850,    280,    850,    280,    850}
1297 ,{    280,    280,    280,    280,    280}
1298 ,{    850,    280,    850,    280,    850}
1299 ,{    280,    280,    280,    280,    280}
1300 ,{    850,    280,    850,    280,    850}
1301 }
1302 ,{{  850,    850,    850,    850, -160}
1303 ,{    850,    850,    850,    850, -160}
1304 ,{    850,    850,    850,    850, -160}
1305 ,{    850,    850,    850,    850, -160}
1306 ,{   -160,   -160,   -160,   -160,   -160}
1307 }
1308 }
1309 ,{{{  850,    850,    850,    850,    850}
1310 ,{    850,    850,    850,    850,    850}
1311 ,{    850,    850,    850,    850,    850}
1312 ,{    850,    850,    850,    850,    850}
1313 ,{    850,    850,    850,    850,    850}
1314 }
1315 ,{{  850,    850,    850,    850,    850}
1316 ,{    850,    850,    850,    850,    850}
1317 ,{    850,    850,    850,    850,    850}
1318 ,{    280,    280,    280,    280,    280}
1319 ,{    850,    850,    850,    850,    850}
1320 }
1321 ,{{  850,    850,    850,    850,    850}
```

```
1322 ,{ 850, 850, 850, 850, 850}
1323 ,{ 850, 850, 850, 850, 850}
1324 ,{ 850, 850, 850, 850, 850}
1325 ,{ 850, 850, 850, 850, 850}
1326 }
1327 ,{{ 850, 280, 850, 280, 850}
1328 ,{ 850, 280, 850, 280, 850}
1329 ,{ 850, 280, 850, 280, 850}
1330 ,{ 280, 280, 280, 280, 280}
1331 ,{ 850, 280, 850, 280, 850}
1332 }
1333 ,{{ 850, 850, 850, 850, -160}
1334 ,{ 850, 850, 850, 850, -160}
1335 ,{ 850, 850, 850, 850, -160}
1336 ,{ 850, 850, 850, 850, -160}
1337 ,{ -160, -160, -160, -160, -160}
1338 }
1339 }
1340 ,{{{{ 1350, 1350, 1350, 1350, 1350}
1341 ,{ 1350, 1350, 1350, 1350, 1350}
1342 ,{ 1350, 1350, 1350, 1350, 1350}
1343 ,{ 1350, 1350, 1350, 1350, 1350}
1344 ,{ 1350, 1350, 1350, 1350, 1350}
1345 }
1346 ,{{ 1350, 1350, 1350, 1350, 1350}
1347 ,{ 1350, 1350, 1350, 1350, 1350}
1348 ,{ 1350, 1350, 1350, 1350, 1350}
1349 ,{ 780, 780, 780, 780, 780}
1350 ,{ 1350, 1350, 1350, 1350, 1350}
1351 }
1352 ,{{ 1350, 1350, 1350, 1350, 1350}
1353 ,{ 1350, 1350, 1350, 1350, 1350}
1354 ,{ 1350, 1350, 1350, 1350, 1350}
1355 ,{ 1350, 1350, 1350, 1350, 1350}
1356 ,{ 1350, 1350, 1350, 1350, 1350}
1357 }
1358 ,{{ 1350, 780, 1350, 780, 1350}
1359 ,{ 1350, 780, 1350, 780, 1350}
1360 ,{ 1350, 780, 1350, 780, 1350}
1361 ,{ 780, 780, 780, 780, 780}
1362 ,{ 1350, 780, 1350, 780, 1350}
1363 }
1364 ,{{ 1350, 1350, 1350, 1350, 340}
1365 ,{ 1350, 1350, 1350, 1350, 340}
1366 ,{ 1350, 1350, 1350, 1350, 340}
1367 ,{ 1350, 1350, 1350, 1350, 340}
1368 ,{ 340, 340, 340, 340, 340}
1369 }
1370 }
1371 ,{{{{ 1350, 1350, 1350, 1350, 1350}
1372 ,{ 1350, 1350, 1350, 1350, 1350}
1373 ,{ 1350, 1350, 1350, 1350, 1350}
1374 ,{ 1350, 1350, 1350, 1350, 1350}
1375 ,{ 1350, 1350, 1350, 1350, 1350}
1376 }
1377 ,{{ 1350, 1350, 1350, 1350, 1350}
1378 ,{ 1350, 1350, 1350, 1350, 1350}
1379 ,{ 1350, 1350, 1350, 1350, 1350}
1380 ,{ 780, 780, 780, 780, 780}
1381 ,{ 1350, 1350, 1350, 1350, 1350}
1382 }
1383 ,{{ 1350, 1350, 1350, 1350, 1350}
1384 ,{ 1350, 1350, 1350, 1350, 1350}
1385 ,{ 1350, 1350, 1350, 1350, 1350}
1386 ,{ 1350, 1350, 1350, 1350, 1350}
1387 ,{ 1350, 1350, 1350, 1350, 1350}
1388 }
1389 ,{{ 1350, 780, 1350, 780, 1350}
1390 ,{ 780, 780, 780, 780, 780}
1391 ,{ 1350, 780, 1350, 780, 1350}
1392 ,{ 780, 780, 780, 780, 780}
1393 ,{ 1350, 780, 1350, 780, 1350}
1394 }
1395 ,{{ 1350, 1350, 1350, 1350, 340}
1396 ,{ 1350, 1350, 1350, 1350, 340}
1397 ,{ 1350, 1350, 1350, 1350, 340}
1398 ,{ 1350, 1350, 1350, 1350, 340}
1399 ,{ 340, 340, 340, 340, 340}
1400 }
1401 }
1402 ,{{{{ 1350, 1350, 1350, 1350, 1350}
1403 ,{ 1350, 1350, 1350, 1350, 1350}
1404 ,{ 1350, 1350, 1350, 1350, 1350}
1405 ,{ 1350, 1350, 1350, 1350, 1350}
1406 ,{ 1350, 1350, 1350, 1350, 1350}
1407 }
1408 ,{{ 1350, 1350, 1350, 1350, 1350}
```

```
1409 ,{ 1350, 1350, 1350, 1350, 1350}
1410 ,{ 1350, 1350, 1350, 1350, 1350}
1411 ,{ 780, 780, 780, 780, 780}
1412 ,{ 1350, 1350, 1350, 1350, 1350}
1413 }
1414 ,{{ 1350, 1350, 1350, 1350, 1350}
1415 ,{ 1350, 1350, 1350, 1350, 1350}
1416 ,{ 1350, 1350, 1350, 1350, 1350}
1417 ,{ 1350, 1350, 1350, 1350, 1350}
1418 ,{ 1350, 1350, 1350, 1350, 1350}
1419 }
1420 ,{{ 1350, 780, 1350, 780, 1350}
1421 ,{ 1350, 780, 1350, 780, 1350}
1422 ,{ 1350, 780, 1350, 780, 1350}
1423 ,{ 780, 780, 780, 780, 780}
1424 ,{ 1350, 780, 1350, 780, 1350}
1425 }
1426 ,{{ 1350, 1350, 1350, 1350, 340}
1427 ,{ 1350, 1350, 1350, 1350, 340}
1428 ,{ 1350, 1350, 1350, 1350, 340}
1429 ,{ 1350, 1350, 1350, 1350, 340}
1430 ,{ 340, 340, 340, 340, 340}
1431 }
1432 }
1433 ,{{{{ 1350, 1350, 1350, 1350, 1350}
1434 ,{ 1350, 1350, 1350, 1350, 1350}
1435 ,{ 1350, 1350, 1350, 1350, 1350}
1436 ,{ 1350, 1350, 1350, 1350, 1350}
1437 ,{ 1350, 1350, 1350, 1350, 1350}
1438 }
1439 ,{{ 1350, 1350, 1350, 1350, 1350}
1440 ,{ 1350, 1350, 1350, 1350, 1350}
1441 ,{ 1350, 1350, 1350, 1350, 1350}
1442 ,{ 780, 780, 780, 780, 780}
1443 ,{ 1350, 1350, 1350, 1350, 1350}
1444 }
1445 ,{{ 1350, 1350, 1350, 1350, 1350}
1446 ,{ 1350, 1350, 1350, 1350, 1350}
1447 ,{ 1350, 1350, 1350, 1350, 1350}
1448 ,{ 1350, 1350, 1350, 1350, 1350}
1449 ,{ 1350, 1350, 1350, 1350, 1350}
1450 }
1451 ,{{ 1350, 780, 1350, 780, 1350}
1452 ,{ 780, 780, 780, 780, 780}
1453 ,{ 1350, 780, 1350, 780, 1350}
1454 ,{ 780, 780, 780, 780, 780}
1455 ,{ 1350, 780, 1350, 780, 1350}
1456 }
1457 ,{{ 1350, 1350, 1350, 1350, 340}
1458 ,{ 1350, 1350, 1350, 1350, 340}
1459 ,{ 1350, 1350, 1350, 1350, 340}
1460 ,{ 1350, 1350, 1350, 1350, 340}
1461 ,{ 340, 340, 340, 340, 340}
1462 }
1463 }
1464 ,{{{{ 1350, 1350, 1350, 1350, 1350}
1465 ,{ 1350, 1350, 1350, 1350, 1350}
1466 ,{ 1350, 1350, 1350, 1350, 1350}
1467 ,{ 1350, 1350, 1350, 1350, 1350}
1468 ,{ 1350, 1350, 1350, 1350, 1350}
1469 }
1470 ,{{ 1350, 1350, 1350, 1350, 1350}
1471 ,{ 1350, 1350, 1350, 1350, 1350}
1472 ,{ 1350, 1350, 1350, 1350, 1350}
1473 ,{ 780, 780, 780, 780, 780}
1474 ,{ 1350, 1350, 1350, 1350, 1350}
1475 }
1476 ,{{ 1350, 1350, 1350, 1350, 1350}
1477 ,{ 1350, 1350, 1350, 1350, 1350}
1478 ,{ 1350, 1350, 1350, 1350, 1350}
1479 ,{ 1350, 1350, 1350, 1350, 1350}
1480 ,{ 1350, 1350, 1350, 1350, 1350}
1481 }
1482 ,{{ 1350, 780, 1350, 780, 1350}
1483 ,{ 1350, 780, 1350, 780, 1350}
1484 ,{ 1350, 780, 1350, 780, 1350}
1485 ,{ 780, 780, 780, 780, 780}
1486 ,{ 1350, 780, 1350, 780, 1350}
1487 }
1488 ,{{ 1350, 1350, 1350, 1350, 340}
1489 ,{ 1350, 1350, 1350, 1350, 340}
1490 ,{ 1350, 1350, 1350, 1350, 340}
1491 ,{ 1350, 1350, 1350, 1350, 340}
1492 ,{ 340, 340, 340, 340, 340}
1493 }
1494 }
1495 }
```

```
1496 ,{{{{ INF, INF, INF, INF, INF
1497 ,{ INF, INF, INF, INF, INF}
1498 ,{ INF, INF, INF, INF, INF}
1499 ,{ INF, INF, INF, INF, INF}
1500 ,{ INF, INF, INF, INF, INF}
1501 }
1502 ,{{ INF, INF, INF, INF, INF}
1503 ,{ INF, INF, INF, INF, INF}
1504 ,{ INF, INF, INF, INF, INF}
1505 ,{ INF, INF, INF, INF, INF}
1506 ,{ INF, INF, INF, INF, INF}
1507 }
1508 ,{{ INF, INF, INF, INF, INF}
1509 ,{ INF, INF, INF, INF, INF}
1510 ,{ INF, INF, INF, INF, INF}
1511 ,{ INF, INF, INF, INF, INF}
1512 ,{ INF, INF, INF, INF, INF}
1513 }
1514 ,{{ INF, INF, INF, INF, INF}
1515 ,{ INF, INF, INF, INF, INF}
1516 ,{ INF, INF, INF, INF, INF}
1517 ,{ INF, INF, INF, INF, INF}
1518 ,{ INF, INF, INF, INF, INF}
1519 }
1520 ,{{ INF, INF, INF, INF, INF}
1521 ,{ INF, INF, INF, INF, INF}
1522 ,{ INF, INF, INF, INF, INF}
1523 ,{ INF, INF, INF, INF, INF}
1524 ,{ INF, INF, INF, INF, INF}
1525 }
1526 }
1527 ,{{{ 850, 850, 850, 850, 850}
1528 ,{ 850, 850, 850, 850, 850}
1529 ,{ 850, 850, 850, 850, 850}
1530 ,{ 850, 850, 850, 850, 850}
1531 ,{ 850, 850, 850, 850, 850}
1532 }
1533 ,{{ 850, 850, 850, 280, 850}
1534 ,{ 850, 850, 850, 280, 850}
1535 ,{ 850, 850, 850, 280, 850}
1536 ,{ 280, 280, 280, 280, 280}
1537 ,{ 850, 850, 850, 280, 850}
1538 }
1539 ,{{ 850, 850, 850, 850, 850}
1540 ,{ 850, 850, 850, 850, 850}
1541 ,{ 850, 850, 850, 850, 850}
1542 ,{ 850, 850, 850, 850, 850}
1543 ,{ 850, 850, 850, 850, 850}
1544 }
1545 ,{{ 850, 280, 850, 280, 850}
1546 ,{ 280, 280, 280, 280, 280}
1547 ,{ 850, 280, 850, 280, 850}
1548 ,{ 280, 280, 280, 280, 280}
1549 ,{ 850, 280, 850, 280, 850}
1550 }
1551 ,{{ 850, 850, 850, 850, -160}
1552 ,{ 850, 850, 850, 850, -160}
1553 ,{ 850, 850, 850, 850, -160}
1554 ,{ 850, 850, 850, 850, -160}
1555 ,{ -160, -160, -160, -160, -160}
1556 }
1557 }
1558 ,{{{ 850, 850, 850, 850, 850}
1559 ,{ 850, 850, 850, 850, 850}
1560 ,{ 850, 850, 850, 850, 850}
1561 ,{ 850, 850, 850, 850, 850}
1562 ,{ 850, 850, 850, 850, 850}
1563 }
1564 ,{{ 850, 850, 850, 280, 850}
1565 ,{ 850, 850, 850, 280, 850}
1566 ,{ 850, 850, 850, 280, 850}
1567 ,{ 280, 280, 280, 280, 280}
1568 ,{ 850, 850, 850, 280, 850}
1569 }
1570 ,{{ 850, 850, 850, 850, 850}
1571 ,{ 850, 850, 850, 850, 850}
1572 ,{ 850, 850, 850, 850, 850}
1573 ,{ 850, 850, 850, 850, 850}
1574 ,{ 850, 850, 850, 850, 850}
1575 }
1576 ,{{ 850, 280, 850, 280, 850}
1577 ,{ 850, 280, 850, 280, 850}
1578 ,{ 850, 280, 850, 280, 850}
1579 ,{ 280, 280, 280, 280, 280}
1580 ,{ 850, 280, 850, 280, 850}
1581 }
1582 ,{{ 850, 850, 850, 850, -160}
```

```
1583 ,{ 850, 850, 850, 850, -160}
1584 ,{ 850, 850, 850, 850, -160}
1585 ,{ 850, 850, 850, 850, -160}
1586 ,{ -160, -160, -160, -160, -160}
1587 }
1588 }
1589 ,{{ 1350, 1350, 1350, 1350, 1350}
1590 ,{ 1350, 1350, 1350, 1350, 1350}
1591 ,{ 1350, 1350, 1350, 1350, 1350}
1592 ,{ 1350, 1350, 1350, 1350, 1350}
1593 ,{ 1350, 1350, 1350, 1350, 1350}
1594 }
1595 ,{{ 1350, 1350, 1350, 780, 1350}
1596 ,{ 1350, 1350, 1350, 780, 1350}
1597 ,{ 1350, 1350, 1350, 780, 1350}
1598 ,{ 780, 780, 780, 780, 780}
1599 ,{ 1350, 1350, 1350, 780, 1350}
1600 }
1601 ,{{ 1350, 1350, 1350, 1350, 1350}
1602 ,{ 1350, 1350, 1350, 1350, 1350}
1603 ,{ 1350, 1350, 1350, 1350, 1350}
1604 ,{ 1350, 1350, 1350, 1350, 1350}
1605 ,{ 1350, 1350, 1350, 1350, 1350}
1606 }
1607 ,{{ 1350, 780, 1350, 780, 1350}
1608 ,{ 1350, 780, 1350, 780, 1350}
1609 ,{ 1350, 780, 1350, 780, 1350}
1610 ,{ 780, 780, 780, 780, 780}
1611 ,{ 1350, 780, 1350, 780, 1350}
1612 }
1613 ,{{ 1350, 1350, 1350, 1350, 340}
1614 ,{ 1350, 1350, 1350, 1350, 340}
1615 ,{ 1350, 1350, 1350, 1350, 340}
1616 ,{ 1350, 1350, 1350, 1350, 340}
1617 ,{ 340, 340, 340, 340, 340}
1618 }
1619 }
1620 ,{{ 1350, 1350, 1350, 1350, 1350}
1621 ,{ 1350, 1350, 1350, 1350, 1350}
1622 ,{ 1350, 1350, 1350, 1350, 1350}
1623 ,{ 1350, 1350, 1350, 1350, 1350}
1624 ,{ 1350, 1350, 1350, 1350, 1350}
1625 }
1626 ,{{ 1350, 1350, 1350, 780, 1350}
1627 ,{ 1350, 1350, 1350, 780, 1350}
1628 ,{ 1350, 1350, 1350, 780, 1350}
1629 ,{ 780, 780, 780, 780, 780}
1630 ,{ 1350, 1350, 1350, 780, 1350}
1631 }
1632 ,{{ 1350, 1350, 1350, 1350, 1350}
1633 ,{ 1350, 1350, 1350, 1350, 1350}
1634 ,{ 1350, 1350, 1350, 1350, 1350}
1635 ,{ 1350, 1350, 1350, 1350, 1350}
1636 ,{ 1350, 1350, 1350, 1350, 1350}
1637 }
1638 ,{{ 1350, 780, 1350, 780, 1350}
1639 ,{ 780, 780, 780, 780, 780}
1640 ,{ 1350, 780, 1350, 780, 1350}
1641 ,{ 780, 780, 780, 780, 780}
1642 ,{ 1350, 780, 1350, 780, 1350}
1643 }
1644 ,{{ 1350, 1350, 1350, 1350, 340}
1645 ,{ 1350, 1350, 1350, 1350, 340}
1646 ,{ 1350, 1350, 1350, 1350, 340}
1647 ,{ 1350, 1350, 1350, 1350, 340}
1648 ,{ 340, 340, 340, 340, 340}
1649 }
1650 }
1651 ,{{ 1350, 1350, 1350, 1350, 1350}
1652 ,{ 1350, 1350, 1350, 1350, 1350}
1653 ,{ 1350, 1350, 1350, 1350, 1350}
1654 ,{ 1350, 1350, 1350, 1350, 1350}
1655 ,{ 1350, 1350, 1350, 1350, 1350}
1656 }
1657 ,{{ 1350, 1350, 1350, 780, 1350}
1658 ,{ 1350, 1350, 1350, 780, 1350}
1659 ,{ 1350, 1350, 1350, 780, 1350}
1660 ,{ 780, 780, 780, 780, 780}
1661 ,{ 1350, 1350, 1350, 780, 1350}
1662 }
1663 ,{{ 1350, 1350, 1350, 1350, 1350}
1664 ,{ 1350, 1350, 1350, 1350, 1350}
1665 ,{ 1350, 1350, 1350, 1350, 1350}
1666 ,{ 1350, 1350, 1350, 1350, 1350}
1667 ,{ 1350, 1350, 1350, 1350, 1350}
1668 }
1669 ,{{ 1350, 780, 1350, 780, 1350}}
```

```
1670 ,{ 1350,    780,   1350,    780,   1350}
1671 ,{ 1350,    780,   1350,    780,   1350}
1672 ,{ 780,    780,   1350,    780,   780}
1673 ,{ 1350,    780,   1350,    780,   1350}
1674 }
1675 ,{{ 1350,   1350,   1350,   1350,   340}
1676 ,{ 1350,   1350,   1350,   1350,   340}
1677 ,{ 1350,   1350,   1350,   1350,   340}
1678 ,{ 1350,   1350,   1350,   1350,   340}
1679 ,{ 340,    340,    340,    340,    340}
1680 }
1681 }
1682 ,{{ { 1350,   1350,   1350,   1350,   1350}
1683 ,{ 1350,   1350,   1350,   1350,   1350}
1684 ,{ 1350,   1350,   1350,   1350,   1350}
1685 ,{ 1350,   1350,   1350,   1350,   1350}
1686 ,{ 1350,   1350,   1350,   1350,   1350}
1687 }
1688 ,{{ { 1350,   1350,   1350,   780,   1350}
1689 ,{ 1350,   1350,   1350,   780,   1350}
1690 ,{ 1350,   1350,   1350,   780,   1350}
1691 ,{ 780,    780,   1350,   780,   780}
1692 ,{ 1350,   1350,   1350,   780,   1350}
1693 }
1694 ,{{ { 1350,   1350,   1350,   1350,   1350}
1695 ,{ 1350,   1350,   1350,   1350,   1350}
1696 ,{ 1350,   1350,   1350,   1350,   1350}
1697 ,{ 1350,   1350,   1350,   1350,   1350}
1698 ,{ 1350,   1350,   1350,   1350,   1350}
1699 }
1700 ,{{ { 1350,   780,   1350,   780,   1350}
1701 ,{ 780,    780,   1350,   780,   780}
1702 ,{ 1350,   780,   1350,   780,   1350}
1703 ,{ 780,    780,   1350,   780,   780}
1704 ,{ 1350,   780,   1350,   780,   1350}
1705 }
1706 ,{{ { 1350,   1350,   1350,   1350,   340}
1707 ,{ 1350,   1350,   1350,   1350,   340}
1708 ,{ 1350,   1350,   1350,   1350,   340}
1709 ,{ 1350,   1350,   1350,   1350,   340}
1710 ,{ 340,    340,    340,    340,    340}
1711 }
1712 }
1713 ,{{ { { 1350,   1350,   1350,   1350,   1350}
1714 ,{ 1350,   1350,   1350,   1350,   1350}
1715 ,{ 1350,   1350,   1350,   1350,   1350}
1716 ,{ 1350,   1350,   1350,   1350,   1350}
1717 ,{ 1350,   1350,   1350,   1350,   1350}
1718 }
1719 ,{{ { { 1350,   1350,   1350,   780,   1350}
1720 ,{ 1350,   1350,   1350,   780,   1350}
1721 ,{ 1350,   1350,   1350,   780,   1350}
1722 ,{ 780,    780,   1350,   780,   780}
1723 ,{ 1350,   1350,   1350,   780,   1350}
1724 }
1725 ,{{ { { 1350,   1350,   1350,   1350,   1350}
1726 ,{ 1350,   1350,   1350,   1350,   1350}
1727 ,{ 1350,   1350,   1350,   1350,   1350}
1728 ,{ 1350,   1350,   1350,   1350,   1350}
1729 ,{ 1350,   1350,   1350,   1350,   1350}
1730 }
1731 ,{{ { { 1350,   780,   1350,   780,   1350}
1732 ,{ 1350,   780,   1350,   780,   1350}
1733 ,{ 1350,   780,   1350,   780,   1350}
1734 ,{ 780,    780,   1350,   780,   780}
1735 ,{ 1350,   780,   1350,   780,   1350}
1736 }
1737 ,{{ { { 1350,   1350,   1350,   1350,   340}
1738 ,{ 1350,   1350,   1350,   1350,   340}
1739 ,{ 1350,   1350,   1350,   1350,   340}
1740 ,{ 1350,   1350,   1350,   1350,   340}
1741 ,{ 340,    340,    340,    340,    340}
1742 }
1743 }
1744 }
1745 ,{{ {{ { INF,     INF,     INF,     INF,     INF}
1746 ,{ INF,     INF,     INF,     INF,     INF}
1747 ,{ INF,     INF,     INF,     INF,     INF}
1748 ,{ INF,     INF,     INF,     INF,     INF}
1749 ,{ INF,     INF,     INF,     INF,     INF}
1750 }
1751 ,{{ { { INF,     INF,     INF,     INF,     INF}
1752 ,{ INF,     INF,     INF,     INF,     INF}
1753 ,{ INF,     INF,     INF,     INF,     INF}
1754 ,{ INF,     INF,     INF,     INF,     INF}
1755 ,{ INF,     INF,     INF,     INF,     INF}
1756 }
```

```
1757 ,{ { INF, INF, INF, INF, INF }
1758 ,{ INF, INF, INF, INF, INF }
1759 ,{ INF, INF, INF, INF, INF }
1760 ,{ INF, INF, INF, INF, INF }
1761 ,{ INF, INF, INF, INF, INF }
1762 }
1763 ,{ { INF, INF, INF, INF, INF }
1764 ,{ INF, INF, INF, INF, INF }
1765 ,{ INF, INF, INF, INF, INF }
1766 ,{ INF, INF, INF, INF, INF }
1767 ,{ INF, INF, INF, INF, INF }
1768 }
1769 ,{ { INF, INF, INF, INF, INF }
1770 ,{ INF, INF, INF, INF, INF }
1771 ,{ INF, INF, INF, INF, INF }
1772 ,{ INF, INF, INF, INF, INF }
1773 ,{ INF, INF, INF, INF, INF }
1774 }
1775 }
1776 ,{ { 850, 850, 850, 850, 850 }
1777 ,{ 850, 850, 850, 850, 850 }
1778 ,{ 850, 850, 850, 850, 850 }
1779 ,{ 850, 850, 850, 850, 850 }
1780 ,{ 850, 850, 850, 850, 850 }
1781 }
1782 ,{ { 850, 850, 850, 850, 850 }
1783 ,{ 850, 850, 850, 850, 850 }
1784 ,{ 850, 850, 850, 850, 850 }
1785 ,{ 280, 280, 280, 280, 280 }
1786 ,{ 850, 850, 850, 850, 850 }
1787 }
1788 ,{ { 850, 850, 850, 850, 850 }
1789 ,{ 850, 850, 850, 850, 850 }
1790 ,{ 850, 850, 850, 850, 850 }
1791 ,{ 850, 850, 850, 850, 850 }
1792 ,{ 850, 850, 850, 850, 850 }
1793 }
1794 ,{ { 850, 280, 850, 280, 850 }
1795 ,{ 280, 280, 280, 280, 280 }
1796 ,{ 850, 280, 850, 280, 850 }
1797 ,{ 280, 280, 280, 280, 280 }
1798 ,{ 850, 280, 850, 280, 850 }
1799 }
1800 ,{ { 850, 850, 850, 850, -160 }
1801 ,{ 850, 850, 850, 850, -160 }
1802 ,{ 850, 850, 850, 850, -160 }
1803 ,{ 850, 850, 850, 850, -160 }
1804 ,{ -160, -160, -160, -160, -160 }
1805 }
1806 }
1807 ,{ { { 850, 850, 850, 850, 850 }
1808 ,{ 850, 850, 850, 850, 850 }
1809 ,{ 850, 850, 850, 850, 850 }
1810 ,{ 850, 850, 850, 850, 850 }
1811 ,{ 850, 850, 850, 850, 850 }
1812 }
1813 ,{ { { 850, 850, 850, 850, 850 }
1814 ,{ 850, 850, 850, 850, 850 }
1815 ,{ 850, 850, 850, 850, 850 }
1816 ,{ 280, 280, 280, 280, 280 }
1817 ,{ 850, 850, 850, 850, 850 }
1818 }
1819 ,{ { { 850, 850, 850, 850, 850 }
1820 ,{ 850, 850, 850, 850, 850 }
1821 ,{ 850, 850, 850, 850, 850 }
1822 ,{ 850, 850, 850, 850, 850 }
1823 ,{ 850, 850, 850, 850, 850 }
1824 }
1825 ,{ { { 850, 280, 850, 280, 850 }
1826 ,{ 850, 280, 850, 280, 850 }
1827 ,{ 850, 280, 850, 280, 850 }
1828 ,{ 280, 280, 280, 280, 280 }
1829 ,{ 850, 280, 850, 280, 850 }
1830 }
1831 ,{ { { 850, 850, 850, 850, -160 }
1832 ,{ 850, 850, 850, 850, -160 }
1833 ,{ 850, 850, 850, 850, -160 }
1834 ,{ 850, 850, 850, 850, -160 }
1835 ,{ -160, -160, -160, -160, -160 }
1836 }
1837 }
1838 ,{ { { 1350, 1350, 1350, 1350, 1350 }
1839 ,{ 1350, 1350, 1350, 1350, 1350 }
1840 ,{ 1350, 1350, 1350, 1350, 1350 }
1841 ,{ 1350, 1350, 1350, 1350, 1350 }
1842 ,{ 1350, 1350, 1350, 1350, 1350 }
1843 }
```

```
1844 ,{{ 1350, 1350, 1350, 1350, 1350}
1845 ,{ 1350, 1350, 1350, 1350, 1350}
1846 ,{ 1350, 1350, 1350, 1350, 1350}
1847 ,{ 780, 780, 780, 780, 780}
1848 ,{ 1350, 1350, 1350, 1350, 1350}
1849 }
1850 ,{{ 1350, 1350, 1350, 1350, 1350}
1851 ,{ 1350, 1350, 1350, 1350, 1350}
1852 ,{ 1350, 1350, 1350, 1350, 1350}
1853 ,{ 1350, 1350, 1350, 1350, 1350}
1854 ,{ 1350, 1350, 1350, 1350, 1350}
1855 }
1856 ,{{ 1350, 780, 1350, 780, 1350}
1857 ,{ 1350, 780, 1350, 780, 1350}
1858 ,{ 1350, 780, 1350, 780, 1350}
1859 ,{ 780, 780, 780, 780, 780}
1860 ,{ 1350, 780, 1350, 780, 1350}
1861 }
1862 ,{{ 1350, 1350, 1350, 1350, 340}
1863 ,{ 1350, 1350, 1350, 1350, 340}
1864 ,{ 1350, 1350, 1350, 1350, 340}
1865 ,{ 1350, 1350, 1350, 1350, 340}
1866 ,{ 340, 340, 340, 340, 340}
1867 }
1868 }
1869 ,{{{{ 1350, 1350, 1350, 1350, 1350}
1870 ,{ 1350, 1350, 1350, 1350, 1350}
1871 ,{ 1350, 1350, 1350, 1350, 1350}
1872 ,{ 1350, 1350, 1350, 1350, 1350}
1873 ,{ 1350, 1350, 1350, 1350, 1350}
1874 }
1875 ,{{{ 1350, 1350, 1350, 1350, 1350}
1876 ,{ 1350, 1350, 1350, 1350, 1350}
1877 ,{ 1350, 1350, 1350, 1350, 1350}
1878 ,{ 780, 780, 780, 780, 780}
1879 ,{ 1350, 1350, 1350, 1350, 1350}
1880 }
1881 ,{{ 1350, 1350, 1350, 1350, 1350}
1882 ,{ 1350, 1350, 1350, 1350, 1350}
1883 ,{ 1350, 1350, 1350, 1350, 1350}
1884 ,{ 1350, 1350, 1350, 1350, 1350}
1885 ,{ 1350, 1350, 1350, 1350, 1350}
1886 }
1887 ,{{ 1350, 780, 1350, 780, 1350}
1888 ,{ 780, 780, 780, 780, 780}
1889 ,{ 1350, 780, 1350, 780, 1350}
1890 ,{ 780, 780, 780, 780, 780}
1891 ,{ 1350, 780, 1350, 780, 1350}
1892 }
1893 ,{{{{ 1350, 1350, 1350, 1350, 340}
1894 ,{ 1350, 1350, 1350, 1350, 340}
1895 ,{ 1350, 1350, 1350, 1350, 340}
1896 ,{ 1350, 1350, 1350, 1350, 340}
1897 ,{ 340, 340, 340, 340, 340}
1898 }
1899 }
1900 ,{{{{ 1350, 1350, 1350, 1350, 1350}
1901 ,{ 1350, 1350, 1350, 1350, 1350}
1902 ,{ 1350, 1350, 1350, 1350, 1350}
1903 ,{ 1350, 1350, 1350, 1350, 1350}
1904 ,{ 1350, 1350, 1350, 1350, 1350}
1905 }
1906 ,{{ 1350, 1350, 1350, 1350, 1350}
1907 ,{ 1350, 1350, 1350, 1350, 1350}
1908 ,{ 1350, 1350, 1350, 1350, 1350}
1909 ,{ 780, 780, 780, 780, 780}
1910 ,{ 1350, 1350, 1350, 1350, 1350}
1911 }
1912 ,{{ 1350, 1350, 1350, 1350, 1350}
1913 ,{ 1350, 1350, 1350, 1350, 1350}
1914 ,{ 1350, 1350, 1350, 1350, 1350}
1915 ,{ 1350, 1350, 1350, 1350, 1350}
1916 ,{ 1350, 1350, 1350, 1350, 1350}
1917 }
1918 ,{{ 1350, 780, 1350, 780, 1350}
1919 ,{ 1350, 780, 1350, 780, 1350}
1920 ,{ 1350, 780, 1350, 780, 1350}
1921 ,{ 780, 780, 780, 780, 780}
1922 ,{ 1350, 780, 1350, 780, 1350}
1923 }
1924 ,{{ 1350, 1350, 1350, 1350, 340}
1925 ,{ 1350, 1350, 1350, 1350, 340}
1926 ,{ 1350, 1350, 1350, 1350, 340}
1927 ,{ 1350, 1350, 1350, 1350, 340}
1928 ,{ 340, 340, 340, 340, 340}
1929 }
1930 }
```

```

1931 ,{{{{ 1350, 1350, 1350, 1350, 1350}
1932 ,{ 1350, 1350, 1350, 1350, 1350}
1933 ,{ 1350, 1350, 1350, 1350, 1350}
1934 ,{ 1350, 1350, 1350, 1350, 1350}
1935 ,{ 1350, 1350, 1350, 1350, 1350}
1936 }
1937 ,{{ 1350, 1350, 1350, 1350, 1350}
1938 ,{ 1350, 1350, 1350, 1350, 1350}
1939 ,{ 1350, 1350, 1350, 1350, 1350}
1940 ,{ 780, 780, 780, 780, 780}
1941 ,{ 1350, 1350, 1350, 1350, 1350}
1942 }
1943 ,{{ 1350, 1350, 1350, 1350, 1350}
1944 ,{ 1350, 1350, 1350, 1350, 1350}
1945 ,{ 1350, 1350, 1350, 1350, 1350}
1946 ,{ 1350, 1350, 1350, 1350, 1350}
1947 ,{ 1350, 1350, 1350, 1350, 1350}
1948 }
1949 ,{{ 1350, 780, 1350, 780, 1350}
1950 ,{ 780, 780, 780, 780, 780}
1951 ,{ 1350, 780, 1350, 780, 1350}
1952 ,{ 780, 780, 780, 780, 780}
1953 ,{ 1350, 780, 1350, 780, 1350}
1954 }
1955 ,{{ 1350, 1350, 1350, 1350, 340}
1956 ,{ 1350, 1350, 1350, 1350, 340}
1957 ,{ 1350, 1350, 1350, 1350, 340}
1958 ,{ 1350, 1350, 1350, 1350, 340}
1959 ,{ 340, 340, 340, 340, 340}
1960 }
1961 }
1962 ,{{{{ 1350, 1350, 1350, 1350, 1350}
1963 ,{ 1350, 1350, 1350, 1350, 1350}
1964 ,{ 1350, 1350, 1350, 1350, 1350}
1965 ,{ 1350, 1350, 1350, 1350, 1350}
1966 ,{ 1350, 1350, 1350, 1350, 1350}
1967 }
1968 ,{{ 1350, 1350, 1350, 1350, 1350}
1969 ,{ 1350, 1350, 1350, 1350, 1350}
1970 ,{ 1350, 1350, 1350, 1350, 1350}
1971 ,{ 780, 780, 780, 780, 780}
1972 ,{ 1350, 1350, 1350, 1350, 1350}
1973 }
1974 ,{{ 1350, 1350, 1350, 1350, 1350}
1975 ,{ 1350, 1350, 1350, 1350, 1350}
1976 ,{ 1350, 1350, 1350, 1350, 1350}
1977 ,{ 1350, 1350, 1350, 1350, 1350}
1978 ,{ 1350, 1350, 1350, 1350, 1350}
1979 }
1980 ,{{ 1350, 780, 1350, 780, 1350}
1981 ,{ 1350, 780, 1350, 780, 1350}
1982 ,{ 1350, 780, 1350, 780, 1350}
1983 ,{ 780, 780, 780, 780, 780}
1984 ,{ 1350, 780, 1350, 780, 1350}
1985 }
1986 ,{{ 1350, 1350, 1350, 1350, 340}
1987 ,{ 1350, 1350, 1350, 1350, 340}
1988 ,{ 1350, 1350, 1350, 1350, 340}
1989 ,{ 1350, 1350, 1350, 1350, 340}
1990 ,{ 340, 340, 340, 340, 340}
1991 }
1992 }
1993 }};

```

## 18.168 intl22.h

```

1 PUBLIC int int22_37[NBPAIRS+1][NBPAIRS+1][5][5][5][5] =
2 {{{{{{{ INF, INF, INF, INF, INF}
3 ,{ INF, INF, INF, INF, INF}
4 ,{ INF, INF, INF, INF, INF}
5 ,{ INF, INF, INF, INF, INF}
6 ,{ INF, INF, INF, INF, INF}
7 }
8 ,{{ INF, INF, INF, INF, INF}
9 ,{ INF, INF, INF, INF, INF}
10 ,{ INF, INF, INF, INF, INF}
11 ,{ INF, INF, INF, INF, INF}
12 ,{ INF, INF, INF, INF, INF}
13 }
14 ,{{ INF, INF, INF, INF, INF}
15 ,{ INF, INF, INF, INF, INF}
16 ,{ INF, INF, INF, INF, INF}
17 ,{ INF, INF, INF, INF, INF}
18 ,{ INF, INF, INF, INF, INF}
19 }

```

```
20 ,{{ INF, INF, INF, INF, INF}
21 ,{ INF, INF, INF, INF, INF}
22 ,{ INF, INF, INF, INF, INF}
23 ,{ INF, INF, INF, INF, INF}
24 ,{ INF, INF, INF, INF, INF}
25 }
26 ,{{ INF, INF, INF, INF, INF}
27 ,{ INF, INF, INF, INF, INF}
28 ,{ INF, INF, INF, INF, INF}
29 ,{ INF, INF, INF, INF, INF}
30 ,{ INF, INF, INF, INF, INF}
31 }
32 }
33 ,{{{ INF, INF, INF, INF, INF}
34 ,{ INF, INF, INF, INF, INF}
35 ,{ INF, INF, INF, INF, INF}
36 ,{ INF, INF, INF, INF, INF}
37 ,{ INF, INF, INF, INF, INF}
38 }
39 ,{{ INF, INF, INF, INF, INF}
40 ,{ INF, INF, INF, INF, INF}
41 ,{ INF, INF, INF, INF, INF}
42 ,{ INF, INF, INF, INF, INF}
43 ,{ INF, INF, INF, INF, INF}
44 }
45 ,{{ INF, INF, INF, INF, INF}
46 ,{ INF, INF, INF, INF, INF}
47 ,{ INF, INF, INF, INF, INF}
48 ,{ INF, INF, INF, INF, INF}
49 ,{ INF, INF, INF, INF, INF}
50 }
51 ,{{ INF, INF, INF, INF, INF}
52 ,{ INF, INF, INF, INF, INF}
53 ,{ INF, INF, INF, INF, INF}
54 ,{ INF, INF, INF, INF, INF}
55 ,{ INF, INF, INF, INF, INF}
56 }
57 ,{{ INF, INF, INF, INF, INF}
58 ,{ INF, INF, INF, INF, INF}
59 ,{ INF, INF, INF, INF, INF}
60 ,{ INF, INF, INF, INF, INF}
61 ,{ INF, INF, INF, INF, INF}
62 }
63 }
64 ,{{{ INF, INF, INF, INF, INF}
65 ,{ INF, INF, INF, INF, INF}
66 ,{ INF, INF, INF, INF, INF}
67 ,{ INF, INF, INF, INF, INF}
68 ,{ INF, INF, INF, INF, INF}
69 }
70 ,{{ INF, INF, INF, INF, INF}
71 ,{ INF, INF, INF, INF, INF}
72 ,{ INF, INF, INF, INF, INF}
73 ,{ INF, INF, INF, INF, INF}
74 ,{ INF, INF, INF, INF, INF}
75 }
76 ,{{ INF, INF, INF, INF, INF}
77 ,{ INF, INF, INF, INF, INF}
78 ,{ INF, INF, INF, INF, INF}
79 ,{ INF, INF, INF, INF, INF}
80 ,{ INF, INF, INF, INF, INF}
81 }
82 ,{{ INF, INF, INF, INF, INF}
83 ,{ INF, INF, INF, INF, INF}
84 ,{ INF, INF, INF, INF, INF}
85 ,{ INF, INF, INF, INF, INF}
86 ,{ INF, INF, INF, INF, INF}
87 }
88 ,{{ INF, INF, INF, INF, INF}
89 ,{ INF, INF, INF, INF, INF}
90 ,{ INF, INF, INF, INF, INF}
91 ,{ INF, INF, INF, INF, INF}
92 ,{ INF, INF, INF, INF, INF}
93 }
94 }
95 ,{{{ INF, INF, INF, INF, INF}
96 ,{ INF, INF, INF, INF, INF}
97 ,{ INF, INF, INF, INF, INF}
98 ,{ INF, INF, INF, INF, INF}
99 ,{ INF, INF, INF, INF, INF}
100 }
101 ,{{ INF, INF, INF, INF, INF}
102 ,{ INF, INF, INF, INF, INF}
103 ,{ INF, INF, INF, INF, INF}
104 ,{ INF, INF, INF, INF, INF}
105 ,{ INF, INF, INF, INF, INF}
106 }
```

```
107 , {{ INF, INF, INF, INF, INF }
108 , { INF, INF, INF, INF, INF }
109 , { INF, INF, INF, INF, INF }
110 , { INF, INF, INF, INF, INF }
111 , { INF, INF, INF, INF, INF }
112 }
113 , {{ INF, INF, INF, INF, INF }
114 , { INF, INF, INF, INF, INF }
115 , { INF, INF, INF, INF, INF }
116 , { INF, INF, INF, INF, INF }
117 , { INF, INF, INF, INF, INF }
118 }
119 , {{ INF, INF, INF, INF, INF }
120 , { INF, INF, INF, INF, INF }
121 , { INF, INF, INF, INF, INF }
122 , { INF, INF, INF, INF, INF }
123 , { INF, INF, INF, INF, INF }
124 }
125 }
126 , {{{ INF, INF, INF, INF, INF }
127 , { INF, INF, INF, INF, INF }
128 , { INF, INF, INF, INF, INF }
129 , { INF, INF, INF, INF, INF }
130 , { INF, INF, INF, INF, INF }
131 }
132 , {{ INF, INF, INF, INF, INF }
133 , { INF, INF, INF, INF, INF }
134 , { INF, INF, INF, INF, INF }
135 , { INF, INF, INF, INF, INF }
136 , { INF, INF, INF, INF, INF }
137 }
138 , {{ INF, INF, INF, INF, INF }
139 , { INF, INF, INF, INF, INF }
140 , { INF, INF, INF, INF, INF }
141 , { INF, INF, INF, INF, INF }
142 , { INF, INF, INF, INF, INF }
143 }
144 , {{ INF, INF, INF, INF, INF }
145 , { INF, INF, INF, INF, INF }
146 , { INF, INF, INF, INF, INF }
147 , { INF, INF, INF, INF, INF }
148 , { INF, INF, INF, INF, INF }
149 }
150 , {{ INF, INF, INF, INF, INF }
151 , { INF, INF, INF, INF, INF }
152 , { INF, INF, INF, INF, INF }
153 , { INF, INF, INF, INF, INF }
154 , { INF, INF, INF, INF, INF }
155 }
156 }
157 }
158 , {{{ INF, INF, INF, INF, INF }
159 , { INF, INF, INF, INF, INF }
160 , { INF, INF, INF, INF, INF }
161 , { INF, INF, INF, INF, INF }
162 , { INF, INF, INF, INF, INF }
163 }
164 , {{ INF, INF, INF, INF, INF }
165 , { INF, INF, INF, INF, INF }
166 , { INF, INF, INF, INF, INF }
167 , { INF, INF, INF, INF, INF }
168 , { INF, INF, INF, INF, INF }
169 }
170 , {{ INF, INF, INF, INF, INF }
171 , { INF, INF, INF, INF, INF }
172 , { INF, INF, INF, INF, INF }
173 , { INF, INF, INF, INF, INF }
174 , { INF, INF, INF, INF, INF }
175 }
176 , {{ INF, INF, INF, INF, INF }
177 , { INF, INF, INF, INF, INF }
178 , { INF, INF, INF, INF, INF }
179 , { INF, INF, INF, INF, INF }
180 , { INF, INF, INF, INF, INF }
181 }
182 , {{ INF, INF, INF, INF, INF }
183 , { INF, INF, INF, INF, INF }
184 , { INF, INF, INF, INF, INF }
185 , { INF, INF, INF, INF, INF }
186 , { INF, INF, INF, INF, INF }
187 }
188 }
189 , {{{ INF, INF, INF, INF, INF }
190 , { INF, INF, INF, INF, INF }
191 , { INF, INF, INF, INF, INF }
192 , { INF, INF, INF, INF, INF }
193 , { INF, INF, INF, INF, INF }
```

```
194      }
195      ,{ {   INF,    INF,    INF,    INF,    INF}
196      ,{   INF,    INF,    INF,    INF,    INF}
197      ,{   INF,    INF,    INF,    INF,    INF}
198      ,{   INF,    INF,    INF,    INF,    INF}
199      ,{   INF,    INF,    INF,    INF,    INF}
200      }
201      ,{ {   INF,    INF,    INF,    INF,    INF}
202      ,{   INF,    INF,    INF,    INF,    INF}
203      ,{   INF,    INF,    INF,    INF,    INF}
204      ,{   INF,    INF,    INF,    INF,    INF}
205      ,{   INF,    INF,    INF,    INF,    INF}
206      }
207      ,{ {   INF,    INF,    INF,    INF,    INF}
208      ,{   INF,    INF,    INF,    INF,    INF}
209      ,{   INF,    INF,    INF,    INF,    INF}
210      ,{   INF,    INF,    INF,    INF,    INF}
211      ,{   INF,    INF,    INF,    INF,    INF}
212      }
213      ,{ {   INF,    INF,    INF,    INF,    INF}
214      ,{   INF,    INF,    INF,    INF,    INF}
215      ,{   INF,    INF,    INF,    INF,    INF}
216      ,{   INF,    INF,    INF,    INF,    INF}
217      ,{   INF,    INF,    INF,    INF,    INF}
218      }
219      }
220      ,{ {{  INF,    INF,    INF,    INF,    INF}
221      ,{   INF,    INF,    INF,    INF,    INF}
222      ,{   INF,    INF,    INF,    INF,    INF}
223      ,{   INF,    INF,    INF,    INF,    INF}
224      ,{   INF,    INF,    INF,    INF,    INF}
225      }
226      ,{ {{  INF,    INF,    INF,    INF,    INF}
227      ,{   INF,    INF,    INF,    INF,    INF}
228      ,{   INF,    INF,    INF,    INF,    INF}
229      ,{   INF,    INF,    INF,    INF,    INF}
230      ,{   INF,    INF,    INF,    INF,    INF}
231      }
232      ,{ {{  INF,    INF,    INF,    INF,    INF}
233      ,{   INF,    INF,    INF,    INF,    INF}
234      ,{   INF,    INF,    INF,    INF,    INF}
235      ,{   INF,    INF,    INF,    INF,    INF}
236      ,{   INF,    INF,    INF,    INF,    INF}
237      }
238      ,{ {{  INF,    INF,    INF,    INF,    INF}
239      ,{   INF,    INF,    INF,    INF,    INF}
240      ,{   INF,    INF,    INF,    INF,    INF}
241      ,{   INF,    INF,    INF,    INF,    INF}
242      ,{   INF,    INF,    INF,    INF,    INF}
243      }
244      ,{ {{  INF,    INF,    INF,    INF,    INF}
245      ,{   INF,    INF,    INF,    INF,    INF}
246      ,{   INF,    INF,    INF,    INF,    INF}
247      ,{   INF,    INF,    INF,    INF,    INF}
248      ,{   INF,    INF,    INF,    INF,    INF}
249      }
250      }
251      ,{ {{  INF,    INF,    INF,    INF,    INF}
252      ,{   INF,    INF,    INF,    INF,    INF}
253      ,{   INF,    INF,    INF,    INF,    INF}
254      ,{   INF,    INF,    INF,    INF,    INF}
255      ,{   INF,    INF,    INF,    INF,    INF}
256      }
257      ,{ {{  INF,    INF,    INF,    INF,    INF}
258      ,{   INF,    INF,    INF,    INF,    INF}
259      ,{   INF,    INF,    INF,    INF,    INF}
260      ,{   INF,    INF,    INF,    INF,    INF}
261      ,{   INF,    INF,    INF,    INF,    INF}
262      }
263      ,{ {{  INF,    INF,    INF,    INF,    INF}
264      ,{   INF,    INF,    INF,    INF,    INF}
265      ,{   INF,    INF,    INF,    INF,    INF}
266      ,{   INF,    INF,    INF,    INF,    INF}
267      ,{   INF,    INF,    INF,    INF,    INF}
268      }
269      ,{ {{  INF,    INF,    INF,    INF,    INF}
270      ,{   INF,    INF,    INF,    INF,    INF}
271      ,{   INF,    INF,    INF,    INF,    INF}
272      ,{   INF,    INF,    INF,    INF,    INF}
273      ,{   INF,    INF,    INF,    INF,    INF}
274      }
275      ,{ {{  INF,    INF,    INF,    INF,    INF}
276      ,{   INF,    INF,    INF,    INF,    INF}
277      ,{   INF,    INF,    INF,    INF,    INF}
278      ,{   INF,    INF,    INF,    INF,    INF}
279      ,{   INF,    INF,    INF,    INF,    INF}
280      }
```

```
281      }
282 ,{{{{ INF, INF, INF, INF, INF
283 ,{ INF, INF, INF, INF, INF
284 ,{ INF, INF, INF, INF, INF
285 ,{ INF, INF, INF, INF, INF
286 ,{ INF, INF, INF, INF, INF
287 }
288 ,{{ INF, INF, INF, INF, INF
289 ,{ INF, INF, INF, INF, INF
290 ,{ INF, INF, INF, INF, INF
291 ,{ INF, INF, INF, INF, INF
292 ,{ INF, INF, INF, INF, INF
293 }
294 ,{{ INF, INF, INF, INF, INF
295 ,{ INF, INF, INF, INF, INF
296 ,{ INF, INF, INF, INF, INF
297 ,{ INF, INF, INF, INF, INF
298 ,{ INF, INF, INF, INF, INF
299 }
300 ,{{{{ INF, INF, INF, INF, INF
301 ,{ INF, INF, INF, INF, INF
302 ,{ INF, INF, INF, INF, INF
303 ,{ INF, INF, INF, INF, INF
304 ,{ INF, INF, INF, INF, INF
305 }
306 ,{{{{ INF, INF, INF, INF, INF
307 ,{ INF, INF, INF, INF, INF
308 ,{ INF, INF, INF, INF, INF
309 ,{ INF, INF, INF, INF, INF
310 ,{ INF, INF, INF, INF, INF
311 }
312 }
313 }
314 ,{{{{ INF, INF, INF, INF, INF
315 ,{ INF, INF, INF, INF, INF
316 ,{ INF, INF, INF, INF, INF
317 ,{ INF, INF, INF, INF, INF
318 ,{ INF, INF, INF, INF, INF
319 }
320 ,{{{{ INF, INF, INF, INF, INF
321 ,{ INF, INF, INF, INF, INF
322 ,{ INF, INF, INF, INF, INF
323 ,{ INF, INF, INF, INF, INF
324 ,{ INF, INF, INF, INF, INF
325 }
326 ,{{{{ INF, INF, INF, INF, INF
327 ,{ INF, INF, INF, INF, INF
328 ,{ INF, INF, INF, INF, INF
329 ,{ INF, INF, INF, INF, INF
330 ,{ INF, INF, INF, INF, INF
331 }
332 ,{{{{ INF, INF, INF, INF, INF
333 ,{ INF, INF, INF, INF, INF
334 ,{ INF, INF, INF, INF, INF
335 ,{ INF, INF, INF, INF, INF
336 ,{ INF, INF, INF, INF, INF
337 }
338 ,{{{{ INF, INF, INF, INF, INF
339 ,{ INF, INF, INF, INF, INF
340 ,{ INF, INF, INF, INF, INF
341 ,{ INF, INF, INF, INF, INF
342 ,{ INF, INF, INF, INF, INF
343 }
344 }
345 ,{{{{ INF, INF, INF, INF, INF
346 ,{ INF, INF, INF, INF, INF
347 ,{ INF, INF, INF, INF, INF
348 ,{ INF, INF, INF, INF, INF
349 ,{ INF, INF, INF, INF, INF
350 }
351 ,{{{{ INF, INF, INF, INF, INF
352 ,{ INF, INF, INF, INF, INF
353 ,{ INF, INF, INF, INF, INF
354 ,{ INF, INF, INF, INF, INF
355 ,{ INF, INF, INF, INF, INF
356 }
357 ,{{{{ INF, INF, INF, INF, INF
358 ,{ INF, INF, INF, INF, INF
359 ,{ INF, INF, INF, INF, INF
360 ,{ INF, INF, INF, INF, INF
361 ,{ INF, INF, INF, INF, INF
362 }
363 ,{{{{ INF, INF, INF, INF, INF
364 ,{ INF, INF, INF, INF, INF
365 ,{ INF, INF, INF, INF, INF
366 ,{ INF, INF, INF, INF, INF
367 ,{ INF, INF, INF, INF, INF}
```

```
368      }
369 ,{{ INF, INF, INF, INF, INF}
370 ,{ INF, INF, INF, INF, INF}
371 ,{ INF, INF, INF, INF, INF}
372 ,{ INF, INF, INF, INF, INF}
373 ,{ INF, INF, INF, INF, INF}
374 }
375 }
376 ,{{ INF, INF, INF, INF, INF}
377 ,{ INF, INF, INF, INF, INF}
378 ,{ INF, INF, INF, INF, INF}
379 ,{ INF, INF, INF, INF, INF}
380 ,{ INF, INF, INF, INF, INF}
381 }
382 ,{{ INF, INF, INF, INF, INF}
383 ,{ INF, INF, INF, INF, INF}
384 ,{ INF, INF, INF, INF, INF}
385 ,{ INF, INF, INF, INF, INF}
386 ,{ INF, INF, INF, INF, INF}
387 }
388 ,{{ INF, INF, INF, INF, INF}
389 ,{ INF, INF, INF, INF, INF}
390 ,{ INF, INF, INF, INF, INF}
391 ,{ INF, INF, INF, INF, INF}
392 ,{ INF, INF, INF, INF, INF}
393 }
394 ,{{ INF, INF, INF, INF, INF}
395 ,{ INF, INF, INF, INF, INF}
396 ,{ INF, INF, INF, INF, INF}
397 ,{ INF, INF, INF, INF, INF}
398 ,{ INF, INF, INF, INF, INF}
399 }
400 ,{{ INF, INF, INF, INF, INF}
401 ,{ INF, INF, INF, INF, INF}
402 ,{ INF, INF, INF, INF, INF}
403 ,{ INF, INF, INF, INF, INF}
404 ,{ INF, INF, INF, INF, INF}
405 }
406 }
407 ,{{ INF, INF, INF, INF, INF}
408 ,{ INF, INF, INF, INF, INF}
409 ,{ INF, INF, INF, INF, INF}
410 ,{ INF, INF, INF, INF, INF}
411 ,{ INF, INF, INF, INF, INF}
412 }
413 ,{{ INF, INF, INF, INF, INF}
414 ,{ INF, INF, INF, INF, INF}
415 ,{ INF, INF, INF, INF, INF}
416 ,{ INF, INF, INF, INF, INF}
417 ,{ INF, INF, INF, INF, INF}
418 }
419 ,{{ INF, INF, INF, INF, INF}
420 ,{ INF, INF, INF, INF, INF}
421 ,{ INF, INF, INF, INF, INF}
422 ,{ INF, INF, INF, INF, INF}
423 ,{ INF, INF, INF, INF, INF}
424 }
425 ,{{ INF, INF, INF, INF, INF}
426 ,{ INF, INF, INF, INF, INF}
427 ,{ INF, INF, INF, INF, INF}
428 ,{ INF, INF, INF, INF, INF}
429 ,{ INF, INF, INF, INF, INF}
430 }
431 ,{{ INF, INF, INF, INF, INF}
432 ,{ INF, INF, INF, INF, INF}
433 ,{ INF, INF, INF, INF, INF}
434 ,{ INF, INF, INF, INF, INF}
435 ,{ INF, INF, INF, INF, INF}
436 }
437 }
438 ,{{ INF, INF, INF, INF, INF}
439 ,{ INF, INF, INF, INF, INF}
440 ,{ INF, INF, INF, INF, INF}
441 ,{ INF, INF, INF, INF, INF}
442 ,{ INF, INF, INF, INF, INF}
443 }
444 ,{{ INF, INF, INF, INF, INF}
445 ,{ INF, INF, INF, INF, INF}
446 ,{ INF, INF, INF, INF, INF}
447 ,{ INF, INF, INF, INF, INF}
448 ,{ INF, INF, INF, INF, INF}
449 }
450 ,{{ INF, INF, INF, INF, INF}
451 ,{ INF, INF, INF, INF, INF}
452 ,{ INF, INF, INF, INF, INF}
453 ,{ INF, INF, INF, INF, INF}
454 ,{ INF, INF, INF, INF, INF}
```

```
455      }
456 ,{ {   INF,    INF,    INF,    INF,    INF}
457 ,{ {   INF,    INF,    INF,    INF,    INF}
458 ,{ {   INF,    INF,    INF,    INF,    INF}
459 ,{ {   INF,    INF,    INF,    INF,    INF}
460 ,{ {   INF,    INF,    INF,    INF,    INF}
461 }
462 ,{ {   INF,    INF,    INF,    INF,    INF}
463 ,{ {   INF,    INF,    INF,    INF,    INF}
464 ,{ {   INF,    INF,    INF,    INF,    INF}
465 ,{ {   INF,    INF,    INF,    INF,    INF}
466 ,{ {   INF,    INF,    INF,    INF,    INF}
467 }
468 }
469 }
470 ,{ { { {   INF,    INF,    INF,    INF,    INF}
471 ,{ { { {   INF,    INF,    INF,    INF,    INF}
472 ,{ { { {   INF,    INF,    INF,    INF,    INF}
473 ,{ { { {   INF,    INF,    INF,    INF,    INF}
474 ,{ { { {   INF,    INF,    INF,    INF,    INF}
475 }
476 ,{ { { {   INF,    INF,    INF,    INF,    INF}
477 ,{ { { {   INF,    INF,    INF,    INF,    INF}
478 ,{ { { {   INF,    INF,    INF,    INF,    INF}
479 ,{ { { {   INF,    INF,    INF,    INF,    INF}
480 ,{ { { {   INF,    INF,    INF,    INF,    INF}
481 }
482 ,{ { { {   INF,    INF,    INF,    INF,    INF}
483 ,{ { { {   INF,    INF,    INF,    INF,    INF}
484 ,{ { { {   INF,    INF,    INF,    INF,    INF}
485 ,{ { { {   INF,    INF,    INF,    INF,    INF}
486 ,{ { { {   INF,    INF,    INF,    INF,    INF}
487 }
488 ,{ { { {   INF,    INF,    INF,    INF,    INF}
489 ,{ { { {   INF,    INF,    INF,    INF,    INF}
490 ,{ { { {   INF,    INF,    INF,    INF,    INF}
491 ,{ { { {   INF,    INF,    INF,    INF,    INF}
492 ,{ { { {   INF,    INF,    INF,    INF,    INF}
493 }
494 ,{ { { {   INF,    INF,    INF,    INF,    INF}
495 ,{ { { {   INF,    INF,    INF,    INF,    INF}
496 ,{ { { {   INF,    INF,    INF,    INF,    INF}
497 ,{ { { {   INF,    INF,    INF,    INF,    INF}
498 ,{ { { {   INF,    INF,    INF,    INF,    INF}
499 }
500 }
501 ,{ { { {   INF,    INF,    INF,    INF,    INF}
502 ,{ { { {   INF,    INF,    INF,    INF,    INF}
503 ,{ { { {   INF,    INF,    INF,    INF,    INF}
504 ,{ { { {   INF,    INF,    INF,    INF,    INF}
505 ,{ { { {   INF,    INF,    INF,    INF,    INF}
506 }
507 ,{ { { {   INF,    INF,    INF,    INF,    INF}
508 ,{ { { {   INF,    INF,    INF,    INF,    INF}
509 ,{ { { {   INF,    INF,    INF,    INF,    INF}
510 ,{ { { {   INF,    INF,    INF,    INF,    INF}
511 ,{ { { {   INF,    INF,    INF,    INF,    INF}
512 }
513 ,{ { { {   INF,    INF,    INF,    INF,    INF}
514 ,{ { { {   INF,    INF,    INF,    INF,    INF}
515 ,{ { { {   INF,    INF,    INF,    INF,    INF}
516 ,{ { { {   INF,    INF,    INF,    INF,    INF}
517 ,{ { { {   INF,    INF,    INF,    INF,    INF}
518 }
519 ,{ { { {   INF,    INF,    INF,    INF,    INF}
520 ,{ { { {   INF,    INF,    INF,    INF,    INF}
521 ,{ { { {   INF,    INF,    INF,    INF,    INF}
522 ,{ { { {   INF,    INF,    INF,    INF,    INF}
523 ,{ { { {   INF,    INF,    INF,    INF,    INF}
524 }
525 ,{ { { {   INF,    INF,    INF,    INF,    INF}
526 ,{ { { {   INF,    INF,    INF,    INF,    INF}
527 ,{ { { {   INF,    INF,    INF,    INF,    INF}
528 ,{ { { {   INF,    INF,    INF,    INF,    INF}
529 ,{ { { {   INF,    INF,    INF,    INF,    INF}
530 }
531 }
532 ,{ { { {   INF,    INF,    INF,    INF,    INF}
533 ,{ { { {   INF,    INF,    INF,    INF,    INF}
534 ,{ { { {   INF,    INF,    INF,    INF,    INF}
535 ,{ { { {   INF,    INF,    INF,    INF,    INF}
536 ,{ { { {   INF,    INF,    INF,    INF,    INF}
537 }
538 ,{ { { {   INF,    INF,    INF,    INF,    INF}
539 ,{ { { {   INF,    INF,    INF,    INF,    INF}
540 ,{ { { {   INF,    INF,    INF,    INF,    INF}
541 ,{ { { {   INF,    INF,    INF,    INF,    INF}
```

```
542 , { INF, INF, INF, INF, INF }
543 }
544 , { { INF, INF, INF, INF, INF }
545 , { INF, INF, INF, INF, INF }
546 , { INF, INF, INF, INF, INF }
547 , { INF, INF, INF, INF, INF }
548 , { INF, INF, INF, INF, INF }
549 }
550 , { { INF, INF, INF, INF, INF }
551 , { INF, INF, INF, INF, INF }
552 , { INF, INF, INF, INF, INF }
553 , { INF, INF, INF, INF, INF }
554 , { INF, INF, INF, INF, INF }
555 }
556 , { { INF, INF, INF, INF, INF }
557 , { INF, INF, INF, INF, INF }
558 , { INF, INF, INF, INF, INF }
559 , { INF, INF, INF, INF, INF }
560 , { INF, INF, INF, INF, INF }
561 }
562 }
563 , { { { INF, INF, INF, INF, INF }
564 , { INF, INF, INF, INF, INF }
565 , { INF, INF, INF, INF, INF }
566 , { INF, INF, INF, INF, INF }
567 , { INF, INF, INF, INF, INF }
568 }
569 , { { INF, INF, INF, INF, INF }
570 , { INF, INF, INF, INF, INF }
571 , { INF, INF, INF, INF, INF }
572 , { INF, INF, INF, INF, INF }
573 , { INF, INF, INF, INF, INF }
574 }
575 , { { INF, INF, INF, INF, INF }
576 , { INF, INF, INF, INF, INF }
577 , { INF, INF, INF, INF, INF }
578 , { INF, INF, INF, INF, INF }
579 , { INF, INF, INF, INF, INF }
580 }
581 , { { { INF, INF, INF, INF, INF }
582 , { INF, INF, INF, INF, INF }
583 , { INF, INF, INF, INF, INF }
584 , { INF, INF, INF, INF, INF }
585 , { INF, INF, INF, INF, INF }
586 }
587 , { { { INF, INF, INF, INF, INF }
588 , { INF, INF, INF, INF, INF }
589 , { INF, INF, INF, INF, INF }
590 , { INF, INF, INF, INF, INF }
591 , { INF, INF, INF, INF, INF }
592 }
593 }
594 , { { { INF, INF, INF, INF, INF }
595 , { INF, INF, INF, INF, INF }
596 , { INF, INF, INF, INF, INF }
597 , { INF, INF, INF, INF, INF }
598 , { INF, INF, INF, INF, INF }
599 }
600 , { { { INF, INF, INF, INF, INF }
601 , { INF, INF, INF, INF, INF }
602 , { INF, INF, INF, INF, INF }
603 , { INF, INF, INF, INF, INF }
604 , { INF, INF, INF, INF, INF }
605 }
606 , { { { INF, INF, INF, INF, INF }
607 , { INF, INF, INF, INF, INF }
608 , { INF, INF, INF, INF, INF }
609 , { INF, INF, INF, INF, INF }
610 , { INF, INF, INF, INF, INF }
611 }
612 , { { { INF, INF, INF, INF, INF }
613 , { INF, INF, INF, INF, INF }
614 , { INF, INF, INF, INF, INF }
615 , { INF, INF, INF, INF, INF }
616 , { INF, INF, INF, INF, INF }
617 }
618 , { { { INF, INF, INF, INF, INF }
619 , { INF, INF, INF, INF, INF }
620 , { INF, INF, INF, INF, INF }
621 , { INF, INF, INF, INF, INF }
622 , { INF, INF, INF, INF, INF }
623 }
624 }
625 }
626 , { { { { INF, INF, INF, INF, INF }
627 , { INF, INF, INF, INF, INF }
628 , { INF, INF, INF, INF, INF }
```

```
629 , { INF, INF, INF, INF, INF }
630 , { INF, INF, INF, INF, INF }
631 }
632 , { { INF, INF, INF, INF, INF }
633 , { INF, INF, INF, INF, INF }
634 , { INF, INF, INF, INF, INF }
635 , { INF, INF, INF, INF, INF }
636 , { INF, INF, INF, INF, INF }
637 }
638 , { { INF, INF, INF, INF, INF }
639 , { INF, INF, INF, INF, INF }
640 , { INF, INF, INF, INF, INF }
641 , { INF, INF, INF, INF, INF }
642 , { INF, INF, INF, INF, INF }
643 }
644 , { { INF, INF, INF, INF, INF }
645 , { INF, INF, INF, INF, INF }
646 , { INF, INF, INF, INF, INF }
647 , { INF, INF, INF, INF, INF }
648 , { INF, INF, INF, INF, INF }
649 }
650 , { { INF, INF, INF, INF, INF }
651 , { INF, INF, INF, INF, INF }
652 , { INF, INF, INF, INF, INF }
653 , { INF, INF, INF, INF, INF }
654 , { INF, INF, INF, INF, INF }
655 }
656 }
657 , { { { INF, INF, INF, INF, INF }
658 , { INF, INF, INF, INF, INF }
659 , { INF, INF, INF, INF, INF }
660 , { INF, INF, INF, INF, INF }
661 , { INF, INF, INF, INF, INF }
662 }
663 , { { { INF, INF, INF, INF, INF }
664 , { INF, INF, INF, INF, INF }
665 , { INF, INF, INF, INF, INF }
666 , { INF, INF, INF, INF, INF }
667 , { INF, INF, INF, INF, INF }
668 }
669 , { { { INF, INF, INF, INF, INF }
670 , { INF, INF, INF, INF, INF }
671 , { INF, INF, INF, INF, INF }
672 , { INF, INF, INF, INF, INF }
673 , { INF, INF, INF, INF, INF }
674 }
675 , { { { INF, INF, INF, INF, INF }
676 , { INF, INF, INF, INF, INF }
677 , { INF, INF, INF, INF, INF }
678 , { INF, INF, INF, INF, INF }
679 , { INF, INF, INF, INF, INF }
680 }
681 , { { { INF, INF, INF, INF, INF }
682 , { INF, INF, INF, INF, INF }
683 , { INF, INF, INF, INF, INF }
684 , { INF, INF, INF, INF, INF }
685 , { INF, INF, INF, INF, INF }
686 }
687 }
688 , { { { { INF, INF, INF, INF, INF }
689 , { { INF, INF, INF, INF, INF }
690 , { { INF, INF, INF, INF, INF }
691 , { { INF, INF, INF, INF, INF }
692 , { { INF, INF, INF, INF, INF }
693 }
694 , { { { { INF, INF, INF, INF, INF }
695 , { { INF, INF, INF, INF, INF }
696 , { { INF, INF, INF, INF, INF }
697 , { { INF, INF, INF, INF, INF }
698 , { { INF, INF, INF, INF, INF }
699 }
700 , { { { { INF, INF, INF, INF, INF }
701 , { { INF, INF, INF, INF, INF }
702 , { { INF, INF, INF, INF, INF }
703 , { { INF, INF, INF, INF, INF }
704 , { { INF, INF, INF, INF, INF }
705 }
706 , { { { { INF, INF, INF, INF, INF }
707 , { { INF, INF, INF, INF, INF }
708 , { { INF, INF, INF, INF, INF }
709 , { { INF, INF, INF, INF, INF }
710 , { { INF, INF, INF, INF, INF }
711 }
712 , { { { { INF, INF, INF, INF, INF }
713 , { { INF, INF, INF, INF, INF }
714 , { { INF, INF, INF, INF, INF }
715 , { { INF, INF, INF, INF, INF }
```

```
716 , { INF, INF, INF, INF, INF }
717 }
718 }
719 , {{ INF, INF, INF, INF, INF }
720 , { INF, INF, INF, INF, INF }
721 , { INF, INF, INF, INF, INF }
722 , { INF, INF, INF, INF, INF }
723 , { INF, INF, INF, INF, INF }
724 }
725 , {{ INF, INF, INF, INF, INF }
726 , { INF, INF, INF, INF, INF }
727 , { INF, INF, INF, INF, INF }
728 , { INF, INF, INF, INF, INF }
729 , { INF, INF, INF, INF, INF }
730 }
731 , {{ INF, INF, INF, INF, INF }
732 , { INF, INF, INF, INF, INF }
733 , { INF, INF, INF, INF, INF }
734 , { INF, INF, INF, INF, INF }
735 , { INF, INF, INF, INF, INF }
736 }
737 , {{ INF, INF, INF, INF, INF }
738 , { INF, INF, INF, INF, INF }
739 , { INF, INF, INF, INF, INF }
740 , { INF, INF, INF, INF, INF }
741 , { INF, INF, INF, INF, INF }
742 }
743 , {{ INF, INF, INF, INF, INF }
744 , { INF, INF, INF, INF, INF }
745 , { INF, INF, INF, INF, INF }
746 , { INF, INF, INF, INF, INF }
747 , { INF, INF, INF, INF, INF }
748 }
749 }
750 , {{ { INF, INF, INF, INF, INF }
751 , { INF, INF, INF, INF, INF }
752 , { INF, INF, INF, INF, INF }
753 , { INF, INF, INF, INF, INF }
754 , { INF, INF, INF, INF, INF }
755 }
756 , {{ INF, INF, INF, INF, INF }
757 , { INF, INF, INF, INF, INF }
758 , { INF, INF, INF, INF, INF }
759 , { INF, INF, INF, INF, INF }
760 , { INF, INF, INF, INF, INF }
761 }
762 , {{ INF, INF, INF, INF, INF }
763 , { INF, INF, INF, INF, INF }
764 , { INF, INF, INF, INF, INF }
765 , { INF, INF, INF, INF, INF }
766 , { INF, INF, INF, INF, INF }
767 }
768 , {{ INF, INF, INF, INF, INF }
769 , { INF, INF, INF, INF, INF }
770 , { INF, INF, INF, INF, INF }
771 , { INF, INF, INF, INF, INF }
772 , { INF, INF, INF, INF, INF }
773 }
774 , {{ INF, INF, INF, INF, INF }
775 , { INF, INF, INF, INF, INF }
776 , { INF, INF, INF, INF, INF }
777 , { INF, INF, INF, INF, INF }
778 , { INF, INF, INF, INF, INF }
779 }
780 }
781 }
782 , {{{ { INF, INF, INF, INF, INF }
783 , { INF, INF, INF, INF, INF }
784 , { INF, INF, INF, INF, INF }
785 , { INF, INF, INF, INF, INF }
786 , { INF, INF, INF, INF, INF }
787 }
788 , {{ { INF, INF, INF, INF, INF }
789 , { INF, INF, INF, INF, INF }
790 , { INF, INF, INF, INF, INF }
791 , { INF, INF, INF, INF, INF }
792 , { INF, INF, INF, INF, INF }
793 }
794 , {{ { INF, INF, INF, INF, INF }
795 , { INF, INF, INF, INF, INF }
796 , { INF, INF, INF, INF, INF }
797 , { INF, INF, INF, INF, INF }
798 , { INF, INF, INF, INF, INF }
799 }
800 , {{ { INF, INF, INF, INF, INF }
801 , { INF, INF, INF, INF, INF }
802 , { INF, INF, INF, INF, INF }
```

```
803 , { INF, INF, INF, INF, INF }
804 , { INF, INF, INF, INF, INF }
805 }
806 , { { INF, INF, INF, INF, INF }
807 , { INF, INF, INF, INF, INF }
808 , { INF, INF, INF, INF, INF }
809 , { INF, INF, INF, INF, INF }
810 , { INF, INF, INF, INF, INF }
811 }
812 }
813 , { { { INF, INF, INF, INF, INF }
814 , { INF, INF, INF, INF, INF }
815 , { INF, INF, INF, INF, INF }
816 , { INF, INF, INF, INF, INF }
817 , { INF, INF, INF, INF, INF }
818 }
819 , { { { INF, INF, INF, INF, INF }
820 , { INF, INF, INF, INF, INF }
821 , { INF, INF, INF, INF, INF }
822 , { INF, INF, INF, INF, INF }
823 , { INF, INF, INF, INF, INF }
824 }
825 , { { { INF, INF, INF, INF, INF }
826 , { INF, INF, INF, INF, INF }
827 , { INF, INF, INF, INF, INF }
828 , { INF, INF, INF, INF, INF }
829 , { INF, INF, INF, INF, INF }
830 }
831 , { { { INF, INF, INF, INF, INF }
832 , { INF, INF, INF, INF, INF }
833 , { INF, INF, INF, INF, INF }
834 , { INF, INF, INF, INF, INF }
835 , { INF, INF, INF, INF, INF }
836 }
837 , { { { INF, INF, INF, INF, INF }
838 , { INF, INF, INF, INF, INF }
839 , { INF, INF, INF, INF, INF }
840 , { INF, INF, INF, INF, INF }
841 , { INF, INF, INF, INF, INF }
842 }
843 }
844 , { { { INF, INF, INF, INF, INF }
845 , { INF, INF, INF, INF, INF }
846 , { INF, INF, INF, INF, INF }
847 , { INF, INF, INF, INF, INF }
848 , { INF, INF, INF, INF, INF }
849 }
850 , { { { INF, INF, INF, INF, INF }
851 , { INF, INF, INF, INF, INF }
852 , { INF, INF, INF, INF, INF }
853 , { INF, INF, INF, INF, INF }
854 , { INF, INF, INF, INF, INF }
855 }
856 , { { { INF, INF, INF, INF, INF }
857 , { INF, INF, INF, INF, INF }
858 , { INF, INF, INF, INF, INF }
859 , { INF, INF, INF, INF, INF }
860 , { INF, INF, INF, INF, INF }
861 }
862 , { { { INF, INF, INF, INF, INF }
863 , { INF, INF, INF, INF, INF }
864 , { INF, INF, INF, INF, INF }
865 , { INF, INF, INF, INF, INF }
866 , { INF, INF, INF, INF, INF }
867 }
868 , { { { INF, INF, INF, INF, INF }
869 , { INF, INF, INF, INF, INF }
870 , { INF, INF, INF, INF, INF }
871 , { INF, INF, INF, INF, INF }
872 , { INF, INF, INF, INF, INF }
873 }
874 }
875 , { { { INF, INF, INF, INF, INF }
876 , { INF, INF, INF, INF, INF }
877 , { INF, INF, INF, INF, INF }
878 , { INF, INF, INF, INF, INF }
879 , { INF, INF, INF, INF, INF }
880 }
881 , { { { INF, INF, INF, INF, INF }
882 , { INF, INF, INF, INF, INF }
883 , { INF, INF, INF, INF, INF }
884 , { INF, INF, INF, INF, INF }
885 , { INF, INF, INF, INF, INF }
886 }
887 , { { { INF, INF, INF, INF, INF }
888 , { INF, INF, INF, INF, INF }
889 , { INF, INF, INF, INF, INF }
```

```
890 , { INF, INF, INF, INF, INF }
891 , { INF, INF, INF, INF, INF }
892 }
893 , { { INF, INF, INF, INF, INF }
894 , { INF, INF, INF, INF, INF }
895 , { INF, INF, INF, INF, INF }
896 , { INF, INF, INF, INF, INF }
897 , { INF, INF, INF, INF, INF }
898 }
899 , { { INF, INF, INF, INF, INF }
900 , { INF, INF, INF, INF, INF }
901 , { INF, INF, INF, INF, INF }
902 , { INF, INF, INF, INF, INF }
903 , { INF, INF, INF, INF, INF }
904 }
905 }
906 , { { { INF, INF, INF, INF, INF }
907 , { INF, INF, INF, INF, INF }
908 , { INF, INF, INF, INF, INF }
909 , { INF, INF, INF, INF, INF }
910 , { INF, INF, INF, INF, INF }
911 }
912 , { { INF, INF, INF, INF, INF }
913 , { INF, INF, INF, INF, INF }
914 , { INF, INF, INF, INF, INF }
915 , { INF, INF, INF, INF, INF }
916 , { INF, INF, INF, INF, INF }
917 }
918 , { { INF, INF, INF, INF, INF }
919 , { INF, INF, INF, INF, INF }
920 , { INF, INF, INF, INF, INF }
921 , { INF, INF, INF, INF, INF }
922 , { INF, INF, INF, INF, INF }
923 }
924 , { { INF, INF, INF, INF, INF }
925 , { INF, INF, INF, INF, INF }
926 , { INF, INF, INF, INF, INF }
927 , { INF, INF, INF, INF, INF }
928 , { INF, INF, INF, INF, INF }
929 }
930 , { { INF, INF, INF, INF, INF }
931 , { INF, INF, INF, INF, INF }
932 , { INF, INF, INF, INF, INF }
933 , { INF, INF, INF, INF, INF }
934 , { INF, INF, INF, INF, INF }
935 }
936 }
937 }
938 , { { { { INF, INF, INF, INF, INF }
939 , { INF, INF, INF, INF, INF }
940 , { INF, INF, INF, INF, INF }
941 , { INF, INF, INF, INF, INF }
942 , { INF, INF, INF, INF, INF }
943 }
944 , { { INF, INF, INF, INF, INF }
945 , { INF, INF, INF, INF, INF }
946 , { INF, INF, INF, INF, INF }
947 , { INF, INF, INF, INF, INF }
948 , { INF, INF, INF, INF, INF }
949 }
950 , { { INF, INF, INF, INF, INF }
951 , { INF, INF, INF, INF, INF }
952 , { INF, INF, INF, INF, INF }
953 , { INF, INF, INF, INF, INF }
954 , { INF, INF, INF, INF, INF }
955 }
956 , { { INF, INF, INF, INF, INF }
957 , { INF, INF, INF, INF, INF }
958 , { INF, INF, INF, INF, INF }
959 , { INF, INF, INF, INF, INF }
960 , { INF, INF, INF, INF, INF }
961 }
962 , { { INF, INF, INF, INF, INF }
963 , { INF, INF, INF, INF, INF }
964 , { INF, INF, INF, INF, INF }
965 , { INF, INF, INF, INF, INF }
966 , { INF, INF, INF, INF, INF }
967 }
968 }
969 , { { { INF, INF, INF, INF, INF }
970 , { INF, INF, INF, INF, INF }
971 , { INF, INF, INF, INF, INF }
972 , { INF, INF, INF, INF, INF }
973 , { INF, INF, INF, INF, INF }
974 }
975 , { { INF, INF, INF, INF, INF }
976 , { INF, INF, INF, INF, INF }
```

```
977 , { INF, INF, INF, INF, INF }
978 , { INF, INF, INF, INF, INF }
979 , { INF, INF, INF, INF, INF }
980 }
981 , { { INF, INF, INF, INF, INF }
982 , { { INF, INF, INF, INF, INF }
983 , { { INF, INF, INF, INF, INF }
984 , { { INF, INF, INF, INF, INF }
985 , { { INF, INF, INF, INF, INF }
986 }
987 , { { { INF, INF, INF, INF, INF }
988 , { { { INF, INF, INF, INF, INF }
989 , { { { INF, INF, INF, INF, INF }
990 , { { { INF, INF, INF, INF, INF }
991 , { { { INF, INF, INF, INF, INF }
992 }
993 , { { { INF, INF, INF, INF, INF }
994 , { { { INF, INF, INF, INF, INF }
995 , { { { INF, INF, INF, INF, INF }
996 , { { { INF, INF, INF, INF, INF }
997 , { { { INF, INF, INF, INF, INF }
998 }
999 }

1000 , { { { { INF, INF, INF, INF, INF }
1001 , { { { { INF, INF, INF, INF, INF }
1002 , { { { { INF, INF, INF, INF, INF }
1003 , { { { { INF, INF, INF, INF, INF }
1004 , { { { { INF, INF, INF, INF, INF }
1005 }
1006 , { { { { INF, INF, INF, INF, INF }
1007 , { { { { INF, INF, INF, INF, INF }
1008 , { { { { INF, INF, INF, INF, INF }
1009 , { { { { INF, INF, INF, INF, INF }
1010 , { { { { INF, INF, INF, INF, INF }
1011 }
1012 , { { { { INF, INF, INF, INF, INF }
1013 , { { { { INF, INF, INF, INF, INF }
1014 , { { { { INF, INF, INF, INF, INF }
1015 , { { { { INF, INF, INF, INF, INF }
1016 , { { { { INF, INF, INF, INF, INF }
1017 }
1018 , { { { { INF, INF, INF, INF, INF }
1019 , { { { { INF, INF, INF, INF, INF }
1020 , { { { { INF, INF, INF, INF, INF }
1021 , { { { { INF, INF, INF, INF, INF }
1022 , { { { { INF, INF, INF, INF, INF }
1023 }
1024 , { { { { INF, INF, INF, INF, INF }
1025 , { { { { INF, INF, INF, INF, INF }
1026 , { { { { INF, INF, INF, INF, INF }
1027 , { { { { INF, INF, INF, INF, INF }
1028 , { { { { INF, INF, INF, INF, INF }
1029 }
1030 }
1031 , { { { { { INF, INF, INF, INF, INF }
1032 , { { { { { INF, INF, INF, INF, INF }
1033 , { { { { { INF, INF, INF, INF, INF }
1034 , { { { { { INF, INF, INF, INF, INF }
1035 , { { { { { INF, INF, INF, INF, INF }
1036 }
1037 , { { { { { INF, INF, INF, INF, INF }
1038 , { { { { { INF, INF, INF, INF, INF }
1039 , { { { { { INF, INF, INF, INF, INF }
1040 , { { { { { INF, INF, INF, INF, INF }
1041 , { { { { { INF, INF, INF, INF, INF }
1042 }
1043 , { { { { { INF, INF, INF, INF, INF }
1044 , { { { { { INF, INF, INF, INF, INF }
1045 , { { { { { INF, INF, INF, INF, INF }
1046 , { { { { { INF, INF, INF, INF, INF }
1047 , { { { { { INF, INF, INF, INF, INF }
1048 }
1049 , { { { { { INF, INF, INF, INF, INF }
1050 , { { { { { INF, INF, INF, INF, INF }
1051 , { { { { { INF, INF, INF, INF, INF }
1052 , { { { { { INF, INF, INF, INF, INF }
1053 , { { { { { INF, INF, INF, INF, INF }
1054 }
1055 , { { { { { INF, INF, INF, INF, INF }
1056 , { { { { { INF, INF, INF, INF, INF }
1057 , { { { { { INF, INF, INF, INF, INF }
1058 , { { { { { INF, INF, INF, INF, INF }
1059 , { { { { { INF, INF, INF, INF, INF }
1060 }
1061 }
1062 , { { { { { INF, INF, INF, INF, INF }
1063 , { { { { { INF, INF, INF, INF, INF }}
```

```
1064 , { INF, INF, INF, INF, INF }
1065 , { INF, INF, INF, INF, INF }
1066 , { INF, INF, INF, INF, INF }
1067 }
1068 , {{ INF, INF, INF, INF, INF }
1069 , { INF, INF, INF, INF, INF }
1070 , { INF, INF, INF, INF, INF }
1071 , { INF, INF, INF, INF, INF }
1072 , { INF, INF, INF, INF, INF }
1073 }
1074 , {{ INF, INF, INF, INF, INF }
1075 , { INF, INF, INF, INF, INF }
1076 , { INF, INF, INF, INF, INF }
1077 , { INF, INF, INF, INF, INF }
1078 , { INF, INF, INF, INF, INF }
1079 }
1080 , {{ INF, INF, INF, INF, INF }
1081 , { INF, INF, INF, INF, INF }
1082 , { INF, INF, INF, INF, INF }
1083 , { INF, INF, INF, INF, INF }
1084 , { INF, INF, INF, INF, INF }
1085 }
1086 , {{ INF, INF, INF, INF, INF }
1087 , { INF, INF, INF, INF, INF }
1088 , { INF, INF, INF, INF, INF }
1089 , { INF, INF, INF, INF, INF }
1090 , { INF, INF, INF, INF, INF }
1091 }
1092 }
1093 }
1094 , {{{ INF, INF, INF, INF, INF }
1095 , { INF, INF, INF, INF, INF }
1096 , { INF, INF, INF, INF, INF }
1097 , { INF, INF, INF, INF, INF }
1098 , { INF, INF, INF, INF, INF }
1099 }
1100 , {{ INF, INF, INF, INF, INF }
1101 , { INF, INF, INF, INF, INF }
1102 , { INF, INF, INF, INF, INF }
1103 , { INF, INF, INF, INF, INF }
1104 , { INF, INF, INF, INF, INF }
1105 }
1106 , {{ INF, INF, INF, INF, INF }
1107 , { INF, INF, INF, INF, INF }
1108 , { INF, INF, INF, INF, INF }
1109 , { INF, INF, INF, INF, INF }
1110 , { INF, INF, INF, INF, INF }
1111 }
1112 , {{ INF, INF, INF, INF, INF }
1113 , { INF, INF, INF, INF, INF }
1114 , { INF, INF, INF, INF, INF }
1115 , { INF, INF, INF, INF, INF }
1116 , { INF, INF, INF, INF, INF }
1117 }
1118 , {{ INF, INF, INF, INF, INF }
1119 , { INF, INF, INF, INF, INF }
1120 , { INF, INF, INF, INF, INF }
1121 , { INF, INF, INF, INF, INF }
1122 , { INF, INF, INF, INF, INF }
1123 }
1124 }
1125 , {{{ INF, INF, INF, INF, INF }
1126 , { INF, INF, INF, INF, INF }
1127 , { INF, INF, INF, INF, INF }
1128 , { INF, INF, INF, INF, INF }
1129 , { INF, INF, INF, INF, INF }
1130 }
1131 , {{ INF, INF, INF, INF, INF }
1132 , { INF, INF, INF, INF, INF }
1133 , { INF, INF, INF, INF, INF }
1134 , { INF, INF, INF, INF, INF }
1135 , { INF, INF, INF, INF, INF }
1136 }
1137 , {{ INF, INF, INF, INF, INF }
1138 , { INF, INF, INF, INF, INF }
1139 , { INF, INF, INF, INF, INF }
1140 , { INF, INF, INF, INF, INF }
1141 , { INF, INF, INF, INF, INF }
1142 }
1143 , {{ INF, INF, INF, INF, INF }
1144 , { INF, INF, INF, INF, INF }
1145 , { INF, INF, INF, INF, INF }
1146 , { INF, INF, INF, INF, INF }
1147 , { INF, INF, INF, INF, INF }
1148 }
1149 , {{ INF, INF, INF, INF, INF }
1150 , { INF, INF, INF, INF, INF }
```

```
1151 , { INF, INF, INF, INF, INF }
1152 , { INF, INF, INF, INF, INF }
1153 , { INF, INF, INF, INF, INF }
1154 }
1155 }
1156 , {{ { INF, INF, INF, INF, INF }
1157 , { INF, INF, INF, INF, INF }
1158 , { INF, INF, INF, INF, INF }
1159 , { INF, INF, INF, INF, INF }
1160 , { INF, INF, INF, INF, INF }
1161 }
1162 , {{ { INF, INF, INF, INF, INF }
1163 , { INF, INF, INF, INF, INF }
1164 , { INF, INF, INF, INF, INF }
1165 , { INF, INF, INF, INF, INF }
1166 , { INF, INF, INF, INF, INF }
1167 }
1168 , {{ { INF, INF, INF, INF, INF }
1169 , { INF, INF, INF, INF, INF }
1170 , { INF, INF, INF, INF, INF }
1171 , { INF, INF, INF, INF, INF }
1172 , { INF, INF, INF, INF, INF }
1173 }
1174 , {{ { INF, INF, INF, INF, INF }
1175 , { INF, INF, INF, INF, INF }
1176 , { INF, INF, INF, INF, INF }
1177 , { INF, INF, INF, INF, INF }
1178 , { INF, INF, INF, INF, INF }
1179 }
1180 , {{ { INF, INF, INF, INF, INF }
1181 , { INF, INF, INF, INF, INF }
1182 , { INF, INF, INF, INF, INF }
1183 , { INF, INF, INF, INF, INF }
1184 , { INF, INF, INF, INF, INF }
1185 }
1186 }
1187 , {{ { INF, INF, INF, INF, INF }
1188 , { INF, INF, INF, INF, INF }
1189 , { INF, INF, INF, INF, INF }
1190 , { INF, INF, INF, INF, INF }
1191 , { INF, INF, INF, INF, INF }
1192 }
1193 , {{ { INF, INF, INF, INF, INF }
1194 , { INF, INF, INF, INF, INF }
1195 , { INF, INF, INF, INF, INF }
1196 , { INF, INF, INF, INF, INF }
1197 , { INF, INF, INF, INF, INF }
1198 }
1199 , {{ { INF, INF, INF, INF, INF }
1200 , { INF, INF, INF, INF, INF }
1201 , { INF, INF, INF, INF, INF }
1202 , { INF, INF, INF, INF, INF }
1203 , { INF, INF, INF, INF, INF }
1204 }
1205 , {{ { INF, INF, INF, INF, INF }
1206 , { INF, INF, INF, INF, INF }
1207 , { INF, INF, INF, INF, INF }
1208 , { INF, INF, INF, INF, INF }
1209 , { INF, INF, INF, INF, INF }
1210 }
1211 , {{ { INF, INF, INF, INF, INF }
1212 , { INF, INF, INF, INF, INF }
1213 , { INF, INF, INF, INF, INF }
1214 , { INF, INF, INF, INF, INF }
1215 , { INF, INF, INF, INF, INF }
1216 }
1217 }
1218 , {{ { INF, INF, INF, INF, INF }
1219 , { INF, INF, INF, INF, INF }
1220 , { INF, INF, INF, INF, INF }
1221 , { INF, INF, INF, INF, INF }
1222 , { INF, INF, INF, INF, INF }
1223 }
1224 , {{ { INF, INF, INF, INF, INF }
1225 , { INF, INF, INF, INF, INF }
1226 , { INF, INF, INF, INF, INF }
1227 , { INF, INF, INF, INF, INF }
1228 , { INF, INF, INF, INF, INF }
1229 }
1230 , {{ { INF, INF, INF, INF, INF }
1231 , { INF, INF, INF, INF, INF }
1232 , { INF, INF, INF, INF, INF }
1233 , { INF, INF, INF, INF, INF }
1234 , { INF, INF, INF, INF, INF }
1235 }
1236 , {{ { INF, INF, INF, INF, INF }
1237 , { INF, INF, INF, INF, INF }}
```

```
1238 , { INF, INF, INF, INF, INF }
1239 , { INF, INF, INF, INF, INF }
1240 , { INF, INF, INF, INF, INF }
1241 }
1242 , {{ INF, INF, INF, INF, INF }
1243 , { INF, INF, INF, INF, INF }
1244 , { INF, INF, INF, INF, INF }
1245 , { INF, INF, INF, INF, INF }
1246 , { INF, INF, INF, INF, INF }
1247 }
1248 }
1249 }
1250 }
1251 , {{{{ INF, INF, INF, INF, INF }
1252 , { INF, INF, INF, INF, INF }
1253 , { INF, INF, INF, INF, INF }
1254 , { INF, INF, INF, INF, INF }
1255 , { INF, INF, INF, INF, INF }
1256 }
1257 , {{ INF, INF, INF, INF, INF }
1258 , { INF, INF, INF, INF, INF }
1259 , { INF, INF, INF, INF, INF }
1260 , { INF, INF, INF, INF, INF }
1261 , { INF, INF, INF, INF, INF }
1262 }
1263 , {{ INF, INF, INF, INF, INF }
1264 , { INF, INF, INF, INF, INF }
1265 , { INF, INF, INF, INF, INF }
1266 , { INF, INF, INF, INF, INF }
1267 , { INF, INF, INF, INF, INF }
1268 }
1269 , {{ INF, INF, INF, INF, INF }
1270 , { INF, INF, INF, INF, INF }
1271 , { INF, INF, INF, INF, INF }
1272 , { INF, INF, INF, INF, INF }
1273 , { INF, INF, INF, INF, INF }
1274 }
1275 , {{ INF, INF, INF, INF, INF }
1276 , { INF, INF, INF, INF, INF }
1277 , { INF, INF, INF, INF, INF }
1278 , { INF, INF, INF, INF, INF }
1279 , { INF, INF, INF, INF, INF }
1280 }
1281 }
1282 , {{ INF, INF, INF, INF, INF }
1283 , { INF, INF, INF, INF, INF }
1284 , { INF, INF, INF, INF, INF }
1285 , { INF, INF, INF, INF, INF }
1286 , { INF, INF, INF, INF, INF }
1287 }
1288 , {{ INF, INF, INF, INF, INF }
1289 , { INF, INF, INF, INF, INF }
1290 , { INF, INF, INF, INF, INF }
1291 , { INF, INF, INF, INF, INF }
1292 , { INF, INF, INF, INF, INF }
1293 }
1294 , {{ INF, INF, INF, INF, INF }
1295 , { INF, INF, INF, INF, INF }
1296 , { INF, INF, INF, INF, INF }
1297 , { INF, INF, INF, INF, INF }
1298 , { INF, INF, INF, INF, INF }
1299 }
1300 , {{ INF, INF, INF, INF, INF }
1301 , { INF, INF, INF, INF, INF }
1302 , { INF, INF, INF, INF, INF }
1303 , { INF, INF, INF, INF, INF }
1304 , { INF, INF, INF, INF, INF }
1305 }
1306 , {{ INF, INF, INF, INF, INF }
1307 , { INF, INF, INF, INF, INF }
1308 , { INF, INF, INF, INF, INF }
1309 , { INF, INF, INF, INF, INF }
1310 , { INF, INF, INF, INF, INF }
1311 }
1312 }
1313 , {{{ INF, INF, INF, INF, INF }
1314 , { INF, INF, INF, INF, INF }
1315 , { INF, INF, INF, INF, INF }
1316 , { INF, INF, INF, INF, INF }
1317 , { INF, INF, INF, INF, INF }
1318 }
1319 , {{ INF, INF, INF, INF, INF }
1320 , { INF, INF, INF, INF, INF }
1321 , { INF, INF, INF, INF, INF }
1322 , { INF, INF, INF, INF, INF }
1323 , { INF, INF, INF, INF, INF }
1324 }
```

```
1325 , {{ INF, INF, INF, INF, INF }
1326 , { INF, INF, INF, INF, INF }
1327 , { INF, INF, INF, INF, INF }
1328 , { INF, INF, INF, INF, INF }
1329 , { INF, INF, INF, INF, INF }
1330 }
1331 , {{ INF, INF, INF, INF, INF }
1332 , { INF, INF, INF, INF, INF }
1333 , { INF, INF, INF, INF, INF }
1334 , { INF, INF, INF, INF, INF }
1335 , { INF, INF, INF, INF, INF }
1336 }
1337 , {{ INF, INF, INF, INF, INF }
1338 , { INF, INF, INF, INF, INF }
1339 , { INF, INF, INF, INF, INF }
1340 , { INF, INF, INF, INF, INF }
1341 , { INF, INF, INF, INF, INF }
1342 }
1343 }
1344 , {{{ INF, INF, INF, INF, INF }
1345 , { INF, INF, INF, INF, INF }
1346 , { INF, INF, INF, INF, INF }
1347 , { INF, INF, INF, INF, INF }
1348 , { INF, INF, INF, INF, INF }
1349 }
1350 , {{ INF, INF, INF, INF, INF }
1351 , { INF, INF, INF, INF, INF }
1352 , { INF, INF, INF, INF, INF }
1353 , { INF, INF, INF, INF, INF }
1354 , { INF, INF, INF, INF, INF }
1355 }
1356 , {{ INF, INF, INF, INF, INF }
1357 , { INF, INF, INF, INF, INF }
1358 , { INF, INF, INF, INF, INF }
1359 , { INF, INF, INF, INF, INF }
1360 , { INF, INF, INF, INF, INF }
1361 }
1362 , {{ INF, INF, INF, INF, INF }
1363 , { INF, INF, INF, INF, INF }
1364 , { INF, INF, INF, INF, INF }
1365 , { INF, INF, INF, INF, INF }
1366 , { INF, INF, INF, INF, INF }
1367 }
1368 , {{ INF, INF, INF, INF, INF }
1369 , { INF, INF, INF, INF, INF }
1370 , { INF, INF, INF, INF, INF }
1371 , { INF, INF, INF, INF, INF }
1372 , { INF, INF, INF, INF, INF }
1373 }
1374 }
1375 , {{{ INF, INF, INF, INF, INF }
1376 , { INF, INF, INF, INF, INF }
1377 , { INF, INF, INF, INF, INF }
1378 , { INF, INF, INF, INF, INF }
1379 , { INF, INF, INF, INF, INF }
1380 }
1381 , {{ INF, INF, INF, INF, INF }
1382 , { INF, INF, INF, INF, INF }
1383 , { INF, INF, INF, INF, INF }
1384 , { INF, INF, INF, INF, INF }
1385 , { INF, INF, INF, INF, INF }
1386 }
1387 , {{ INF, INF, INF, INF, INF }
1388 , { INF, INF, INF, INF, INF }
1389 , { INF, INF, INF, INF, INF }
1390 , { INF, INF, INF, INF, INF }
1391 , { INF, INF, INF, INF, INF }
1392 }
1393 , {{ INF, INF, INF, INF, INF }
1394 , { INF, INF, INF, INF, INF }
1395 , { INF, INF, INF, INF, INF }
1396 , { INF, INF, INF, INF, INF }
1397 , { INF, INF, INF, INF, INF }
1398 }
1399 , {{ INF, INF, INF, INF, INF }
1400 , { INF, INF, INF, INF, INF }
1401 , { INF, INF, INF, INF, INF }
1402 , { INF, INF, INF, INF, INF }
1403 , { INF, INF, INF, INF, INF }
1404 }
1405 }
1406 }
1407 , {{{{ 200, 160, 200, 150, 200}}
1408 , { 200, 160, 200, 150, 200}
1409 , { 180, 140, 180, 140, 180}
1410 , { 200, 160, 200, 150, 200}
1411 , { 170, 130, 170, 120, 170}}
```

```
1412      }
1413 ,{{ 160, 120, 160, 110, 160}
1414 ,{ 160, 120, 160, 110, 160}
1415 ,{ 150, 110, 150, 110, 150}
1416 ,{ 110, 20, 110, 20, 90}
1417 ,{ 150, 110, 150, 110, 150}
1418 }
1419 ,{{ 200, 160, 200, 150, 200}
1420 ,{ 200, 160, 200, 150, 200}
1421 ,{ 180, 140, 180, 140, 180}
1422 ,{ 200, 160, 200, 150, 200}
1423 ,{ 170, 130, 170, 120, 170}
1424 }
1425 ,{{ 150, 110, 150, 110, 150}
1426 ,{ 110, 20, 110, 20, 90}
1427 ,{ 150, 110, 150, 110, 150}
1428 ,{ 80, 0, 10, 80, 20}
1429 ,{ 150, 110, 150, 110, 150}
1430 }
1431 ,{{ 200, 160, 200, 150, 200}
1432 ,{ 200, 160, 200, 150, 200}
1433 ,{ 170, 130, 170, 120, 170}
1434 ,{ 200, 160, 200, 150, 200}
1435 ,{ 100, 100, 80, 30, 80}
1436 }
1437 }
1438 ,{{{{ 200, 160, 200, 110, 200}
1439 ,{ 200, 160, 200, 60, 200}
1440 ,{ 180, 140, 180, 110, 180}
1441 ,{ 200, 160, 200, 60, 200}
1442 ,{ 170, 130, 170, 90, 170}
1443 }
1444 ,{{ 160, 120, 160, 20, 160}
1445 ,{ 160, 120, 160, 20, 160}
1446 ,{ 150, 110, 150, 20, 150}
1447 ,{ 60, 20, 60, -70, 60}
1448 ,{ 150, 110, 150, 20, 150}
1449 }
1450 ,{{ 200, 160, 200, 110, 200}
1451 ,{ 200, 160, 200, 60, 200}
1452 ,{ 180, 140, 180, 110, 180}
1453 ,{ 200, 160, 200, 60, 200}
1454 ,{ 170, 130, 170, 90, 170}
1455 }
1456 ,{{ 150, 110, 150, 20, 150}
1457 ,{ 60, 20, 60, -70, 60}
1458 ,{ 150, 110, 150, 20, 150}
1459 ,{ 10, -30, 10, 0, 10}
1460 ,{ 150, 110, 150, 20, 150}
1461 }
1462 ,{{ 200, 160, 200, 90, 200}
1463 ,{ 200, 160, 200, 60, 200}
1464 ,{ 170, 130, 170, 90, 170}
1465 ,{ 200, 160, 200, 60, 200}
1466 ,{ 100, 100, 80, -50, 80}
1467 }
1468 }
1469 ,{{{{ 180, 150, 180, 150, 170}
1470 ,{ 180, 150, 180, 150, 170}
1471 ,{ 170, 140, 170, 140, 150}
1472 ,{ 180, 150, 180, 150, 170}
1473 ,{ 150, 120, 150, 120, 140}
1474 }
1475 ,{{ 140, 110, 140, 110, 130}
1476 ,{ 140, 110, 140, 110, 130}
1477 ,{ 140, 110, 140, 110, 120}
1478 ,{ 110, 20, 110, 20, 90}
1479 ,{ 140, 110, 140, 110, 120}
1480 }
1481 ,{{ 180, 150, 180, 150, 170}
1482 ,{ 180, 150, 180, 150, 170}
1483 ,{ 170, 140, 170, 140, 150}
1484 ,{ 180, 150, 180, 150, 170}
1485 ,{ 150, 120, 150, 120, 140}
1486 }
1487 ,{{ 140, 110, 140, 110, 120}
1488 ,{ 110, 20, 110, 20, 90}
1489 ,{ 140, 110, 140, 110, 120}
1490 ,{ -10, -40, -10, -40, -20}
1491 ,{ 140, 110, 140, 110, 120}
1492 }
1493 ,{{ 180, 150, 180, 150, 170}
1494 ,{ 180, 150, 180, 150, 170}
1495 ,{ 150, 120, 150, 120, 140}
1496 ,{ 180, 150, 180, 150, 170}
1497 ,{ 60, 30, 60, 30, 50}
1498 }
```

```
1499    }
1500 ,{{ 200,   110,   200,     80,   200}
1501 ,{ 200,     60,   200,     10,   200}
1502 ,{ 180,   110,   180,    -10,   180}
1503 ,{ 200,     60,   200,     80,   200}
1504 ,{ 170,     90,   170,     20,   170}
1505 }
1506 ,{{ 160,     20,   160,      0,   160}
1507 ,{ 160,     20,   160,    -30,   160}
1508 ,{ 150,     20,   150,    -40,   150}
1509 ,{  60,    -70,     60,      0,   60}
1510 ,{ 150,     20,   150,    -40,   150}
1511 }
1512 ,{{ 200,   110,   200,     10,   200}
1513 ,{ 200,     60,   200,     10,   200}
1514 ,{ 180,   110,   180,    -10,   180}
1515 ,{ 200,     60,   200,     10,   200}
1516 ,{ 170,     90,   170,    -20,   170}
1517 }
1518 ,{{ 150,     20,   150,     80,   150}
1519 ,{  60,    -70,     60,      0,   60}
1520 ,{ 150,     20,   150,    -40,   150}
1521 ,{  80,      0,     10,     80,   10}
1522 ,{ 150,     20,   150,    -40,   150}
1523 }
1524 ,{{ 200,     90,   200,     20,   200}
1525 ,{ 200,     60,   200,     10,   200}
1526 ,{ 170,     90,   170,    -20,   170}
1527 ,{ 200,     60,   200,     10,   200}
1528 ,{  80,    -50,     80,     20,   80}
1529 }
1530 }
1531 ,{{{{ 170,   150,   170,   150,   100}
1532 ,{ 170,   150,   170,   150,   100}
1533 ,{ 150,   140,   150,   140,   60}
1534 ,{ 170,   150,   170,   150,   80}
1535 ,{ 140,   120,   140,   120,   50}
1536 }
1537 ,{{ 130,   110,   130,   110,   100}
1538 ,{ 130,   110,   130,   110,   100}
1539 ,{ 120,   110,   120,   110,   30}
1540 ,{  90,     20,     90,     20,  -50}
1541 ,{ 120,   110,   120,   110,   30}
1542 }
1543 ,{{ 170,   150,   170,   150,   80}
1544 ,{ 170,   150,   170,   150,   80}
1545 ,{ 150,   140,   150,   140,   60}
1546 ,{ 170,   150,   170,   150,   80}
1547 ,{ 140,   120,   140,   120,   50}
1548 }
1549 ,{{ 120,   110,   120,   110,   30}
1550 ,{  90,     20,     90,     20,  -50}
1551 ,{ 120,   110,   120,   110,   30}
1552 ,{  20,    -40,    -20,    -40,   20}
1553 ,{ 120,   110,   120,   110,   30}
1554 }
1555 ,{{ 170,   150,   170,   150,   80}
1556 ,{ 170,   150,   170,   150,   80}
1557 ,{ 140,   120,   140,   120,   50}
1558 ,{ 170,   150,   170,   150,   80}
1559 ,{  50,     30,     50,     30,  -40}
1560 }
1561 }
1562 }
1563 {{{{{ 220,   150,   220,   140,   170}
1564 ,{ 220,   130,   220,   130,   170}
1565 ,{ 150,   110,   150,   110,   150}
1566 ,{ 140,   100,   140,   100,   140}
1567 ,{ 170,   150,   150,   140,   170}
1568 }
1569 ,{{ 220,   130,   220,   130,   170}
1570 ,{ 220,   130,   220,   130,   170}
1571 ,{ 150,   110,   150,   100,   150}
1572 ,{  70,    -30,     70,    -70,   50}
1573 ,{ 150,   110,   150,   100,   150}
1574 }
1575 ,{{ 190,   110,   190,   100,   170}
1576 ,{ 190,   110,   190,   100,   140}
1577 ,{ 150,   110,   150,   100,   150}
1578 ,{ 140,   100,   140,   100,   140}
1579 ,{ 170,   110,   150,   100,   170}
1580 }
1581 ,{{ 150,   110,   150,   100,   150}
1582 ,{ 140,     70,     70,    -10,   140}
1583 ,{ 150,   110,   150,   100,   150}
1584 ,{  80,    -30,     10,     80,   70}
1585 ,{ 150,   110,   150,   100,   150}
```

```
1586      }
1587 ,{{ 150,   150,   150,   140,   150}
1588 ,{ 140,   100,   140,   100,   140}
1589 ,{ 150,   110,   150,   110,   150}
1590 ,{ 140,   100,   140,   100,   140}
1591 ,{ 150,   150,    70,   140,    70}
1592 }
1593 }
1594 ,{{{ 170,   150,   150,    90,   170}
1595 ,{ 170,   130,   140,    10,   170}
1596 ,{ 150,   110,   150,    80,   150}
1597 ,{ 140,   100,   140,    10,   140}
1598 ,{ 150,   150,   150,    90,   150}
1599 }
1600 ,{{ 170,   130,   150,    10,   170}
1601 ,{ 170,   130,     60,     0,   170}
1602 ,{ 150,   110,   150,   -70,   150}
1603 ,{ 10,   -30,     10,  -160,   -30}
1604 ,{ 150,   110,   150,    10,   150}
1605 }
1606 ,{{ 150,   110,   150,    70,   150}
1607 ,{ 140,   100,     50,  -100,   140}
1608 ,{ 150,   110,   150,   -60,   150}
1609 ,{ 140,   100,   140,    10,   140}
1610 ,{ 150,   110,   150,    70,   150}
1611 }
1612 ,{{ 150,   110,   150,    10,   150}
1613 ,{ 40,     40,     30,   -70,    30}
1614 ,{ 150,   110,   150,    10,   150}
1615 ,{ 10,   -30,   -30,     0,   10}
1616 ,{ 150,   110,   150,    10,   150}
1617 }
1618 ,{{ 150,   150,   150,    90,   150}
1619 ,{ 140,   100,   140,    10,   140}
1620 ,{ 150,   110,   150,    80,   150}
1621 ,{ 140,   100,   140,    10,   140}
1622 ,{ 150,   150,     0,    90,    70}
1623 }
1624 }
1625 ,{{{ 220,   130,   220,   130,   170}
1626 ,{ 220,   130,   220,   130,   140}
1627 ,{ 140,   110,   140,   110,   120}
1628 ,{ 130,   100,   130,   100,   110}
1629 ,{ 170,   100,   130,   100,   170}
1630 }
1631 ,{{ 220,   130,   220,   130,   140}
1632 ,{ 220,   130,   220,   130,   140}
1633 ,{ 130,   100,   130,   100,   120}
1634 ,{ 70,   -70,     70,   -70,     0}
1635 ,{ 130,   100,   130,   100,   120}
1636 }
1637 ,{{ 190,   110,   190,   100,   170}
1638 ,{ 190,   110,   190,   100,   110}
1639 ,{ 130,   100,   130,   100,   120}
1640 ,{ 130,   100,   130,   100,   110}
1641 ,{ 170,   100,   130,   100,   170}
1642 }
1643 ,{{ 130,   100,   130,   100,   120}
1644 ,{ 70,     70,     70,   -10,    60}
1645 ,{ 130,   100,   130,   100,   120}
1646 ,{ 20,   -40,   -10,   -40,    20}
1647 ,{ 130,   100,   130,   100,   120}
1648 }
1649 ,{{ 140,   110,   140,   110,   120}
1650 ,{ 130,   100,   130,   100,   110}
1651 ,{ 140,   110,   140,   110,   120}
1652 ,{ 130,   100,   130,   100,   110}
1653 ,{ 30,   -20,   -10,    30,    20}
1654 }
1655 }
1656 ,{{{ 170,     90,   170,   140,   170}
1657 ,{ 170,     70,   170,   -10,   170}
1658 ,{ 150,     80,   150,   -40,   150}
1659 ,{ 140,     10,   140,    80,   140}
1660 ,{ 150,     90,   150,   140,   150}
1661 }
1662 ,{{ 170,     10,   170,   -10,   170}
1663 ,{ 170,   -20,   170,   -10,   170}
1664 ,{ 150,   -40,   150,   -40,   150}
1665 ,{ -30,  -170,   -30,   -90,   -30}
1666 ,{ 150,     10,   150,   -40,   150}
1667 }
1668 ,{{ 150,     70,   150,    20,   150}
1669 ,{ 140,     70,   140,   -50,   140}
1670 ,{ 150,     70,   150,   -40,   150}
1671 ,{ 140,     10,   140,   -50,   140}
1672 ,{ 150,     70,   150,    20,   150}
```

```
1673    }
1674 ,{{ 150,   10,   150,    80,   150}
1675 ,{ 30, -50,   30,   -30,   30}
1676 ,{ 150,   10,   150,   -40,   150}
1677 ,{ 80, -30,   10,    80,   10}
1678 ,{ 150,   10,   150,   -40,   150}
1679 }
1680 ,{{ 150,    90,   150,   140,   150}
1681 ,{ 140,   10,   140,   -50,   140}
1682 ,{ 150,   80,   150,   -50,   150}
1683 ,{ 140,   10,   140,   -50,   140}
1684 ,{ 140,   90,    70,   140,    70}
1685 }
1686 }
1687 ,{{{ 140,   130,   140,   130,   140}
1688 ,{ 140,   130,   140,   130,   140}
1689 ,{ 120,   110,   120,   110,   30}
1690 ,{ 110,   100,   110,   100,   70}
1691 ,{ 120,   100,   120,   100,   30}
1692 }
1693 ,{{ 140,   130,   140,   130,   140}
1694 ,{ 140,   130,   140,   130,   140}
1695 ,{ 120,   100,   120,   100,   30}
1696 ,{ 50,   -70,    0,   -70,   50}
1697 ,{ 120,   100,   120,   100,   30}
1698 }
1699 ,{{ 120,   100,   120,   100,   30}
1700 ,{ 110,   100,   110,   100,   30}
1701 ,{ 120,   100,   120,   100,   30}
1702 ,{ 110,   100,   110,   100,   20}
1703 ,{ 120,   100,   120,   100,   30}
1704 }
1705 ,{{ 140,   100,   120,   100,   140}
1706 ,{ 140,   -10,    50,   -10,   140}
1707 ,{ 120,   100,   120,   100,   30}
1708 ,{ 70,   -40,   -60,   -40,   70}
1709 ,{ 120,   100,   120,   100,   30}
1710 }
1711 ,{{ 120,   110,   120,   110,   30}
1712 ,{ 110,   100,   110,   100,   20}
1713 ,{ 120,   110,   120,   110,   30}
1714 ,{ 110,   100,   110,   100,   20}
1715 ,{ 40,    30,    40,    30,   -60}
1716 }
1717 }
1718 }
1719 ,{{{ 300,   290,   300,   260,   300}
1720 ,{ 300,   270,   300,   260,   300}
1721 ,{ 270,   230,   270,   220,   270}
1722 ,{ 270,   230,   270,   220,   270}
1723 ,{ 290,   290,   270,   220,   270}
1724 }
1725 ,{{ 300,   270,   300,   260,   300}
1726 ,{ 300,   270,   300,   260,   300}
1727 ,{ 270,   230,   270,   220,   270}
1728 ,{ 230,   150,   230,   140,   220}
1729 ,{ 270,   230,   270,   220,   270}
1730 }
1731 ,{{ 270,   230,   270,   220,   270}
1732 ,{ 270,   230,   270,   220,   270}
1733 ,{ 270,   230,   270,   220,   270}
1734 ,{ 270,   230,   270,   220,   270}
1735 ,{ 270,   230,   270,   220,   270}
1736 }
1737 ,{{ 270,   230,   270,   220,   270}
1738 ,{ 270,   190,   270,   180,   260}
1739 ,{ 270,   230,   270,   220,   270}
1740 ,{ 210,   130,   140,   210,   150}
1741 ,{ 270,   230,   270,   220,   270}
1742 }
1743 ,{{ 290,   290,   270,   220,   270}
1744 ,{ 270,   230,   270,   220,   270}
1745 ,{ 270,   230,   270,   220,   270}
1746 ,{ 270,   230,   270,   220,   270}
1747 ,{ 290,   290,   270,   220,   270}
1748 }
1749 }
1750 ,{{{ 300,   290,   300,   190,   300}
1751 ,{ 300,   270,   300,   170,   300}
1752 ,{ 270,   230,   270,   190,   270}
1753 ,{ 270,   230,   270,   130,   270}
1754 ,{ 290,   290,   270,   190,   270}
1755 }
1756 ,{{ 300,   270,   300,   170,   300}
1757 ,{ 300,   270,   300,   170,   300}
1758 ,{ 270,   230,   270,   130,   270}
1759 ,{ 190,   150,   190,    50,   190}}
```

```
1760 , { 270, 230, 270, 130, 270}
1761 }
1762 , {{ 270, 230, 270, 190, 270}
1763 , { 270, 230, 270, 130, 270}
1764 , { 270, 230, 270, 190, 270}
1765 , { 270, 230, 270, 130, 270}
1766 , { 270, 230, 270, 190, 270}
1767 }
1768 , {{ 270, 230, 270, 130, 270}
1769 , { 230, 190, 230, 90, 230}
1770 , { 270, 230, 270, 130, 270}
1771 , { 140, 100, 140, 130, 140}
1772 , { 270, 230, 270, 130, 270}
1773 }
1774 , {{ 290, 290, 270, 190, 270}
1775 , { 270, 230, 270, 130, 270}
1776 , { 270, 230, 270, 190, 270}
1777 , { 270, 230, 270, 130, 270}
1778 , { 290, 290, 270, 130, 270}
1779 }
1780 }
1781 , {{{ 290, 260, 290, 260, 270}
1782 , { 290, 260, 290, 260, 270}
1783 , { 250, 220, 250, 220, 240}
1784 , { 250, 220, 250, 220, 240}
1785 , { 250, 220, 250, 220, 240}
1786 }
1787 , {{ 290, 260, 290, 260, 270}
1788 , { 290, 260, 290, 260, 270}
1789 , { 250, 220, 250, 220, 240}
1790 , { 230, 140, 230, 140, 220}
1791 , { 250, 220, 250, 220, 240}
1792 }
1793 , {{ 250, 220, 250, 220, 240}
1794 , { 250, 220, 250, 220, 240}
1795 , { 250, 220, 250, 220, 240}
1796 , { 250, 220, 250, 220, 240}
1797 , { 250, 220, 250, 220, 240}
1798 }
1799 , {{ 270, 220, 270, 220, 260}
1800 , { 270, 180, 270, 180, 260}
1801 , { 250, 220, 250, 220, 240}
1802 , { 120, 90, 120, 90, 110}
1803 , { 250, 220, 250, 220, 240}
1804 }
1805 , {{ 250, 220, 250, 220, 240}
1806 , { 250, 220, 250, 220, 240}
1807 , { 250, 220, 250, 220, 240}
1808 , { 250, 220, 250, 220, 240}
1809 , { 250, 220, 250, 220, 240}
1810 }
1811 }
1812 , {{{ 300, 190, 300, 210, 300}
1813 , { 300, 170, 300, 170, 300}
1814 , { 270, 190, 270, 80, 270}
1815 , { 270, 130, 270, 210, 270}
1816 , { 270, 190, 270, 210, 270}
1817 }
1818 , {{ 300, 170, 300, 130, 300}
1819 , { 300, 170, 300, 110, 300}
1820 , { 270, 130, 270, 80, 270}
1821 , { 190, 50, 190, 130, 190}
1822 , { 270, 130, 270, 80, 270}
1823 }
1824 , {{ 270, 190, 270, 80, 270}
1825 , { 270, 130, 270, 80, 270}
1826 , { 270, 190, 270, 80, 270}
1827 , { 270, 130, 270, 80, 270}
1828 , { 270, 190, 270, 80, 270}
1829 }
1830 , {{ 270, 130, 270, 210, 270}
1831 , { 230, 90, 230, 170, 230}
1832 , { 270, 130, 270, 80, 270}
1833 , { 210, 130, 140, 210, 140}
1834 , { 270, 130, 270, 80, 270}
1835 }
1836 , {{ 270, 190, 270, 210, 270}
1837 , { 270, 130, 270, 80, 270}
1838 , { 270, 190, 270, 80, 270}
1839 , { 270, 130, 270, 80, 270}
1840 , { 270, 130, 270, 210, 270}
1841 }
1842 }
1843 , {{{ 270, 260, 270, 260, 240}
1844 , { 270, 260, 270, 260, 240}
1845 , { 240, 220, 240, 220, 150}
1846 , { 240, 220, 240, 220, 150}
```

```
1847 , { 240, 220, 240, 220, 150}
1848 }
1849 , {{ 270, 260, 270, 260, 240}
1850 , { 270, 260, 270, 260, 240}
1851 , { 240, 220, 240, 220, 150}
1852 , { 220, 140, 220, 140, 70}
1853 , { 240, 220, 240, 220, 150}
1854 }
1855 , {{ 240, 220, 240, 220, 150}
1856 , { 240, 220, 240, 220, 150}
1857 , { 240, 220, 240, 220, 150}
1858 , { 240, 220, 240, 220, 150}
1859 , { 240, 220, 240, 220, 150}
1860 }
1861 , {{ 260, 220, 260, 220, 150}
1862 , { 260, 180, 260, 180, 110}
1863 , { 240, 220, 240, 220, 150}
1864 , { 150, 90, 110, 90, 150}
1865 , { 240, 220, 240, 220, 150}
1866 }
1867 , {{ 240, 220, 240, 220, 150}
1868 , { 240, 220, 240, 220, 150}
1869 , { 240, 220, 240, 220, 150}
1870 , { 240, 220, 240, 220, 150}
1871 , { 240, 220, 240, 220, 150}
1872 }
1873 }
1874 }
1875 , {{{ 310, 260, 310, 220, 300}
1876 , { 310, 230, 310, 220, 300}
1877 , { 240, 200, 240, 190, 240}
1878 , { 240, 200, 240, 190, 240}
1879 , { 260, 260, 240, 190, 240}
1880 }
1881 , {{ 240, 200, 240, 190, 240}
1882 , { 200, 160, 200, 160, 200}
1883 , { 240, 200, 240, 190, 240}
1884 , { 150, 60, 150, 60, 130}
1885 , { 240, 200, 240, 190, 240}
1886 }
1887 , {{ 240, 200, 240, 190, 240}
1888 , { 240, 200, 240, 190, 240}
1889 , { 240, 200, 240, 190, 240}
1890 , { 240, 200, 240, 190, 240}
1891 , { 240, 200, 240, 190, 240}
1892 }
1893 , {{ 310, 230, 310, 220, 300}
1894 , { 310, 230, 310, 220, 300}
1895 , { 240, 200, 240, 190, 240}
1896 , { 180, 100, 110, 180, 120}
1897 , { 240, 200, 240, 190, 240}
1898 }
1899 , {{ 260, 260, 240, 190, 240}
1900 , { 240, 200, 240, 190, 240}
1901 , { 240, 200, 240, 190, 240}
1902 , { 240, 200, 240, 190, 240}
1903 , { 260, 260, 240, 190, 240}
1904 }
1905 }
1906 , {{{ 270, 260, 270, 160, 270}
1907 , { 270, 230, 270, 130, 270}
1908 , { 240, 200, 240, 160, 240}
1909 , { 240, 200, 240, 100, 240}
1910 , { 260, 260, 240, 160, 240}
1911 }
1912 , {{ 240, 200, 240, 100, 240}
1913 , { 200, 160, 200, 70, 200}
1914 , { 240, 200, 240, 100, 240}
1915 , { 100, 60, 100, -30, 100}
1916 , { 240, 200, 240, 100, 240}
1917 }
1918 , {{ 240, 200, 240, 160, 240}
1919 , { 240, 200, 240, 100, 240}
1920 , { 240, 200, 240, 160, 240}
1921 , { 240, 200, 240, 100, 240}
1922 , { 240, 200, 240, 160, 240}
1923 }
1924 , {{ 270, 230, 270, 130, 270}
1925 , { 270, 230, 270, 130, 270}
1926 , { 240, 200, 240, 100, 240}
1927 , { 110, 70, 110, 100, 110}
1928 , { 240, 200, 240, 100, 240}
1929 }
1930 , {{ 260, 260, 240, 160, 240}
1931 , { 240, 200, 240, 100, 240}
1932 , { 240, 200, 240, 160, 240}
1933 , { 240, 200, 240, 100, 240}
```

```
1934 , { 260, 260, 240, 100, 240}
1935 }
1936 }
1937 , {{ 310, 220, 310, 220, 300}
1938 , { 310, 220, 310, 220, 300}
1939 , { 220, 190, 220, 190, 210}
1940 , { 220, 190, 220, 190, 210}
1941 , { 220, 190, 220, 190, 210}
1942 }
1943 , {{ 220, 190, 220, 190, 210}
1944 , { 190, 160, 190, 160, 170}
1945 , { 220, 190, 220, 190, 210}
1946 , { 150, 60, 150, 60, 130}
1947 , { 220, 190, 220, 190, 210}
1948 }
1949 , {{ 220, 190, 220, 190, 210}
1950 , { 220, 190, 220, 190, 210}
1951 , { 220, 190, 220, 190, 210}
1952 , { 220, 190, 220, 190, 210}
1953 , { 220, 190, 220, 190, 210}
1954 }
1955 , {{ 310, 220, 310, 220, 300}
1956 , { 310, 220, 310, 220, 300}
1957 , { 220, 190, 220, 190, 210}
1958 , { 90, 60, 90, 60, 80}
1959 , { 220, 190, 220, 190, 210}
1960 }
1961 , {{ 220, 190, 220, 190, 210}
1962 , { 220, 190, 220, 190, 210}
1963 , { 220, 190, 220, 190, 210}
1964 , { 220, 190, 220, 190, 210}
1965 , { 220, 190, 220, 190, 210}
1966 }
1967 }
1968 , {{ 270, 160, 270, 210, 270}
1969 , { 270, 130, 270, 210, 270}
1970 , { 240, 160, 240, 50, 240}
1971 , { 240, 100, 240, 180, 240}
1972 , { 240, 160, 240, 180, 240}
1973 }
1974 , {{ 240, 100, 240, 50, 240}
1975 , { 200, 70, 200, 10, 200}
1976 , { 240, 100, 240, 50, 240}
1977 , { 100, -30, 100, 40, 100}
1978 , { 240, 100, 240, 50, 240}
1979 }
1980 , {{ 240, 160, 240, 50, 240}
1981 , { 240, 100, 240, 50, 240}
1982 , { 240, 160, 240, 50, 240}
1983 , { 240, 100, 240, 50, 240}
1984 , { 240, 160, 240, 50, 240}
1985 }
1986 , {{ 270, 130, 270, 210, 270}
1987 , { 270, 130, 270, 210, 270}
1988 , { 240, 100, 240, 50, 240}
1989 , { 180, 100, 110, 180, 110}
1990 , { 240, 100, 240, 50, 240}
1991 }
1992 , {{ 240, 160, 240, 180, 240}
1993 , { 240, 100, 240, 50, 240}
1994 , { 240, 160, 240, 50, 240}
1995 , { 240, 100, 240, 50, 240}
1996 , { 240, 100, 240, 180, 240}
1997 }
1998 }
1999 , {{ 300, 220, 300, 220, 150}
2000 , { 300, 220, 300, 220, 150}
2001 , { 210, 190, 210, 190, 120}
2002 , { 210, 190, 210, 190, 120}
2003 , { 210, 190, 210, 190, 120}
2004 }
2005 , {{ 210, 190, 210, 190, 140}
2006 , { 170, 160, 170, 160, 140}
2007 , { 210, 190, 210, 190, 120}
2008 , { 130, 60, 130, 60, -10}
2009 , { 210, 190, 210, 190, 120}
2010 }
2011 , {{ 210, 190, 210, 190, 120}
2012 , { 210, 190, 210, 190, 120}
2013 , { 210, 190, 210, 190, 120}
2014 , { 210, 190, 210, 190, 120}
2015 , { 210, 190, 210, 190, 120}
2016 }
2017 , {{ 300, 220, 300, 220, 150}
2018 , { 300, 220, 300, 220, 150}
2019 , { 210, 190, 210, 190, 120}
2020 , { 120, 60, 80, 60, 120}
```

```
2021 , { 210, 190, 210, 190, 120}
2022 }
2023 , {{ 210, 190, 210, 190, 120}
2024 , { 210, 190, 210, 190, 120}
2025 , { 210, 190, 210, 190, 120}
2026 , { 210, 190, 210, 190, 120}
2027 , { 210, 190, 210, 190, 120}
2028 }
2029 }
2030 }
2031 , {{{{ 240, 200, 240, 190, 240}
2032 , { 240, 200, 240, 190, 240}
2033 , { 220, 180, 220, 170, 220}
2034 , { 220, 180, 220, 180, 220}
2035 , { 220, 180, 220, 170, 220}
2036 }
2037 , {{ 240, 200, 240, 190, 240}
2038 , { 240, 200, 240, 190, 240}
2039 , { 210, 170, 210, 170, 210}
2040 , { 160, 70, 160, 70, 140}
2041 , { 210, 170, 210, 170, 210}
2042 }
2043 , {{ 220, 180, 220, 180, 220}
2044 , { 220, 180, 220, 180, 220}
2045 , { 220, 180, 220, 170, 220}
2046 , { 220, 180, 220, 180, 220}
2047 , { 220, 180, 220, 170, 220}
2048 }
2049 , {{ 230, 170, 230, 170, 210}
2050 , { 230, 140, 230, 140, 210}
2051 , { 210, 170, 210, 170, 210}
2052 , { 130, 60, 60, 130, 70}
2053 , { 210, 170, 210, 170, 210}
2054 }
2055 , {{ 220, 180, 220, 180, 220}
2056 , { 220, 180, 220, 180, 220}
2057 , { 220, 180, 220, 170, 220}
2058 , { 220, 180, 220, 180, 220}
2059 , { 150, 150, 130, 80, 130}
2060 }
2061 }
2062 , {{{{ 240, 200, 240, 140, 240}
2063 , { 240, 200, 240, 100, 240}
2064 , { 220, 180, 220, 140, 220}
2065 , { 220, 180, 220, 90, 220}
2066 , { 220, 180, 220, 140, 220}
2067 }
2068 , {{ 240, 200, 240, 100, 240}
2069 , { 240, 200, 240, 100, 240}
2070 , { 210, 170, 210, 80, 210}
2071 , { 110, 70, 110, -20, 110}
2072 , { 210, 170, 210, 80, 210}
2073 }
2074 , {{ 220, 180, 220, 140, 220}
2075 , { 220, 180, 220, 90, 220}
2076 , { 220, 180, 220, 140, 220}
2077 , { 220, 180, 220, 90, 220}
2078 , { 220, 180, 220, 140, 220}
2079 }
2080 , {{ 210, 170, 210, 80, 210}
2081 , { 180, 140, 180, 50, 180}
2082 , { 210, 170, 210, 80, 210}
2083 , { 60, 20, 60, 60, 60}
2084 , { 210, 170, 210, 80, 210}
2085 }
2086 , {{ 220, 180, 220, 140, 220}
2087 , { 220, 180, 220, 90, 220}
2088 , { 220, 180, 220, 140, 220}
2089 , { 220, 180, 220, 90, 220}
2090 , { 150, 150, 130, 0, 130}
2091 }
2092 }
2093 , {{{{ 230, 190, 230, 190, 210}
2094 , { 230, 190, 230, 190, 210}
2095 , { 200, 170, 200, 170, 190}
2096 , { 210, 180, 210, 180, 190}
2097 , { 200, 170, 200, 170, 190}
2098 }
2099 , {{ 220, 190, 220, 190, 210}
2100 , { 220, 190, 220, 190, 210}
2101 , { 200, 170, 200, 170, 180}
2102 , { 160, 70, 160, 70, 140}
2103 , { 200, 170, 200, 170, 180}
2104 }
2105 , {{ 210, 180, 210, 180, 190}
2106 , { 210, 180, 210, 180, 190}
2107 , { 200, 170, 200, 170, 190}
```

```
2108 , { 210, 180, 210, 180, 190}
2109 , { 200, 170, 200, 170, 190}
2110 }
2111 , {{ 230, 170, 230, 170, 210}
2112 , { 230, 140, 230, 140, 210}
2113 , { 200, 170, 200, 170, 180}
2114 , { 50, 20, 50, 20, 30}
2115 , { 200, 170, 200, 170, 180}
2116 }
2117 , {{ 210, 180, 210, 180, 190}
2118 , { 210, 180, 210, 180, 190}
2119 , { 200, 170, 200, 170, 190}
2120 , { 210, 180, 210, 180, 190}
2121 , { 110, 80, 110, 80, 100}
2122 }
2123 }
2124 , {{{ 240, 140, 240, 130, 240}
2125 , { 240, 100, 240, 120, 240}
2126 , { 220, 140, 220, 30, 220}
2127 , { 220, 90, 220, 130, 220}
2128 , { 220, 140, 220, 70, 220}
2129 }
2130 , {{ 240, 100, 240, 50, 240}
2131 , { 240, 100, 240, 50, 240}
2132 , { 210, 80, 210, 20, 210}
2133 , { 110, -20, 110, 50, 110}
2134 , { 210, 80, 210, 20, 210}
2135 }
2136 , {{ 220, 140, 220, 30, 220}
2137 , { 220, 90, 220, 30, 220}
2138 , { 220, 140, 220, 30, 220}
2139 , { 220, 90, 220, 30, 220}
2140 , { 220, 140, 220, 30, 220}
2141 }
2142 , {{ 210, 80, 210, 130, 210}
2143 , { 180, 50, 180, 120, 180}
2144 , { 210, 80, 210, 20, 210}
2145 , { 130, 60, 60, 130, 60}
2146 , { 210, 80, 210, 20, 210}
2147 }
2148 , {{ 220, 140, 220, 70, 220}
2149 , { 220, 90, 220, 30, 220}
2150 , { 220, 140, 220, 30, 220}
2151 , { 220, 90, 220, 30, 220}
2152 , { 130, 0, 130, 70, 130}
2153 }
2154 }
2155 , {{{ 210, 190, 210, 190, 180}
2156 , { 210, 190, 210, 190, 180}
2157 , { 190, 170, 190, 170, 100}
2158 , { 190, 180, 190, 180, 100}
2159 , { 190, 170, 190, 170, 100}
2160 }
2161 , {{ 210, 190, 210, 190, 180}
2162 , { 210, 190, 210, 190, 180}
2163 , { 180, 170, 180, 170, 90}
2164 , { 140, 70, 140, 70, 0}
2165 , { 180, 170, 180, 170, 90}
2166 }
2167 , {{ 190, 180, 190, 180, 100}
2168 , { 190, 180, 190, 180, 100}
2169 , { 190, 170, 190, 170, 100}
2170 , { 190, 180, 190, 180, 100}
2171 , { 190, 170, 190, 170, 100}
2172 }
2173 , {{ 210, 170, 210, 170, 90}
2174 , { 210, 140, 210, 140, 60}
2175 , { 180, 170, 180, 170, 90}
2176 , { 70, 20, 30, 20, 70}
2177 , { 180, 170, 180, 170, 90}
2178 }
2179 , {{ 190, 180, 190, 180, 100}
2180 , { 190, 180, 190, 180, 100}
2181 , { 190, 170, 190, 170, 100}
2182 , { 190, 180, 190, 180, 100}
2183 , { 100, 80, 100, 80, 10}
2184 }
2185 }
2186 }
2187 , {{{{ 240, 200, 240, 190, 240}
2188 , { 240, 200, 240, 190, 240}
2189 , { 240, 200, 240, 190, 240}
2190 , { 240, 200, 240, 190, 240}
2191 , { 240, 200, 240, 190, 240}
2192 }
2193 , {{ 240, 200, 240, 190, 240}
2194 , { 240, 200, 240, 190, 240}
```

```
2195 , { 190, 150, 190, 150, 190}
2196 , { 180, 90, 180, 90, 160}
2197 , { 190, 150, 190, 150, 190}
2198 }
2199 , {{ 240, 200, 240, 190, 240}
2200 , { 240, 200, 240, 190, 240}
2201 , { 240, 200, 240, 190, 240}
2202 , { 240, 200, 240, 190, 240}
2203 , { 240, 200, 240, 190, 240}
2204 }
2205 , {{ 190, 150, 190, 150, 190}
2206 , { 190, 100, 190, 100, 170}
2207 , { 190, 150, 190, 150, 190}
2208 , { 150, 80, 80, 150, 90}
2209 , { 190, 150, 190, 150, 190}
2210 }
2211 , {{ 240, 200, 240, 190, 240}
2212 , { 240, 200, 240, 190, 240}
2213 , { 210, 170, 210, 160, 210}
2214 , { 240, 200, 240, 190, 240}
2215 , { 170, 170, 150, 110, 150}
2216 }
2217 }
2218 , {{{ 240, 200, 240, 160, 240}
2219 , { 240, 200, 240, 100, 240}
2220 , { 240, 200, 240, 160, 240}
2221 , { 240, 200, 240, 100, 240}
2222 , { 240, 200, 240, 160, 240}
2223 }
2224 , {{ 240, 200, 240, 100, 240}
2225 , { 240, 200, 240, 100, 240}
2226 , { 190, 150, 190, 60, 190}
2227 , { 130, 90, 130, 0, 130}
2228 , { 190, 150, 190, 60, 190}
2229 }
2230 , {{ 240, 200, 240, 160, 240}
2231 , { 240, 200, 240, 100, 240}
2232 , { 240, 200, 240, 160, 240}
2233 , { 240, 200, 240, 100, 240}
2234 , { 240, 200, 240, 160, 240}
2235 }
2236 , {{ 190, 150, 190, 80, 190}
2237 , { 140, 100, 140, 10, 140}
2238 , { 190, 150, 190, 60, 190}
2239 , { 80, 40, 80, 80, 80}
2240 , { 190, 150, 190, 60, 190}
2241 }
2242 , {{ 240, 200, 240, 130, 240}
2243 , { 240, 200, 240, 100, 240}
2244 , { 210, 170, 210, 130, 210}
2245 , { 240, 200, 240, 100, 240}
2246 , { 170, 170, 150, 20, 150}
2247 }
2248 }
2249 , {{{ 220, 190, 220, 190, 210}
2250 , { 220, 190, 220, 190, 210}
2251 , { 220, 190, 220, 190, 210}
2252 , { 220, 190, 220, 190, 210}
2253 , { 220, 190, 220, 190, 210}
2254 }
2255 , {{ 220, 190, 220, 190, 210}
2256 , { 220, 190, 220, 190, 210}
2257 , { 180, 150, 180, 150, 160}
2258 , { 180, 90, 180, 90, 160}
2259 , { 180, 150, 180, 150, 160}
2260 }
2261 , {{ 220, 190, 220, 190, 210}
2262 , { 220, 190, 220, 190, 210}
2263 , { 220, 190, 220, 190, 210}
2264 , { 220, 190, 220, 190, 210}
2265 , { 220, 190, 220, 190, 210}
2266 }
2267 , {{ 190, 150, 190, 150, 170}
2268 , { 190, 100, 190, 100, 170}
2269 , { 180, 150, 180, 150, 160}
2270 , { 70, 40, 70, 40, 50}
2271 , { 180, 150, 180, 150, 160}
2272 }
2273 , {{ 220, 190, 220, 190, 210}
2274 , { 220, 190, 220, 190, 210}
2275 , { 190, 160, 190, 160, 180}
2276 , { 220, 190, 220, 190, 210}
2277 , { 140, 110, 140, 110, 120}
2278 }
2279 }
2280 , {{{ 240, 160, 240, 150, 240}
2281 , { 240, 100, 240, 80, 240}
```

```
2282 , { 240, 160, 240, 50, 240}
2283 , { 240, 100, 240, 150, 240}
2284 , { 240, 160, 240, 90, 240}
2285 }
2286 , {{ 240, 100, 240, 70, 240}
2287 , { 240, 100, 240, 50, 240}
2288 , { 190, 60, 190, 0, 190}
2289 , { 130, 0, 130, 70, 130}
2290 , { 190, 60, 190, 0, 190}
2291 }
2292 , {{ 240, 160, 240, 50, 240}
2293 , { 240, 100, 240, 50, 240}
2294 , { 240, 160, 240, 50, 240}
2295 , { 240, 100, 240, 50, 240}
2296 , { 240, 160, 240, 50, 240}
2297 }
2298 , {{ 190, 80, 190, 150, 190}
2299 , { 140, 10, 140, 80, 140}
2300 , { 190, 60, 190, 0, 190}
2301 , { 150, 80, 80, 150, 80}
2302 , { 190, 60, 190, 0, 190}
2303 }
2304 , {{ 240, 130, 240, 90, 240}
2305 , { 240, 100, 240, 50, 240}
2306 , { 210, 130, 210, 20, 210}
2307 , { 240, 100, 240, 50, 240}
2308 , { 150, 20, 150, 90, 150}
2309 }
2310 }
2311 , {{{ 210, 190, 210, 190, 180}
2312 , { 210, 190, 210, 190, 180}
2313 , { 210, 190, 210, 190, 120}
2314 , { 210, 190, 210, 190, 120}
2315 , { 210, 190, 210, 190, 120}
2316 }
2317 , {{ 210, 190, 210, 190, 180}
2318 , { 210, 190, 210, 190, 180}
2319 , { 160, 150, 160, 150, 70}
2320 , { 160, 90, 160, 90, 10}
2321 , { 160, 150, 160, 150, 70}
2322 }
2323 , {{ 210, 190, 210, 190, 120}
2324 , { 210, 190, 210, 190, 120}
2325 , { 210, 190, 210, 190, 120}
2326 , { 210, 190, 210, 190, 120}
2327 , { 210, 190, 210, 190, 120}
2328 }
2329 , {{ 170, 150, 170, 150, 90}
2330 , { 170, 100, 170, 100, 20}
2331 , { 160, 150, 160, 150, 70}
2332 , { 90, 40, 50, 40, 90}
2333 , { 160, 150, 160, 150, 70}
2334 }
2335 , {{ 210, 190, 210, 190, 120}
2336 , { 210, 190, 210, 190, 120}
2337 , { 180, 160, 180, 160, 90}
2338 , { 210, 190, 210, 190, 120}
2339 , { 120, 110, 120, 110, 30}
2340 }
2341 }
2342 }
2343 , {{{{ 310, 290, 310, 260, 300}
2344 , { 310, 270, 310, 260, 300}
2345 , { 270, 230, 270, 220, 270}
2346 , { 270, 230, 270, 220, 270}
2347 , { 290, 290, 270, 220, 270}
2348 }
2349 , {{ 300, 270, 300, 260, 300}
2350 , { 300, 270, 300, 260, 300}
2351 , { 270, 230, 270, 220, 270}
2352 , { 230, 150, 230, 140, 220}
2353 , { 270, 230, 270, 220, 270}
2354 }
2355 , {{ 270, 230, 270, 220, 270}
2356 , { 270, 230, 270, 220, 270}
2357 , { 270, 230, 270, 220, 270}
2358 , { 270, 230, 270, 220, 270}
2359 , { 270, 230, 270, 220, 270}
2360 }
2361 , {{ 310, 230, 310, 220, 300}
2362 , { 310, 230, 310, 220, 300}
2363 , { 270, 230, 270, 220, 270}
2364 , { 210, 130, 140, 210, 150}
2365 , { 270, 230, 270, 220, 270}
2366 }
2367 , {{ 290, 290, 270, 220, 270}
2368 , { 270, 230, 270, 220, 270}
```

```
2369 , { 270, 230, 270, 220, 270}
2370 , { 270, 230, 270, 220, 270}
2371 , { 290, 290, 270, 220, 270}
2372 }
2373 }
2374 , {{ { 300, 290, 300, 190, 300}
2375 , { 300, 270, 300, 170, 300}
2376 , { 270, 230, 270, 190, 270}
2377 , { 270, 230, 270, 130, 270}
2378 , { 290, 290, 270, 190, 270}
2379 }
2380 , {{ { 300, 270, 300, 170, 300}
2381 , { 300, 270, 300, 170, 300}
2382 , { 270, 230, 270, 130, 270}
2383 , { 190, 150, 190, 50, 190}
2384 , { 270, 230, 270, 130, 270}
2385 }
2386 , {{ { 270, 230, 270, 190, 270}
2387 , { 270, 230, 270, 130, 270}
2388 , { 270, 230, 270, 190, 270}
2389 , { 270, 230, 270, 130, 270}
2390 , { 270, 230, 270, 190, 270}
2391 }
2392 , {{ { 270, 230, 270, 130, 270}
2393 , { 270, 230, 270, 130, 270}
2394 , { 270, 230, 270, 130, 270}
2395 , { 140, 100, 140, 130, 140}
2396 , { 270, 230, 270, 130, 270}
2397 }
2398 , {{ { 290, 290, 270, 190, 270}
2399 , { 270, 230, 270, 130, 270}
2400 , { 270, 230, 270, 190, 270}
2401 , { 270, 230, 270, 130, 270}
2402 , { 290, 290, 270, 130, 270}
2403 }
2404 }
2405 , {{ { 310, 260, 310, 260, 300}
2406 , { 310, 260, 310, 260, 300}
2407 , { 250, 220, 250, 220, 240}
2408 , { 250, 220, 250, 220, 240}
2409 , { 250, 220, 250, 220, 240}
2410 }
2411 , {{ { 290, 260, 290, 260, 270}
2412 , { 290, 260, 290, 260, 270}
2413 , { 250, 220, 250, 220, 240}
2414 , { 230, 140, 230, 140, 220}
2415 , { 250, 220, 250, 220, 240}
2416 }
2417 , {{ { 250, 220, 250, 220, 240}
2418 , { 250, 220, 250, 220, 240}
2419 , { 250, 220, 250, 220, 240}
2420 , { 250, 220, 250, 220, 240}
2421 , { 250, 220, 250, 220, 240}
2422 }
2423 , {{ { 310, 220, 310, 220, 300}
2424 , { 310, 220, 310, 220, 300}
2425 , { 250, 220, 250, 220, 240}
2426 , { 120, 90, 120, 90, 110}
2427 , { 250, 220, 250, 220, 240}
2428 }
2429 , {{ { 250, 220, 250, 220, 240}
2430 , { 250, 220, 250, 220, 240}
2431 , { 250, 220, 250, 220, 240}
2432 , { 250, 220, 250, 220, 240}
2433 , { 250, 220, 250, 220, 240}
2434 }
2435 }
2436 , {{ { 300, 190, 300, 210, 300}
2437 , { 300, 170, 300, 210, 300}
2438 , { 270, 190, 270, 80, 270}
2439 , { 270, 130, 270, 210, 270}
2440 , { 270, 190, 270, 210, 270}
2441 }
2442 , {{ { 300, 170, 300, 130, 300}
2443 , { 300, 170, 300, 110, 300}
2444 , { 270, 130, 270, 80, 270}
2445 , { 190, 50, 190, 130, 190}
2446 , { 270, 130, 270, 80, 270}
2447 }
2448 , {{ { 270, 190, 270, 80, 270}
2449 , { 270, 130, 270, 80, 270}
2450 , { 270, 190, 270, 80, 270}
2451 , { 270, 130, 270, 80, 270}
2452 , { 270, 190, 270, 80, 270}
2453 }
2454 , {{ { 270, 130, 270, 210, 270}
2455 , { 270, 130, 270, 210, 270}}
```

```
2456 , { 270, 130, 270, 80, 270}
2457 , { 210, 130, 140, 210, 140}
2458 , { 270, 130, 270, 80, 270}
2459 }
2460 , {{ 270, 190, 270, 210, 270}
2461 , { 270, 130, 270, 80, 270}
2462 , { 270, 190, 270, 80, 270}
2463 , { 270, 130, 270, 80, 270}
2464 , { 270, 130, 270, 210, 270}
2465 }
2466 }
2467 , {{{ 300, 260, 300, 260, 240}
2468 , { 300, 260, 300, 260, 240}
2469 , { 240, 220, 240, 220, 150}
2470 , { 240, 220, 240, 220, 150}
2471 , { 240, 220, 240, 220, 150}
2472 }
2473 , {{ 270, 260, 270, 260, 240}
2474 , { 270, 260, 270, 260, 240}
2475 , { 240, 220, 240, 220, 150}
2476 , { 220, 140, 220, 140, 70}
2477 , { 240, 220, 240, 220, 150}
2478 }
2479 , {{ 240, 220, 240, 220, 150}
2480 , { 240, 220, 240, 220, 150}
2481 , { 240, 220, 240, 220, 150}
2482 , { 240, 220, 240, 220, 150}
2483 , { 240, 220, 240, 220, 150}
2484 }
2485 , {{ 300, 220, 300, 220, 150}
2486 , { 300, 220, 300, 220, 150}
2487 , { 240, 220, 240, 220, 150}
2488 , { 150, 90, 110, 90, 150}
2489 , { 240, 220, 240, 220, 150}
2490 }
2491 , {{ 240, 220, 240, 220, 150}
2492 , { 240, 220, 240, 220, 150}
2493 , { 240, 220, 240, 220, 150}
2494 , { 240, 220, 240, 220, 150}
2495 , { 240, 220, 240, 220, 150}
2496 }
2497 }
2498 }
2499 }
2500 , {{{{{ INF, INF, INF, INF, INF
2501 , { INF, INF, INF, INF, INF
2502 , { INF, INF, INF, INF, INF
2503 , { INF, INF, INF, INF, INF
2504 , { INF, INF, INF, INF, INF
2505 }
2506 , {{ INF, INF, INF, INF, INF
2507 , { INF, INF, INF, INF, INF
2508 , { INF, INF, INF, INF, INF
2509 , { INF, INF, INF, INF, INF
2510 , { INF, INF, INF, INF, INF
2511 }
2512 , {{ INF, INF, INF, INF, INF
2513 , { INF, INF, INF, INF, INF
2514 , { INF, INF, INF, INF, INF
2515 , { INF, INF, INF, INF, INF
2516 , { INF, INF, INF, INF, INF
2517 }
2518 , {{ INF, INF, INF, INF, INF
2519 , { INF, INF, INF, INF, INF
2520 , { INF, INF, INF, INF, INF
2521 , { INF, INF, INF, INF, INF
2522 , { INF, INF, INF, INF, INF
2523 }
2524 , {{ INF, INF, INF, INF, INF
2525 , { INF, INF, INF, INF, INF
2526 , { INF, INF, INF, INF, INF
2527 , { INF, INF, INF, INF, INF
2528 , { INF, INF, INF, INF, INF
2529 }
2530 }
2531 , {{{ INF, INF, INF, INF, INF
2532 , { INF, INF, INF, INF, INF
2533 , { INF, INF, INF, INF, INF
2534 , { INF, INF, INF, INF, INF
2535 , { INF, INF, INF, INF, INF
2536 }
2537 , {{ INF, INF, INF, INF, INF
2538 , { INF, INF, INF, INF, INF
2539 , { INF, INF, INF, INF, INF
2540 , { INF, INF, INF, INF, INF
2541 , { INF, INF, INF, INF, INF
2542 }
```

```
2543 , {{ INF, INF, INF, INF, INF }
2544 , { INF, INF, INF, INF, INF }
2545 , { INF, INF, INF, INF, INF }
2546 , { INF, INF, INF, INF, INF }
2547 , { INF, INF, INF, INF, INF }
2548 }
2549 , {{ INF, INF, INF, INF, INF }
2550 , { INF, INF, INF, INF, INF }
2551 , { INF, INF, INF, INF, INF }
2552 , { INF, INF, INF, INF, INF }
2553 , { INF, INF, INF, INF, INF }
2554 }
2555 , {{ INF, INF, INF, INF, INF }
2556 , { INF, INF, INF, INF, INF }
2557 , { INF, INF, INF, INF, INF }
2558 , { INF, INF, INF, INF, INF }
2559 , { INF, INF, INF, INF, INF }
2560 }
2561 }
2562 , {{{ INF, INF, INF, INF, INF }
2563 , { INF, INF, INF, INF, INF }
2564 , { INF, INF, INF, INF, INF }
2565 , { INF, INF, INF, INF, INF }
2566 , { INF, INF, INF, INF, INF }
2567 }
2568 , {{ INF, INF, INF, INF, INF }
2569 , { INF, INF, INF, INF, INF }
2570 , { INF, INF, INF, INF, INF }
2571 , { INF, INF, INF, INF, INF }
2572 , { INF, INF, INF, INF, INF }
2573 }
2574 , {{ INF, INF, INF, INF, INF }
2575 , { INF, INF, INF, INF, INF }
2576 , { INF, INF, INF, INF, INF }
2577 , { INF, INF, INF, INF, INF }
2578 , { INF, INF, INF, INF, INF }
2579 }
2580 , {{ INF, INF, INF, INF, INF }
2581 , { INF, INF, INF, INF, INF }
2582 , { INF, INF, INF, INF, INF }
2583 , { INF, INF, INF, INF, INF }
2584 , { INF, INF, INF, INF, INF }
2585 }
2586 , {{ INF, INF, INF, INF, INF }
2587 , { INF, INF, INF, INF, INF }
2588 , { INF, INF, INF, INF, INF }
2589 , { INF, INF, INF, INF, INF }
2590 , { INF, INF, INF, INF, INF }
2591 }
2592 }
2593 , {{{ INF, INF, INF, INF, INF }
2594 , { INF, INF, INF, INF, INF }
2595 , { INF, INF, INF, INF, INF }
2596 , { INF, INF, INF, INF, INF }
2597 , { INF, INF, INF, INF, INF }
2598 }
2599 , {{ INF, INF, INF, INF, INF }
2600 , { INF, INF, INF, INF, INF }
2601 , { INF, INF, INF, INF, INF }
2602 , { INF, INF, INF, INF, INF }
2603 , { INF, INF, INF, INF, INF }
2604 }
2605 , {{ INF, INF, INF, INF, INF }
2606 , { INF, INF, INF, INF, INF }
2607 , { INF, INF, INF, INF, INF }
2608 , { INF, INF, INF, INF, INF }
2609 , { INF, INF, INF, INF, INF }
2610 }
2611 , {{ INF, INF, INF, INF, INF }
2612 , { INF, INF, INF, INF, INF }
2613 , { INF, INF, INF, INF, INF }
2614 , { INF, INF, INF, INF, INF }
2615 , { INF, INF, INF, INF, INF }
2616 }
2617 , {{ INF, INF, INF, INF, INF }
2618 , { INF, INF, INF, INF, INF }
2619 , { INF, INF, INF, INF, INF }
2620 , { INF, INF, INF, INF, INF }
2621 , { INF, INF, INF, INF, INF }
2622 }
2623 }
2624 , {{{ INF, INF, INF, INF, INF }
2625 , { INF, INF, INF, INF, INF }
2626 , { INF, INF, INF, INF, INF }
2627 , { INF, INF, INF, INF, INF }
2628 , { INF, INF, INF, INF, INF }
2629 }
```

```
2630 , {{ INF, INF, INF, INF, INF }
2631 , { INF, INF, INF, INF, INF }
2632 , { INF, INF, INF, INF, INF }
2633 , { INF, INF, INF, INF, INF }
2634 , { INF, INF, INF, INF, INF }
2635 }
2636 , {{ INF, INF, INF, INF, INF }
2637 , { INF, INF, INF, INF, INF }
2638 , { INF, INF, INF, INF, INF }
2639 , { INF, INF, INF, INF, INF }
2640 , { INF, INF, INF, INF, INF }
2641 }
2642 , {{ INF, INF, INF, INF, INF }
2643 , { INF, INF, INF, INF, INF }
2644 , { INF, INF, INF, INF, INF }
2645 , { INF, INF, INF, INF, INF }
2646 , { INF, INF, INF, INF, INF }
2647 }
2648 , {{ INF, INF, INF, INF, INF }
2649 , { INF, INF, INF, INF, INF }
2650 , { INF, INF, INF, INF, INF }
2651 , { INF, INF, INF, INF, INF }
2652 , { INF, INF, INF, INF, INF }
2653 }
2654 }
2655 }
2656 , {{{ 220, 220, 190, 150, 150 }
2657 , { 170, 170, 150, 150, 150 }
2658 , { 220, 220, 190, 130, 140 }
2659 , { 170, 170, 150, 150, 150 }
2660 , { 140, 140, 120, 140, 120 }
2661 }
2662 , {{ 150, 130, 110, 110, 150 }
2663 , { 150, 130, 110, 110, 150 }
2664 , { 130, 130, 110, 100, 110 }
2665 , { 90, 10, 70, 10, 90 }
2666 , { 130, 130, 100, 100, 110 }
2667 }
2668 , {{ 220, 220, 190, 150, 150 }
2669 , { 150, 150, 150, 150, 150 }
2670 , { 220, 220, 190, 130, 140 }
2671 , { 170, 170, 150, 150, 150 }
2672 , { 140, 140, 120, 120, 120 }
2673 }
2674 , {{ 140, 130, 100, 100, 140 }
2675 , { 90, 10, 70, 10, 90 }
2676 , { 130, 130, 100, 100, 110 }
2677 , { 140, -10, 20, 80, 140 }
2678 , { 130, 130, 100, 100, 110 }
2679 }
2680 , {{ 170, 170, 170, 150, 150 }
2681 , { 170, 170, 150, 150, 150 }
2682 , { 170, 140, 170, 120, 120 }
2683 , { 170, 170, 150, 150, 150 }
2684 , { 140, 140, 30, 140, 30 }
2685 }
2686 }
2687 , {{{ 220, 220, 190, 140, 140 }
2688 , { 170, 170, 140, 40, 140 }
2689 , { 220, 220, 190, 70, 130 }
2690 , { 170, 170, 140, 30, 140 }
2691 , { 140, 140, 110, 140, 110 }
2692 }
2693 , {{ 130, 130, 110, 70, 100 }
2694 , { 130, 130, 100, 40, 100 }
2695 , { 130, 130, 110, 70, 100 }
2696 , { 70, -20, 70, -50, 10 }
2697 , { 130, 130, 100, -10, 100 }
2698 }
2699 , {{ 220, 220, 190, 70, 140 }
2700 , { 140, 60, 50, 30, 140 }
2701 , { 220, 220, 190, 70, 130 }
2702 , { 170, 170, 140, 30, 140 }
2703 , { 140, 140, 110, 50, 110 }
2704 }
2705 , {{ 130, 130, 100, -10, 100 }
2706 , { 10, 0, -100, -70, 10 }
2707 , { 130, 130, 100, -10, 100 }
2708 , { -10, -10, -50, -30, -50 }
2709 , { 130, 130, 100, -10, 100 }
2710 }
2711 , {{ 170, 170, 140, 140, 140 }
2712 , { 170, 170, 140, 30, 140 }
2713 , { 140, 140, 110, 60, 110 }
2714 , { 170, 170, 140, 30, 140 }
2715 , { 140, 140, 30, 140, 20 }
2716 }
```

```
2717    }
2718 ,{{ 150, 150, 150, 150, 150}
2719 ,{ 150, 150, 150, 150, 150}
2720 ,{ 140, 130, 130, 130, 140}
2721 ,{ 150, 150, 150, 150, 150}
2722 ,{ 120, 120, 120, 120, 120}
2723 }
2724 ,{{ 110, 110, 110, 110, 110}
2725 ,{ 110, 110, 110, 110, 110}
2726 ,{ 110, 100, 100, 100, 110}
2727 ,{ 80, -40, 70, 10, 80}
2728 ,{ 110, 100, 100, 100, 110}
2729 }
2730 ,{{ 150, 150, 150, 150, 150}
2731 ,{ 150, 150, 150, 150, 150}
2732 ,{ 140, 130, 130, 130, 140}
2733 ,{ 150, 150, 150, 150, 150}
2734 ,{ 120, 120, 120, 120, 120}
2735 }
2736 ,{{ 110, 100, 100, 100, 110}
2737 ,{ 80, -70, -60, 10, 80}
2738 ,{ 110, 100, 100, 100, 110}
2739 ,{ -40, -40, -40, -40, -50}
2740 ,{ 110, 100, 100, 100, 110}
2741 }
2742 ,{{ 150, 150, 150, 150, 150}
2743 ,{ 150, 150, 150, 150, 150}
2744 ,{ 120, 120, 120, 120, 120}
2745 ,{ 150, 150, 150, 150, 150}
2746 ,{ 30, 30, 30, 30, 30}
2747 }
2748 }
2749 ,{{ { 140, 70, 140, 80, 140}
2750 ,{ 140, 10, 140, 10, 140}
2751 ,{ 130, 70, 130, 20, 130}
2752 ,{ 140, -30, 140, 80, 140}
2753 ,{ 110, 50, 110, 70, 110}
2754 }
2755 ,{{ { 100, -30, 100, -30, 100}
2756 ,{ 100, -30, 100, -30, 100}
2757 ,{ 100, -70, 100, -40, 100}
2758 ,{ 10, -170, 10, -30, 10}
2759 ,{ 100, -70, 100, -40, 100}
2760 }
2761 ,{{ { 140, 70, 140, 10, 140}
2762 ,{ 140, 10, 140, -30, 140}
2763 ,{ 130, 70, 130, -10, 130}
2764 ,{ 140, -30, 140, 10, 140}
2765 ,{ 110, 0, 110, -60, 110}
2766 }
2767 ,{{ { 100, -70, 100, 80, 100}
2768 ,{ 10, -160, 10, 0, 10}
2769 ,{ 100, -70, 100, -40, 100}
2770 ,{ 80, -90, -50, 80, -50}
2771 ,{ 100, -70, 100, -40, 100}
2772 }
2773 ,{{ { 140, 50, 140, 70, 140}
2774 ,{ 140, -30, 140, 10, 140}
2775 ,{ 110, 0, 110, 20, 110}
2776 ,{ 140, -30, 140, 10, 140}
2777 ,{ 70, 50, 20, 70, 20}
2778 }
2779 }
2780 ,{{ { 170, 150, 170, 150, 150}
2781 ,{ 150, 150, 150, 150, 150}
2782 ,{ 170, 130, 170, 130, 30}
2783 ,{ 150, 150, 150, 150, 140}
2784 ,{ 120, 120, 120, 120, 40}
2785 }
2786 ,{{ { 150, 110, 110, 110, 150}
2787 ,{ 150, 110, 110, 110, 150}
2788 ,{ 100, 100, 100, 100, -20}
2789 ,{ 90, 10, 70, 10, 90}
2790 ,{ 100, 100, 100, 100, 30}
2791 }
2792 ,{{ { 150, 150, 150, 150, 70}
2793 ,{ 150, 150, 150, 150, 0}
2794 ,{ 130, 130, 130, 130, -10}
2795 ,{ 150, 150, 150, 150, 70}
2796 ,{ 120, 120, 120, 120, 40}
2797 }
2798 ,{{ { 140, 100, 100, 100, 140}
2799 ,{ 90, 10, 70, 10, 90}
2800 ,{ 100, 100, 100, 100, 30}
2801 ,{ 140, -40, 20, -40, 140}
2802 ,{ 100, 100, 100, 100, 30}
2803 }
```

```
2804 , {{ 170,   150,   170,   150,    70}
2805 , { 150,   150,   150,   150,    70}
2806 , { 170,   120,   170,   120,   20}
2807 , { 150,   150,   150,   150,   70}
2808 , { 30,    30,    30,    30,  -60}
2809 }
2810 }
2811 }
2812 , {{{{ 150,   150,   120,   120,  130}
2813 , { 150,   150,   120,   120,  130}
2814 , { 130,   130,   100,   100, 110}
2815 , { 120,   120,    90,    90, 100}
2816 , { 120,   120,   100,   100, 100}
2817 }
2818 , {{ 150,   150,   120,   120,  130}
2819 , { 150,   150,   120,   120,  130}
2820 , { 120,   120,   100,   100, 100}
2821 , { -10,   -50,   -20,   -80, -10}
2822 , { 120,   120,   100,   100, 100}
2823 }
2824 , {{ 120,   120,   100,   100, 100}
2825 , { 120,   120,    90,    90, 100}
2826 , { 120,   120,   100,   100, 100}
2827 , { 120,   120,    90,    90, 100}
2828 , { 120,   120,   100,   100, 100}
2829 }
2830 , {{ 120,   120,   100,   100, 100}
2831 , { 50,    10,    50,   -10,  50}
2832 , { 120,   120,   100,   100, 100}
2833 , { 80,   -20,   -40,    80, 10}
2834 , { 120,   120,   100,   100, 100}
2835 }
2836 , {{ 130,   130,   100,   100, 110}
2837 , { 120,   120,    90,    90, 100}
2838 , { 130,   130,   100,   100, 110}
2839 , { 120,   120,    90,    90, 100}
2840 , { 110,   110,    20,     20, 30}
2841 }
2842 }
2843 , {{{{ 150,   150,   120,    50, 120}
2844 , { 150,   150,   120,    10, 120}
2845 , { 130,   130,   100,    50, 100}
2846 , { 120,   120,    90,   -20,  90}
2847 , { 120,   120,    90,     50, 90}
2848 }
2849 , {{ 150,   150,   120,     10, 120}
2850 , { 150,   150,   120,     10, 120}
2851 , { 120,   120,    90,   -10,  90}
2852 , { -50,   -50,   -80,  -190, -80}
2853 , { 120,   120,    90,   -10,  90}
2854 }
2855 , {{ 120,   120,    90,     50, 90}
2856 , { 120,   120,    90,   -20,  90}
2857 , { 120,   120,    90,     50, 90}
2858 , { 120,   120,    90,   -20,  90}
2859 , { 120,   120,    90,     50, 90}
2860 }
2861 , {{ 120,   120,    90,   -10,  90}
2862 , { 10,    10,   -20,  -130, -20}
2863 , { 120,   120,    90,   -10,  90}
2864 , { -20,   -20,   -50,  -20, -50}
2865 , { 120,   120,    90,   -10,  90}
2866 }
2867 , {{ 130,   130,   100,     50, 100}
2868 , { 120,   120,    90,   -20,  90}
2869 , { 130,   130,   100,     50, 100}
2870 , { 120,   120,    90,   -20,  90}
2871 , { 110,   110,    20,   -90, 20}
2872 }
2873 }
2874 , {{{{ 130,   120,   120,   120, 130}
2875 , { 130,   120,   120,   120, 130}
2876 , { 110,   100,   100,   100, 110}
2877 , { 100,   90,    90,    90, 100}
2878 , { 100,   100,   100,   100, 100}
2879 }
2880 , {{ 130,   120,   120,   120, 130}
2881 , { 130,   120,   120,   120, 130}
2882 , { 100,   100,   100,   100, 100}
2883 , { -10,   -80,   -20,   -80, -10}
2884 , { 100,   100,   100,   100, 100}
2885 }
2886 , {{ 100,   100,   100,   100, 100}
2887 , { 100,    90,    90,    90, 100}
2888 , { 100,   100,   100,   100, 100}
2889 , { 100,   90,    90,    90, 100}
2890 , { 100,   100,   100,   100, 100}
```

```
2891      }
2892 ,{{ 100, 100, 100, 100, 100}
2893 ,{ 50, -10, 50, -10, 50}
2894 ,{ 100, 100, 100, 100, 100}
2895 ,{ -40, -40, -40, -40, -40}
2896 ,{ 100, 100, 100, 100, 100}
2897 }
2898 ,{{ 110, 100, 100, 100, 110}
2899 ,{ 100, 90, 90, 90, 100}
2900 ,{ 110, 100, 100, 100, 110}
2901 ,{ 100, 90, 90, 90, 100}
2902 ,{ 30, 20, 20, 20, 30}
2903 }
2904 }
2905 ,{{{ 120, -10, 120, 80, 120}
2906 ,{ 120, -50, 120, -20, 120}
2907 ,{ 100, -10, 100, -40, 100}
2908 ,{ 90, -80, 90, 80, 90}
2909 ,{ 90, -20, 90, 10, 90}
2910 }
2911 ,{{ 120, -50, 120, -20, 120}
2912 ,{ 120, -50, 120, -20, 120}
2913 ,{ 90, -80, 90, -40, 90}
2914 ,{ -80, -260, -80, -90, -80}
2915 ,{ 90, -80, 90, -40, 90}
2916 }
2917 ,{{ 90, -20, 90, -40, 90}
2918 ,{ 90, -80, 90, -50, 90}
2919 ,{ 90, -20, 90, -40, 90}
2920 ,{ 90, -80, 90, -50, 90}
2921 ,{ 90, -20, 90, -40, 90}
2922 }
2923 ,{{ 90, -80, 90, 80, 90}
2924 ,{ -20, -190, -20, -20, -20}
2925 ,{ 90, -80, 90, -40, 90}
2926 ,{ 80, -90, -50, 80, -50}
2927 ,{ 90, -80, 90, -40, 90}
2928 }
2929 ,{{ 100, -10, 100, 10, 100}
2930 ,{ 90, -80, 90, -50, 90}
2931 ,{ 100, -10, 100, -40, 100}
2932 ,{ 90, -80, 90, -50, 90}
2933 ,{ 20, -150, 20, 10, 20}
2934 }
2935 }
2936 ,{{{ 120, 120, 120, 120, 110}
2937 ,{ 120, 120, 120, 120, 110}
2938 ,{ 100, 100, 100, 100, 30}
2939 ,{ 90, 90, 90, 90, 20}
2940 ,{ 100, 100, 100, 100, 20}
2941 }
2942 ,{{ 120, 120, 120, 120, 110}
2943 ,{ 120, 120, 120, 120, 110}
2944 ,{ 100, 100, 100, 100, 20}
2945 ,{ -20, -80, -20, -80, -150}
2946 ,{ 100, 100, 100, 100, 20}
2947 }
2948 ,{{ 100, 100, 100, 100, 20}
2949 ,{ 90, 90, 90, 90, 20}
2950 ,{ 100, 100, 100, 100, 20}
2951 ,{ 90, 90, 90, 90, 20}
2952 ,{ 100, 100, 100, 100, 20}
2953 }
2954 ,{{ 100, 100, 100, 100, 20}
2955 ,{ 50, -10, 50, -10, -90}
2956 ,{ 100, 100, 100, 100, 20}
2957 ,{ 10, -40, -40, -40, 10}
2958 ,{ 100, 100, 100, 100, 20}
2959 }
2960 ,{{ 100, 100, 100, 100, 30}
2961 ,{ 90, 90, 90, 90, 20}
2962 ,{ 100, 100, 100, 100, 30}
2963 ,{ 90, 90, 90, 90, 20}
2964 ,{ 20, 20, 20, 20, -50}
2965 }
2966 }
2967 }
2968 ,{{{ 300, 300, 250, 250, 260}
2969 ,{ 280, 280, 250, 250, 260}
2970 ,{ 240, 240, 220, 220, 220}
2971 ,{ 240, 240, 220, 220, 220}
2972 ,{ 300, 300, 220, 220, 220}
2973 }
2974 ,{{ 280, 280, 250, 250, 260}
2975 ,{ 280, 280, 250, 250, 260}
2976 ,{ 240, 240, 220, 220, 220}
2977 ,{ 200, 160, 200, 140, 200}}
```

```
2978 , { 240, 240, 220, 220, 220}
2979 }
2980 , {{ 240, 240, 220, 220, 220}
2981 , { 240, 240, 220, 220, 220}
2982 , { 240, 240, 220, 220, 220}
2983 , { 240, 240, 220, 220, 220}
2984 , { 240, 240, 220, 220, 220}
2985 }
2986 , {{ 240, 240, 240, 220, 240}
2987 , { 240, 200, 240, 180, 240}
2988 , { 240, 240, 220, 220, 220}
2989 , { 210, 110, 90, 210, 140}
2990 , { 240, 240, 220, 220, 220}
2991 }
2992 , {{ 300, 300, 220, 220, 220}
2993 , { 240, 240, 220, 220, 220}
2994 , { 240, 240, 220, 220, 220}
2995 , { 240, 240, 220, 220, 220}
2996 , { 300, 300, 220, 220, 220}
2997 }
2998 }
2999 , {{{ 300, 300, 250, 160, 250}
3000 , { 280, 280, 250, 140, 250}
3001 , { 240, 240, 210, 160, 210}
3002 , { 240, 240, 210, 100, 210}
3003 , { 300, 300, 210, 160, 210}
3004 }
3005 , {{ 280, 280, 250, 140, 250}
3006 , { 280, 280, 250, 140, 250}
3007 , { 240, 240, 210, 100, 210}
3008 , { 160, 160, 130, 20, 130}
3009 , { 240, 240, 210, 100, 210}
3010 }
3011 , {{ 240, 240, 210, 160, 210}
3012 , { 240, 240, 210, 100, 210}
3013 , { 240, 240, 210, 160, 210}
3014 , { 240, 240, 210, 100, 210}
3015 , { 240, 240, 210, 160, 210}
3016 }
3017 , {{ 240, 240, 210, 100, 210}
3018 , { 200, 200, 170, 60, 170}
3019 , { 240, 240, 210, 100, 210}
3020 , { 110, 110, 80, 100, 80}
3021 , { 240, 240, 210, 100, 210}
3022 }
3023 , {{ 300, 300, 210, 160, 210}
3024 , { 240, 240, 210, 100, 210}
3025 , { 240, 240, 210, 160, 210}
3026 , { 240, 240, 210, 100, 210}
3027 , { 300, 300, 210, 100, 210}
3028 }
3029 }
3030 , {{{ 260, 250, 250, 250, 260}
3031 , { 260, 250, 250, 250, 260}
3032 , { 220, 220, 220, 220, 220}
3033 , { 220, 220, 220, 220, 220}
3034 , { 220, 220, 220, 220, 220}
3035 }
3036 , {{ 260, 250, 250, 250, 260}
3037 , { 260, 250, 250, 250, 260}
3038 , { 220, 220, 220, 220, 220}
3039 , { 200, 140, 200, 140, 200}
3040 , { 220, 220, 220, 220, 220}
3041 }
3042 , {{ 220, 220, 220, 220, 220}
3043 , { 220, 220, 220, 220, 220}
3044 , { 220, 220, 220, 220, 220}
3045 , { 220, 220, 220, 220, 220}
3046 , { 220, 220, 220, 220, 220}
3047 }
3048 , {{ 240, 220, 240, 220, 240}
3049 , { 240, 180, 240, 180, 240}
3050 , { 220, 220, 220, 220, 220}
3051 , { 90, 90, 90, 90, 90}
3052 , { 220, 220, 220, 220, 220}
3053 }
3054 , {{ 220, 220, 220, 220, 220}
3055 , { 220, 220, 220, 220, 220}
3056 , { 220, 220, 220, 220, 220}
3057 , { 220, 220, 220, 220, 220}
3058 , { 220, 220, 220, 220, 220}
3059 }
3060 }
3061 , {{{ 250, 100, 250, 210, 250}
3062 , { 250, 70, 250, 170, 250}
3063 , { 210, 100, 210, 80, 210}
3064 , { 210, 40, 210, 210, 210}
```

```
3065 , { 210, 100, 210, 210, 210}
3066 }
3067 , {{ 250, 70, 250, 130, 250}
3068 , { 250, 70, 250, 110, 250}
3069 , { 210, 40, 210, 80, 210}
3070 , { 130, -40, 130, 130, 130}
3071 , { 210, 40, 210, 80, 210}
3072 }
3073 , {{ 210, 100, 210, 80, 210}
3074 , { 210, 40, 210, 80, 210}
3075 , { 210, 100, 210, 80, 210}
3076 , { 210, 40, 210, 80, 210}
3077 , { 210, 100, 210, 80, 210}
3078 }
3079 , {{ 210, 40, 210, 210, 210}
3080 , { 170, 0, 170, 170, 170}
3081 , { 210, 40, 210, 80, 210}
3082 , { 210, 40, 80, 210, 80}
3083 , { 210, 40, 210, 80, 210}
3084 }
3085 , {{ 210, 100, 210, 210, 210}
3086 , { 210, 40, 210, 80, 210}
3087 , { 210, 100, 210, 80, 210}
3088 , { 210, 40, 210, 80, 210}
3089 , { 210, 40, 210, 210, 210}
3090 }
3091 }
3092 , {{{ 250, 250, 250, 250, 240}
3093 , { 250, 250, 250, 250, 240}
3094 , { 220, 220, 220, 220, 140}
3095 , { 220, 220, 220, 220, 140}
3096 , { 220, 220, 220, 220, 140}
3097 }
3098 , {{ 250, 250, 250, 250, 240}
3099 , { 250, 250, 250, 250, 240}
3100 , { 220, 220, 220, 220, 140}
3101 , { 200, 140, 200, 140, 60}
3102 , { 220, 220, 220, 220, 140}
3103 }
3104 , {{ 220, 220, 220, 220, 140}
3105 , { 220, 220, 220, 220, 140}
3106 , { 220, 220, 220, 220, 140}
3107 , { 220, 220, 220, 220, 140}
3108 , { 220, 220, 220, 220, 140}
3109 }
3110 , {{ 240, 220, 240, 220, 140}
3111 , { 240, 180, 240, 180, 100}
3112 , { 220, 220, 220, 220, 140}
3113 , { 140, 90, 90, 90, 140}
3114 , { 220, 220, 220, 220, 140}
3115 }
3116 , {{ 220, 220, 220, 220, 140}
3117 , { 220, 220, 220, 220, 140}
3118 , { 220, 220, 220, 220, 140}
3119 , { 220, 220, 220, 220, 140}
3120 , { 220, 220, 220, 220, 140}
3121 }
3122 }
3123 }
3124 , {{{ 280, 270, 280, 220, 280}
3125 , { 280, 240, 280, 220, 280}
3126 , { 210, 210, 190, 190, 190}
3127 , { 210, 210, 190, 190, 190}
3128 , { 270, 270, 190, 190, 190}
3129 }
3130 , {{ 210, 210, 190, 190, 190}
3131 , { 190, 190, 150, 150, 160}
3132 , { 210, 210, 190, 190, 190}
3133 , { 120, 80, 110, 50, 120}
3134 , { 210, 210, 190, 190, 190}
3135 }
3136 , {{ 210, 210, 190, 190, 190}
3137 , { 210, 210, 190, 190, 190}
3138 , { 210, 210, 190, 190, 190}
3139 , { 210, 210, 190, 190, 190}
3140 , { 210, 210, 190, 190, 190}
3141 }
3142 , {{ 280, 240, 280, 220, 280}
3143 , { 280, 240, 280, 220, 280}
3144 , { 210, 210, 190, 190, 190}
3145 , { 180, 80, 60, 180, 110}
3146 , { 210, 210, 190, 190, 190}
3147 }
3148 , {{ 270, 270, 190, 190, 190}
3149 , { 210, 210, 190, 190, 190}
3150 , { 210, 210, 190, 190, 190}
3151 , { 210, 210, 190, 190, 190}
```

```
3152 , { 270, 270, 190, 190, 190}
3153 }
3154 }
3155 ,{{ { 270, 270, 210, 130, 210}
3156 ,{ 240, 240, 210, 100, 210}
3157 ,{ 210, 210, 180, 130, 180}
3158 ,{ 210, 210, 180, 70, 180}
3159 ,{ 270, 270, 180, 130, 180}
3160 }
3161 ,{{ { 210, 210, 180, 70, 180}
3162 ,{ 190, 190, 150, 40, 150}
3163 ,{ 210, 210, 180, 70, 180}
3164 ,{ 80, 80, 50, -60, 50}
3165 ,{ 210, 210, 180, 70, 180}
3166 }
3167 ,{{ { 210, 210, 180, 130, 180}
3168 ,{ 210, 210, 180, 70, 180}
3169 ,{ 210, 210, 180, 130, 180}
3170 ,{ 210, 210, 180, 70, 180}
3171 ,{ 210, 210, 180, 130, 180}
3172 }
3173 ,{{ { 240, 240, 210, 100, 210}
3174 ,{ 240, 240, 210, 100, 210}
3175 ,{ 210, 210, 180, 70, 180}
3176 ,{ 80, 80, 50, 70, 50}
3177 ,{ 210, 210, 180, 70, 180}
3178 }
3179 ,{{ { 270, 270, 180, 130, 180}
3180 ,{ 210, 210, 180, 70, 180}
3181 ,{ 210, 210, 180, 130, 180}
3182 ,{ 210, 210, 180, 70, 180}
3183 ,{ 270, 270, 180, 70, 180}
3184 }
3185 }
3186 ,{{ { 280, 220, 280, 220, 280}
3187 ,{ 280, 220, 280, 220, 280}
3188 ,{ 190, 190, 190, 190, 190}
3189 ,{ 190, 190, 190, 190, 190}
3190 ,{ 190, 190, 190, 190, 190}
3191 }
3192 ,{{ { 190, 190, 190, 190, 190}
3193 ,{ 160, 150, 150, 150, 160}
3194 ,{ 190, 190, 190, 190, 190}
3195 ,{ 120, 50, 110, 50, 120}
3196 ,{ 190, 190, 190, 190, 190}
3197 }
3198 ,{{ { 190, 190, 190, 190, 190}
3199 ,{ 190, 190, 190, 190, 190}
3200 ,{ 190, 190, 190, 190, 190}
3201 ,{ 190, 190, 190, 190, 190}
3202 ,{ 190, 190, 190, 190, 190}
3203 }
3204 ,{{ { 280, 220, 280, 220, 280}
3205 ,{ 280, 220, 280, 220, 280}
3206 ,{ 190, 190, 190, 190, 190}
3207 ,{ 60, 60, 60, 60, 60}
3208 ,{ 190, 190, 190, 190, 190}
3209 }
3210 ,{{ { 190, 190, 190, 190, 190}
3211 ,{ 190, 190, 190, 190, 190}
3212 ,{ 190, 190, 190, 190, 190}
3213 ,{ 190, 190, 190, 190, 190}
3214 ,{ 190, 190, 190, 190, 190}
3215 }
3216 }
3217 ,{{ { 210, 70, 210, 210, 210}
3218 ,{ 210, 40, 210, 210, 210}
3219 ,{ 180, 70, 180, 50, 180}
3220 ,{ 180, 10, 180, 180, 180}
3221 ,{ 180, 70, 180, 180, 180}
3222 }
3223 ,{{ { 180, 10, 180, 50, 180}
3224 ,{ 150, -20, 150, 10, 150}
3225 ,{ 180, 10, 180, 50, 180}
3226 ,{ 50, -120, 50, 40, 50}
3227 ,{ 180, 10, 180, 50, 180}
3228 }
3229 ,{{ { 180, 70, 180, 50, 180}
3230 ,{ 180, 10, 180, 50, 180}
3231 ,{ 180, 70, 180, 50, 180}
3232 ,{ 180, 10, 180, 50, 180}
3233 ,{ 180, 70, 180, 50, 180}
3234 }
3235 ,{{ { 210, 40, 210, 210, 210}
3236 ,{ 210, 40, 210, 210, 210}
3237 ,{ 180, 10, 180, 50, 180}
3238 ,{ 180, 10, 50, 180, 50}
```

```
3239 , { 180,   10,   180,    50,   180}
3240   }
3241 , {{ 180,    70,   180,   180,   180}
3242 , { 180,   10,   180,    50,   180}
3243 , { 180,    70,   180,    50,   180}
3244 , { 180,   10,   180,    50,   180}
3245 , { 180,   10,   180,   180,   180}
3246   }
3247   }
3248 , {{{ 280,   220,   280,   220,   140}
3249 , { 280,   220,   280,   220,   140}
3250 , { 190,   190,   190,   190,   110}
3251 , { 190,   190,   190,   190,   110}
3252 , { 190,   190,   190,   190,   110}
3253   }
3254 , {{ 190,   190,   190,   190,   140}
3255 , { 150,   150,   150,   150,   140}
3256 , { 190,   190,   190,   190,   110}
3257 , { 110,    50,   110,    50,   -20}
3258 , { 190,   190,   190,   190,   110}
3259   }
3260 , {{ 190,   190,   190,   190,   110}
3261 , { 190,   190,   190,   190,   110}
3262 , { 190,   190,   190,   190,   110}
3263 , { 190,   190,   190,   190,   110}
3264 , { 190,   190,   190,   190,   110}
3265   }
3266 , {{ 280,   220,   280,   220,   140}
3267 , { 280,   220,   280,   220,   140}
3268 , { 190,   190,   190,   190,   110}
3269 , { 110,    60,    60,    60,   110}
3270 , { 190,   190,   190,   190,   110}
3271   }
3272 , {{ 190,   190,   190,   190,   110}
3273 , { 190,   190,   190,   190,   110}
3274 , { 190,   190,   190,   190,   110}
3275 , { 190,   190,   190,   190,   110}
3276 , { 190,   190,   190,   190,   110}
3277   }
3278   }
3279   }
3280 , {{{{{ 210,   210,   190,   190,   200}
3281 , { 210,   210,   190,   190,   200}
3282 , { 190,   190,   170,   170,   170}
3283 , { 200,   200,   170,   170,   180}
3284 , { 190,   190,   170,   170,   170}
3285   }
3286 , {{ 210,   210,   190,   190,   190}
3287 , { 210,   210,   190,   190,   190}
3288 , { 190,   190,   160,   160,   170}
3289 , { 130,    90,   120,    60,   130}
3290 , { 190,   190,   160,   160,   170}
3291   }
3292 , {{ 200,   200,   170,   170,   180}
3293 , { 200,   200,   170,   170,   180}
3294 , { 190,   190,   170,   170,   170}
3295 , { 200,   200,   170,   170,   180}
3296 , { 190,   190,   170,   170,   170}
3297   }
3298 , {{ 200,   190,   190,   160,   200}
3299 , { 200,   160,   190,   130,   200}
3300 , { 190,   190,   160,   160,   170}
3301 , { 130,    40,    10,   130,    70}
3302 , { 190,   190,   160,   160,   170}
3303   }
3304 , {{ 200,   200,   170,   170,   180}
3305 , { 200,   200,   170,   170,   180}
3306 , { 190,   190,   170,   170,   170}
3307 , { 200,   200,   170,   170,   180}
3308 , { 160,   160,    80,    80,    80}
3309   }
3310   }
3311 , {{{{ 210,   210,   180,   110,   180}
3312 , { 210,   210,   180,    70,   180}
3313 , { 190,   190,   160,   110,   160}
3314 , { 200,   200,   170,    60,   170}
3315 , { 190,   190,   160,   110,   160}
3316   }
3317 , {{ 210,   210,   180,    70,   180}
3318 , { 210,   210,   180,    70,   180}
3319 , { 190,   190,   160,    50,   160}
3320 , { 90,    90,    60,   -50,    60}
3321 , { 190,   190,   160,    50,   160}
3322   }
3323 , {{ 200,   200,   170,   110,   170}
3324 , { 200,   200,   170,    60,   170}
3325 , { 190,   190,   160,   110,   160}
```

```
3326 , { 200, 200, 170, 60, 170}
3327 , { 190, 190, 160, 110, 160}
3328 }
3329 , {{ 190, 190, 160, 50, 160}
3330 , { 160, 160, 130, 20, 130}
3331 , { 190, 190, 160, 50, 160}
3332 , { 40, 40, 10, 30, 10}
3333 , { 190, 190, 160, 50, 160}
3334 }
3335 , {{ 200, 200, 170, 110, 170}
3336 , { 200, 200, 170, 60, 170}
3337 , { 190, 190, 160, 110, 160}
3338 , { 200, 200, 170, 60, 170}
3339 , { 160, 160, 70, -30, 70}
3340 }
3341 }
3342 , {{{ 200, 190, 190, 190, 200}
3343 , { 200, 190, 190, 190, 200}
3344 , { 170, 170, 170, 170, 170}
3345 , { 180, 170, 170, 170, 180}
3346 , { 170, 170, 170, 170, 170}
3347 }
3348 , {{ 190, 190, 190, 190, 190}
3349 , { 190, 190, 190, 190, 190}
3350 , { 170, 160, 160, 160, 170}
3351 , { 130, 60, 120, 60, 130}
3352 , { 170, 160, 160, 160, 170}
3353 }
3354 , {{ 180, 170, 170, 170, 180}
3355 , { 180, 170, 170, 170, 180}
3356 , { 170, 170, 170, 170, 170}
3357 , { 180, 170, 170, 170, 180}
3358 , { 170, 170, 170, 170, 170}
3359 }
3360 , {{ 200, 160, 190, 160, 200}
3361 , { 200, 130, 190, 130, 200}
3362 , { 170, 160, 160, 160, 170}
3363 , { 20, 10, 10, 10, 20}
3364 , { 170, 160, 160, 160, 170}
3365 }
3366 , {{ 180, 170, 170, 170, 180}
3367 , { 180, 170, 170, 170, 180}
3368 , { 170, 170, 170, 170, 170}
3369 , { 180, 170, 170, 170, 180}
3370 , { 80, 80, 80, 80, 80}
3371 }
3372 }
3373 , {{{ 180, 50, 180, 130, 180}
3374 , { 180, 10, 180, 120, 180}
3375 , { 160, 50, 160, 30, 160}
3376 , { 170, 0, 170, 130, 170}
3377 , { 160, 50, 160, 70, 160}
3378 }
3379 , {{ 180, 10, 180, 50, 180}
3380 , { 180, 10, 180, 50, 180}
3381 , { 160, -10, 160, 20, 160}
3382 , { 60, -110, 60, 50, 60}
3383 , { 160, -10, 160, 20, 160}
3384 }
3385 , {{ 170, 50, 170, 30, 170}
3386 , { 170, 0, 170, 30, 170}
3387 , { 160, 50, 160, 30, 160}
3388 , { 170, 0, 170, 30, 170}
3389 , { 160, 50, 160, 30, 160}
3390 }
3391 , {{ 160, -10, 160, 130, 160}
3392 , { 130, -40, 130, 120, 130}
3393 , { 160, -10, 160, 20, 160}
3394 , { 130, -30, 10, 130, 10}
3395 , { 160, -10, 160, 20, 160}
3396 }
3397 , {{ 170, 50, 170, 70, 170}
3398 , { 170, 0, 170, 30, 170}
3399 , { 160, 50, 160, 30, 160}
3400 , { 170, 0, 170, 30, 170}
3401 , { 70, -100, 70, 70, 70}
3402 }
3403 }
3404 , {{{ 190, 190, 190, 190, 170}
3405 , { 190, 190, 190, 190, 170}
3406 , { 170, 170, 170, 170, 90}
3407 , { 170, 170, 170, 170, 100}
3408 , { 170, 170, 170, 170, 90}
3409 }
3410 , {{ 190, 190, 190, 190, 170}
3411 , { 190, 190, 190, 190, 170}
3412 , { 160, 160, 160, 160, 90}
```

```
3413 , { 120,   60,   120,    60,   -10}
3414 , { 160,  160,  160,  160,    90}
3415 }
3416 , {{ 170,  170,  170,  170,   100}
3417 , { 170,  170,  170,  170,   100}
3418 , { 170,  170,  170,  170,    90}
3419 , { 170,  170,  170,  170,   100}
3420 , { 170,  170,  170,  170,    90}
3421 }
3422 , {{ 190,  160,  190,  160,    90}
3423 , { 190,  130,  190,  130,    60}
3424 , { 160,  160,  160,  160,    90}
3425 , { 70,    10,   10,   10,    70}
3426 , { 160,  160,  160,  160,    90}
3427 }
3428 , {{ 170,  170,  170,  170,   100}
3429 , { 170,  170,  170,  170,   100}
3430 , { 170,  170,  170,  170,    90}
3431 , { 170,  170,  170,  170,   100}
3432 , { 80,    80,   80,   80,     0}
3433 }
3434 }
3435 }
3436 , {{{ 210,  210,  190,  190,  190}
3437 , { 210,  210,  190,  190,  190}
3438 , { 210,  210,  190,  190,  190}
3439 , { 210,  210,  190,  190,  190}
3440 , { 210,  210,  190,  190,  190}
3441 }
3442 , {{ 210,  210,  190,  190,  190}
3443 , { 210,  210,  190,  190,  190}
3444 , { 170,  170,  140,  140,  150}
3445 , { 150,  110,  140,   80,  150}
3446 , { 170,  170,  140,  140,  150}
3447 }
3448 , {{ 210,  210,  190,  190,  190}
3449 , { 210,  210,  190,  190,  190}
3450 , { 210,  210,  190,  190,  190}
3451 , { 210,  210,  190,  190,  190}
3452 , { 210,  210,  190,  190,  190}
3453 }
3454 , {{ 170,  170,  150,  150,  160}
3455 , { 160,  120,  150,   90,  160}
3456 , { 170,  170,  140,  140,  150}
3457 , { 150,   60,   30,  150,   90}
3458 , { 170,  170,  140,  140,  150}
3459 }
3460 , {{ 210,  210,  190,  190,  190}
3461 , { 210,  210,  190,  190,  190}
3462 , { 180,  180,  160,  160,  160}
3463 , { 210,  210,  190,  190,  190}
3464 , { 190,  190,  100,  100,  110}
3465 }
3466 }
3467 , {{{ 210,  210,  180,  130,  180}
3468 , { 210,  210,  180,   70,  180}
3469 , { 210,  210,  180,  130,  180}
3470 , { 210,  210,  180,   70,  180}
3471 , { 210,  210,  180,  130,  180}
3472 }
3473 , {{ 210,  210,  180,   70,  180}
3474 , { 210,  210,  180,   70,  180}
3475 , { 170,  170,  140,   30,  140}
3476 , { 110,  110,   80,  -30,   80}
3477 , { 170,  170,  140,   30,  140}
3478 }
3479 , {{ 210,  210,  180,  130,  180}
3480 , { 210,  210,  180,   70,  180}
3481 , { 210,  210,  180,  130,  180}
3482 , { 210,  210,  180,   70,  180}
3483 , { 210,  210,  180,  130,  180}
3484 }
3485 , {{ 170,  170,  140,   50,  140}
3486 , { 120,  120,   90,  -20,   90}
3487 , { 170,  170,  140,   30,  140}
3488 , { 60,    60,   30,   50,   30}
3489 , { 170,  170,  140,   30,  140}
3490 }
3491 , {{ 210,  210,  180,  100,  180}
3492 , { 210,  210,  180,   70,  180}
3493 , { 180,  180,  150,  100,  150}
3494 , { 210,  210,  180,   70,  180}
3495 , { 190,  190,  100,  -10,  100}
3496 }
3497 }
3498 , {{{ 190,  190,  190,  190,  190}
3499 , { 190,  190,  190,  190,  190}}
```

```
3500 , { 190, 190, 190, 190, 190}
3501 , { 190, 190, 190, 190, 190}
3502 , { 190, 190, 190, 190, 190}
3503 }
3504 , {{ 190, 190, 190, 190, 190}
3505 , { 190, 190, 190, 190, 190}
3506 , { 150, 140, 140, 140, 150}
3507 , { 150, 80, 140, 80, 150}
3508 , { 150, 140, 140, 140, 150}
3509 }
3510 , {{ 190, 190, 190, 190, 190}
3511 , { 190, 190, 190, 190, 190}
3512 , { 190, 190, 190, 190, 190}
3513 , { 190, 190, 190, 190, 190}
3514 , { 190, 190, 190, 190, 190}
3515 }
3516 , {{ 160, 140, 150, 140, 160}
3517 , { 160, 90, 150, 90, 160}
3518 , { 150, 140, 140, 140, 150}
3519 , { 40, 30, 30, 30, 40}
3520 , { 150, 140, 140, 140, 150}
3521 }
3522 , {{ 190, 190, 190, 190, 190}
3523 , { 190, 190, 190, 190, 190}
3524 , { 160, 160, 160, 160, 160}
3525 , { 190, 190, 190, 190, 190}
3526 , { 110, 100, 100, 100, 110}
3527 }
3528 }
3529 , {{{ 180, 70, 180, 150, 180}
3530 , { 180, 10, 180, 80, 180}
3531 , { 180, 70, 180, 50, 180}
3532 , { 180, 10, 180, 150, 180}
3533 , { 180, 70, 180, 90, 180}
3534 }
3535 , {{ 180, 10, 180, 70, 180}
3536 , { 180, 10, 180, 50, 180}
3537 , { 140, -30, 140, 0, 140}
3538 , { 80, -90, 80, 70, 80}
3539 , { 140, -30, 140, 0, 140}
3540 }
3541 , {{ 180, 70, 180, 50, 180}
3542 , { 180, 10, 180, 50, 180}
3543 , { 180, 70, 180, 50, 180}
3544 , { 180, 10, 180, 50, 180}
3545 , { 180, 70, 180, 50, 180}
3546 }
3547 , {{ 150, -10, 140, 150, 140}
3548 , { 90, -80, 90, 80, 90}
3549 , { 140, -30, 140, 0, 140}
3550 , { 150, -10, 30, 150, 30}
3551 , { 140, -30, 140, 0, 140}
3552 }
3553 , {{ 180, 40, 180, 90, 180}
3554 , { 180, 10, 180, 50, 180}
3555 , { 150, 40, 150, 20, 150}
3556 , { 180, 10, 180, 50, 180}
3557 , { 100, -70, 100, 90, 100}
3558 }
3559 }
3560 , {{{ 190, 190, 190, 190, 170}
3561 , { 190, 190, 190, 190, 170}
3562 , { 190, 190, 190, 190, 110}
3563 , { 190, 190, 190, 190, 110}
3564 , { 190, 190, 190, 190, 110}
3565 }
3566 , {{ 190, 190, 190, 190, 170}
3567 , { 190, 190, 190, 190, 170}
3568 , { 140, 140, 140, 140, 70}
3569 , { 140, 80, 140, 80, 10}
3570 , { 140, 140, 140, 140, 70}
3571 }
3572 , {{ 190, 190, 190, 190, 110}
3573 , { 190, 190, 190, 190, 110}
3574 , { 190, 190, 190, 190, 110}
3575 , { 190, 190, 190, 190, 110}
3576 , { 190, 190, 190, 190, 110}
3577 }
3578 , {{ 150, 140, 150, 140, 90}
3579 , { 150, 90, 150, 90, 20}
3580 , { 140, 140, 140, 140, 70}
3581 , { 90, 30, 30, 30, 90}
3582 , { 140, 140, 140, 140, 70}
3583 }
3584 , {{ 190, 190, 190, 190, 110}
3585 , { 190, 190, 190, 190, 110}
3586 , { 160, 160, 160, 160, 80}
```

```
3587 , { 190, 190, 190, 190, 110}
3588 , { 100, 100, 100, 100, 30}
3589 }
3590 }
3591 }
3592 ,{ {{ 300, 300, 280, 250, 280}
3593 , { 280, 280, 280, 250, 280}
3594 , { 240, 240, 220, 220, 220}
3595 , { 240, 240, 220, 220, 220}
3596 , { 300, 300, 220, 220, 220}
3597 }
3598 ,{ { 280, 280, 250, 250, 260}
3599 , { 280, 280, 250, 250, 260}
3600 , { 240, 240, 220, 220, 220}
3601 , { 200, 160, 200, 140, 200}
3602 , { 240, 240, 220, 220, 220}
3603 }
3604 ,{ { 240, 240, 220, 220, 220}
3605 , { 240, 240, 220, 220, 220}
3606 , { 240, 240, 220, 220, 220}
3607 , { 240, 240, 220, 220, 220}
3608 , { 240, 240, 220, 220, 220}
3609 }
3610 ,{ {{ 280, 240, 280, 220, 280}
3611 , { 280, 240, 280, 220, 280}
3612 , { 240, 240, 220, 220, 220}
3613 , { 210, 110, 90, 210, 140}
3614 , { 240, 240, 220, 220, 220}
3615 }
3616 ,{ {{ 300, 300, 220, 220, 220}
3617 , { 240, 240, 220, 220, 220}
3618 , { 240, 240, 220, 220, 220}
3619 , { 240, 240, 220, 220, 220}
3620 , { 300, 300, 220, 220, 220}
3621 }
3622 }
3623 ,{ {{ 300, 300, 250, 160, 250}
3624 , { 280, 280, 250, 140, 250}
3625 , { 240, 240, 210, 160, 210}
3626 , { 240, 240, 210, 100, 210}
3627 , { 300, 300, 210, 160, 210}
3628 }
3629 ,{ {{ 280, 280, 250, 140, 250}
3630 , { 280, 280, 250, 140, 250}
3631 , { 240, 240, 210, 100, 210}
3632 , { 160, 160, 130, 20, 130}
3633 , { 240, 240, 210, 100, 210}
3634 }
3635 ,{ {{ 240, 240, 210, 160, 210}
3636 , { 240, 240, 210, 100, 210}
3637 , { 240, 240, 210, 160, 210}
3638 , { 240, 240, 210, 100, 210}
3639 , { 240, 240, 210, 160, 210}
3640 }
3641 ,{ {{ 240, 240, 210, 100, 210}
3642 , { 240, 240, 210, 100, 210}
3643 , { 240, 240, 210, 100, 210}
3644 , { 110, 110, 80, 100, 80}
3645 , { 240, 240, 210, 100, 210}
3646 }
3647 ,{ {{ 300, 300, 210, 160, 210}
3648 , { 240, 240, 210, 100, 210}
3649 , { 240, 240, 210, 160, 210}
3650 , { 240, 240, 210, 100, 210}
3651 , { 300, 300, 210, 140, 210}
3652 }
3653 }
3654 ,{ {{ 280, 250, 280, 250, 280}
3655 , { 280, 250, 280, 250, 280}
3656 , { 220, 220, 220, 220, 220}
3657 , { 220, 220, 220, 220, 220}
3658 , { 220, 220, 220, 220, 220}
3659 }
3660 ,{ {{ 260, 250, 250, 250, 260}
3661 , { 260, 250, 250, 250, 260}
3662 , { 220, 220, 220, 220, 220}
3663 , { 200, 140, 200, 140, 200}
3664 , { 220, 220, 220, 220, 220}
3665 }
3666 ,{ {{ 220, 220, 220, 220, 220}
3667 , { 220, 220, 220, 220, 220}
3668 , { 220, 220, 220, 220, 220}
3669 , { 220, 220, 220, 220, 220}
3670 , { 220, 220, 220, 220, 220}
3671 }
3672 ,{ {{ 280, 220, 280, 220, 280}
3673 , { 280, 220, 280, 220, 280}}
```

```
3674 , { 220, 220, 220, 220, 220}
3675 , { 90, 90, 90, 90, 90}
3676 , { 220, 220, 220, 220, 220}
3677 }
3678 , {{ 220, 220, 220, 220, 220}
3679 , { 220, 220, 220, 220, 220}
3680 , { 220, 220, 220, 220, 220}
3681 , { 220, 220, 220, 220, 220}
3682 , { 220, 220, 220, 220, 220}
3683 }
3684 }
3685 , {{{ 250, 100, 250, 210, 250}
3686 , { 250, 70, 250, 210, 250}
3687 , { 210, 100, 210, 80, 210}
3688 , { 210, 40, 210, 210, 210}
3689 , { 210, 100, 210, 210, 210}
3690 }
3691 , {{ 250, 70, 250, 130, 250}
3692 , { 250, 70, 250, 110, 250}
3693 , { 210, 40, 210, 80, 210}
3694 , { 130, -40, 130, 130, 130}
3695 , { 210, 40, 210, 80, 210}
3696 }
3697 , {{ 210, 100, 210, 80, 210}
3698 , { 210, 40, 210, 80, 210}
3699 , { 210, 100, 210, 80, 210}
3700 , { 210, 40, 210, 80, 210}
3701 , { 210, 100, 210, 80, 210}
3702 }
3703 , {{ 210, 40, 210, 210, 210}
3704 , { 210, 40, 210, 210, 210}
3705 , { 210, 40, 210, 80, 210}
3706 , { 210, 40, 80, 210, 80}
3707 , { 210, 40, 210, 80, 210}
3708 }
3709 , {{ 210, 100, 210, 210, 210}
3710 , { 210, 40, 210, 80, 210}
3711 , { 210, 100, 210, 80, 210}
3712 , { 210, 40, 210, 80, 210}
3713 , { 210, 50, 210, 210, 210}
3714 }
3715 }
3716 , {{{ 280, 250, 280, 250, 240}
3717 , { 280, 250, 280, 250, 240}
3718 , { 220, 220, 220, 220, 140}
3719 , { 220, 220, 220, 220, 140}
3720 , { 220, 220, 220, 220, 140}
3721 }
3722 , {{ 250, 250, 250, 250, 240}
3723 , { 250, 250, 250, 250, 240}
3724 , { 220, 220, 220, 220, 140}
3725 , { 200, 140, 200, 140, 90}
3726 , { 220, 220, 220, 220, 140}
3727 }
3728 , {{ 220, 220, 220, 220, 140}
3729 , { 220, 220, 220, 220, 140}
3730 , { 220, 220, 220, 220, 140}
3731 , { 220, 220, 220, 220, 140}
3732 , { 220, 220, 220, 220, 140}
3733 }
3734 , {{ 280, 220, 280, 220, 140}
3735 , { 280, 220, 280, 220, 140}
3736 , { 220, 220, 220, 220, 140}
3737 , { 140, 90, 90, 90, 140}
3738 , { 220, 220, 220, 220, 140}
3739 }
3740 , {{ 220, 220, 220, 220, 140}
3741 , { 220, 220, 220, 220, 140}
3742 , { 220, 220, 220, 220, 140}
3743 , { 220, 220, 220, 220, 140}
3744 , { 220, 220, 220, 220, 140}
3745 }
3746 }
3747 }
3748 }
3749 , {{{{{ INF, INF, INF, INF, INF}
3750 , { INF, INF, INF, INF, INF}
3751 , { INF, INF, INF, INF, INF}
3752 , { INF, INF, INF, INF, INF}
3753 , { INF, INF, INF, INF, INF}
3754 }
3755 , {{ INF, INF, INF, INF, INF}
3756 , { INF, INF, INF, INF, INF}
3757 , { INF, INF, INF, INF, INF}
3758 , { INF, INF, INF, INF, INF}
3759 , { INF, INF, INF, INF, INF}
3760 }
```

```
3761 , {{ INF, INF, INF, INF, INF }
3762 , { INF, INF, INF, INF, INF }
3763 , { INF, INF, INF, INF, INF }
3764 , { INF, INF, INF, INF, INF }
3765 , { INF, INF, INF, INF, INF }
3766 }
3767 , {{ INF, INF, INF, INF, INF }
3768 , { INF, INF, INF, INF, INF }
3769 , { INF, INF, INF, INF, INF }
3770 , { INF, INF, INF, INF, INF }
3771 , { INF, INF, INF, INF, INF }
3772 }
3773 , {{ INF, INF, INF, INF, INF }
3774 , { INF, INF, INF, INF, INF }
3775 , { INF, INF, INF, INF, INF }
3776 , { INF, INF, INF, INF, INF }
3777 , { INF, INF, INF, INF, INF }
3778 }
3779 }
3780 , {{{ INF, INF, INF, INF, INF }
3781 , { INF, INF, INF, INF, INF }
3782 , { INF, INF, INF, INF, INF }
3783 , { INF, INF, INF, INF, INF }
3784 , { INF, INF, INF, INF, INF }
3785 }
3786 , {{ INF, INF, INF, INF, INF }
3787 , { INF, INF, INF, INF, INF }
3788 , { INF, INF, INF, INF, INF }
3789 , { INF, INF, INF, INF, INF }
3790 , { INF, INF, INF, INF, INF }
3791 }
3792 , {{ INF, INF, INF, INF, INF }
3793 , { INF, INF, INF, INF, INF }
3794 , { INF, INF, INF, INF, INF }
3795 , { INF, INF, INF, INF, INF }
3796 , { INF, INF, INF, INF, INF }
3797 }
3798 , {{ INF, INF, INF, INF, INF }
3799 , { INF, INF, INF, INF, INF }
3800 , { INF, INF, INF, INF, INF }
3801 , { INF, INF, INF, INF, INF }
3802 , { INF, INF, INF, INF, INF }
3803 }
3804 , {{ INF, INF, INF, INF, INF }
3805 , { INF, INF, INF, INF, INF }
3806 , { INF, INF, INF, INF, INF }
3807 , { INF, INF, INF, INF, INF }
3808 , { INF, INF, INF, INF, INF }
3809 }
3810 }
3811 , {{{ INF, INF, INF, INF, INF }
3812 , { INF, INF, INF, INF, INF }
3813 , { INF, INF, INF, INF, INF }
3814 , { INF, INF, INF, INF, INF }
3815 , { INF, INF, INF, INF, INF }
3816 }
3817 , {{ INF, INF, INF, INF, INF }
3818 , { INF, INF, INF, INF, INF }
3819 , { INF, INF, INF, INF, INF }
3820 , { INF, INF, INF, INF, INF }
3821 , { INF, INF, INF, INF, INF }
3822 }
3823 , {{ INF, INF, INF, INF, INF }
3824 , { INF, INF, INF, INF, INF }
3825 , { INF, INF, INF, INF, INF }
3826 , { INF, INF, INF, INF, INF }
3827 , { INF, INF, INF, INF, INF }
3828 }
3829 , {{ INF, INF, INF, INF, INF }
3830 , { INF, INF, INF, INF, INF }
3831 , { INF, INF, INF, INF, INF }
3832 , { INF, INF, INF, INF, INF }
3833 , { INF, INF, INF, INF, INF }
3834 }
3835 , {{ INF, INF, INF, INF, INF }
3836 , { INF, INF, INF, INF, INF }
3837 , { INF, INF, INF, INF, INF }
3838 , { INF, INF, INF, INF, INF }
3839 , { INF, INF, INF, INF, INF }
3840 }
3841 }
3842 , {{{ INF, INF, INF, INF, INF }
3843 , { INF, INF, INF, INF, INF }
3844 , { INF, INF, INF, INF, INF }
3845 , { INF, INF, INF, INF, INF }
3846 , { INF, INF, INF, INF, INF }
3847 }
```

```
3848 , {{ INF, INF, INF, INF, INF }
3849 , { INF, INF, INF, INF, INF }
3850 , { INF, INF, INF, INF, INF }
3851 , { INF, INF, INF, INF, INF }
3852 , { INF, INF, INF, INF, INF }
3853 }
3854 , {{ INF, INF, INF, INF, INF }
3855 , { INF, INF, INF, INF, INF }
3856 , { INF, INF, INF, INF, INF }
3857 , { INF, INF, INF, INF, INF }
3858 , { INF, INF, INF, INF, INF }
3859 }
3860 , {{ INF, INF, INF, INF, INF }
3861 , { INF, INF, INF, INF, INF }
3862 , { INF, INF, INF, INF, INF }
3863 , { INF, INF, INF, INF, INF }
3864 , { INF, INF, INF, INF, INF }
3865 }
3866 , {{ INF, INF, INF, INF, INF }
3867 , { INF, INF, INF, INF, INF }
3868 , { INF, INF, INF, INF, INF }
3869 , { INF, INF, INF, INF, INF }
3870 , { INF, INF, INF, INF, INF }
3871 }
3872 }
3873 , {{{ INF, INF, INF, INF, INF }
3874 , { INF, INF, INF, INF, INF }
3875 , { INF, INF, INF, INF, INF }
3876 , { INF, INF, INF, INF, INF }
3877 , { INF, INF, INF, INF, INF }
3878 }
3879 , {{ INF, INF, INF, INF, INF }
3880 , { INF, INF, INF, INF, INF }
3881 , { INF, INF, INF, INF, INF }
3882 , { INF, INF, INF, INF, INF }
3883 , { INF, INF, INF, INF, INF }
3884 }
3885 , {{ INF, INF, INF, INF, INF }
3886 , { INF, INF, INF, INF, INF }
3887 , { INF, INF, INF, INF, INF }
3888 , { INF, INF, INF, INF, INF }
3889 , { INF, INF, INF, INF, INF }
3890 }
3891 , {{ INF, INF, INF, INF, INF }
3892 , { INF, INF, INF, INF, INF }
3893 , { INF, INF, INF, INF, INF }
3894 , { INF, INF, INF, INF, INF }
3895 , { INF, INF, INF, INF, INF }
3896 }
3897 , {{ INF, INF, INF, INF, INF }
3898 , { INF, INF, INF, INF, INF }
3899 , { INF, INF, INF, INF, INF }
3900 , { INF, INF, INF, INF, INF }
3901 , { INF, INF, INF, INF, INF }
3902 }
3903 }
3904 }
3905 , {{{{ 300, 300, 270, 270, 290}
3906 , { 300, 300, 270, 270, 290}
3907 , { 290, 290, 250, 270, 250}
3908 , { 300, 300, 270, 270, 270}
3909 , { 270, 270, 240, 260, 240}
3910 }
3911 , {{ 290, 270, 230, 230, 290}
3912 , { 290, 270, 230, 230, 290}
3913 , { 260, 260, 220, 220, 220}
3914 , { 190, 170, 190, 130, 190}
3915 , { 260, 260, 220, 220, 220}
3916 }
3917 , {{ 300, 300, 270, 270, 270}
3918 , { 300, 300, 270, 270, 270}
3919 , { 290, 290, 250, 270, 250}
3920 , { 300, 300, 270, 270, 270}
3921 , { 270, 270, 240, 260, 240}
3922 }
3923 , {{ 260, 260, 220, 220, 220}
3924 , { 190, 170, 190, 130, 190}
3925 , { 260, 260, 220, 220, 220}
3926 , { 210, 130, 80, 210, 210}
3927 , { 260, 260, 220, 220, 220}
3928 }
3929 , {{ 300, 300, 270, 270, 270}
3930 , { 300, 300, 270, 270, 270}
3931 , { 270, 270, 240, 260, 240}
3932 , { 300, 300, 270, 270, 270}
3933 , { 240, 240, 150, 150, 150}
3934 }
```

```
3935      }
3936 ,{{ 300, 300, 270, 270, 270}
3937 ,{ 300, 300, 270, 230, 270}
3938 ,{ 290, 290, 250, 270, 250}
3939 ,{ 300, 300, 270, 230, 270}
3940 ,{ 270, 270, 240, 260, 240}
3941 }
3942 ,{{ 270, 270, 230, 190, 230}
3943 ,{ 270, 270, 230, 190, 230}
3944 ,{ 260, 260, 220, 180, 220}
3945 ,{ 170, 170, 130, 90, 130}
3946 ,{ 260, 260, 220, 180, 220}
3947 }
3948 ,{{ 300, 300, 270, 270, 270}
3949 ,{ 300, 300, 270, 230, 270}
3950 ,{ 290, 290, 250, 270, 250}
3951 ,{ 300, 300, 270, 230, 270}
3952 ,{ 270, 270, 240, 260, 240}
3953 }
3954 ,{{ 260, 260, 220, 180, 220}
3955 ,{ 170, 170, 130, 90, 130}
3956 ,{ 260, 260, 220, 180, 220}
3957 ,{ 170, 110, 80, 170, 80}
3958 ,{ 260, 260, 220, 180, 220}
3959 }
3960 ,{{ 300, 300, 270, 260, 270}
3961 ,{ 300, 300, 270, 230, 270}
3962 ,{ 270, 270, 240, 260, 240}
3963 ,{ 300, 300, 270, 230, 270}
3964 ,{ 240, 240, 150, 110, 150}
3965 }
3966 }
3967 ,{{ 270, 270, 270, 270, 270}
3968 ,{ 270, 270, 270, 270, 270}
3969 ,{ 250, 250, 250, 250, 250}
3970 ,{ 270, 270, 270, 270, 270}
3971 ,{ 240, 240, 240, 240, 240}
3972 }
3973 ,{{ 230, 230, 230, 230, 230}
3974 ,{ 230, 230, 230, 230, 230}
3975 ,{ 220, 220, 220, 220, 220}
3976 ,{ 190, 130, 190, 130, 190}
3977 ,{ 220, 220, 220, 220, 220}
3978 }
3979 ,{{ 270, 270, 270, 270, 270}
3980 ,{ 270, 270, 270, 270, 270}
3981 ,{ 250, 250, 250, 250, 250}
3982 ,{ 270, 270, 270, 270, 270}
3983 ,{ 240, 240, 240, 240, 240}
3984 }
3985 ,{{ 220, 220, 220, 220, 220}
3986 ,{ 190, 130, 190, 130, 190}
3987 ,{ 220, 220, 220, 220, 220}
3988 ,{ 80, 80, 80, 80, 80}
3989 ,{ 220, 220, 220, 220, 220}
3990 }
3991 ,{{ 270, 270, 270, 270, 270}
3992 ,{ 270, 270, 270, 270, 270}
3993 ,{ 240, 240, 240, 240, 240}
3994 ,{ 270, 270, 270, 270, 270}
3995 ,{ 150, 150, 150, 150, 150}
3996 }
3997 }
3998 ,{{ 270, 230, 270, 210, 270}
3999 ,{ 270, 190, 270, 140, 270}
4000 ,{ 250, 230, 250, 120, 250}
4001 ,{ 270, 190, 270, 210, 270}
4002 ,{ 240, 220, 240, 150, 240}
4003 }
4004 ,{{ 230, 150, 230, 130, 230}
4005 ,{ 230, 150, 230, 100, 230}
4006 ,{ 220, 140, 220, 90, 220}
4007 ,{ 130, 50, 130, 130, 130}
4008 ,{ 220, 140, 220, 90, 220}
4009 }
4010 ,{{ 270, 230, 270, 140, 270}
4011 ,{ 270, 190, 270, 140, 270}
4012 ,{ 250, 230, 250, 120, 250}
4013 ,{ 270, 190, 270, 140, 270}
4014 ,{ 240, 220, 240, 110, 240}
4015 }
4016 ,{{ 220, 140, 220, 210, 220}
4017 ,{ 130, 50, 130, 130, 130}
4018 ,{ 220, 140, 220, 90, 220}
4019 ,{ 210, 130, 80, 210, 80}
4020 ,{ 220, 140, 220, 90, 220}
4021 }
```

```
4022 , {{ 270, 220, 270, 150, 270} }
4023 , { 270, 190, 270, 140, 270}
4024 , { 240, 220, 240, 110, 240}
4025 , { 270, 190, 270, 140, 270}
4026 , { 150, 70, 150, 150, 150}
4027 }
4028 }
4029 , {{{ 290, 270, 270, 270, 290} }
4030 , { 290, 270, 270, 270, 290}
4031 , { 250, 250, 250, 250, 250}
4032 , { 270, 270, 270, 270, 270}
4033 , { 240, 240, 240, 240, 240}
4034 }
4035 , {{ 290, 230, 230, 230, 290} }
4036 , { 290, 230, 230, 230, 290}
4037 , { 220, 220, 220, 220, 220}
4038 , { 190, 130, 190, 130, 130}
4039 , { 220, 220, 220, 220, 220}
4040 }
4041 , {{ 270, 270, 270, 270, 270} }
4042 , { 270, 270, 270, 270, 270}
4043 , { 250, 250, 250, 250, 250}
4044 , { 270, 270, 270, 270, 270}
4045 , { 240, 240, 240, 240, 240}
4046 }
4047 , {{ 220, 220, 220, 220, 220} }
4048 , { 190, 130, 190, 130, 130}
4049 , { 220, 220, 220, 220, 220}
4050 , { 210, 80, 80, 80, 210}
4051 , { 220, 220, 220, 220, 220}
4052 }
4053 , {{ 270, 270, 270, 270, 270} }
4054 , { 270, 270, 270, 270, 270}
4055 , { 240, 240, 240, 240, 240}
4056 , { 270, 270, 270, 270, 270}
4057 , { 150, 150, 150, 150, 150}
4058 }
4059 }
4060 }
4061 , {{{{ 300, 280, 240, 240, 300} }
4062 , { 300, 280, 240, 240, 300}
4063 , { 260, 260, 220, 240, 220}
4064 , { 250, 250, 210, 210, 210}
4065 , { 250, 250, 220, 240, 220}
4066 }
4067 , {{ 300, 280, 240, 240, 300} }
4068 , { 300, 280, 240, 240, 300}
4069 , { 250, 250, 220, 220, 220}
4070 , { 100, 70, 100, 40, 100}
4071 , { 250, 250, 220, 220, 220}
4072 }
4073 , {{ 250, 250, 220, 240, 220} }
4074 , { 250, 250, 210, 210, 210}
4075 , { 250, 250, 220, 240, 220}
4076 , { 250, 250, 210, 210, 210}
4077 , { 250, 250, 220, 240, 220}
4078 }
4079 , {{ 250, 250, 220, 220, 220} }
4080 , { 160, 140, 160, 100, 160}
4081 , { 250, 250, 220, 220, 220}
4082 , { 210, 130, 80, 210, 210}
4083 , { 250, 250, 220, 220, 220}
4084 }
4085 , {{ 260, 260, 220, 240, 220} }
4086 , { 250, 250, 210, 210, 210}
4087 , { 260, 260, 220, 240, 220}
4088 , { 250, 250, 210, 210, 210}
4089 , { 240, 240, 140, 140, 140}
4090 }
4091 }
4092 , {{{ 280, 280, 240, 240, 240} }
4093 , { 280, 280, 240, 200, 240}
4094 , { 260, 260, 220, 240, 220}
4095 , { 250, 250, 210, 170, 210}
4096 , { 250, 250, 220, 240, 220}
4097 }
4098 , {{ 280, 280, 240, 200, 240} }
4099 , { 280, 280, 240, 200, 240}
4100 , { 250, 250, 220, 180, 220}
4101 , { 70, 70, 40, 0, 40}
4102 , { 250, 250, 220, 180, 220}
4103 }
4104 , {{ 250, 250, 220, 240, 220} }
4105 , { 250, 250, 210, 170, 210}
4106 , { 250, 250, 220, 240, 220}
4107 , { 250, 250, 210, 170, 210}
4108 , { 250, 250, 220, 240, 220}
```

```
4109      }
4110 ,{{ 250, 250, 220, 180, 220}
4111 ,{ 140, 140, 100, 60, 100}
4112 ,{ 250, 250, 220, 180, 220}
4113 ,{ 170, 110, 80, 170, 80}
4114 ,{ 250, 250, 220, 180, 220}
4115 }
4116 ,{{ 260, 260, 220, 240, 220}
4117 ,{ 250, 250, 210, 170, 210}
4118 ,{ 260, 260, 220, 240, 220}
4119 ,{ 250, 250, 210, 170, 210}
4120 ,{ 240, 240, 140, 100, 140}
4121 }
4122 }
4123 ,{{{{ 240, 240, 240, 240, 240}
4124 ,{ 240, 240, 240, 240, 240}
4125 ,{ 220, 220, 220, 220, 220}
4126 ,{ 210, 210, 210, 210, 210}
4127 ,{ 220, 220, 220, 220, 220}
4128 }
4129 ,{{ 240, 240, 240, 240, 240}
4130 ,{ 240, 240, 240, 240, 240}
4131 ,{ 220, 220, 220, 220, 220}
4132 ,{ 100, 40, 100, 40, 100}
4133 ,{ 220, 220, 220, 220, 220}
4134 }
4135 ,{{ 220, 220, 220, 220, 220}
4136 ,{ 210, 210, 210, 210, 210}
4137 ,{ 220, 220, 220, 220, 220}
4138 ,{ 210, 210, 210, 210, 210}
4139 ,{ 220, 220, 220, 220, 220}
4140 }
4141 ,{{ 220, 220, 220, 220, 220}
4142 ,{ 160, 100, 160, 100, 160}
4143 ,{ 220, 220, 220, 220, 220}
4144 ,{ 80, 80, 80, 80, 80}
4145 ,{ 220, 220, 220, 220, 220}
4146 }
4147 ,{{ 220, 220, 220, 220, 220}
4148 ,{ 210, 210, 210, 210, 210}
4149 ,{ 220, 220, 220, 220, 220}
4150 ,{ 210, 210, 210, 210, 210}
4151 ,{ 140, 140, 140, 140, 140}
4152 }
4153 }
4154 ,{{{{ 240, 200, 240, 210, 240}
4155 ,{ 240, 160, 240, 110, 240}
4156 ,{ 220, 200, 220, 90, 220}
4157 ,{ 210, 130, 210, 210, 210}
4158 ,{ 220, 200, 220, 140, 220}
4159 }
4160 ,{{ 240, 160, 240, 110, 240}
4161 ,{ 240, 160, 240, 110, 240}
4162 ,{ 220, 140, 220, 90, 220}
4163 ,{ 40, -40, 40, 40, 40}
4164 ,{ 220, 140, 220, 90, 220}
4165 }
4166 ,{{ 220, 200, 220, 90, 220}
4167 ,{ 210, 130, 210, 80, 210}
4168 ,{ 220, 200, 220, 90, 220}
4169 ,{ 210, 130, 210, 80, 210}
4170 ,{ 220, 200, 220, 90, 220}
4171 }
4172 ,{{ 220, 140, 220, 210, 220}
4173 ,{ 100, 20, 100, 100, 100}
4174 ,{ 220, 140, 220, 90, 220}
4175 ,{ 210, 130, 80, 210, 80}
4176 ,{ 220, 140, 220, 90, 220}
4177 }
4178 ,{{ 220, 200, 220, 140, 220}
4179 ,{ 210, 130, 210, 80, 210}
4180 ,{ 220, 200, 220, 90, 220}
4181 ,{ 210, 130, 210, 80, 210}
4182 ,{ 140, 60, 140, 140, 140}
4183 }
4184 }
4185 ,{{{{ 300, 240, 240, 240, 300}
4186 ,{ 300, 240, 240, 240, 300}
4187 ,{ 220, 220, 220, 220, 220}
4188 ,{ 210, 210, 210, 210, 210}
4189 ,{ 220, 220, 220, 220, 220}
4190 }
4191 ,{{ 300, 240, 240, 240, 300}
4192 ,{ 300, 240, 240, 240, 300}
4193 ,{ 220, 220, 220, 220, 220}
4194 ,{ 100, 40, 100, 40, 40}
4195 ,{ 220, 220, 220, 220, 220}
```

```
4196      }
4197 ,{{ 220, 220, 220, 220, 220}
4198 ,{ 210, 210, 210, 210, 210}
4199 ,{ 220, 220, 220, 220, 220}
4200 ,{ 210, 210, 210, 210, 210}
4201 ,{ 220, 220, 220, 220, 220}
4202 }
4203 ,{{ 220, 220, 220, 220, 220}
4204 ,{ 160, 100, 160, 100, 100}
4205 ,{ 220, 220, 220, 220, 220}
4206 ,{ 210, 80, 80, 80, 210}
4207 ,{ 220, 220, 220, 220, 220}
4208 }
4209 ,{{ 220, 220, 220, 220, 220}
4210 ,{ 210, 210, 210, 210, 210}
4211 ,{ 220, 220, 220, 220, 220}
4212 ,{ 210, 210, 210, 210, 210}
4213 ,{ 140, 140, 140, 140, 140}
4214 }
4215 }
4216 }
4217 ,{{{{ 430, 430, 370, 370, 430}
4218 ,{ 430, 410, 370, 370, 430}
4219 ,{ 370, 370, 340, 360, 340}
4220 ,{ 370, 370, 340, 340, 340}
4221 ,{ 430, 430, 340, 360, 340}
4222 }
4223 ,{ { 430, 410, 370, 370, 430}
4224 ,{ { 430, 410, 370, 370, 430}
4225 ,{ { 370, 370, 340, 340, 340}
4226 ,{ { 320, 290, 320, 260, 320}
4227 ,{ { 370, 370, 340, 340, 340}
4228 }
4229 ,{{ 370, 370, 340, 360, 340}
4230 ,{ { 370, 370, 340, 340, 340}
4231 ,{ { 370, 370, 340, 360, 340}
4232 ,{ { 370, 370, 340, 340, 340}
4233 ,{ { 370, 370, 340, 360, 340}
4234 }
4235 ,{{ { 370, 370, 360, 340, 360}
4236 ,{ { 360, 330, 360, 300, 360}
4237 ,{ { 370, 370, 340, 340, 340}
4238 ,{ { 340, 260, 210, 340, 340}
4239 ,{ { 370, 370, 340, 340, 340}
4240 }
4241 ,{{ { 430, 430, 340, 360, 340}
4242 ,{ { 370, 370, 340, 340, 340}
4243 ,{ { 370, 370, 340, 360, 340}
4244 ,{ { 370, 370, 340, 340, 340}
4245 ,{ { 430, 430, 340, 340, 340}
4246 }
4247 }
4248 ,{{{{ 430, 430, 370, 360, 370}
4249 ,{ { 410, 410, 370, 330, 370}
4250 ,{ { 370, 370, 340, 360, 340}
4251 ,{ { 370, 370, 340, 300, 340}
4252 ,{ { 430, 430, 340, 360, 340}
4253 }
4254 ,{{ { 410, 410, 370, 330, 370}
4255 ,{ { 410, 410, 370, 330, 370}
4256 ,{ { 370, 370, 340, 300, 340}
4257 ,{ { 290, 290, 260, 220, 260}
4258 ,{ { 370, 370, 340, 300, 340}
4259 }
4260 ,{{ { 370, 370, 340, 360, 340}
4261 ,{ { 370, 370, 340, 300, 340}
4262 ,{ { 370, 370, 340, 360, 340}
4263 ,{ { 370, 370, 340, 300, 340}
4264 ,{ { 370, 370, 340, 360, 340}
4265 }
4266 ,{{ { 370, 370, 340, 300, 340}
4267 ,{ { 330, 330, 300, 260, 300}
4268 ,{ { 370, 370, 340, 300, 340}
4269 ,{ { 300, 240, 210, 300, 210}
4270 ,{ { 370, 370, 340, 300, 340}
4271 }
4272 ,{{ { 430, 430, 340, 360, 340}
4273 ,{ { 370, 370, 340, 300, 340}
4274 ,{ { 370, 370, 340, 360, 340}
4275 ,{ { 370, 370, 340, 300, 340}
4276 ,{ { 430, 430, 340, 300, 340}
4277 }
4278 }
4279 ,{{{{ 370, 370, 370, 370, 370}
4280 ,{ { 370, 370, 370, 370, 370}
4281 ,{ { 340, 340, 340, 340, 340}
4282 ,{ { 340, 340, 340, 340, 340}
```

```
4283 , { 340, 340, 340, 340, 340}
4284 }
4285 , {{ 370, 370, 370, 370, 370}
4286 , { 370, 370, 370, 370, 370}
4287 , { 340, 340, 340, 340, 340}
4288 , { 320, 260, 320, 260, 320}
4289 , { 340, 340, 340, 340, 340}
4290 }
4291 , {{ 340, 340, 340, 340, 340}
4292 , { 340, 340, 340, 340, 340}
4293 , { 340, 340, 340, 340, 340}
4294 , { 340, 340, 340, 340, 340}
4295 , { 340, 340, 340, 340, 340}
4296 }
4297 , {{ 360, 340, 360, 340, 360}
4298 , { 360, 300, 360, 300, 360}
4299 , { 340, 340, 340, 340, 340}
4300 , { 210, 210, 210, 210, 210}
4301 , { 340, 340, 340, 340, 340}
4302 }
4303 , {{ 340, 340, 340, 340, 340}
4304 , { 340, 340, 340, 340, 340}
4305 , { 340, 340, 340, 340, 340}
4306 , { 340, 340, 340, 340, 340}
4307 , { 340, 340, 340, 340, 340}
4308 }
4309 }
4310 , {{ { 370, 320, 370, 340, 370}
4311 , { 370, 290, 370, 300, 370}
4312 , { 340, 320, 340, 210, 340}
4313 , { 340, 260, 340, 340, 340}
4314 , { 340, 320, 340, 340, 340}
4315 }
4316 , {{ 370, 290, 370, 260, 370}
4317 , { 370, 290, 370, 240, 370}
4318 , { 340, 260, 340, 210, 340}
4319 , { 260, 180, 260, 260, 260}
4320 , { 340, 260, 340, 210, 340}
4321 }
4322 , {{ 340, 320, 340, 210, 340}
4323 , { 340, 260, 340, 210, 340}
4324 , { 340, 320, 340, 210, 340}
4325 , { 340, 260, 340, 210, 340}
4326 , { 340, 320, 340, 210, 340}
4327 }
4328 , {{ 340, 260, 340, 340, 340}
4329 , { 300, 220, 300, 300, 300}
4330 , { 340, 260, 340, 210, 340}
4331 , { 340, 260, 210, 340, 210}
4332 , { 340, 260, 340, 210, 340}
4333 }
4334 , {{ 340, 320, 340, 340, 340}
4335 , { 340, 260, 340, 210, 340}
4336 , { 340, 320, 340, 210, 340}
4337 , { 340, 260, 340, 210, 340}
4338 , { 340, 260, 340, 340, 340}
4339 }
4340 }
4341 , {{ { 430, 370, 370, 370, 430}
4342 , { 430, 370, 370, 370, 430}
4343 , { 340, 340, 340, 340, 340}
4344 , { 340, 340, 340, 340, 340}
4345 , { 340, 340, 340, 340, 340}
4346 }
4347 , {{ 430, 370, 370, 370, 430}
4348 , { 430, 370, 370, 370, 430}
4349 , { 340, 340, 340, 340, 340}
4350 , { 320, 260, 320, 260, 260}
4351 , { 340, 340, 340, 340, 340}
4352 }
4353 , {{ { 340, 340, 340, 340, 340}
4354 , { 340, 340, 340, 340, 340}
4355 , { 340, 340, 340, 340, 340}
4356 , { 340, 340, 340, 340, 340}
4357 , { 340, 340, 340, 340, 340}
4358 }
4359 , {{ { 360, 340, 360, 340, 340}
4360 , { 360, 300, 360, 300, 300}
4361 , { 340, 340, 340, 340, 340}
4362 , { 340, 210, 210, 210, 340}
4363 , { 340, 340, 340, 340, 340}
4364 }
4365 , {{ { 340, 340, 340, 340, 340}
4366 , { 340, 340, 340, 340, 340}
4367 , { 340, 340, 340, 340, 340}
4368 , { 340, 340, 340, 340, 340}
4369 , { 340, 340, 340, 340, 340}
```

```
4370      }
4371      }
4372  }
4373 ,{{{{ 400, 400, 400, 360, 400}
4374 ,{ 400, 370, 400, 360, 400}
4375 ,{ 340, 340, 310, 330, 310}
4376 ,{ 340, 340, 310, 310, 310}
4377 ,{ 400, 400, 310, 330, 310}
4378 }
4379 ,{{ 360, 360, 310, 360, 330}
4380 ,{ 360, 360, 270, 360, 330}
4381 ,{ 340, 340, 310, 310, 310}
4382 ,{ 230, 220, 230, 170, 230}
4383 ,{ 340, 340, 310, 310, 310}
4384 }
4385 ,{{ 340, 340, 310, 330, 310}
4386 ,{ 340, 340, 310, 310, 310}
4387 ,{ 340, 340, 310, 330, 310}
4388 ,{ 340, 340, 310, 310, 310}
4389 ,{ 340, 340, 310, 330, 310}
4390 }
4391 ,{{ 400, 370, 400, 340, 400}
4392 ,{ 400, 370, 400, 340, 400}
4393 ,{ 340, 340, 310, 310, 310}
4394 ,{ 310, 230, 180, 310, 310}
4395 ,{ 340, 340, 310, 310, 310}
4396 }
4397 ,{{ 400, 400, 310, 330, 310}
4398 ,{ 340, 340, 310, 310, 310}
4399 ,{ 340, 340, 310, 330, 310}
4400 ,{ 340, 340, 310, 310, 310}
4401 ,{ 400, 400, 310, 310, 310}
4402 }
4403 }
4404 ,{{{{ 400, 400, 340, 360, 340}
4405 ,{ 370, 370, 340, 360, 340}
4406 ,{ 340, 340, 310, 330, 310}
4407 ,{ 340, 340, 310, 270, 310}
4408 ,{ 400, 400, 310, 330, 310}
4409 }
4410 ,{{ 360, 360, 310, 360, 310}
4411 ,{ 360, 360, 270, 360, 270}
4412 ,{ 340, 340, 310, 270, 310}
4413 ,{ 220, 220, 170, 130, 170}
4414 ,{ 340, 340, 310, 270, 310}
4415 }
4416 ,{{ 340, 340, 310, 330, 310}
4417 ,{ 340, 340, 310, 270, 310}
4418 ,{ 340, 340, 310, 330, 310}
4419 ,{ 340, 340, 310, 270, 310}
4420 ,{ 340, 340, 310, 330, 310}
4421 }
4422 ,{{ 370, 370, 340, 300, 340}
4423 ,{ 370, 370, 340, 300, 340}
4424 ,{ 340, 340, 310, 270, 310}
4425 ,{ 270, 210, 180, 270, 180}
4426 ,{ 340, 340, 310, 270, 310}
4427 }
4428 ,{{ 400, 400, 310, 330, 310}
4429 ,{ 340, 340, 310, 270, 310}
4430 ,{ 340, 340, 310, 330, 310}
4431 ,{ 340, 340, 310, 270, 310}
4432 ,{ 400, 400, 310, 270, 310}
4433 }
4434 }
4435 ,{{{{ 400, 340, 400, 340, 400}
4436 ,{ 400, 340, 400, 340, 400}
4437 ,{ 310, 310, 310, 310, 310}
4438 ,{ 310, 310, 310, 310, 310}
4439 ,{ 310, 310, 310, 310, 310}
4440 }
4441 ,{{ 310, 310, 310, 310, 310}
4442 ,{ 270, 270, 270, 270, 270}
4443 ,{ 310, 310, 310, 310, 310}
4444 ,{ 230, 170, 230, 170, 230}
4445 ,{ 310, 310, 310, 310, 310}
4446 }
4447 ,{{ 310, 310, 310, 310, 310}
4448 ,{ 310, 310, 310, 310, 310}
4449 ,{ 310, 310, 310, 310, 310}
4450 ,{ 310, 310, 310, 310, 310}
4451 ,{ 310, 310, 310, 310, 310}
4452 }
4453 ,{{ 400, 340, 400, 340, 400}
4454 ,{ 400, 340, 400, 340, 400}
4455 ,{ 310, 310, 310, 310, 310}
4456 ,{ 180, 180, 180, 180, 180}
```

```
4457 , { 310, 310, 310, 310, 310}
4458 }
4459 , {{ 310, 310, 310, 310, 310}
4460 , { 310, 310, 310, 310, 310}
4461 , { 310, 310, 310, 310, 310}
4462 , { 310, 310, 310, 310, 310}
4463 , { 310, 310, 310, 310, 310}
4464 }
4465 }
4466 , {{{ 340, 290, 340, 340, 340}
4467 , { 340, 260, 340, 340, 340}
4468 , { 310, 290, 310, 180, 310}
4469 , { 310, 230, 310, 310, 310}
4470 , { 310, 290, 310, 310, 310}
4471 }
4472 , {{ 310, 230, 310, 180, 310}
4473 , { 270, 190, 270, 140, 270}
4474 , { 310, 230, 310, 180, 310}
4475 , { 170, 20, 170, 170, 170}
4476 , { 310, 230, 310, 180, 310}
4477 }
4478 , {{ 310, 290, 310, 180, 310}
4479 , { 310, 230, 310, 180, 310}
4480 , { 310, 290, 310, 180, 310}
4481 , { 310, 230, 310, 180, 310}
4482 , { 310, 290, 310, 180, 310}
4483 }
4484 , {{ 340, 260, 340, 340, 340}
4485 , { 340, 260, 340, 340, 340}
4486 , { 310, 230, 310, 180, 310}
4487 , { 310, 230, 180, 310, 180}
4488 , { 310, 230, 310, 180, 310}
4489 }
4490 , {{ 310, 290, 310, 310, 310}
4491 , { 310, 230, 310, 180, 310}
4492 , { 310, 290, 310, 180, 310}
4493 , { 310, 230, 310, 180, 310}
4494 , { 310, 230, 310, 310, 310}
4495 }
4496 }
4497 , {{{ 400, 340, 400, 340, 340}
4498 , { 400, 340, 400, 340, 340}
4499 , { 310, 310, 310, 310, 310}
4500 , { 310, 310, 310, 310, 310}
4501 , { 310, 310, 310, 310, 310}
4502 }
4503 , {{ 330, 310, 310, 310, 330}
4504 , { 330, 270, 270, 270, 330}
4505 , { 310, 310, 310, 310, 310}
4506 , { 230, 170, 230, 170, 170}
4507 , { 310, 310, 310, 310, 310}
4508 }
4509 , {{ 310, 310, 310, 310, 310}
4510 , { 310, 310, 310, 310, 310}
4511 , { 310, 310, 310, 310, 310}
4512 , { 310, 310, 310, 310, 310}
4513 , { 310, 310, 310, 310, 310}
4514 }
4515 , {{ 400, 340, 400, 340, 340}
4516 , { 400, 340, 400, 340, 340}
4517 , { 310, 310, 310, 310, 310}
4518 , { 310, 180, 180, 180, 310}
4519 , { 310, 310, 310, 310, 310}
4520 }
4521 , {{ 310, 310, 310, 310, 310}
4522 , { 310, 310, 310, 310, 310}
4523 , { 310, 310, 310, 310, 310}
4524 , { 310, 310, 310, 310, 310}
4525 , { 310, 310, 310, 310, 310}
4526 }
4527 }
4528 }
4529 , {{{{ 370, 340, 310, 310, 370}
4530 , { 370, 340, 310, 310, 370}
4531 , { 320, 320, 290, 310, 290}
4532 , { 330, 330, 290, 290, 290}
4533 , { 320, 320, 290, 310, 290}
4534 }
4535 , {{ 370, 340, 310, 310, 370}
4536 , { 370, 340, 310, 310, 370}
4537 , { 320, 320, 280, 280, 280}
4538 , { 240, 220, 240, 180, 240}
4539 , { 320, 320, 280, 280, 280}
4540 }
4541 , {{ 330, 330, 290, 310, 290}
4542 , { 330, 330, 290, 290, 290}
4543 , { 320, 320, 290, 310, 290}
```

```
4544 , { 330, 330, 290, 290, 290}
4545 , { 320, 320, 290, 310, 290}
4546 }
4547 , {{ 320, 320, 310, 280, 310}
4548 , { 310, 290, 310, 250, 310}
4549 , { 320, 320, 280, 280, 280}
4550 , { 260, 180, 130, 260, 260}
4551 , { 320, 320, 280, 280, 280}
4552 }
4553 , {{ 330, 330, 290, 310, 290}
4554 , { 330, 330, 290, 290, 290}
4555 , { 320, 320, 290, 310, 290}
4556 , { 330, 330, 290, 290, 290}
4557 , { 290, 290, 200, 200, 200}
4558 }
4559 }
4560 , {{{ 340, 340, 310, 310, 310}
4561 , { 340, 340, 310, 270, 310}
4562 , { 320, 320, 290, 310, 290}
4563 , { 330, 330, 290, 250, 290}
4564 , { 320, 320, 290, 310, 290}
4565 }
4566 , {{ 340, 340, 310, 270, 310}
4567 , { 340, 340, 310, 270, 310}
4568 , { 320, 320, 280, 240, 280}
4569 , { 220, 220, 180, 140, 180}
4570 , { 320, 320, 280, 240, 280}
4571 }
4572 , {{ 330, 330, 290, 310, 290}
4573 , { 330, 330, 290, 250, 290}
4574 , { 320, 320, 290, 310, 290}
4575 , { 330, 330, 290, 250, 290}
4576 , { 320, 320, 290, 310, 290}
4577 }
4578 , {{ 320, 320, 280, 240, 280}
4579 , { 290, 290, 250, 210, 250}
4580 , { 320, 320, 280, 240, 280}
4581 , { 220, 170, 130, 220, 130}
4582 , { 320, 320, 280, 240, 280}
4583 }
4584 , {{ 330, 330, 290, 310, 290}
4585 , { 330, 330, 290, 250, 290}
4586 , { 320, 320, 290, 310, 290}
4587 , { 330, 330, 290, 250, 290}
4588 , { 290, 290, 200, 160, 200}
4589 }
4590 }
4591 , {{{ 310, 310, 310, 310, 310}
4592 , { 310, 310, 310, 310, 310}
4593 , { 290, 290, 290, 290, 290}
4594 , { 290, 290, 290, 290, 290}
4595 , { 290, 290, 290, 290, 290}
4596 }
4597 , {{ 310, 310, 310, 310, 310}
4598 , { 310, 310, 310, 310, 310}
4599 , { 280, 280, 280, 280, 280}
4600 , { 240, 180, 240, 180, 240}
4601 , { 280, 280, 280, 280, 280}
4602 }
4603 , {{ 290, 290, 290, 290, 290}
4604 , { 290, 290, 290, 290, 290}
4605 , { 290, 290, 290, 290, 290}
4606 , { 290, 290, 290, 290, 290}
4607 , { 290, 290, 290, 290, 290}
4608 }
4609 , {{ 310, 280, 310, 280, 310}
4610 , { 310, 250, 310, 250, 310}
4611 , { 280, 280, 280, 280, 280}
4612 , { 130, 130, 130, 130, 130}
4613 , { 280, 280, 280, 280, 280}
4614 }
4615 , {{ 290, 290, 290, 290, 290}
4616 , { 290, 290, 290, 290, 290}
4617 , { 290, 290, 290, 290, 290}
4618 , { 290, 290, 290, 290, 290}
4619 , { 200, 200, 200, 200, 200}
4620 }
4621 }
4622 , {{{ 310, 270, 310, 260, 310}
4623 , { 310, 230, 310, 250, 310}
4624 , { 290, 270, 290, 160, 290}
4625 , { 290, 210, 290, 260, 290}
4626 , { 290, 270, 290, 200, 290}
4627 }
4628 , {{ 310, 230, 310, 180, 310}
4629 , { 310, 230, 310, 180, 310}
4630 , { 280, 200, 280, 150, 280}
```

```
4631 , { 180, 100, 180, 180, 180}
4632 , { 280, 200, 280, 150, 280}
4633 }
4634 , {{ 290, 270, 290, 160, 290}
4635 , { 290, 210, 290, 160, 290}
4636 , { 290, 270, 290, 160, 290}
4637 , { 290, 210, 290, 160, 290}
4638 , { 290, 270, 290, 160, 290}
4639 }
4640 , {{ 280, 200, 280, 260, 280}
4641 , { 250, 170, 250, 250, 250}
4642 , { 280, 200, 280, 150, 280}
4643 , { 260, 180, 130, 260, 130}
4644 , { 280, 200, 280, 150, 280}
4645 }
4646 , {{ 290, 270, 290, 200, 290}
4647 , { 290, 210, 290, 160, 290}
4648 , { 290, 270, 290, 160, 290}
4649 , { 290, 210, 290, 160, 290}
4650 , { 200, 120, 200, 200, 200}
4651 }
4652 }
4653 , {{{ 370, 310, 310, 310, 370}
4654 , { 370, 310, 310, 310, 370}
4655 , { 290, 290, 290, 290, 290}
4656 , { 290, 290, 290, 290, 290}
4657 , { 290, 290, 290, 290, 290}
4658 }
4659 , {{ 370, 310, 310, 310, 370}
4660 , { 370, 310, 310, 310, 370}
4661 , { 280, 280, 280, 280, 280}
4662 , { 240, 180, 240, 180, 180}
4663 , { 280, 280, 280, 280, 280}
4664 }
4665 , {{ 290, 290, 290, 290, 290}
4666 , { 290, 290, 290, 290, 290}
4667 , { 290, 290, 290, 290, 290}
4668 , { 290, 290, 290, 290, 290}
4669 , { 290, 290, 290, 290, 290}
4670 }
4671 , {{ 310, 280, 310, 280, 280}
4672 , { 310, 250, 310, 250, 250}
4673 , { 280, 280, 280, 280, 280}
4674 , { 260, 130, 130, 130, 260}
4675 , { 280, 280, 280, 280, 280}
4676 }
4677 , {{ 290, 290, 290, 290, 290}
4678 , { 290, 290, 290, 290, 290}
4679 , { 290, 290, 290, 290, 290}
4680 , { 290, 290, 290, 290, 290}
4681 , { 200, 200, 200, 200, 200}
4682 }
4683 }
4684 }
4685 , {{{{ 370, 340, 310, 330, 370}
4686 , { 370, 340, 310, 310, 370}
4687 , { 340, 340, 310, 330, 310}
4688 , { 340, 340, 310, 310, 310}
4689 , { 340, 340, 310, 330, 310}
4690 }
4691 , {{ 370, 340, 310, 310, 370}
4692 , { 370, 340, 310, 310, 370}
4693 , { 300, 300, 260, 260, 260}
4694 , { 260, 240, 260, 200, 260}
4695 , { 300, 300, 260, 260, 260}
4696 }
4697 , {{ 340, 340, 310, 330, 310}
4698 , { 340, 340, 310, 310, 310}
4699 , { 340, 340, 310, 330, 310}
4700 , { 340, 340, 310, 310, 310}
4701 , { 340, 340, 310, 330, 310}
4702 }
4703 , {{ 300, 300, 270, 280, 280}
4704 , { 270, 250, 270, 210, 270}
4705 , { 300, 300, 260, 260, 260}
4706 , { 280, 200, 150, 280, 280}
4707 , { 300, 300, 260, 260, 260}
4708 }
4709 , {{ 340, 340, 310, 310, 310}
4710 , { 340, 340, 310, 310, 310}
4711 , { 310, 310, 280, 300, 280}
4712 , { 340, 340, 310, 310, 310}
4713 , { 320, 320, 220, 220, 220}
4714 }
4715 }
4716 , {{{ 340, 340, 310, 330, 310}
4717 , { 340, 340, 310, 270, 310}}
```

```
4718 , { 340, 340, 310, 330, 310}
4719 , { 340, 340, 310, 270, 310}
4720 , { 340, 340, 310, 330, 310}
4721 }
4722 , {{ 340, 340, 310, 270, 310}
4723 , { 340, 340, 310, 270, 310}
4724 , { 300, 300, 260, 220, 260}
4725 , { 240, 240, 200, 160, 200}
4726 , { 300, 300, 260, 220, 260}
4727 }
4728 , {{ 340, 340, 310, 330, 310}
4729 , { 340, 340, 310, 270, 310}
4730 , { 340, 340, 310, 330, 310}
4731 , { 340, 340, 310, 270, 310}
4732 , { 340, 340, 310, 330, 310}
4733 }
4734 , {{ 300, 300, 260, 240, 260}
4735 , { 250, 250, 210, 170, 210}
4736 , { 300, 300, 260, 220, 260}
4737 , { 240, 190, 150, 240, 150}
4738 , { 300, 300, 260, 220, 260}
4739 }
4740 , {{ 340, 340, 310, 300, 310}
4741 , { 340, 340, 310, 270, 310}
4742 , { 310, 310, 280, 300, 280}
4743 , { 340, 340, 310, 270, 310}
4744 , { 320, 320, 220, 180, 220}
4745 }
4746 }
4747 , {{{ 310, 310, 310, 310, 310}
4748 , { 310, 310, 310, 310, 310}
4749 , { 310, 310, 310, 310, 310}
4750 , { 310, 310, 310, 310, 310}
4751 , { 310, 310, 310, 310, 310}
4752 }
4753 , {{ 310, 310, 310, 310, 310}
4754 , { 310, 310, 310, 310, 310}
4755 , { 260, 260, 260, 260, 260}
4756 , { 260, 200, 260, 200, 260}
4757 , { 260, 260, 260, 260, 260}
4758 }
4759 , {{ 310, 310, 310, 310, 310}
4760 , { 310, 310, 310, 310, 310}
4761 , { 310, 310, 310, 310, 310}
4762 , { 310, 310, 310, 310, 310}
4763 , { 310, 310, 310, 310, 310}
4764 }
4765 , {{ 270, 260, 270, 260, 270}
4766 , { 270, 210, 270, 210, 270}
4767 , { 260, 260, 260, 260, 260}
4768 , { 150, 150, 150, 150, 150}
4769 , { 260, 260, 260, 260, 260}
4770 }
4771 , {{ 310, 310, 310, 310, 310}
4772 , { 310, 310, 310, 310, 310}
4773 , { 280, 280, 280, 280, 280}
4774 , { 310, 310, 310, 310, 310}
4775 , { 220, 220, 220, 220, 220}
4776 }
4777 }
4778 , {{{ 310, 290, 310, 280, 310}
4779 , { 310, 230, 310, 210, 310}
4780 , { 310, 290, 310, 180, 310}
4781 , { 310, 230, 310, 280, 310}
4782 , { 310, 290, 310, 220, 310}
4783 }
4784 , {{ 310, 230, 310, 200, 310}
4785 , { 310, 230, 310, 180, 310}
4786 , { 260, 180, 260, 130, 260}
4787 , { 200, 120, 200, 200, 200}
4788 , { 260, 180, 260, 130, 260}
4789 }
4790 , {{ 310, 290, 310, 180, 310}
4791 , { 310, 230, 310, 180, 310}
4792 , { 310, 290, 310, 180, 310}
4793 , { 310, 230, 310, 180, 310}
4794 , { 310, 290, 310, 180, 310}
4795 }
4796 , {{ 280, 200, 260, 280, 260}
4797 , { 210, 130, 210, 210, 210}
4798 , { 260, 180, 260, 130, 260}
4799 , { 280, 200, 150, 280, 150}
4800 , { 260, 180, 260, 130, 260}
4801 }
4802 , {{ 310, 260, 310, 220, 310}
4803 , { 310, 230, 310, 180, 310}
4804 , { 280, 260, 280, 150, 280}
```

```
4805 , { 310, 230, 310, 180, 310}
4806 , { 220, 140, 220, 220, 220}
4807 }
4808 }
4809 , {{ { 370, 310, 310, 310, 370}
4810 , { 370, 310, 310, 310, 370}
4811 , { 310, 310, 310, 310, 310}
4812 , { 310, 310, 310, 310, 310}
4813 , { 310, 310, 310, 310, 310}
4814 }
4815 , {{ { 370, 310, 310, 310, 370}
4816 , { 370, 310, 310, 310, 370}
4817 , { 260, 260, 260, 260, 260}
4818 , { 260, 200, 260, 200, 200}
4819 , { 260, 260, 260, 260, 260}
4820 }
4821 , {{ { 310, 310, 310, 310, 310}
4822 , { 310, 310, 310, 310, 310}
4823 , { 310, 310, 310, 310, 310}
4824 , { 310, 310, 310, 310, 310}
4825 , { 310, 310, 310, 310, 310}
4826 }
4827 , {{ { 280, 260, 270, 260, 280}
4828 , { 270, 210, 270, 210, 210}
4829 , { 260, 260, 260, 260, 260}
4830 , { 280, 150, 150, 150, 280}
4831 , { 260, 260, 260, 260, 260}
4832 }
4833 , {{ { 310, 310, 310, 310, 310}
4834 , { 310, 310, 310, 310, 310}
4835 , { 280, 280, 280, 280, 280}
4836 , { 310, 310, 310, 310, 310}
4837 , { 220, 220, 220, 220, 220}
4838 }
4839 }
4840 }
4841 , {{{ { 430, 430, 400, 370, 430}
4842 , { 430, 410, 400, 370, 430}
4843 , { 370, 370, 340, 360, 340}
4844 , { 370, 370, 340, 340, 340}
4845 , { 430, 430, 340, 360, 340}
4846 }
4847 , {{ { 430, 410, 370, 370, 430}
4848 , { 430, 410, 370, 370, 430}
4849 , { 370, 370, 340, 340, 340}
4850 , { 320, 290, 320, 260, 320}
4851 , { 370, 370, 340, 340, 340}
4852 }
4853 , {{ { 370, 370, 340, 360, 340}
4854 , { 370, 370, 340, 340, 340}
4855 , { 370, 370, 340, 360, 340}
4856 , { 370, 370, 340, 340, 340}
4857 , { 370, 370, 340, 360, 340}
4858 }
4859 , {{ { 400, 370, 400, 340, 400}
4860 , { 400, 370, 400, 340, 400}
4861 , { 370, 370, 340, 340, 340}
4862 , { 340, 260, 210, 340, 340}
4863 , { 370, 370, 340, 340, 340}
4864 }
4865 , {{ { 430, 430, 340, 360, 340}
4866 , { 370, 370, 340, 340, 340}
4867 , { 370, 370, 340, 360, 340}
4868 , { 370, 370, 340, 340, 340}
4869 , { 430, 430, 340, 340, 340}
4870 }
4871 }
4872 , {{{ { 430, 430, 370, 360, 370}
4873 , { 410, 410, 370, 360, 370}
4874 , { 370, 370, 340, 360, 340}
4875 , { 370, 370, 340, 300, 340}
4876 , { 430, 430, 340, 360, 340}
4877 }
4878 , {{ { 410, 410, 370, 360, 370}
4879 , { 410, 410, 370, 360, 370}
4880 , { 370, 370, 340, 300, 340}
4881 , { 290, 290, 260, 220, 260}
4882 , { 370, 370, 340, 300, 340}
4883 }
4884 , {{ { 370, 370, 340, 360, 340}
4885 , { 370, 370, 340, 300, 340}
4886 , { 370, 370, 340, 360, 340}
4887 , { 370, 370, 340, 300, 340}
4888 , { 370, 370, 340, 360, 340}
4889 }
4890 , {{ { 370, 370, 340, 300, 340}
4891 , { 370, 370, 340, 300, 340}
```

```
4892 , { 370, 370, 340, 300, 340}
4893 , { 300, 240, 210, 300, 210}
4894 , { 370, 370, 340, 300, 340}
4895 }
4896 , {{ 430, 430, 340, 360, 340}
4897 , { 370, 370, 340, 300, 340}
4898 , { 370, 370, 340, 360, 340}
4899 , { 370, 370, 340, 300, 340}
4900 , { 430, 430, 340, 300, 340}
4901 }
4902 }
4903 , {{ { 400, 370, 400, 370, 400}
4904 , { 400, 370, 400, 370, 400}
4905 , { 340, 340, 340, 340, 340}
4906 , { 340, 340, 340, 340, 340}
4907 , { 340, 340, 340, 340, 340}
4908 }
4909 , {{ { 370, 370, 370, 370, 370}
4910 , { 370, 370, 370, 370, 370}
4911 , { 340, 340, 340, 340, 340}
4912 , { 320, 260, 320, 260, 320}
4913 , { 340, 340, 340, 340, 340}
4914 }
4915 , {{ { 340, 340, 340, 340, 340}
4916 , { 340, 340, 340, 340, 340}
4917 , { 340, 340, 340, 340, 340}
4918 , { 340, 340, 340, 340, 340}
4919 , { 340, 340, 340, 340, 340}
4920 }
4921 , {{ { 400, 340, 400, 340, 400}
4922 , { 400, 340, 400, 340, 400}
4923 , { 340, 340, 340, 340, 340}
4924 , { 210, 210, 210, 210, 210}
4925 , { 340, 340, 340, 340, 340}
4926 }
4927 , {{ { 340, 340, 340, 340, 340}
4928 , { 340, 340, 340, 340, 340}
4929 , { 340, 340, 340, 340, 340}
4930 , { 340, 340, 340, 340, 340}
4931 , { 340, 340, 340, 340, 340}
4932 }
4933 }
4934 , {{ { 370, 320, 370, 340, 370}
4935 , { 370, 290, 370, 340, 370}
4936 , { 340, 320, 340, 210, 340}
4937 , { 340, 260, 340, 340, 340}
4938 , { 340, 320, 340, 340, 340}
4939 }
4940 , {{ { 370, 290, 370, 260, 370}
4941 , { 370, 290, 370, 240, 370}
4942 , { 340, 260, 340, 210, 340}
4943 , { 260, 180, 260, 260, 260}
4944 , { 340, 260, 340, 210, 340}
4945 }
4946 , {{ { 340, 320, 340, 210, 340}
4947 , { 340, 260, 340, 210, 340}
4948 , { 340, 320, 340, 210, 340}
4949 , { 340, 260, 340, 210, 340}
4950 , { 340, 320, 340, 210, 340}
4951 }
4952 , {{ { 340, 260, 340, 340, 340}
4953 , { 340, 260, 340, 340, 340}
4954 , { 340, 260, 340, 210, 340}
4955 , { 340, 260, 210, 340, 210}
4956 , { 340, 260, 340, 210, 340}
4957 }
4958 , {{ { 340, 320, 340, 340, 340}
4959 , { 340, 260, 340, 210, 340}
4960 , { 340, 320, 340, 210, 340}
4961 , { 340, 260, 340, 210, 340}
4962 , { 340, 260, 340, 340, 340}
4963 }
4964 }
4965 , {{ { 430, 370, 400, 370, 430}
4966 , { 430, 370, 400, 370, 430}
4967 , { 340, 340, 340, 340, 340}
4968 , { 340, 340, 340, 340, 340}
4969 , { 340, 340, 340, 340, 340}
4970 }
4971 , {{ { 430, 370, 370, 370, 430}
4972 , { 430, 370, 370, 370, 430}
4973 , { 340, 340, 340, 340, 340}
4974 , { 320, 260, 320, 260, 260}
4975 , { 340, 340, 340, 340, 340}
4976 }
4977 , {{ { 340, 340, 340, 340, 340}
4978 , { 340, 340, 340, 340, 340}
```

```
4979 , { 340, 340, 340, 340, 340}
4980 , { 340, 340, 340, 340, 340}
4981 , { 340, 340, 340, 340, 340}
4982 }
4983 , {{ 400, 340, 400, 340, 340}
4984 , { 400, 340, 400, 340, 340}
4985 , { 340, 340, 340, 340, 340}
4986 , { 340, 210, 210, 210, 340}
4987 , { 340, 340, 340, 340, 340}
4988 }
4989 , {{ 340, 340, 340, 340, 340}
4990 , { 340, 340, 340, 340, 340}
4991 , { 340, 340, 340, 340, 340}
4992 , { 340, 340, 340, 340, 340}
4993 , { 340, 340, 340, 340, 340}
4994 }
4995 }
4996 }
4997 }
4998 , {{{{ INF, INF, INF, INF, INF }
4999 , { INF, INF, INF, INF, INF }
5000 , { INF, INF, INF, INF, INF }
5001 , { INF, INF, INF, INF, INF }
5002 , { INF, INF, INF, INF, INF }
5003 }
5004 , {{ INF, INF, INF, INF, INF }
5005 , { INF, INF, INF, INF, INF }
5006 , { INF, INF, INF, INF, INF }
5007 , { INF, INF, INF, INF, INF }
5008 , { INF, INF, INF, INF, INF }
5009 }
5010 , {{ INF, INF, INF, INF, INF }
5011 , { INF, INF, INF, INF, INF }
5012 , { INF, INF, INF, INF, INF }
5013 , { INF, INF, INF, INF, INF }
5014 , { INF, INF, INF, INF, INF }
5015 }
5016 , {{ INF, INF, INF, INF, INF }
5017 , { INF, INF, INF, INF, INF }
5018 , { INF, INF, INF, INF, INF }
5019 , { INF, INF, INF, INF, INF }
5020 , { INF, INF, INF, INF, INF }
5021 }
5022 , {{ INF, INF, INF, INF, INF }
5023 , { INF, INF, INF, INF, INF }
5024 , { INF, INF, INF, INF, INF }
5025 , { INF, INF, INF, INF, INF }
5026 , { INF, INF, INF, INF, INF }
5027 }
5028 }
5029 , {{{ INF, INF, INF, INF, INF }
5030 , { INF, INF, INF, INF, INF }
5031 , { INF, INF, INF, INF, INF }
5032 , { INF, INF, INF, INF, INF }
5033 , { INF, INF, INF, INF, INF }
5034 }
5035 , {{ INF, INF, INF, INF, INF }
5036 , { INF, INF, INF, INF, INF }
5037 , { INF, INF, INF, INF, INF }
5038 , { INF, INF, INF, INF, INF }
5039 , { INF, INF, INF, INF, INF }
5040 }
5041 , {{ INF, INF, INF, INF, INF }
5042 , { INF, INF, INF, INF, INF }
5043 , { INF, INF, INF, INF, INF }
5044 , { INF, INF, INF, INF, INF }
5045 , { INF, INF, INF, INF, INF }
5046 }
5047 , {{ INF, INF, INF, INF, INF }
5048 , { INF, INF, INF, INF, INF }
5049 , { INF, INF, INF, INF, INF }
5050 , { INF, INF, INF, INF, INF }
5051 , { INF, INF, INF, INF, INF }
5052 }
5053 , {{ INF, INF, INF, INF, INF }
5054 , { INF, INF, INF, INF, INF }
5055 , { INF, INF, INF, INF, INF }
5056 , { INF, INF, INF, INF, INF }
5057 , { INF, INF, INF, INF, INF }
5058 }
5059 }
5060 , {{{{ INF, INF, INF, INF, INF }
5061 , { INF, INF, INF, INF, INF }
5062 , { INF, INF, INF, INF, INF }
5063 , { INF, INF, INF, INF, INF }
5064 , { INF, INF, INF, INF, INF }
5065 }
```

```
5066 , {{ INF, INF, INF, INF, INF }
5067 , { INF, INF, INF, INF, INF }
5068 , { INF, INF, INF, INF, INF }
5069 , { INF, INF, INF, INF, INF }
5070 , { INF, INF, INF, INF, INF }
5071 }
5072 , {{ INF, INF, INF, INF, INF }
5073 , { INF, INF, INF, INF, INF }
5074 , { INF, INF, INF, INF, INF }
5075 , { INF, INF, INF, INF, INF }
5076 , { INF, INF, INF, INF, INF }
5077 }
5078 , {{ INF, INF, INF, INF, INF }
5079 , { INF, INF, INF, INF, INF }
5080 , { INF, INF, INF, INF, INF }
5081 , { INF, INF, INF, INF, INF }
5082 , { INF, INF, INF, INF, INF }
5083 }
5084 , {{ INF, INF, INF, INF, INF }
5085 , { INF, INF, INF, INF, INF }
5086 , { INF, INF, INF, INF, INF }
5087 , { INF, INF, INF, INF, INF }
5088 , { INF, INF, INF, INF, INF }
5089 }
5090 }
5091 , {{{ INF, INF, INF, INF, INF }
5092 , { INF, INF, INF, INF, INF }
5093 , { INF, INF, INF, INF, INF }
5094 , { INF, INF, INF, INF, INF }
5095 , { INF, INF, INF, INF, INF }
5096 }
5097 , {{ INF, INF, INF, INF, INF }
5098 , { INF, INF, INF, INF, INF }
5099 , { INF, INF, INF, INF, INF }
5100 , { INF, INF, INF, INF, INF }
5101 , { INF, INF, INF, INF, INF }
5102 }
5103 , {{ INF, INF, INF, INF, INF }
5104 , { INF, INF, INF, INF, INF }
5105 , { INF, INF, INF, INF, INF }
5106 , { INF, INF, INF, INF, INF }
5107 , { INF, INF, INF, INF, INF }
5108 }
5109 , {{ INF, INF, INF, INF, INF }
5110 , { INF, INF, INF, INF, INF }
5111 , { INF, INF, INF, INF, INF }
5112 , { INF, INF, INF, INF, INF }
5113 , { INF, INF, INF, INF, INF }
5114 }
5115 , {{ INF, INF, INF, INF, INF }
5116 , { INF, INF, INF, INF, INF }
5117 , { INF, INF, INF, INF, INF }
5118 , { INF, INF, INF, INF, INF }
5119 , { INF, INF, INF, INF, INF }
5120 }
5121 }
5122 , {{{ INF, INF, INF, INF, INF }
5123 , { INF, INF, INF, INF, INF }
5124 , { INF, INF, INF, INF, INF }
5125 , { INF, INF, INF, INF, INF }
5126 , { INF, INF, INF, INF, INF }
5127 }
5128 , {{ INF, INF, INF, INF, INF }
5129 , { INF, INF, INF, INF, INF }
5130 , { INF, INF, INF, INF, INF }
5131 , { INF, INF, INF, INF, INF }
5132 , { INF, INF, INF, INF, INF }
5133 }
5134 , {{ INF, INF, INF, INF, INF }
5135 , { INF, INF, INF, INF, INF }
5136 , { INF, INF, INF, INF, INF }
5137 , { INF, INF, INF, INF, INF }
5138 , { INF, INF, INF, INF, INF }
5139 }
5140 , {{ INF, INF, INF, INF, INF }
5141 , { INF, INF, INF, INF, INF }
5142 , { INF, INF, INF, INF, INF }
5143 , { INF, INF, INF, INF, INF }
5144 , { INF, INF, INF, INF, INF }
5145 }
5146 , {{ INF, INF, INF, INF, INF }
5147 , { INF, INF, INF, INF, INF }
5148 , { INF, INF, INF, INF, INF }
5149 , { INF, INF, INF, INF, INF }
5150 , { INF, INF, INF, INF, INF }
5151 }
5152 }
```

```
5153 }
5154 ,{{{{ 310, 240, 240, 310, 260}
5155 ,{ 270, 240, 240, 270, 260}
5156 ,{ 310, 220, 220, 310, 220}
5157 ,{ 270, 240, 240, 270, 240}
5158 ,{ 300, 210, 210, 300, 210}
5159 }
5160 ,{{ 260, 200, 200, 230, 260}
5161 ,{ 260, 200, 200, 230, 260}
5162 ,{ 220, 190, 190, 220, 190}
5163 ,{ 160, 100, 160, 130, 160}
5164 ,{ 220, 190, 190, 220, 190}
5165 }
5166 ,{{ 310, 240, 240, 310, 240}
5167 ,{ 270, 240, 240, 270, 240}
5168 ,{ 310, 220, 220, 310, 220}
5169 ,{ 270, 240, 240, 270, 240}
5170 ,{ 300, 210, 210, 300, 210}
5171 }
5172 ,{{ 220, 190, 190, 220, 190}
5173 ,{ 160, 100, 160, 130, 160}
5174 ,{ 220, 190, 190, 220, 190}
5175 ,{ 210, 50, 50, 210, 180}
5176 ,{ 220, 190, 190, 220, 190}
5177 }
5178 ,{{ 300, 240, 240, 300, 240}
5179 ,{ 270, 240, 240, 270, 240}
5180 ,{ 300, 210, 210, 300, 210}
5181 ,{ 270, 240, 240, 270, 240}
5182 ,{ 150, 140, 120, 150, 120}
5183 }
5184 }
5185 ,{{{{ 310, 200, 240, 310, 240}
5186 ,{ 270, 200, 240, 270, 240}
5187 ,{ 310, 190, 220, 310, 220}
5188 ,{ 270, 200, 240, 270, 240}
5189 ,{ 300, 170, 210, 300, 210}
5190 }
5191 ,{{ 230, 160, 200, 230, 200}
5192 ,{ 230, 160, 200, 230, 200}
5193 ,{ 220, 160, 190, 220, 190}
5194 ,{ 130, 70, 100, 130, 100}
5195 ,{ 220, 160, 190, 220, 190}
5196 }
5197 ,{{ 310, 200, 240, 310, 240}
5198 ,{ 270, 200, 240, 270, 240}
5199 ,{ 310, 190, 220, 310, 220}
5200 ,{ 270, 200, 240, 270, 240}
5201 ,{ 300, 170, 210, 300, 210}
5202 }
5203 ,{{ 220, 160, 190, 220, 190}
5204 ,{ 130, 70, 100, 130, 100}
5205 ,{ 220, 160, 190, 220, 190}
5206 ,{ 210, 10, 50, 210, 50}
5207 ,{ 220, 160, 190, 220, 190}
5208 }
5209 ,{{ 300, 200, 240, 300, 240}
5210 ,{ 270, 200, 240, 270, 240}
5211 ,{ 300, 170, 210, 300, 210}
5212 ,{ 270, 200, 240, 270, 240}
5213 ,{ 150, 140, 120, 150, 120}
5214 }
5215 }
5216 ,{{{{ 240, 240, 240, 240, 240}
5217 ,{ 240, 240, 240, 240, 240}
5218 ,{ 220, 220, 220, 220, 220}
5219 ,{ 240, 240, 240, 240, 240}
5220 ,{ 210, 210, 210, 210, 210}
5221 }
5222 ,{{ 200, 200, 200, 200, 200}
5223 ,{ 200, 200, 200, 200, 200}
5224 ,{ 190, 190, 190, 190, 190}
5225 ,{ 160, 100, 160, 100, 160}
5226 ,{ 190, 190, 190, 190, 190}
5227 }
5228 ,{{ 240, 240, 240, 240, 240}
5229 ,{ 240, 240, 240, 240, 240}
5230 ,{ 220, 220, 220, 220, 220}
5231 ,{ 240, 240, 240, 240, 240}
5232 ,{ 210, 210, 210, 210, 210}
5233 }
5234 ,{{ 190, 190, 190, 190, 190}
5235 ,{ 160, 100, 160, 100, 160}
5236 ,{ 190, 190, 190, 190, 190}
5237 ,{ 50, 50, 50, 50, 50}
5238 ,{ 190, 190, 190, 190, 190}
5239 }
```

```
5240 , {{ 240, 240, 240, 240, 240}
5241 , { 240, 240, 240, 240, 240}
5242 , { 210, 210, 210, 210, 210}
5243 , { 240, 240, 240, 240, 240}
5244 , { 120, 120, 120, 120, 120}
5245 }
5246 }
5247 , {{{ 240, 150, 240, 180, 240}
5248 , { 240, 100, 240, 110, 240}
5249 , { 220, 150, 220, 90, 220}
5250 , { 240, 100, 240, 180, 240}
5251 , { 210, 130, 210, 120, 210}
5252 }
5253 , {{ 200, 60, 200, 100, 200}
5254 , { 200, 60, 200, 70, 200}
5255 , { 190, 60, 190, 60, 190}
5256 , { 100, -30, 100, 100, 100}
5257 , { 190, 60, 190, 60, 190}
5258 }
5259 , {{ 240, 150, 240, 110, 240}
5260 , { 240, 100, 240, 110, 240}
5261 , { 220, 150, 220, 90, 220}
5262 , { 240, 100, 240, 110, 240}
5263 , { 210, 130, 210, 80, 210}
5264 }
5265 , {{ 190, 60, 190, 180, 190}
5266 , { 100, -30, 100, 100, 100}
5267 , { 190, 60, 190, 60, 190}
5268 , { 180, 40, 50, 180, 50}
5269 , { 190, 60, 190, 60, 190}
5270 }
5271 , {{ 240, 130, 240, 120, 240}
5272 , { 240, 100, 240, 110, 240}
5273 , { 210, 130, 210, 80, 210}
5274 , { 240, 100, 240, 110, 240}
5275 , { 120, -10, 120, 120, 120}
5276 }
5277 }
5278 , {{{ 260, 240, 240, 240, 260}
5279 , { 260, 240, 240, 240, 260}
5280 , { 220, 220, 220, 220, 220}
5281 , { 240, 240, 240, 240, 240}
5282 , { 210, 210, 210, 210, 210}
5283 }
5284 , {{ 260, 200, 200, 200, 260}
5285 , { 260, 200, 200, 200, 260}
5286 , { 190, 190, 190, 190, 190}
5287 , { 160, 100, 160, 100, 100}
5288 , { 190, 190, 190, 190, 190}
5289 }
5290 , {{ 240, 240, 240, 240, 240}
5291 , { 240, 240, 240, 240, 240}
5292 , { 220, 220, 220, 220, 220}
5293 , { 240, 240, 240, 240, 240}
5294 , { 210, 210, 210, 210, 210}
5295 }
5296 , {{ 190, 190, 190, 190, 190}
5297 , { 160, 100, 160, 100, 100}
5298 , { 190, 190, 190, 190, 190}
5299 , { 180, 50, 50, 50, 180}
5300 , { 190, 190, 190, 190, 190}
5301 }
5302 , {{ 240, 240, 240, 240, 240}
5303 , { 240, 240, 240, 240, 240}
5304 , { 210, 210, 210, 210, 210}
5305 , { 240, 240, 240, 240, 240}
5306 , { 120, 120, 120, 120, 120}
5307 }
5308 }
5309 }
5310 , {{{{ 280, 210, 210, 280, 270}
5311 , { 270, 210, 210, 240, 270}
5312 , { 280, 190, 190, 280, 190}
5313 , { 210, 180, 180, 210, 180}
5314 , { 280, 190, 190, 280, 190}
5315 }
5316 , {{ 270, 210, 210, 240, 270}
5317 , { 270, 210, 210, 240, 270}
5318 , { 220, 190, 190, 220, 190}
5319 , { 70, 10, 70, 40, 70}
5320 , { 220, 190, 190, 220, 190}
5321 }
5322 , {{ 280, 190, 190, 280, 190}
5323 , { 210, 180, 180, 210, 180}
5324 , { 280, 190, 190, 280, 190}
5325 , { 210, 180, 180, 210, 180}
5326 , { 280, 190, 190, 280, 190}
```

```
5327    }
5328 ,{{ 220,   190,   190,   220,   190}
5329 ,{ 130,    70,   130,   100,   130}
5330 ,{ 220,   190,   190,   220,   190}
5331 ,{ 210,    50,    50,   210,   180}
5332 ,{ 220,   190,   190,   220,   190}
5333 }
5334 ,{{ 280,   190,   190,   280,   190}
5335 ,{ 210,   180,   180,   210,   180}
5336 ,{ 280,   190,   190,   280,   190}
5337 ,{ 210,   180,   180,   210,   180}
5338 ,{ 140,   140,   110,   140,   110}
5339 }
5340 }
5341 ,{{{ 280,   190,   210,   280,   210}
5342 ,{ 240,   190,   210,   240,   210}
5343 ,{ 280,   160,   190,   280,   190}
5344 ,{ 210,   150,   180,   210,   180}
5345 ,{ 280,   150,   190,   280,   190}
5346 }
5347 ,{{ 240,   190,   210,   240,   210}
5348 ,{ 240,   190,   210,   240,   210}
5349 ,{ 220,   150,   190,   220,   190}
5350 ,{ 40,   -20,    10,    40,    10}
5351 ,{ 220,   150,   190,   220,   190}
5352 }
5353 ,{{ 280,   150,   190,   280,   190}
5354 ,{ 210,   150,   180,   210,   180}
5355 ,{ 280,   150,   190,   280,   190}
5356 ,{ 210,   150,   180,   210,   180}
5357 ,{ 280,   150,   190,   280,   190}
5358 }
5359 ,{{ 220,   150,   190,   220,   190}
5360 ,{ 100,    40,    70,   100,    70}
5361 ,{ 220,   150,   190,   220,   190}
5362 ,{ 210,    10,    50,   210,    50}
5363 ,{ 220,   150,   190,   220,   190}
5364 }
5365 ,{{ 280,   160,   190,   280,   190}
5366 ,{ 210,   150,   180,   210,   180}
5367 ,{ 280,   160,   190,   280,   190}
5368 ,{ 210,   150,   180,   210,   180}
5369 ,{ 140,   140,   110,   140,   110}
5370 }
5371 }
5372 ,{{{ 210,   210,   210,   210,   210}
5373 ,{ 210,   210,   210,   210,   210}
5374 ,{ 190,   190,   190,   190,   190}
5375 ,{ 180,   180,   180,   180,   180}
5376 ,{ 190,   190,   190,   190,   190}
5377 }
5378 ,{{ 210,   210,   210,   210,   210}
5379 ,{ 210,   210,   210,   210,   210}
5380 ,{ 190,   190,   190,   190,   190}
5381 ,{ 70,    10,    70,    10,    70}
5382 ,{ 190,   190,   190,   190,   190}
5383 }
5384 ,{{ 190,   190,   190,   190,   190}
5385 ,{ 180,   180,   180,   180,   180}
5386 ,{ 190,   190,   190,   190,   190}
5387 ,{ 180,   180,   180,   180,   180}
5388 ,{ 190,   190,   190,   190,   190}
5389 }
5390 ,{{ 190,   190,   190,   190,   190}
5391 ,{ 130,    70,   130,    70,   130}
5392 ,{ 190,   190,   190,   190,   190}
5393 ,{ 50,    50,    50,    50,    50}
5394 ,{ 190,   190,   190,   190,   190}
5395 }
5396 ,{{ 190,   190,   190,   190,   190}
5397 ,{ 180,   180,   180,   180,   180}
5398 ,{ 190,   190,   190,   190,   190}
5399 ,{ 180,   180,   180,   180,   180}
5400 ,{ 110,   110,   110,   110,   110}
5401 }
5402 }
5403 ,{{{ 210,   120,   210,   180,   210}
5404 ,{ 210,    80,   210,    80,   210}
5405 ,{ 190,   120,   190,    60,   190}
5406 ,{ 180,    50,   180,   180,   180}
5407 ,{ 190,   110,   190,   110,   190}
5408 }
5409 ,{{ 210,    80,   210,    80,   210}
5410 ,{ 210,    80,   210,    80,   210}
5411 ,{ 190,    50,   190,    60,   190}
5412 ,{ 10,   -120,    10,    10,    10}
5413 ,{ 190,    50,   190,    60,   190}
```

```
5414      }
5415 ,{{ 190,    110,    190,     60,   190}
5416 ,{ 180,     50,    180,     50,   180}
5417 ,{ 190,    110,    190,     60,   190}
5418 ,{ 180,     50,    180,     50,   180}
5419 ,{ 190,    110,    190,     60,   190}
5420 }
5421 ,{{ 190,     50,    190,   180,   190}
5422 ,{ 70,    -60,     70,     70,    70}
5423 ,{ 190,     50,    190,     60,   190}
5424 ,{ 180,     40,     50,   180,     50}
5425 ,{ 190,     50,    190,     60,   190}
5426 }
5427 ,{{ 190,    120,    190,   110,   190}
5428 ,{ 180,     50,    180,     50,   180}
5429 ,{ 190,    120,    190,     60,   190}
5430 ,{ 180,     50,    180,     50,   180}
5431 ,{ 110,    -20,    110,   110,   110}
5432 }
5433 }
5434 ,{{{ 270,    210,    210,   210,   270}
5435 ,{ 270,    210,    210,   210,   270}
5436 ,{ 190,    190,    190,   190,   190}
5437 ,{ 180,    180,    180,   180,   180}
5438 ,{ 190,    190,    190,   190,   190}
5439 }
5440 ,{{ 270,    210,    210,   210,   270}
5441 ,{ 270,    210,    210,   210,   270}
5442 ,{ 190,    190,    190,   190,   190}
5443 ,{ 70,     10,     70,     10,    10}
5444 ,{ 190,    190,    190,   190,   190}
5445 }
5446 ,{{ 190,    190,    190,   190,   190}
5447 ,{ 180,    180,    180,   180,   180}
5448 ,{ 190,    190,    190,   190,   190}
5449 ,{ 180,    180,    180,   180,   180}
5450 ,{ 190,    190,    190,   190,   190}
5451 }
5452 ,{{ 190,    190,    190,   190,   190}
5453 ,{ 130,     70,    130,     70,    70}
5454 ,{ 190,    190,    190,   190,   190}
5455 ,{ 180,     50,     50,     50,   180}
5456 ,{ 190,    190,    190,   190,   190}
5457 }
5458 ,{{ 190,    190,    190,   190,   190}
5459 ,{ 180,    180,    180,   180,   180}
5460 ,{ 190,    190,    190,   190,   190}
5461 ,{ 180,    180,    180,   180,   180}
5462 ,{ 110,    110,    110,   110,   110}
5463 }
5464 }
5465 }
5466 ,{{{ 400,    360,    340,   400,   400}
5467 ,{ 400,    360,    340,   370,   400}
5468 ,{ 400,    310,    310,   400,   310}
5469 ,{ 340,    310,    310,   340,   310}
5470 ,{ 400,    330,    310,   400,   310}
5471 }
5472 ,{{ 400,    360,    340,   370,   400}
5473 ,{ 400,    360,    340,   370,   400}
5474 ,{ 340,    310,    310,   340,   310}
5475 ,{ 290,    230,    290,   260,   290}
5476 ,{ 340,    310,    310,   340,   310}
5477 }
5478 ,{{ 400,    310,    310,   400,   310}
5479 ,{ 340,    310,    310,   340,   310}
5480 ,{ 400,    310,    310,   400,   310}
5481 ,{ 340,    310,    310,   340,   310}
5482 ,{ 400,    310,    310,   400,   310}
5483 }
5484 ,{{ 360,    360,    330,   340,   330}
5485 ,{ 360,    360,    330,   300,   330}
5486 ,{ 340,    310,    310,   340,   310}
5487 ,{ 340,    180,    180,   340,   310}
5488 ,{ 340,    310,    310,   340,   310}
5489 }
5490 ,{{ 400,    330,    310,   400,   310}
5491 ,{ 340,    310,    310,   340,   310}
5492 ,{ 400,    310,    310,   400,   310}
5493 ,{ 340,    310,    310,   340,   310}
5494 ,{ 340,    330,    310,   340,   310}
5495 }
5496 }
5497 ,{{{ 400,    360,    340,   400,   340}
5498 ,{ 370,    360,    340,   370,   340}
5499 ,{ 400,    270,    310,   400,   310}
5500 ,{ 340,    270,    310,   340,   310}
```

```
5501 , { 400, 330, 310, 400, 310}
5502 }
5503 , {{ 370, 360, 340, 370, 340}
5504 , { 370, 360, 340, 370, 340}
5505 , { 340, 270, 310, 340, 310}
5506 , { 260, 190, 230, 260, 230}
5507 , { 340, 270, 310, 340, 310}
5508 }
5509 , {{ 400, 270, 310, 400, 310}
5510 , { 340, 270, 310, 340, 310}
5511 , { 400, 270, 310, 400, 310}
5512 , { 340, 270, 310, 340, 310}
5513 , { 400, 270, 310, 400, 310}
5514 }
5515 , {{ 360, 360, 310, 340, 310}
5516 , { 360, 360, 270, 300, 270}
5517 , { 340, 270, 310, 340, 310}
5518 , { 340, 140, 180, 340, 180}
5519 , { 340, 270, 310, 340, 310}
5520 }
5521 , {{ 400, 330, 310, 400, 310}
5522 , { 340, 270, 310, 340, 310}
5523 , { 400, 270, 310, 400, 310}
5524 , { 340, 270, 310, 340, 310}
5525 , { 340, 330, 310, 340, 310}
5526 }
5527 }
5528 , {{ { 340, 340, 340, 340, 340}
5529 , { 340, 340, 340, 340, 340}
5530 , { 310, 310, 310, 310, 310}
5531 , { 310, 310, 310, 310, 310}
5532 , { 310, 310, 310, 310, 310}
5533 }
5534 , {{ { 340, 340, 340, 340, 340}
5535 , { 340, 340, 340, 340, 340}
5536 , { 310, 310, 310, 310, 310}
5537 , { 290, 230, 290, 230, 290}
5538 , { 310, 310, 310, 310, 310}
5539 }
5540 , {{ { 310, 310, 310, 310, 310}
5541 , { 310, 310, 310, 310, 310}
5542 , { 310, 310, 310, 310, 310}
5543 , { 310, 310, 310, 310, 310}
5544 , { 310, 310, 310, 310, 310}
5545 }
5546 , {{ { 330, 310, 330, 310, 330}
5547 , { 330, 270, 330, 270, 330}
5548 , { 310, 310, 310, 310, 310}
5549 , { 180, 180, 180, 180, 180}
5550 , { 310, 310, 310, 310, 310}
5551 }
5552 , {{ { 310, 310, 310, 310, 310}
5553 , { 310, 310, 310, 310, 310}
5554 , { 310, 310, 310, 310, 310}
5555 , { 310, 310, 310, 310, 310}
5556 , { 310, 310, 310, 310, 310}
5557 }
5558 }
5559 , {{ { 340, 230, 340, 310, 340}
5560 , { 340, 220, 340, 270, 340}
5561 , { 310, 230, 310, 180, 310}
5562 , { 310, 170, 310, 310, 310}
5563 , { 310, 230, 310, 310, 310}
5564 }
5565 , {{ { 340, 220, 340, 230, 340}
5566 , { 340, 220, 340, 210, 340}
5567 , { 310, 170, 310, 180, 310}
5568 , { 230, 20, 230, 230, 230}
5569 , { 310, 170, 310, 180, 310}
5570 }
5571 , {{ { 310, 230, 310, 180, 310}
5572 , { 310, 170, 310, 180, 310}
5573 , { 310, 230, 310, 180, 310}
5574 , { 310, 170, 310, 180, 310}
5575 , { 310, 230, 310, 180, 310}
5576 }
5577 , {{ { 310, 170, 310, 310, 310}
5578 , { 270, 130, 270, 270, 270}
5579 , { 310, 170, 310, 180, 310}
5580 , { 310, 170, 180, 310, 180}
5581 , { 310, 170, 310, 180, 310}
5582 }
5583 , {{ { 310, 230, 310, 310, 310}
5584 , { 310, 170, 310, 180, 310}
5585 , { 310, 230, 310, 180, 310}
5586 , { 310, 170, 310, 180, 310}
5587 , { 310, 170, 310, 310, 310}
```

```
5588    }
5589    }
5590 ,{{{{ 400,   340,   340,   340,   400}
5591 ,{ 400,   340,   340,   340,   400}
5592 ,{ 310,   310,   310,   310,   310}
5593 ,{ 310,   310,   310,   310,   310}
5594 ,{ 310,   310,   310,   310,   310}
5595 }
5596 ,{{ 400,   340,   340,   340,   400}
5597 ,{ 400,   340,   340,   340,   400}
5598 ,{ 310,   310,   310,   310,   310}
5599 ,{ 290,   230,   290,   230,   230}
5600 ,{ 310,   310,   310,   310,   310}
5601 }
5602 ,{{ 310,   310,   310,   310,   310}
5603 ,{ 310,   310,   310,   310,   310}
5604 ,{ 310,   310,   310,   310,   310}
5605 ,{ 310,   310,   310,   310,   310}
5606 ,{ 310,   310,   310,   310,   310}
5607 }
5608 ,{{ 330,   310,   330,   310,   310}
5609 ,{ 330,   270,   330,   270,   270}
5610 ,{ 310,   310,   310,   310,   310}
5611 ,{ 310,   180,   180,   180,   310}
5612 ,{ 310,   310,   310,   310,   310}
5613 }
5614 ,{{ 310,   310,   310,   310,   310}
5615 ,{ 310,   310,   310,   310,   310}
5616 ,{ 310,   310,   310,   310,   310}
5617 ,{ 310,   310,   310,   310,   310}
5618 ,{ 310,   310,   310,   310,   310}
5619 }
5620 }
5621 }
5622 ,{{{{ 370,   310,   370,   370,   370}
5623 ,{ 370,   310,   370,   340,   370}
5624 ,{ 370,   280,   280,   370,   280}
5625 ,{ 310,   280,   280,   310,   280}
5626 ,{ 370,   300,   280,   370,   280}
5627 }
5628 ,{{ 310,   280,   280,   310,   300}
5629 ,{ 300,   240,   240,   270,   300}
5630 ,{ 310,   280,   280,   310,   280}
5631 ,{ 200,   140,   200,   170,   200}
5632 ,{ 310,   280,   280,   310,   280}
5633 }
5634 ,{{ 370,   280,   280,   370,   280}
5635 ,{ 310,   280,   280,   310,   280}
5636 ,{ 370,   280,   280,   370,   280}
5637 ,{ 310,   280,   280,   310,   280}
5638 ,{ 370,   280,   280,   370,   280}
5639 }
5640 ,{{ 370,   310,   370,   340,   370}
5641 ,{ 370,   310,   370,   340,   370}
5642 ,{ 310,   280,   280,   310,   280}
5643 ,{ 310,   150,   150,   310,   280}
5644 ,{ 310,   280,   280,   310,   280}
5645 }
5646 ,{{ 370,   300,   280,   370,   280}
5647 ,{ 310,   280,   280,   310,   280}
5648 ,{ 370,   280,   280,   370,   280}
5649 ,{ 310,   280,   280,   310,   280}
5650 ,{ 310,   300,   280,   310,   280}
5651 }
5652 }
5653 ,{{{{ 370,   300,   310,   370,   310}
5654 ,{ 340,   270,   310,   340,   310}
5655 ,{ 370,   240,   280,   370,   280}
5656 ,{ 310,   240,   280,   310,   280}
5657 ,{ 370,   300,   280,   370,   280}
5658 }
5659 ,{{ 310,   240,   280,   310,   280}
5660 ,{ 270,   210,   240,   270,   240}
5661 ,{ 310,   240,   280,   310,   280}
5662 ,{ 170,   110,   140,   170,   140}
5663 ,{ 310,   240,   280,   310,   280}
5664 }
5665 ,{{ 370,   240,   280,   370,   280}
5666 ,{ 310,   240,   280,   310,   280}
5667 ,{ 370,   240,   280,   370,   280}
5668 ,{ 310,   240,   280,   310,   280}
5669 ,{ 370,   240,   280,   370,   280}
5670 }
5671 ,{{ 340,   270,   310,   340,   310}
5672 ,{ 340,   270,   310,   340,   310}
5673 ,{ 310,   240,   280,   310,   280}
5674 ,{ 310,   110,   150,   310,   150}
```

```
5675 , { 310, 240, 280, 310, 280}
5676 }
5677 , {{ 370, 300, 280, 370, 280}
5678 , { 310, 240, 280, 310, 280}
5679 , { 370, 240, 280, 370, 280}
5680 , { 310, 240, 280, 310, 280}
5681 , { 310, 300, 280, 310, 280}
5682 }
5683 }
5684 , {{ { 370, 310, 370, 310, 370}
5685 , { 370, 310, 370, 310, 370}
5686 , { 280, 280, 280, 280, 280}
5687 , { 280, 280, 280, 280, 280}
5688 , { 280, 280, 280, 280, 280}
5689 }
5690 , {{ { 280, 280, 280, 280, 280}
5691 , { 240, 240, 240, 240, 240}
5692 , { 280, 280, 280, 280, 280}
5693 , { 200, 140, 200, 140, 200}
5694 , { 280, 280, 280, 280, 280}
5695 }
5696 , {{ { 280, 280, 280, 280, 280}
5697 , { 280, 280, 280, 280, 280}
5698 , { 280, 280, 280, 280, 280}
5699 , { 280, 280, 280, 280, 280}
5700 , { 280, 280, 280, 280, 280}
5701 }
5702 , {{ { 370, 310, 370, 310, 370}
5703 , { 370, 310, 370, 310, 370}
5704 , { 280, 280, 280, 280, 280}
5705 , { 150, 150, 150, 150, 150}
5706 , { 280, 280, 280, 280, 280}
5707 }
5708 , {{ { 280, 280, 280, 280, 280}
5709 , { 280, 280, 280, 280, 280}
5710 , { 280, 280, 280, 280, 280}
5711 , { 280, 280, 280, 280, 280}
5712 , { 280, 280, 280, 280, 280}
5713 }
5714 }
5715 , {{ { { 310, 200, 310, 310, 310}
5716 , { 310, 170, 310, 310, 310}
5717 , { 280, 200, 280, 150, 280}
5718 , { 280, 140, 280, 280, 280}
5719 , { 280, 200, 280, 280, 280}
5720 }
5721 , {{ { 280, 140, 280, 150, 280}
5722 , { 240, 110, 240, 110, 240}
5723 , { 280, 140, 280, 150, 280}
5724 , { 140, 10, 140, 140, 140}
5725 , { 280, 140, 280, 150, 280}
5726 }
5727 , {{ { 280, 200, 280, 150, 280}
5728 , { 280, 140, 280, 150, 280}
5729 , { 280, 200, 280, 150, 280}
5730 , { 280, 140, 280, 150, 280}
5731 , { 280, 200, 280, 150, 280}
5732 }
5733 , {{ { 310, 170, 310, 310, 310}
5734 , { 310, 170, 310, 310, 310}
5735 , { 280, 140, 280, 150, 280}
5736 , { 280, 140, 150, 280, 150}
5737 , { 280, 140, 280, 150, 280}
5738 }
5739 , {{ { 280, 200, 280, 280, 280}
5740 , { 280, 140, 280, 150, 280}
5741 , { 280, 200, 280, 150, 280}
5742 , { 280, 140, 280, 150, 280}
5743 , { 280, 140, 280, 280, 280}
5744 }
5745 }
5746 , {{ { { 370, 310, 370, 310, 310}
5747 , { 370, 310, 370, 310, 310}
5748 , { 280, 280, 280, 280, 280}
5749 , { 280, 280, 280, 280, 280}
5750 , { 280, 280, 280, 280, 280}
5751 }
5752 , {{ { 300, 280, 280, 280, 300}
5753 , { 300, 240, 240, 240, 300}
5754 , { 280, 280, 280, 280, 280}
5755 , { 200, 140, 200, 140, 140}
5756 , { 280, 280, 280, 280, 280}
5757 }
5758 , {{ { 280, 280, 280, 280, 280}
5759 , { 280, 280, 280, 280, 280}
5760 , { 280, 280, 280, 280, 280}
5761 , { 280, 280, 280, 280, 280}
```

```
5762 , { 280, 280, 280, 280, 280}
5763 }
5764 , {{ 370, 310, 370, 310, 310}
5765 , { 370, 310, 370, 310, 310}
5766 , { 280, 280, 280, 280, 280}
5767 , { 280, 150, 150, 150, 280}
5768 , { 280, 280, 280, 280, 280}
5769 }
5770 , {{ 280, 280, 280, 280, 280}
5771 , { 280, 280, 280, 280, 280}
5772 , { 280, 280, 280, 280, 280}
5773 , { 280, 280, 280, 280, 280}
5774 , { 280, 280, 280, 280, 280}
5775 }
5776 }
5777 }
5778 , {{{{ 350, 280, 280, 350, 340}
5779 , { 340, 280, 280, 310, 340}
5780 , { 350, 260, 260, 350, 260}
5781 , { 290, 260, 260, 290, 260}
5782 , { 350, 260, 260, 350, 260}
5783 }
5784 , {{ 340, 280, 280, 310, 340}
5785 , { 340, 280, 280, 310, 340}
5786 , { 280, 250, 250, 280, 250}
5787 , { 210, 150, 210, 180, 210}
5788 , { 280, 250, 250, 280, 250}
5789 }
5790 , {{ 350, 260, 260, 350, 260}
5791 , { 290, 260, 260, 290, 260}
5792 , { 350, 260, 260, 350, 260}
5793 , { 290, 260, 260, 290, 260}
5794 , { 350, 260, 260, 350, 260}
5795 }
5796 , {{ 280, 250, 280, 280, 280}
5797 , { 280, 220, 280, 250, 280}
5798 , { 280, 250, 250, 280, 250}
5799 , { 260, 100, 100, 260, 230}
5800 , { 280, 250, 250, 280, 250}
5801 }
5802 , {{ 350, 260, 260, 350, 260}
5803 , { 290, 260, 260, 290, 260}
5804 , { 350, 260, 260, 350, 260}
5805 , { 290, 260, 260, 290, 260}
5806 , { 200, 190, 170, 200, 170}
5807 }
5808 }
5809 , {{{{ 350, 240, 280, 350, 280}
5810 , { 310, 240, 280, 310, 280}
5811 , { 350, 220, 260, 350, 260}
5812 , { 290, 230, 260, 290, 260}
5813 , { 350, 220, 260, 350, 260}
5814 }
5815 , {{ 310, 240, 280, 310, 280}
5816 , { 310, 240, 280, 310, 280}
5817 , { 280, 220, 250, 280, 250}
5818 , { 180, 120, 150, 180, 150}
5819 , { 280, 220, 250, 280, 250}
5820 }
5821 , {{ 350, 230, 260, 350, 260}
5822 , { 290, 230, 260, 290, 260}
5823 , { 350, 220, 260, 350, 260}
5824 , { 290, 230, 260, 290, 260}
5825 , { 350, 220, 260, 350, 260}
5826 }
5827 , {{ 280, 220, 250, 280, 250}
5828 , { 250, 190, 220, 250, 220}
5829 , { 280, 220, 250, 280, 250}
5830 , { 260, 70, 100, 260, 100}
5831 , { 280, 220, 250, 280, 250}
5832 }
5833 , {{ 350, 230, 260, 350, 260}
5834 , { 290, 230, 260, 290, 260}
5835 , { 350, 220, 260, 350, 260}
5836 , { 290, 230, 260, 290, 260}
5837 , { 200, 190, 170, 200, 170}
5838 }
5839 }
5840 , {{{{ 280, 280, 280, 280, 280}
5841 , { 280, 280, 280, 280, 280}
5842 , { 260, 260, 260, 260, 260}
5843 , { 260, 260, 260, 260, 260}
5844 , { 260, 260, 260, 260, 260}
5845 }
5846 , {{ 280, 280, 280, 280, 280}
5847 , { 280, 280, 280, 280, 280}
5848 , { 250, 250, 250, 250, 250}
```

```
5849 , { 210, 150, 210, 150, 210}
5850 , { 250, 250, 250, 250, 250}
5851 }
5852 , {{ 260, 260, 260, 260, 260}
5853 , { 260, 260, 260, 260, 260}
5854 , { 260, 260, 260, 260, 260}
5855 , { 260, 260, 260, 260, 260}
5856 , { 260, 260, 260, 260, 260}
5857 }
5858 , {{ 280, 250, 280, 250, 280}
5859 , { 280, 220, 280, 220, 280}
5860 , { 250, 250, 250, 250, 250}
5861 , { 100, 100, 100, 100, 100}
5862 , { 250, 250, 250, 250, 250}
5863 }
5864 , {{ 260, 260, 260, 260, 260}
5865 , { 260, 260, 260, 260, 260}
5866 , { 260, 260, 260, 260, 260}
5867 , { 260, 260, 260, 260, 260}
5868 , { 170, 170, 170, 170, 170}
5869 }
5870 }
5871 , {{{ 280, 180, 280, 230, 280}
5872 , { 280, 140, 280, 220, 280}
5873 , { 260, 180, 260, 130, 260}
5874 , { 260, 130, 260, 230, 260}
5875 , { 260, 180, 260, 170, 260}
5876 }
5877 , {{ 280, 140, 280, 150, 280}
5878 , { 280, 140, 280, 150, 280}
5879 , { 250, 120, 250, 120, 250}
5880 , { 150, 20, 150, 150, 150}
5881 , { 250, 120, 250, 120, 250}
5882 }
5883 , {{ 260, 180, 260, 130, 260}
5884 , { 260, 130, 260, 130, 260}
5885 , { 260, 180, 260, 130, 260}
5886 , { 260, 130, 260, 130, 260}
5887 , { 260, 180, 260, 130, 260}
5888 }
5889 , {{ 250, 120, 250, 230, 250}
5890 , { 220, 90, 220, 220, 220}
5891 , { 250, 120, 250, 120, 250}
5892 , { 230, 100, 100, 230, 100}
5893 , { 250, 120, 250, 120, 250}
5894 }
5895 , {{ 260, 180, 260, 170, 260}
5896 , { 260, 130, 260, 130, 260}
5897 , { 260, 180, 260, 130, 260}
5898 , { 260, 130, 260, 130, 260}
5899 , { 170, 30, 170, 170, 170}
5900 }
5901 }
5902 , {{{ 340, 280, 280, 280, 340}
5903 , { 340, 280, 280, 280, 340}
5904 , { 260, 260, 260, 260, 260}
5905 , { 260, 260, 260, 260, 260}
5906 , { 260, 260, 260, 260, 260}
5907 }
5908 , {{ 340, 280, 280, 280, 340}
5909 , { 340, 280, 280, 280, 340}
5910 , { 250, 250, 250, 250, 250}
5911 , { 210, 150, 210, 150, 150}
5912 , { 250, 250, 250, 250, 250}
5913 }
5914 , {{ 260, 260, 260, 260, 260}
5915 , { 260, 260, 260, 260, 260}
5916 , { 260, 260, 260, 260, 260}
5917 , { 260, 260, 260, 260, 260}
5918 , { 260, 260, 260, 260, 260}
5919 }
5920 , {{ 280, 250, 280, 250, 250}
5921 , { 280, 220, 280, 220, 220}
5922 , { 250, 250, 250, 250, 250}
5923 , { 230, 100, 100, 100, 230}
5924 , { 250, 250, 250, 250, 250}
5925 }
5926 , {{ 260, 260, 260, 260, 260}
5927 , { 260, 260, 260, 260, 260}
5928 , { 260, 260, 260, 260, 260}
5929 , { 260, 260, 260, 260, 260}
5930 , { 170, 170, 170, 170, 170}
5931 }
5932 }
5933 }
5934 , {{{{ 370, 280, 280, 370, 340}
5935 , { 340, 280, 280, 310, 340}
```

```
5936 , { 370, 280, 280, 370, 280}
5937 , { 310, 280, 280, 310, 280}
5938 , { 370, 280, 280, 370, 280}
5939 }
5940 , {{ 340, 280, 280, 310, 340}
5941 , { 340, 280, 280, 310, 340}
5942 , { 260, 230, 230, 260, 230}
5943 , { 230, 170, 230, 200, 230}
5944 , { 260, 230, 230, 260, 230}
5945 }
5946 , {{ 370, 280, 280, 370, 280}
5947 , { 310, 280, 280, 310, 280}
5948 , { 370, 280, 280, 370, 280}
5949 , { 310, 280, 280, 310, 280}
5950 , { 370, 280, 280, 370, 280}
5951 }
5952 , {{ 280, 230, 240, 280, 250}
5953 , { 240, 180, 240, 210, 240}
5954 , { 260, 230, 230, 260, 230}
5955 , { 280, 120, 120, 280, 250}
5956 , { 260, 230, 230, 260, 230}
5957 }
5958 , {{ 340, 280, 280, 340, 280}
5959 , { 310, 280, 280, 310, 280}
5960 , { 340, 250, 250, 340, 250}
5961 , { 310, 280, 280, 310, 280}
5962 , { 220, 220, 190, 220, 190}
5963 }
5964 }
5965 , {{{ 370, 240, 280, 370, 280}
5966 , { 310, 240, 280, 310, 280}
5967 , { 370, 240, 280, 370, 280}
5968 , { 310, 240, 280, 310, 280}
5969 , { 370, 240, 280, 370, 280}
5970 }
5971 , {{ 310, 240, 280, 310, 280}
5972 , { 310, 240, 280, 310, 280}
5973 , { 260, 200, 230, 260, 230}
5974 , { 200, 140, 170, 200, 170}
5975 , { 260, 200, 230, 260, 230}
5976 }
5977 , {{ 370, 240, 280, 370, 280}
5978 , { 310, 240, 280, 310, 280}
5979 , { 370, 240, 280, 370, 280}
5980 , { 310, 240, 280, 310, 280}
5981 , { 370, 240, 280, 370, 280}
5982 }
5983 , {{ 280, 200, 230, 280, 230}
5984 , { 210, 150, 180, 210, 180}
5985 , { 260, 200, 230, 260, 230}
5986 , { 280, 90, 120, 280, 120}
5987 , { 260, 200, 230, 260, 230}
5988 }
5989 , {{ 340, 240, 280, 340, 280}
5990 , { 310, 240, 280, 310, 280}
5991 , { 340, 210, 250, 340, 250}
5992 , { 310, 240, 280, 310, 280}
5993 , { 220, 220, 190, 220, 190}
5994 }
5995 }
5996 , {{{ 280, 280, 280, 280, 280}
5997 , { 280, 280, 280, 280, 280}
5998 , { 280, 280, 280, 280, 280}
5999 , { 280, 280, 280, 280, 280}
6000 , { 280, 280, 280, 280, 280}
6001 }
6002 , {{ 280, 280, 280, 280, 280}
6003 , { 280, 280, 280, 280, 280}
6004 , { 230, 230, 230, 230, 230}
6005 , { 230, 170, 230, 170, 230}
6006 , { 230, 230, 230, 230, 230}
6007 }
6008 , {{ 280, 280, 280, 280, 280}
6009 , { 280, 280, 280, 280, 280}
6010 , { 280, 280, 280, 280, 280}
6011 , { 280, 280, 280, 280, 280}
6012 , { 280, 280, 280, 280, 280}
6013 }
6014 , {{ 240, 230, 240, 230, 240}
6015 , { 240, 180, 240, 180, 240}
6016 , { 230, 230, 230, 230, 230}
6017 , { 120, 120, 120, 120, 120}
6018 , { 230, 230, 230, 230, 230}
6019 }
6020 , {{ 280, 280, 280, 280, 280}
6021 , { 280, 280, 280, 280, 280}
6022 , { 250, 250, 250, 250, 250}
```

```
6023 , { 280, 280, 280, 280, 280}
6024 , { 190, 190, 190, 190, 190}
6025 }
6026 }
6027 , {{ { 280, 200, 280, 250, 280}
6028 , { 280, 140, 280, 180, 280}
6029 , { 280, 200, 280, 150, 280}
6030 , { 280, 140, 280, 250, 280}
6031 , { 280, 200, 280, 190, 280}
6032 }
6033 , {{ { 280, 140, 280, 170, 280}
6034 , { 280, 140, 280, 150, 280}
6035 , { 230, 100, 230, 100, 230}
6036 , { 170, 40, 170, 170, 170}
6037 , { 230, 100, 230, 100, 230}
6038 }
6039 , {{ { 280, 200, 280, 150, 280}
6040 , { 280, 140, 280, 150, 280}
6041 , { 280, 200, 280, 150, 280}
6042 , { 280, 140, 280, 150, 280}
6043 , { 280, 200, 280, 150, 280}
6044 }
6045 , {{ { 250, 120, 230, 250, 230}
6046 , { 180, 50, 180, 180, 180}
6047 , { 230, 100, 230, 100, 230}
6048 , { 250, 120, 120, 250, 120}
6049 , { 230, 100, 230, 100, 230}
6050 }
6051 , {{ { 280, 170, 280, 190, 280}
6052 , { 280, 140, 280, 150, 280}
6053 , { 250, 170, 250, 120, 250}
6054 , { 280, 140, 280, 150, 280}
6055 , { 190, 60, 190, 190, 190}
6056 }
6057 }
6058 , {{ { 340, 280, 280, 280, 340}
6059 , { 340, 280, 280, 280, 340}
6060 , { 280, 280, 280, 280, 280}
6061 , { 280, 280, 280, 280, 280}
6062 , { 280, 280, 280, 280, 280}
6063 }
6064 , {{ { 340, 280, 280, 280, 340}
6065 , { 340, 280, 280, 280, 340}
6066 , { 230, 230, 230, 230, 230}
6067 , { 230, 170, 230, 170, 170}
6068 , { 230, 230, 230, 230, 230}
6069 }
6070 , {{ { 280, 280, 280, 280, 280}
6071 , { 280, 280, 280, 280, 280}
6072 , { 280, 280, 280, 280, 280}
6073 , { 280, 280, 280, 280, 280}
6074 , { 280, 280, 280, 280, 280}
6075 }
6076 , {{ { 250, 230, 240, 230, 250}
6077 , { 240, 180, 240, 180, 180}
6078 , { 230, 230, 230, 230, 230}
6079 , { 250, 120, 120, 120, 250}
6080 , { 230, 230, 230, 230, 230}
6081 }
6082 , {{ { 280, 280, 280, 280, 280}
6083 , { 280, 280, 280, 280, 280}
6084 , { 250, 250, 250, 250, 250}
6085 , { 280, 280, 280, 280, 280}
6086 , { 190, 190, 190, 190, 190}
6087 }
6088 }
6089 }
6090 , {{{ { 400, 360, 370, 400, 400}
6091 , { 400, 360, 370, 370, 400}
6092 , { 400, 310, 310, 400, 310}
6093 , { 340, 310, 310, 340, 310}
6094 , { 400, 330, 310, 400, 310}
6095 }
6096 , {{ { 400, 360, 340, 370, 400}
6097 , { 400, 360, 340, 370, 400}
6098 , { 340, 310, 310, 340, 310}
6099 , { 290, 230, 290, 260, 290}
6100 , { 340, 310, 310, 340, 310}
6101 }
6102 , {{ { 400, 310, 310, 400, 310}
6103 , { 340, 310, 310, 340, 310}
6104 , { 400, 310, 310, 400, 310}
6105 , { 340, 310, 310, 340, 310}
6106 , { 400, 310, 310, 400, 310}
6107 }
6108 , {{ { 370, 360, 370, 340, 370}
6109 , { 370, 360, 370, 340, 370}
```

```
6110 , { 340, 310, 310, 340, 310}
6111 , { 340, 180, 180, 340, 310}
6112 , { 340, 310, 310, 340, 310}
6113 }
6114 , {{ 400, 330, 310, 400, 310}
6115 , { 340, 310, 310, 340, 310}
6116 , { 400, 310, 310, 400, 310}
6117 , { 340, 310, 310, 340, 310}
6118 , { 340, 330, 310, 340, 310}
6119 }
6120 }
6121 , {{{ 400, 360, 340, 400, 340}
6122 , { 370, 360, 340, 370, 340}
6123 , { 400, 270, 310, 400, 310}
6124 , { 340, 270, 310, 340, 310}
6125 , { 400, 330, 310, 400, 310}
6126 }
6127 , {{ 370, 360, 340, 370, 340}
6128 , { 370, 360, 340, 370, 340}
6129 , { 340, 270, 310, 340, 310}
6130 , { 260, 190, 230, 260, 230}
6131 , { 340, 270, 310, 340, 310}
6132 }
6133 , {{ 400, 270, 310, 400, 310}
6134 , { 340, 270, 310, 340, 310}
6135 , { 400, 270, 310, 400, 310}
6136 , { 340, 270, 310, 340, 310}
6137 , { 400, 270, 310, 400, 310}
6138 }
6139 , {{ 360, 360, 310, 340, 310}
6140 , { 360, 360, 310, 340, 310}
6141 , { 340, 270, 310, 340, 310}
6142 , { 340, 140, 180, 340, 180}
6143 , { 340, 270, 310, 340, 310}
6144 }
6145 , {{ 400, 330, 310, 400, 310}
6146 , { 340, 270, 310, 340, 310}
6147 , { 400, 270, 310, 400, 310}
6148 , { 340, 270, 310, 340, 310}
6149 , { 340, 330, 310, 340, 310}
6150 }
6151 }
6152 , {{{ 370, 340, 370, 340, 370}
6153 , { 370, 340, 370, 340, 370}
6154 , { 310, 310, 310, 310, 310}
6155 , { 310, 310, 310, 310, 310}
6156 , { 310, 310, 310, 310, 310}
6157 }
6158 , {{ 340, 340, 340, 340, 340}
6159 , { 340, 340, 340, 340, 340}
6160 , { 310, 310, 310, 310, 310}
6161 , { 290, 230, 290, 230, 290}
6162 , { 310, 310, 310, 310, 310}
6163 }
6164 , {{ 310, 310, 310, 310, 310}
6165 , { 310, 310, 310, 310, 310}
6166 , { 310, 310, 310, 310, 310}
6167 , { 310, 310, 310, 310, 310}
6168 , { 310, 310, 310, 310, 310}
6169 }
6170 , {{ 370, 310, 370, 310, 370}
6171 , { 370, 310, 370, 310, 370}
6172 , { 310, 310, 310, 310, 310}
6173 , { 180, 180, 180, 180, 180}
6174 , { 310, 310, 310, 310, 310}
6175 }
6176 , {{ 310, 310, 310, 310, 310}
6177 , { 310, 310, 310, 310, 310}
6178 , { 310, 310, 310, 310, 310}
6179 , { 310, 310, 310, 310, 310}
6180 , { 310, 310, 310, 310, 310}
6181 }
6182 }
6183 , {{{ 340, 230, 340, 310, 340}
6184 , { 340, 220, 340, 310, 340}
6185 , { 310, 230, 310, 180, 310}
6186 , { 310, 170, 310, 310, 310}
6187 , { 310, 230, 310, 310, 310}
6188 }
6189 , {{ 340, 220, 340, 230, 340}
6190 , { 340, 220, 340, 210, 340}
6191 , { 310, 170, 310, 180, 310}
6192 , { 230, 40, 230, 230, 230}
6193 , { 310, 170, 310, 180, 310}
6194 }
6195 , {{ 310, 230, 310, 180, 310}
6196 , { 310, 170, 310, 180, 310}
```

```
6197 , { 310, 230, 310, 180, 310}
6198 , { 310, 170, 310, 180, 310}
6199 , { 310, 230, 310, 180, 310}
6200 }
6201 , {{ 310, 170, 310, 310, 310}
6202 , { 310, 170, 310, 310, 310}
6203 , { 310, 170, 310, 180, 310}
6204 , { 310, 170, 180, 310, 180}
6205 , { 310, 170, 310, 180, 310}
6206 }
6207 , {{ 310, 230, 310, 310, 310}
6208 , { 310, 170, 310, 180, 310}
6209 , { 310, 230, 310, 180, 310}
6210 , { 310, 170, 310, 180, 310}
6211 , { 310, 170, 310, 310, 310}
6212 }
6213 }
6214 , {{{ 400, 340, 370, 340, 400}
6215 , { 400, 340, 370, 340, 400}
6216 , { 310, 310, 310, 310, 310}
6217 , { 310, 310, 310, 310, 310}
6218 , { 310, 310, 310, 310, 310}
6219 }
6220 , {{ 400, 340, 340, 340, 400}
6221 , { 400, 340, 340, 340, 400}
6222 , { 310, 310, 310, 310, 310}
6223 , { 290, 230, 290, 230, 230}
6224 , { 310, 310, 310, 310, 310}
6225 }
6226 , {{ 310, 310, 310, 310, 310}
6227 , { 310, 310, 310, 310, 310}
6228 , { 310, 310, 310, 310, 310}
6229 , { 310, 310, 310, 310, 310}
6230 , { 310, 310, 310, 310, 310}
6231 }
6232 , {{ 370, 310, 370, 310, 310}
6233 , { 370, 310, 370, 310, 310}
6234 , { 310, 310, 310, 310, 310}
6235 , { 310, 180, 180, 180, 310}
6236 , { 310, 310, 310, 310, 310}
6237 }
6238 , {{ 310, 310, 310, 310, 310}
6239 , { 310, 310, 310, 310, 310}
6240 , { 310, 310, 310, 310, 310}
6241 , { 310, 310, 310, 310, 310}
6242 , { 310, 310, 310, 310, 310}
6243 }
6244 }
6245 }
6246 }
6247 , {{{{{ INF, INF, INF, INF, INF
6248 , { INF, INF, INF, INF, INF
6249 , { INF, INF, INF, INF, INF
6250 , { INF, INF, INF, INF, INF
6251 , { INF, INF, INF, INF, INF
6252 }
6253 , {{ INF, INF, INF, INF, INF
6254 , { INF, INF, INF, INF, INF
6255 , { INF, INF, INF, INF, INF
6256 , { INF, INF, INF, INF, INF
6257 , { INF, INF, INF, INF, INF
6258 }
6259 , {{ INF, INF, INF, INF, INF
6260 , { INF, INF, INF, INF, INF
6261 , { INF, INF, INF, INF, INF
6262 , { INF, INF, INF, INF, INF
6263 , { INF, INF, INF, INF, INF
6264 }
6265 , {{ INF, INF, INF, INF, INF
6266 , { INF, INF, INF, INF, INF
6267 , { INF, INF, INF, INF, INF
6268 , { INF, INF, INF, INF, INF
6269 , { INF, INF, INF, INF, INF
6270 }
6271 , {{ INF, INF, INF, INF, INF
6272 , { INF, INF, INF, INF, INF
6273 , { INF, INF, INF, INF, INF
6274 , { INF, INF, INF, INF, INF
6275 , { INF, INF, INF, INF, INF
6276 }
6277 }
6278 , {{{ INF, INF, INF, INF, INF
6279 , { INF, INF, INF, INF, INF
6280 , { INF, INF, INF, INF, INF
6281 , { INF, INF, INF, INF, INF
6282 , { INF, INF, INF, INF, INF
6283 }
```

```
6284 , {{ INF, INF, INF, INF, INF }
6285 , { INF, INF, INF, INF, INF }
6286 , { INF, INF, INF, INF, INF }
6287 , { INF, INF, INF, INF, INF }
6288 , { INF, INF, INF, INF, INF }
6289 }
6290 , {{ INF, INF, INF, INF, INF }
6291 , { INF, INF, INF, INF, INF }
6292 , { INF, INF, INF, INF, INF }
6293 , { INF, INF, INF, INF, INF }
6294 , { INF, INF, INF, INF, INF }
6295 }
6296 , {{ INF, INF, INF, INF, INF }
6297 , { INF, INF, INF, INF, INF }
6298 , { INF, INF, INF, INF, INF }
6299 , { INF, INF, INF, INF, INF }
6300 , { INF, INF, INF, INF, INF }
6301 }
6302 , {{ INF, INF, INF, INF, INF }
6303 , { INF, INF, INF, INF, INF }
6304 , { INF, INF, INF, INF, INF }
6305 , { INF, INF, INF, INF, INF }
6306 , { INF, INF, INF, INF, INF }
6307 }
6308 }
6309 , {{{ INF, INF, INF, INF, INF }
6310 , { INF, INF, INF, INF, INF }
6311 , { INF, INF, INF, INF, INF }
6312 , { INF, INF, INF, INF, INF }
6313 , { INF, INF, INF, INF, INF }
6314 }
6315 , {{ INF, INF, INF, INF, INF }
6316 , { INF, INF, INF, INF, INF }
6317 , { INF, INF, INF, INF, INF }
6318 , { INF, INF, INF, INF, INF }
6319 , { INF, INF, INF, INF, INF }
6320 }
6321 , {{ INF, INF, INF, INF, INF }
6322 , { INF, INF, INF, INF, INF }
6323 , { INF, INF, INF, INF, INF }
6324 , { INF, INF, INF, INF, INF }
6325 , { INF, INF, INF, INF, INF }
6326 }
6327 , {{ INF, INF, INF, INF, INF }
6328 , { INF, INF, INF, INF, INF }
6329 , { INF, INF, INF, INF, INF }
6330 , { INF, INF, INF, INF, INF }
6331 , { INF, INF, INF, INF, INF }
6332 }
6333 , {{ INF, INF, INF, INF, INF }
6334 , { INF, INF, INF, INF, INF }
6335 , { INF, INF, INF, INF, INF }
6336 , { INF, INF, INF, INF, INF }
6337 , { INF, INF, INF, INF, INF }
6338 }
6339 }
6340 , {{{ INF, INF, INF, INF, INF }
6341 , { INF, INF, INF, INF, INF }
6342 , { INF, INF, INF, INF, INF }
6343 , { INF, INF, INF, INF, INF }
6344 , { INF, INF, INF, INF, INF }
6345 }
6346 , {{ INF, INF, INF, INF, INF }
6347 , { INF, INF, INF, INF, INF }
6348 , { INF, INF, INF, INF, INF }
6349 , { INF, INF, INF, INF, INF }
6350 , { INF, INF, INF, INF, INF }
6351 }
6352 , {{ INF, INF, INF, INF, INF }
6353 , { INF, INF, INF, INF, INF }
6354 , { INF, INF, INF, INF, INF }
6355 , { INF, INF, INF, INF, INF }
6356 , { INF, INF, INF, INF, INF }
6357 }
6358 , {{ INF, INF, INF, INF, INF }
6359 , { INF, INF, INF, INF, INF }
6360 , { INF, INF, INF, INF, INF }
6361 , { INF, INF, INF, INF, INF }
6362 , { INF, INF, INF, INF, INF }
6363 }
6364 , {{ INF, INF, INF, INF, INF }
6365 , { INF, INF, INF, INF, INF }
6366 , { INF, INF, INF, INF, INF }
6367 , { INF, INF, INF, INF, INF }
6368 , { INF, INF, INF, INF, INF }
6369 }
6370 }
```

```
6371 ,{{{{ INF, INF, INF, INF, INF
6372 ,{ INF, INF, INF, INF, INF
6373 ,{ INF, INF, INF, INF, INF
6374 ,{ INF, INF, INF, INF, INF
6375 ,{ INF, INF, INF, INF, INF
6376 }
6377 ,{{ INF, INF, INF, INF, INF
6378 ,{ INF, INF, INF, INF, INF
6379 ,{ INF, INF, INF, INF, INF
6380 ,{ INF, INF, INF, INF, INF
6381 ,{ INF, INF, INF, INF, INF
6382 }
6383 ,{{ INF, INF, INF, INF, INF
6384 ,{ INF, INF, INF, INF, INF
6385 ,{ INF, INF, INF, INF, INF
6386 ,{ INF, INF, INF, INF, INF
6387 ,{ INF, INF, INF, INF, INF
6388 }
6389 ,{{ INF, INF, INF, INF, INF
6390 ,{ INF, INF, INF, INF, INF
6391 ,{ INF, INF, INF, INF, INF
6392 ,{ INF, INF, INF, INF, INF
6393 ,{ INF, INF, INF, INF, INF
6394 }
6395 ,{{ INF, INF, INF, INF, INF
6396 ,{ INF, INF, INF, INF, INF
6397 ,{ INF, INF, INF, INF, INF
6398 ,{ INF, INF, INF, INF, INF
6399 ,{ INF, INF, INF, INF, INF
6400 }
6401 }
6402 }
6403 ,{{{ 240, 240, 220, 230, 220
6404 ,{ 240, 240, 220, 210, 220
6405 ,{ 230, 220, 210, 230, 210
6406 ,{ 240, 240, 220, 210, 220
6407 ,{ 210, 210, 190, 210, 190
6408 }
6409 ,{{ 200, 200, 180, 170, 180
6410 ,{ 200, 200, 180, 170, 180
6411 ,{ 190, 190, 180, 170, 180
6412 ,{ 140, 100, 140, 80, 140
6413 ,{ 190, 190, 180, 170, 180
6414 }
6415 ,{{ 240, 240, 220, 230, 220
6416 ,{ 240, 240, 220, 210, 220
6417 ,{ 230, 220, 210, 230, 210
6418 ,{ 240, 240, 220, 210, 220
6419 ,{ 210, 210, 190, 210, 190
6420 }
6421 ,{{ 190, 190, 180, 170, 180
6422 ,{ 140, 100, 140, 80, 140
6423 ,{ 190, 190, 180, 170, 180
6424 ,{ 130, 50, 30, 130, 70
6425 ,{ 190, 190, 180, 170, 180
6426 }
6427 ,{{ 240, 240, 220, 210, 220
6428 ,{ 240, 240, 220, 210, 220
6429 ,{ 210, 210, 190, 210, 190
6430 ,{ 240, 240, 220, 210, 220
6431 ,{ 180, 180, 100, 90, 100
6432 }
6433 }
6434 ,{{{ 240, 240, 220, 230, 220
6435 ,{ 240, 240, 220, 180, 220
6436 ,{ 230, 220, 210, 230, 210
6437 ,{ 240, 240, 220, 180, 220
6438 ,{ 210, 210, 190, 210, 190
6439 }
6440 ,{{ 200, 200, 180, 140, 180
6441 ,{ 200, 200, 180, 140, 180
6442 ,{ 190, 190, 180, 140, 180
6443 ,{ 100, 100, 90, 50, 90
6444 ,{ 190, 190, 180, 140, 180
6445 }
6446 ,{{ 240, 240, 220, 230, 220
6447 ,{ 240, 240, 220, 180, 220
6448 ,{ 230, 220, 210, 230, 210
6449 ,{ 240, 240, 220, 180, 220
6450 ,{ 210, 210, 190, 210, 190
6451 }
6452 ,{{ 190, 190, 180, 140, 180
6453 ,{ 100, 100, 90, 50, 90
6454 ,{ 190, 190, 180, 140, 180
6455 ,{ 120, 50, 30, 120, 30
6456 ,{ 190, 190, 180, 140, 180
6457 }}
```

```
6458 , {{ 240, 240, 220, 210, 220}
6459 , { 240, 240, 220, 180, 220}
6460 , { 210, 210, 190, 210, 190}
6461 , { 240, 240, 220, 180, 220}
6462 , { 180, 180, 100, 60, 100}
6463 }
6464 }
6465 , {{ 220, 210, 220, 210, 220}
6466 , { 220, 210, 220, 210, 220}
6467 , { 200, 200, 200, 200, 200}
6468 , { 220, 210, 220, 210, 220}
6469 , { 190, 180, 190, 180, 190}
6470 }
6471 , {{ 180, 170, 180, 170, 180}
6472 , { 180, 170, 180, 170, 180}
6473 , { 170, 170, 170, 170, 170}
6474 , { 140, 80, 140, 80, 140}
6475 , { 170, 170, 170, 170, 170}
6476 }
6477 , {{ 220, 210, 220, 210, 220}
6478 , { 220, 210, 220, 210, 220}
6479 , { 200, 200, 200, 200, 200}
6480 , { 220, 210, 220, 210, 220}
6481 , { 190, 180, 190, 180, 190}
6482 }
6483 , {{ 170, 170, 170, 170, 170}
6484 , { 140, 80, 140, 80, 140}
6485 , { 170, 170, 170, 170, 170}
6486 , { 30, 20, 30, 20, 30}
6487 , { 170, 170, 170, 170, 170}
6488 }
6489 , {{ 220, 210, 220, 210, 220}
6490 , { 220, 210, 220, 210, 220}
6491 , { 190, 180, 190, 180, 190}
6492 , { 220, 210, 220, 210, 220}
6493 , { 100, 90, 100, 90, 100}
6494 }
6495 }
6496 , {{ 220, 160, 220, 130, 220}
6497 , { 220, 110, 220, 60, 220}
6498 , { 210, 160, 210, 50, 210}
6499 , { 220, 110, 220, 130, 220}
6500 , { 190, 140, 190, 70, 190}
6501 }
6502 , {{ 180, 70, 180, 60, 180}
6503 , { 180, 70, 180, 20, 180}
6504 , { 180, 70, 180, 20, 180}
6505 , { 90, -20, 90, 60, 90}
6506 , { 180, 70, 180, 20, 180}
6507 }
6508 , {{ 220, 160, 220, 60, 220}
6509 , { 220, 110, 220, 60, 220}
6510 , { 210, 160, 210, 50, 210}
6511 , { 220, 110, 220, 60, 220}
6512 , { 190, 140, 190, 30, 190}
6513 }
6514 , {{ 180, 70, 180, 130, 180}
6515 , { 90, -20, 90, 60, 90}
6516 , { 180, 70, 180, 20, 180}
6517 , { 130, 50, 30, 130, 30}
6518 , { 180, 70, 180, 20, 180}
6519 }
6520 , {{ 220, 140, 220, 70, 220}
6521 , { 220, 110, 220, 60, 220}
6522 , { 190, 140, 190, 30, 190}
6523 , { 220, 110, 220, 60, 220}
6524 , { 100, 0, 100, 70, 100}
6525 }
6526 }
6527 , {{ 220, 210, 220, 210, 150}
6528 , { 220, 210, 220, 210, 150}
6529 , { 200, 200, 200, 200, 110}
6530 , { 220, 210, 220, 210, 130}
6531 , { 190, 180, 190, 180, 100}
6532 }
6533 , {{ 180, 170, 180, 170, 150}
6534 , { 180, 170, 180, 170, 150}
6535 , { 170, 170, 170, 170, 80}
6536 , { 140, 80, 140, 80, 0}
6537 , { 170, 170, 170, 170, 80}
6538 }
6539 , {{ 220, 210, 220, 210, 130}
6540 , { 220, 210, 220, 210, 130}
6541 , { 200, 200, 200, 200, 110}
6542 , { 220, 210, 220, 210, 130}
6543 , { 190, 180, 190, 180, 100}
6544 }
```

```
6545 , {{ 170, 170, 170, 170, 80}
6546 , { 140, 80, 140, 80, 0}
6547 , { 170, 170, 170, 170, 80}
6548 , { 70, 20, 30, 20, 70}
6549 , { 170, 170, 170, 170, 80}
6550 }
6551 , {{ 220, 210, 220, 210, 130}
6552 , { 220, 210, 220, 210, 130}
6553 , { 190, 180, 190, 180, 100}
6554 , { 220, 210, 220, 210, 130}
6555 , { 100, 90, 100, 90, 10}
6556 }
6557 }
6558 }
6559 , {{{{ 210, 210, 200, 200, 200}
6560 , { 210, 210, 200, 190, 200}
6561 , { 200, 190, 180, 200, 180}
6562 , { 180, 180, 170, 160, 170}
6563 , { 190, 190, 170, 190, 170}
6564 }
6565 , {{ 210, 210, 200, 190, 200}
6566 , { 210, 210, 200, 190, 200}
6567 , { 190, 190, 170, 160, 170}
6568 , { 50, 10, 50, -10, 50}
6569 , { 190, 190, 170, 160, 170}
6570 }
6571 , {{ 190, 190, 170, 190, 170}
6572 , { 180, 180, 170, 160, 170}
6573 , { 190, 190, 170, 190, 170}
6574 , { 180, 180, 170, 160, 170}
6575 , { 190, 190, 170, 190, 170}
6576 }
6577 , {{ 190, 190, 170, 160, 170}
6578 , { 110, 70, 110, 50, 110}
6579 , { 190, 190, 170, 160, 170}
6580 , { 130, 50, 30, 130, 70}
6581 , { 190, 190, 170, 160, 170}
6582 }
6583 , {{ 200, 190, 180, 200, 180}
6584 , { 180, 180, 170, 160, 170}
6585 , { 200, 190, 180, 200, 180}
6586 , { 180, 180, 170, 160, 170}
6587 , { 170, 170, 100, 90, 100}
6588 }
6589 }
6590 , {{{{ 210, 210, 200, 200, 200}
6591 , { 210, 210, 200, 160, 200}
6592 , { 200, 190, 180, 200, 180}
6593 , { 180, 180, 170, 130, 170}
6594 , { 190, 190, 170, 190, 170}
6595 }
6596 , {{ 210, 210, 200, 160, 200}
6597 , { 210, 210, 200, 160, 200}
6598 , { 190, 190, 170, 130, 170}
6599 , { 10, 10, 0, -40, 0}
6600 , { 190, 190, 170, 130, 170}
6601 }
6602 , {{ 190, 190, 170, 190, 170}
6603 , { 180, 180, 170, 130, 170}
6604 , { 190, 190, 170, 190, 170}
6605 , { 180, 180, 170, 130, 170}
6606 , { 190, 190, 170, 190, 170}
6607 }
6608 , {{ 190, 190, 170, 130, 170}
6609 , { 70, 70, 60, 20, 60}
6610 , { 190, 190, 170, 130, 170}
6611 , { 120, 50, 30, 120, 30}
6612 , { 190, 190, 170, 130, 170}
6613 }
6614 , {{ 200, 190, 180, 200, 180}
6615 , { 180, 180, 170, 130, 170}
6616 , { 200, 190, 180, 200, 180}
6617 , { 180, 180, 170, 130, 170}
6618 , { 170, 170, 100, 60, 100}
6619 }
6620 }
6621 , {{{{ 190, 190, 190, 190, 190}
6622 , { 190, 190, 190, 190, 190}
6623 , { 170, 170, 170, 170, 170}
6624 , { 160, 160, 160, 160, 160}
6625 , { 170, 160, 170, 160, 170}
6626 }
6627 , {{ 190, 190, 190, 190, 190}
6628 , { 190, 190, 190, 190, 190}
6629 , { 170, 160, 170, 160, 170}
6630 , { 50, -10, 50, -10, 50}
6631 , { 170, 160, 170, 160, 170}
```

```
6632      }
6633 ,{{ 170,   160,   170,   160,   170}
6634 ,{ 160,   160,   160,   160,   160}
6635 ,{ 170,   160,   170,   160,   170}
6636 ,{ 160,   160,   160,   160,   160}
6637 ,{ 170,   160,   170,   160,   170}
6638      }
6639 ,{{ 170,   160,   170,   160,   170}
6640 ,{ 110,    50,   110,    50,   110}
6641 ,{ 170,   160,   170,   160,   170}
6642 ,{ 30,     20,    30,    20,    30}
6643 ,{ 170,   160,   170,   160,   170}
6644      }
6645 ,{{ 170,   170,   170,   170,   170}
6646 ,{ 160,   160,   160,   160,   160}
6647 ,{ 170,   170,   170,   170,   170}
6648 ,{ 160,   160,   160,   160,   160}
6649 ,{ 90,     90,    90,    90,    90}
6650      }
6651      }
6652 ,{{{ 200,   130,   200,   130,   200}
6653 ,{ 200,    90,   200,    40,   200}
6654 ,{ 180,   130,   180,    20,   180}
6655 ,{ 170,    60,   170,   130,   170}
6656 ,{ 170,   120,   170,    70,   170}
6657      }
6658 ,{{ 200,    90,   200,    40,   200}
6659 ,{ 200,    90,   200,    40,   200}
6660 ,{ 170,    60,   170,    10,   170}
6661 ,{ 0,   -110,    0,   -30,    0}
6662 ,{ 170,    60,   170,    10,   170}
6663      }
6664 ,{{ 170,   120,   170,    10,   170}
6665 ,{ 170,    60,   170,    10,   170}
6666 ,{ 170,   120,   170,    10,   170}
6667 ,{ 170,    60,   170,    10,   170}
6668 ,{ 170,   120,   170,    10,   170}
6669      }
6670 ,{{ 170,    60,   170,   130,   170}
6671 ,{ 60,   -50,    60,    30,    60}
6672 ,{ 170,    60,   170,    10,   170}
6673 ,{ 130,    50,    30,   130,    30}
6674 ,{ 170,    60,   170,    10,   170}
6675      }
6676 ,{{ 180,   130,   180,    70,   180}
6677 ,{ 170,    60,   170,    10,   170}
6678 ,{ 180,   130,   180,    20,   180}
6679 ,{ 170,    60,   170,    10,   170}
6680 ,{ 100,   -10,   100,    70,   100}
6681      }
6682      }
6683 ,{{{ 190,   190,   190,   190,   160}
6684 ,{ 190,   190,   190,   190,   160}
6685 ,{ 170,   170,   170,   170,    80}
6686 ,{ 160,   160,   160,   160,    70}
6687 ,{ 170,   160,   170,   160,    80}
6688      }
6689 ,{{ 190,   190,   190,   190,   160}
6690 ,{ 190,   190,   190,   190,   160}
6691 ,{ 170,   160,   170,   160,    80}
6692 ,{ 50,   -10,    50,   -10,  -100}
6693 ,{ 170,   160,   170,   160,    80}
6694      }
6695 ,{{ 170,   160,   170,   160,    80}
6696 ,{ 160,   160,   160,   160,    70}
6697 ,{ 170,   160,   170,   160,    80}
6698 ,{ 160,   160,   160,   160,    70}
6699 ,{ 170,   160,   170,   160,    80}
6700      }
6701 ,{{ 170,   160,   170,   160,    80}
6702 ,{ 110,    50,   110,    50,   -30}
6703 ,{ 170,   160,   170,   160,    80}
6704 ,{ 70,     20,    30,    20,    70}
6705 ,{ 170,   160,   170,   160,    80}
6706      }
6707 ,{{ 170,   170,   170,   170,    80}
6708 ,{ 160,   160,   160,   160,    70}
6709 ,{ 170,   170,   170,   170,    80}
6710 ,{ 160,   160,   160,   160,    70}
6711 ,{ 90,     90,    90,    90,     0}
6712      }
6713      }
6714      }
6715 ,{{{ 370,   370,   330,   320,   330}
6716 ,{ 340,   340,   330,   320,   330}
6717 ,{ 310,   310,   290,   310,   290}
6718 ,{ 310,   310,   290,   280,   290}
```

```
6719 , { 370, 370, 290, 310, 290}
6720 }
6721 , {{ 340, 340, 330, 320, 330}
6722 , { 340, 340, 330, 320, 330}
6723 , { 310, 310, 290, 280, 290}
6724 , { 270, 230, 270, 200, 270}
6725 , { 310, 310, 290, 280, 290}
6726 }
6727 , {{ 310, 310, 290, 310, 290}
6728 , { 310, 310, 290, 280, 290}
6729 , { 310, 310, 290, 310, 290}
6730 , { 310, 310, 290, 280, 290}
6731 , { 310, 310, 290, 310, 290}
6732 }
6733 , {{ 310, 310, 310, 280, 310}
6734 , { 310, 270, 310, 240, 310}
6735 , { 310, 310, 290, 280, 290}
6736 , { 260, 180, 160, 260, 200}
6737 , { 310, 310, 290, 280, 290}
6738 }
6739 , {{ 370, 370, 290, 310, 290}
6740 , { 310, 310, 290, 280, 290}
6741 , { 310, 310, 290, 310, 290}
6742 , { 310, 310, 290, 280, 290}
6743 , { 370, 370, 290, 280, 290}
6744 }
6745 }
6746 , {{{ 370, 370, 330, 310, 330}
6747 , { 340, 340, 330, 290, 330}
6748 , { 310, 310, 290, 310, 290}
6749 , { 310, 310, 290, 250, 290}
6750 , { 370, 370, 290, 310, 290}
6751 }
6752 , {{ 340, 340, 330, 290, 330}
6753 , { 340, 340, 330, 290, 330}
6754 , { 310, 310, 290, 250, 290}
6755 , { 230, 230, 210, 170, 210}
6756 , { 310, 310, 290, 250, 290}
6757 }
6758 , {{ 310, 310, 290, 310, 290}
6759 , { 310, 310, 290, 250, 290}
6760 , { 310, 310, 290, 310, 290}
6761 , { 310, 310, 290, 250, 290}
6762 , { 310, 310, 290, 310, 290}
6763 }
6764 , {{ 310, 310, 290, 250, 290}
6765 , { 270, 270, 250, 210, 250}
6766 , { 310, 310, 290, 250, 290}
6767 , { 250, 180, 160, 250, 160}
6768 , { 310, 310, 290, 250, 290}
6769 }
6770 , {{ 370, 370, 290, 310, 290}
6771 , { 310, 310, 290, 250, 290}
6772 , { 310, 310, 290, 310, 290}
6773 , { 310, 310, 290, 250, 290}
6774 , { 370, 370, 290, 250, 290}
6775 }
6776 }
6777 , {{{ 320, 320, 320, 320, 320}
6778 , { 320, 320, 320, 320, 320}
6779 , { 290, 280, 290, 280, 290}
6780 , { 290, 280, 290, 280, 290}
6781 , { 290, 280, 290, 280, 290}
6782 }
6783 , {{ 320, 320, 320, 320, 320}
6784 , { 320, 320, 320, 320, 320}
6785 , { 290, 280, 290, 280, 290}
6786 , { 270, 200, 270, 200, 270}
6787 , { 290, 280, 290, 280, 290}
6788 }
6789 , {{ 290, 280, 290, 280, 290}
6790 , { 290, 280, 290, 280, 290}
6791 , { 290, 280, 290, 280, 290}
6792 , { 290, 280, 290, 280, 290}
6793 , { 290, 280, 290, 280, 290}
6794 }
6795 , {{ 310, 280, 310, 280, 310}
6796 , { 310, 240, 310, 240, 310}
6797 , { 290, 280, 290, 280, 290}
6798 , { 160, 150, 160, 150, 160}
6799 , { 290, 280, 290, 280, 290}
6800 }
6801 , {{ 290, 280, 290, 280, 290}
6802 , { 290, 280, 290, 280, 290}
6803 , { 290, 280, 290, 280, 290}
6804 , { 290, 280, 290, 280, 290}
6805 , { 290, 280, 290, 280, 290}
```

```
6806      }
6807      }
6808 ,{{{{ 330,    240,    330,    260,    330}
6809 ,{ 330,    220,    330,    220,    330}
6810 ,{ 290,    240,    290,    130,    290}
6811 ,{ 290,    180,    290,    260,    290}
6812 ,{ 290,    240,    290,    260,    290}
6813      }
6814 ,{{ 330,    220,    330,    180,    330}
6815 ,{ 330,    220,    330,    170,    330}
6816 ,{ 290,    180,    290,    130,    290}
6817 ,{ 210,    100,    210,    180,    210}
6818 ,{ 290,    180,    290,    130,    290}
6819      }
6820 ,{{ 290,    240,    290,    130,    290}
6821 ,{ 290,    180,    290,    130,    290}
6822 ,{ 290,    240,    290,    130,    290}
6823 ,{ 290,    180,    290,    130,    290}
6824 ,{ 290,    240,    290,    130,    290}
6825      }
6826 ,{{ 290,    180,    290,    260,    290}
6827 ,{ 250,    140,    250,    220,    250}
6828 ,{ 290,    180,    290,    130,    290}
6829 ,{ 260,    180,    160,    260,    160}
6830 ,{ 290,    180,    290,    130,    290}
6831      }
6832 ,{{ 290,    240,    290,    260,    290}
6833 ,{ 290,    180,    290,    130,    290}
6834 ,{ 290,    240,    290,    130,    290}
6835 ,{ 290,    180,    290,    130,    290}
6836 ,{ 290,    180,    290,    260,    290}
6837      }
6838      }
6839 ,{{{{ 320,    320,    320,    320,    290}
6840 ,{ 320,    320,    320,    320,    290}
6841 ,{ 290,    280,    290,    280,    200}
6842 ,{ 290,    280,    290,    280,    200}
6843 ,{ 290,    280,    290,    280,    200}
6844      }
6845 ,{{{{ 320,    320,    320,    320,    290}
6846 ,{ 320,    320,    320,    320,    290}
6847 ,{ 290,    280,    290,    280,    200}
6848 ,{ 270,    200,    270,    200,    120}
6849 ,{ 290,    280,    290,    280,    200}
6850      }
6851 ,{{{{ 290,    280,    290,    280,    200}
6852 ,{ 290,    280,    290,    280,    200}
6853 ,{ 290,    280,    290,    280,    200}
6854 ,{ 290,    280,    290,    280,    200}
6855 ,{ 290,    280,    290,    280,    200}
6856      }
6857 ,{{{{ 310,    280,    310,    280,    200}
6858 ,{ 310,    240,    310,    240,    160}
6859 ,{ 290,    280,    290,    280,    200}
6860 ,{ 200,    150,    160,    150,    200}
6861 ,{ 290,    280,    290,    280,    200}
6862      }
6863 ,{{{{ 290,    280,    290,    280,    200}
6864 ,{ 290,    280,    290,    280,    200}
6865 ,{ 290,    280,    290,    280,    200}
6866 ,{ 290,    280,    290,    280,    200}
6867 ,{ 290,    280,    290,    280,    200}
6868      }
6869      }
6870      }
6871 ,{{{{{ 350,    340,    350,    280,    350}
6872 ,{ 350,    310,    350,    280,    350}
6873 ,{ 280,    280,    260,    280,    260}
6874 ,{ 280,    280,    260,    250,    260}
6875 ,{ 340,    340,    260,    280,    260}
6876      }
6877 ,{{{{ 280,    280,    260,    250,    260}
6878 ,{ 240,    240,    230,    220,    230}
6879 ,{ 280,    280,    260,    250,    260}
6880 ,{ 180,    140,    180,    120,    180}
6881 ,{ 280,    280,    260,    250,    260}
6882      }
6883 ,{{{{ 280,    280,    260,    280,    260}
6884 ,{ 280,    280,    260,    250,    260}
6885 ,{ 280,    280,    260,    280,    260}
6886 ,{ 280,    280,    260,    250,    260}
6887 ,{ 280,    280,    260,    280,    260}
6888      }
6889 ,{{{{ 350,    310,    350,    280,    350}
6890 ,{ 350,    310,    350,    280,    350}
6891 ,{ 280,    280,    260,    250,    260}
6892 ,{ 230,    150,    130,    230,    170}
```

```
6893 , { 280, 280, 260, 250, 260}
6894 }
6895 , {{ 340, 340, 260, 280, 260}
6896 , { 280, 280, 260, 250, 260}
6897 , { 280, 280, 260, 280, 260}
6898 , { 280, 280, 260, 250, 260}
6899 , { 340, 340, 260, 250, 260}
6900 }
6901 }
6902 , {{ { 340, 340, 290, 280, 290}
6903 , { 310, 310, 290, 250, 290}
6904 , { 280, 280, 260, 280, 260}
6905 , { 280, 280, 260, 220, 260}
6906 , { 340, 340, 260, 280, 260}
6907 }
6908 , {{ { 280, 280, 260, 220, 260}
6909 , { 240, 240, 230, 190, 230}
6910 , { 280, 280, 260, 220, 260}
6911 , { 140, 140, 130, 90, 130}
6912 , { 280, 280, 260, 220, 260}
6913 }
6914 , {{ { 280, 280, 260, 280, 260}
6915 , { 280, 280, 260, 220, 260}
6916 , { 280, 280, 260, 280, 260}
6917 , { 280, 280, 260, 220, 260}
6918 , { 280, 280, 260, 280, 260}
6919 }
6920 , {{ { 310, 310, 290, 250, 290}
6921 , { 310, 310, 290, 250, 290}
6922 , { 280, 280, 260, 220, 260}
6923 , { 220, 150, 130, 220, 130}
6924 , { 280, 280, 260, 220, 260}
6925 }
6926 , {{ { 340, 340, 260, 280, 260}
6927 , { 280, 280, 260, 220, 260}
6928 , { 280, 280, 260, 280, 260}
6929 , { 280, 280, 260, 220, 260}
6930 , { 340, 340, 260, 220, 260}
6931 }
6932 }
6933 , {{ { 350, 280, 350, 280, 350}
6934 , { 350, 280, 350, 280, 350}
6935 , { 260, 250, 260, 250, 260}
6936 , { 260, 250, 260, 250, 260}
6937 , { 260, 250, 260, 250, 260}
6938 }
6939 , {{ { 260, 250, 260, 250, 260}
6940 , { 220, 220, 220, 220, 220}
6941 , { 260, 250, 260, 250, 260}
6942 , { 180, 120, 180, 120, 180}
6943 , { 260, 250, 260, 250, 260}
6944 }
6945 , {{ { 260, 250, 260, 250, 260}
6946 , { 260, 250, 260, 250, 260}
6947 , { 260, 250, 260, 250, 260}
6948 , { 260, 250, 260, 250, 260}
6949 , { 260, 250, 260, 250, 260}
6950 }
6951 , {{ { 350, 280, 350, 280, 350}
6952 , { 350, 280, 350, 280, 350}
6953 , { 260, 250, 260, 250, 260}
6954 , { 130, 120, 130, 120, 130}
6955 , { 260, 250, 260, 250, 260}
6956 }
6957 , {{ { 260, 250, 260, 250, 260}
6958 , { 260, 250, 260, 250, 260}
6959 , { 260, 250, 260, 250, 260}
6960 , { 260, 250, 260, 250, 260}
6961 , { 260, 250, 260, 250, 260}
6962 }
6963 }
6964 , {{ { 290, 210, 290, 260, 290}
6965 , { 290, 180, 290, 260, 290}
6966 , { 260, 210, 260, 100, 260}
6967 , { 260, 150, 260, 230, 260}
6968 , { 260, 210, 260, 230, 260}
6969 }
6970 , {{ { 260, 150, 260, 100, 260}
6971 , { 230, 120, 230, 70, 230}
6972 , { 260, 150, 260, 100, 260}
6973 , { 130, 20, 130, 100, 130}
6974 , { 260, 150, 260, 100, 260}
6975 }
6976 , {{ { 260, 210, 260, 100, 260}
6977 , { 260, 150, 260, 100, 260}
6978 , { 260, 210, 260, 100, 260}
6979 , { 260, 150, 260, 100, 260}
```

```
6980 , { 260, 210, 260, 100, 260}
6981 }
6982 , {{ 290, 180, 290, 260, 290}
6983 , { 290, 180, 290, 260, 290}
6984 , { 260, 150, 260, 100, 260}
6985 , { 230, 150, 130, 230, 130}
6986 , { 260, 150, 260, 100, 260}
6987 }
6988 , {{ 260, 210, 260, 230, 260}
6989 , { 260, 150, 260, 100, 260}
6990 , { 260, 210, 260, 100, 260}
6991 , { 260, 150, 260, 100, 260}
6992 , { 260, 150, 260, 230, 260}
6993 }
6994 }
6995 , {{{ 350, 280, 350, 280, 200}
6996 , { 350, 280, 350, 280, 200}
6997 , { 260, 250, 260, 250, 170}
6998 , { 260, 250, 260, 250, 170}
6999 , { 260, 250, 260, 250, 170}
7000 }
7001 , {{ 260, 250, 260, 250, 190}
7002 , { 220, 220, 220, 220, 190}
7003 , { 260, 250, 260, 250, 170}
7004 , { 180, 120, 180, 120, 30}
7005 , { 260, 250, 260, 250, 170}
7006 }
7007 , {{ 260, 250, 260, 250, 170}
7008 , { 260, 250, 260, 250, 170}
7009 , { 260, 250, 260, 250, 170}
7010 , { 260, 250, 260, 250, 170}
7011 , { 260, 250, 260, 250, 170}
7012 }
7013 , {{ 350, 280, 350, 280, 200}
7014 , { 350, 280, 350, 280, 200}
7015 , { 260, 250, 260, 250, 170}
7016 , { 170, 120, 130, 120, 170}
7017 , { 260, 250, 260, 250, 170}
7018 }
7019 , {{ 260, 250, 260, 250, 170}
7020 , { 260, 250, 260, 250, 170}
7021 , { 260, 250, 260, 250, 170}
7022 , { 260, 250, 260, 250, 170}
7023 , { 260, 250, 260, 250, 170}
7024 }
7025 }
7026 }
7027 , {{{{ 280, 280, 260, 260, 260}
7028 , { 280, 280, 260, 250, 260}
7029 , { 260, 260, 240, 260, 240}
7030 , { 260, 260, 250, 240, 250}
7031 , { 260, 260, 240, 260, 240}
7032 }
7033 , {{ 280, 280, 260, 250, 260}
7034 , { 280, 280, 260, 250, 260}
7035 , { 250, 250, 240, 230, 240}
7036 , { 190, 150, 190, 130, 190}
7037 , { 250, 250, 240, 230, 240}
7038 }
7039 , {{ 260, 260, 250, 260, 250}
7040 , { 260, 260, 250, 240, 250}
7041 , { 260, 260, 240, 260, 240}
7042 , { 260, 260, 250, 240, 250}
7043 , { 260, 260, 240, 260, 240}
7044 }
7045 , {{ 260, 250, 260, 230, 260}
7046 , { 260, 220, 260, 200, 260}
7047 , { 250, 250, 240, 230, 240}
7048 , { 190, 110, 90, 190, 120}
7049 , { 250, 250, 240, 230, 240}
7050 }
7051 , {{ 260, 260, 250, 260, 250}
7052 , { 260, 260, 250, 240, 250}
7053 , { 260, 260, 240, 260, 240}
7054 , { 260, 260, 250, 240, 250}
7055 , { 230, 230, 150, 140, 150}
7056 }
7057 }
7058 , {{{ 280, 280, 260, 260, 260}
7059 , { 280, 280, 260, 220, 260}
7060 , { 260, 260, 240, 260, 240}
7061 , { 260, 260, 250, 210, 250}
7062 , { 260, 260, 240, 260, 240}
7063 }
7064 , {{ 280, 280, 260, 220, 260}
7065 , { 280, 280, 260, 220, 260}
7066 , { 250, 250, 240, 200, 240}
```

```
7067 , { 150, 150, 140, 100, 140}
7068 , { 250, 250, 240, 200, 240}
7069 }
7070 , {{ 260, 260, 250, 260, 250}
7071 , { 260, 260, 250, 210, 250}
7072 , { 260, 260, 240, 260, 240}
7073 , { 260, 260, 250, 210, 250}
7074 , { 260, 260, 240, 260, 240}
7075 }
7076 , {{ 250, 250, 240, 200, 240}
7077 , { 220, 220, 210, 170, 210}
7078 , { 250, 250, 240, 200, 240}
7079 , { 180, 100, 90, 180, 90}
7080 , { 250, 250, 240, 200, 240}
7081 }
7082 , {{ 260, 260, 250, 260, 250}
7083 , { 260, 260, 250, 210, 250}
7084 , { 260, 260, 240, 260, 240}
7085 , { 260, 260, 250, 210, 250}
7086 , { 230, 230, 150, 110, 150}
7087 }
7088 }
7089 , {{ { 260, 250, 260, 250, 260}
7090 , { 260, 250, 260, 250, 260}
7091 , { 240, 230, 240, 230, 240}
7092 , { 240, 240, 240, 240, 240}
7093 , { 240, 230, 240, 230, 240}
7094 }
7095 , {{ { 260, 250, 260, 250, 260}
7096 , { 260, 250, 260, 250, 260}
7097 , { 230, 230, 230, 230, 230}
7098 , { 190, 130, 190, 130, 190}
7099 , { 230, 230, 230, 230, 230}
7100 }
7101 , {{ { 240, 240, 240, 240, 240}
7102 , { 240, 240, 240, 240, 240}
7103 , { 240, 230, 240, 230, 240}
7104 , { 240, 240, 240, 240, 240}
7105 , { 240, 230, 240, 230, 240}
7106 }
7107 , {{ { 260, 230, 260, 230, 260}
7108 , { 260, 200, 260, 200, 260}
7109 , { 230, 230, 230, 230, 230}
7110 , { 80, 80, 80, 80, 80}
7111 , { 230, 230, 230, 230, 230}
7112 }
7113 , {{ { 240, 240, 240, 240, 240}
7114 , { 240, 240, 240, 240, 240}
7115 , { 240, 230, 240, 230, 240}
7116 , { 240, 240, 240, 240, 240}
7117 , { 150, 140, 150, 140, 150}
7118 }
7119 }
7120 , {{ { { 260, 190, 260, 190, 260}
7121 , { 260, 150, 260, 180, 260}
7122 , { 240, 190, 240, 80, 240}
7123 , { 250, 140, 250, 190, 250}
7124 , { 240, 190, 240, 120, 240}
7125 }
7126 , {{ { 260, 150, 260, 110, 260}
7127 , { 260, 150, 260, 100, 260}
7128 , { 240, 130, 240, 80, 240}
7129 , { 140, 30, 140, 110, 140}
7130 , { 240, 130, 240, 80, 240}
7131 }
7132 , {{ { 250, 190, 250, 90, 250}
7133 , { 250, 140, 250, 90, 250}
7134 , { 240, 190, 240, 80, 240}
7135 , { 250, 140, 250, 90, 250}
7136 , { 240, 190, 240, 80, 240}
7137 }
7138 , {{ { 240, 130, 240, 190, 240}
7139 , { 210, 100, 210, 180, 210}
7140 , { 240, 130, 240, 80, 240}
7141 , { 190, 110, 90, 190, 90}
7142 , { 240, 130, 240, 80, 240}
7143 }
7144 , {{ { 250, 190, 250, 120, 250}
7145 , { 250, 140, 250, 90, 250}
7146 , { 240, 190, 240, 80, 240}
7147 , { 250, 140, 250, 90, 250}
7148 , { 150, 40, 150, 120, 150}
7149 }
7150 }
7151 , {{ { { 260, 250, 260, 250, 230}
7152 , { 260, 250, 260, 250, 230}
7153 , { 240, 230, 240, 230, 150}
```

```
7154 , { 240, 240, 240, 240, 150}
7155 , { 240, 230, 240, 230, 150}
7156 }
7157 , {{ 260, 250, 260, 250, 230}
7158 , { 260, 250, 260, 250, 230}
7159 , { 230, 230, 230, 230, 140}
7160 , { 190, 130, 190, 130, 40}
7161 , { 230, 230, 230, 230, 140}
7162 }
7163 , {{ 240, 240, 240, 240, 150}
7164 , { 240, 240, 240, 240, 150}
7165 , { 240, 230, 240, 230, 150}
7166 , { 240, 240, 240, 240, 150}
7167 , { 240, 230, 240, 230, 150}
7168 }
7169 , {{ 260, 230, 260, 230, 140}
7170 , { 260, 200, 260, 200, 110}
7171 , { 230, 230, 230, 230, 140}
7172 , { 120, 80, 80, 80, 120}
7173 , { 230, 230, 230, 230, 140}
7174 }
7175 , {{ 240, 240, 240, 240, 150}
7176 , { 240, 240, 240, 240, 150}
7177 , { 240, 230, 240, 230, 150}
7178 , { 240, 240, 240, 240, 150}
7179 , { 150, 140, 150, 140, 60}
7180 }
7181 }
7182 }
7183 , {{{ 280, 280, 260, 280, 260}
7184 , { 280, 280, 260, 250, 260}
7185 , { 280, 280, 260, 280, 260}
7186 , { 280, 280, 260, 250, 260}
7187 , { 280, 280, 260, 280, 260}
7188 }
7189 , {{ 280, 280, 260, 250, 260}
7190 , { 280, 280, 260, 250, 260}
7191 , { 230, 230, 220, 210, 220}
7192 , { 210, 170, 210, 150, 210}
7193 , { 230, 230, 220, 210, 220}
7194 }
7195 , {{ 280, 280, 260, 280, 260}
7196 , { 280, 280, 260, 250, 260}
7197 , { 280, 280, 260, 280, 260}
7198 , { 280, 280, 260, 250, 260}
7199 , { 280, 280, 260, 280, 260}
7200 }
7201 , {{ 230, 230, 220, 210, 220}
7202 , { 220, 180, 220, 160, 220}
7203 , { 230, 230, 220, 210, 220}
7204 , { 210, 130, 110, 210, 140}
7205 , { 230, 230, 220, 210, 220}
7206 }
7207 , {{ 280, 280, 260, 250, 260}
7208 , { 280, 280, 260, 250, 260}
7209 , { 250, 250, 230, 250, 230}
7210 , { 280, 280, 260, 250, 260}
7211 , { 250, 250, 180, 170, 180}
7212 }
7213 }
7214 , {{{ 280, 280, 260, 280, 260}
7215 , { 280, 280, 260, 220, 260}
7216 , { 280, 280, 260, 280, 260}
7217 , { 280, 280, 260, 220, 260}
7218 , { 280, 280, 260, 280, 260}
7219 }
7220 , {{ 280, 280, 260, 220, 260}
7221 , { 280, 280, 260, 220, 260}
7222 , { 230, 230, 220, 180, 220}
7223 , { 170, 170, 160, 120, 160}
7224 , { 230, 230, 220, 180, 220}
7225 }
7226 , {{ 280, 280, 260, 280, 260}
7227 , { 280, 280, 260, 220, 260}
7228 , { 280, 280, 260, 280, 260}
7229 , { 280, 280, 260, 220, 260}
7230 , { 280, 280, 260, 280, 260}
7231 }
7232 , {{ 230, 230, 220, 200, 220}
7233 , { 180, 180, 170, 130, 170}
7234 , { 230, 230, 220, 180, 220}
7235 , { 200, 120, 110, 200, 110}
7236 , { 230, 230, 220, 180, 220}
7237 }
7238 , {{ 280, 280, 260, 250, 260}
7239 , { 280, 280, 260, 220, 260}
7240 , { 250, 250, 230, 250, 230}
```

```
7241 , { 280, 280, 260, 220, 260}
7242 , { 250, 250, 180, 140, 180}
7243 }
7244 }
7245 , {{ 260, 250, 260, 250, 260}
7246 , { 260, 250, 260, 250, 260}
7247 , { 260, 250, 260, 250, 260}
7248 , { 260, 250, 260, 250, 260}
7249 , { 260, 250, 260, 250, 260}
7250 }
7251 , {{ 260, 250, 260, 250, 260}
7252 , { 260, 250, 260, 250, 260}
7253 , { 210, 210, 210, 210, 210}
7254 , { 210, 150, 210, 150, 210}
7255 , { 210, 210, 210, 210, 210}
7256 }
7257 , {{ 260, 250, 260, 250, 260}
7258 , { 260, 250, 260, 250, 260}
7259 , { 260, 250, 260, 250, 260}
7260 , { 260, 250, 260, 250, 260}
7261 , { 260, 250, 260, 250, 260}
7262 }
7263 , {{ 220, 210, 220, 210, 220}
7264 , { 220, 160, 220, 160, 220}
7265 , { 210, 210, 210, 210, 210}
7266 , { 100, 100, 100, 100, 100}
7267 , { 210, 210, 210, 210, 210}
7268 }
7269 , {{ 260, 250, 260, 250, 260}
7270 , { 260, 250, 260, 250, 260}
7271 , { 230, 220, 230, 220, 230}
7272 , { 260, 250, 260, 250, 260}
7273 , { 170, 170, 170, 170, 170}
7274 }
7275 }
7276 , {{ 260, 210, 260, 210, 260}
7277 , { 260, 150, 260, 140, 260}
7278 , { 260, 210, 260, 100, 260}
7279 , { 260, 150, 260, 210, 260}
7280 , { 260, 210, 260, 150, 260}
7281 }
7282 , {{ 260, 150, 260, 130, 260}
7283 , { 260, 150, 260, 100, 260}
7284 , { 220, 110, 220, 60, 220}
7285 , { 160, 50, 160, 130, 160}
7286 , { 220, 110, 220, 60, 220}
7287 }
7288 , {{ 260, 210, 260, 100, 260}
7289 , { 260, 150, 260, 100, 260}
7290 , { 260, 210, 260, 100, 260}
7291 , { 260, 150, 260, 100, 260}
7292 , { 260, 210, 260, 100, 260}
7293 }
7294 , {{ 220, 130, 220, 210, 220}
7295 , { 170, 60, 170, 140, 170}
7296 , { 220, 110, 220, 60, 220}
7297 , { 210, 130, 110, 210, 110}
7298 , { 220, 110, 220, 60, 220}
7299 }
7300 , {{ 260, 180, 260, 150, 260}
7301 , { 260, 150, 260, 100, 260}
7302 , { 230, 180, 230, 70, 230}
7303 , { 260, 150, 260, 100, 260}
7304 , { 180, 70, 180, 150, 180}
7305 }
7306 }
7307 , {{ 260, 250, 260, 250, 230}
7308 , { 260, 250, 260, 250, 230}
7309 , { 260, 250, 260, 250, 170}
7310 , { 260, 250, 260, 250, 170}
7311 , { 260, 250, 260, 250, 170}
7312 }
7313 , {{ 260, 250, 260, 250, 230}
7314 , { 260, 250, 260, 250, 230}
7315 , { 210, 210, 210, 210, 120}
7316 , { 210, 150, 210, 150, 60}
7317 , { 210, 210, 210, 210, 120}
7318 }
7319 , {{ 260, 250, 260, 250, 170}
7320 , { 260, 250, 260, 250, 170}
7321 , { 260, 250, 260, 250, 170}
7322 , { 260, 250, 260, 250, 170}
7323 , { 260, 250, 260, 250, 170}
7324 }
7325 , {{ 220, 210, 220, 210, 140}
7326 , { 220, 160, 220, 160, 70}
7327 , { 210, 210, 210, 210, 120}
```

```
7328 , { 140, 100, 100, 100, 140}
7329 , { 210, 210, 210, 210, 120}
7330 }
7331 , {{ 260, 250, 260, 250, 170}
7332 , { 260, 250, 260, 250, 170}
7333 , { 230, 220, 230, 220, 140}
7334 , { 260, 250, 260, 250, 170}
7335 , { 170, 170, 170, 170, 80}
7336 }
7337 }
7338 }
7339 , {{{ 370, 370, 350, 320, 350}
7340 , { 350, 340, 350, 320, 350}
7341 , { 310, 310, 290, 310, 290}
7342 , { 310, 310, 290, 280, 290}
7343 , { 370, 370, 290, 310, 290}
7344 }
7345 , {{ 340, 340, 330, 320, 330}
7346 , { 340, 340, 330, 320, 330}
7347 , { 310, 310, 290, 280, 290}
7348 , { 270, 230, 270, 200, 270}
7349 , { 310, 310, 290, 280, 290}
7350 }
7351 , {{ 310, 310, 290, 310, 290}
7352 , { 310, 310, 290, 280, 290}
7353 , { 310, 310, 290, 310, 290}
7354 , { 310, 310, 290, 280, 290}
7355 , { 310, 310, 290, 310, 290}
7356 }
7357 , {{ 350, 310, 350, 280, 350}
7358 , { 350, 310, 350, 280, 350}
7359 , { 310, 310, 290, 280, 290}
7360 , { 260, 180, 160, 260, 200}
7361 , { 310, 310, 290, 280, 290}
7362 }
7363 , {{ 370, 370, 290, 310, 290}
7364 , { 310, 310, 290, 280, 290}
7365 , { 310, 310, 290, 310, 290}
7366 , { 310, 310, 290, 280, 290}
7367 , { 370, 370, 290, 280, 290}
7368 }
7369 }
7370 , {{{ 370, 370, 330, 310, 330}
7371 , { 340, 340, 330, 290, 330}
7372 , { 310, 310, 290, 310, 290}
7373 , { 310, 310, 290, 250, 290}
7374 , { 370, 370, 290, 310, 290}
7375 }
7376 , {{ 340, 340, 330, 290, 330}
7377 , { 340, 340, 330, 290, 330}
7378 , { 310, 310, 290, 250, 290}
7379 , { 230, 230, 210, 170, 210}
7380 , { 310, 310, 290, 250, 290}
7381 }
7382 , {{ 310, 310, 290, 310, 290}
7383 , { 310, 310, 290, 250, 290}
7384 , { 310, 310, 290, 310, 290}
7385 , { 310, 310, 290, 250, 290}
7386 , { 310, 310, 290, 310, 290}
7387 }
7388 , {{ 310, 310, 290, 250, 290}
7389 , { 310, 310, 290, 250, 290}
7390 , { 310, 310, 290, 250, 290}
7391 , { 250, 180, 160, 250, 160}
7392 , { 310, 310, 290, 250, 290}
7393 }
7394 , {{ 370, 370, 290, 310, 290}
7395 , { 310, 310, 290, 250, 290}
7396 , { 310, 310, 290, 310, 290}
7397 , { 310, 310, 290, 250, 290}
7398 , { 370, 370, 290, 250, 290}
7399 }
7400 }
7401 , {{{ 350, 320, 350, 320, 350}
7402 , { 350, 320, 350, 320, 350}
7403 , { 290, 280, 290, 280, 290}
7404 , { 290, 280, 290, 280, 290}
7405 , { 290, 280, 290, 280, 290}
7406 }
7407 , {{ 320, 320, 320, 320, 320}
7408 , { 320, 320, 320, 320, 320}
7409 , { 290, 280, 290, 280, 290}
7410 , { 270, 200, 270, 200, 270}
7411 , { 290, 280, 290, 280, 290}
7412 }
7413 , {{ 290, 280, 290, 280, 290}
7414 , { 290, 280, 290, 280, 290}
```

```
7415 , { 290, 280, 290, 280, 290}
7416 , { 290, 280, 290, 280, 290}
7417 , { 290, 280, 290, 280, 290}
7418 }
7419 , {{ 350, 280, 350, 280, 350}
7420 , { 350, 280, 350, 280, 350}
7421 , { 290, 280, 290, 280, 290}
7422 , { 160, 150, 160, 150, 160}
7423 , { 290, 280, 290, 280, 290}
7424 }
7425 , {{ 290, 280, 290, 280, 290}
7426 , { 290, 280, 290, 280, 290}
7427 , { 290, 280, 290, 280, 290}
7428 , { 290, 280, 290, 280, 290}
7429 , { 290, 280, 290, 280, 290}
7430 }
7431 }
7432 , {{{ 330, 240, 330, 260, 330}
7433 , { 330, 220, 330, 260, 330}
7434 , { 290, 240, 290, 130, 290}
7435 , { 290, 180, 290, 260, 290}
7436 , { 290, 240, 290, 260, 290}
7437 }
7438 , {{ 330, 220, 330, 180, 330}
7439 , { 330, 220, 330, 170, 330}
7440 , { 290, 180, 290, 130, 290}
7441 , { 210, 100, 210, 180, 210}
7442 , { 290, 180, 290, 130, 290}
7443 }
7444 , {{ 290, 240, 290, 130, 290}
7445 , { 290, 180, 290, 130, 290}
7446 , { 290, 240, 290, 130, 290}
7447 , { 290, 180, 290, 130, 290}
7448 , { 290, 240, 290, 130, 290}
7449 }
7450 , {{ 290, 180, 290, 260, 290}
7451 , { 290, 180, 290, 260, 290}
7452 , { 290, 180, 290, 130, 290}
7453 , { 260, 180, 160, 260, 160}
7454 , { 290, 180, 290, 130, 290}
7455 }
7456 , {{ 290, 240, 290, 260, 290}
7457 , { 290, 180, 290, 130, 290}
7458 , { 290, 240, 290, 130, 290}
7459 , { 290, 180, 290, 130, 290}
7460 , { 290, 180, 290, 260, 290}
7461 }
7462 }
7463 , {{{ 350, 320, 350, 320, 290}
7464 , { 350, 320, 350, 320, 290}
7465 , { 290, 280, 290, 280, 200}
7466 , { 290, 280, 290, 280, 200}
7467 , { 290, 280, 290, 280, 200}
7468 }
7469 , {{ 320, 320, 320, 320, 290}
7470 , { 320, 320, 320, 320, 290}
7471 , { 290, 280, 290, 280, 200}
7472 , { 270, 200, 270, 200, 120}
7473 , { 290, 280, 290, 280, 200}
7474 }
7475 , {{ 290, 280, 290, 280, 200}
7476 , { 290, 280, 290, 280, 200}
7477 , { 290, 280, 290, 280, 200}
7478 , { 290, 280, 290, 280, 200}
7479 , { 290, 280, 290, 280, 200}
7480 }
7481 , {{ 350, 280, 350, 280, 200}
7482 , { 350, 280, 350, 280, 200}
7483 , { 290, 280, 290, 280, 200}
7484 , { 200, 150, 160, 150, 200}
7485 , { 290, 280, 290, 280, 200}
7486 }
7487 , {{ 290, 280, 290, 280, 200}
7488 , { 290, 280, 290, 280, 200}
7489 , { 290, 280, 290, 280, 200}
7490 , { 290, 280, 290, 280, 200}
7491 , { 290, 280, 290, 280, 200}
7492 }
7493 }
7494 }
7495 }
7496 , {{{{ INF, INF, INF, INF, INF
7497 , { INF, INF, INF, INF, INF
7498 , { INF, INF, INF, INF, INF
7499 , { INF, INF, INF, INF, INF
7500 , { INF, INF, INF, INF, INF
7501 }}}
```

```
7502 , {{ INF, INF, INF, INF, INF }
7503 , { INF, INF, INF, INF, INF }
7504 , { INF, INF, INF, INF, INF }
7505 , { INF, INF, INF, INF, INF }
7506 , { INF, INF, INF, INF, INF }
7507 }
7508 , {{ INF, INF, INF, INF, INF }
7509 , { INF, INF, INF, INF, INF }
7510 , { INF, INF, INF, INF, INF }
7511 , { INF, INF, INF, INF, INF }
7512 , { INF, INF, INF, INF, INF }
7513 }
7514 , {{ INF, INF, INF, INF, INF }
7515 , { INF, INF, INF, INF, INF }
7516 , { INF, INF, INF, INF, INF }
7517 , { INF, INF, INF, INF, INF }
7518 , { INF, INF, INF, INF, INF }
7519 }
7520 , {{ INF, INF, INF, INF, INF }
7521 , { INF, INF, INF, INF, INF }
7522 , { INF, INF, INF, INF, INF }
7523 , { INF, INF, INF, INF, INF }
7524 , { INF, INF, INF, INF, INF }
7525 }
7526 }
7527 , {{ { INF, INF, INF, INF, INF }
7528 , { INF, INF, INF, INF, INF }
7529 , { INF, INF, INF, INF, INF }
7530 , { INF, INF, INF, INF, INF }
7531 , { INF, INF, INF, INF, INF }
7532 }
7533 , {{ INF, INF, INF, INF, INF }
7534 , { INF, INF, INF, INF, INF }
7535 , { INF, INF, INF, INF, INF }
7536 , { INF, INF, INF, INF, INF }
7537 , { INF, INF, INF, INF, INF }
7538 }
7539 , {{ INF, INF, INF, INF, INF }
7540 , { INF, INF, INF, INF, INF }
7541 , { INF, INF, INF, INF, INF }
7542 , { INF, INF, INF, INF, INF }
7543 , { INF, INF, INF, INF, INF }
7544 }
7545 , {{ INF, INF, INF, INF, INF }
7546 , { INF, INF, INF, INF, INF }
7547 , { INF, INF, INF, INF, INF }
7548 , { INF, INF, INF, INF, INF }
7549 , { INF, INF, INF, INF, INF }
7550 }
7551 , {{ INF, INF, INF, INF, INF }
7552 , { INF, INF, INF, INF, INF }
7553 , { INF, INF, INF, INF, INF }
7554 , { INF, INF, INF, INF, INF }
7555 , { INF, INF, INF, INF, INF }
7556 }
7557 }
7558 , {{ { INF, INF, INF, INF, INF }
7559 , { INF, INF, INF, INF, INF }
7560 , { INF, INF, INF, INF, INF }
7561 , { INF, INF, INF, INF, INF }
7562 , { INF, INF, INF, INF, INF }
7563 }
7564 , {{ INF, INF, INF, INF, INF }
7565 , { INF, INF, INF, INF, INF }
7566 , { INF, INF, INF, INF, INF }
7567 , { INF, INF, INF, INF, INF }
7568 , { INF, INF, INF, INF, INF }
7569 }
7570 , {{ INF, INF, INF, INF, INF }
7571 , { INF, INF, INF, INF, INF }
7572 , { INF, INF, INF, INF, INF }
7573 , { INF, INF, INF, INF, INF }
7574 , { INF, INF, INF, INF, INF }
7575 }
7576 , {{ INF, INF, INF, INF, INF }
7577 , { INF, INF, INF, INF, INF }
7578 , { INF, INF, INF, INF, INF }
7579 , { INF, INF, INF, INF, INF }
7580 , { INF, INF, INF, INF, INF }
7581 }
7582 , {{ INF, INF, INF, INF, INF }
7583 , { INF, INF, INF, INF, INF }
7584 , { INF, INF, INF, INF, INF }
7585 , { INF, INF, INF, INF, INF }
7586 , { INF, INF, INF, INF, INF }
7587 }
7588 }
```

```
7589 ,{{{{ INF, INF, INF, INF, INF
7590 ,{ INF, INF, INF, INF, INF
7591 ,{ INF, INF, INF, INF, INF
7592 ,{ INF, INF, INF, INF, INF
7593 ,{ INF, INF, INF, INF, INF
7594 }
7595 ,{{ INF, INF, INF, INF, INF
7596 ,{ INF, INF, INF, INF, INF
7597 ,{ INF, INF, INF, INF, INF
7598 ,{ INF, INF, INF, INF, INF
7599 ,{ INF, INF, INF, INF, INF
7600 }
7601 ,{{ INF, INF, INF, INF, INF
7602 ,{ INF, INF, INF, INF, INF
7603 ,{ INF, INF, INF, INF, INF
7604 ,{ INF, INF, INF, INF, INF
7605 ,{ INF, INF, INF, INF, INF
7606 }
7607 ,{{ INF, INF, INF, INF, INF
7608 ,{ INF, INF, INF, INF, INF
7609 ,{ INF, INF, INF, INF, INF
7610 ,{ INF, INF, INF, INF, INF
7611 ,{ INF, INF, INF, INF, INF
7612 }
7613 ,{{ INF, INF, INF, INF, INF
7614 ,{ INF, INF, INF, INF, INF
7615 ,{ INF, INF, INF, INF, INF
7616 ,{ INF, INF, INF, INF, INF
7617 ,{ INF, INF, INF, INF, INF
7618 }
7619 }
7620 ,{{{{ INF, INF, INF, INF, INF
7621 ,{ INF, INF, INF, INF, INF
7622 ,{ INF, INF, INF, INF, INF
7623 ,{ INF, INF, INF, INF, INF
7624 ,{ INF, INF, INF, INF, INF
7625 }
7626 ,{{ INF, INF, INF, INF, INF
7627 ,{ INF, INF, INF, INF, INF
7628 ,{ INF, INF, INF, INF, INF
7629 ,{ INF, INF, INF, INF, INF
7630 ,{ INF, INF, INF, INF, INF
7631 }
7632 ,{{ INF, INF, INF, INF, INF
7633 ,{ INF, INF, INF, INF, INF
7634 ,{ INF, INF, INF, INF, INF
7635 ,{ INF, INF, INF, INF, INF
7636 ,{ INF, INF, INF, INF, INF
7637 }
7638 ,{{ INF, INF, INF, INF, INF
7639 ,{ INF, INF, INF, INF, INF
7640 ,{ INF, INF, INF, INF, INF
7641 ,{ INF, INF, INF, INF, INF
7642 ,{ INF, INF, INF, INF, INF
7643 }
7644 ,{{ INF, INF, INF, INF, INF
7645 ,{ INF, INF, INF, INF, INF
7646 ,{ INF, INF, INF, INF, INF
7647 ,{ INF, INF, INF, INF, INF
7648 ,{ INF, INF, INF, INF, INF
7649 }
7650 }
7651 }
7652 ,{{{ 240, 240, 240, 190, 240}
7653 ,{ 240, 240, 240, 190, 240}
7654 ,{ 220, 220, 220, 190, 220}
7655 ,{ 240, 240, 240, 190, 240}
7656 ,{ 210, 210, 210, 170, 210}
7657 }
7658 ,{{ 200, 200, 200, 150, 200}
7659 ,{ 200, 200, 200, 150, 200}
7660 ,{ 190, 190, 190, 150, 190}
7661 ,{ 160, 100, 160, 80, 130}
7662 ,{ 190, 190, 190, 150, 190}
7663 }
7664 ,{{ 240, 240, 240, 190, 240}
7665 ,{ 240, 240, 240, 190, 240}
7666 ,{ 220, 220, 220, 190, 220}
7667 ,{ 240, 240, 240, 190, 240}
7668 ,{ 210, 210, 210, 170, 210}
7669 }
7670 ,{{ 190, 190, 190, 150, 190}
7671 ,{ 160, 100, 160, 80, 130}
7672 ,{ 190, 190, 190, 150, 190}
7673 ,{ 150, 70, 50, 150, 90}
7674 ,{ 190, 190, 190, 150, 190}
7675 }
```

```
7676 , {{ 240, 240, 240, 190, 240}
7677 , { 240, 240, 240, 190, 240}
7678 , { 210, 210, 210, 170, 210}
7679 , { 240, 240, 240, 190, 240}
7680 , { 180, 180, 120, 90, 120}
7681 }
7682 }
7683 , {{ 240, 240, 240, 190, 240}
7684 , { 240, 240, 240, 140, 240}
7685 , { 220, 220, 220, 190, 220}
7686 , { 240, 240, 240, 140, 240}
7687 , { 210, 210, 210, 170, 210}
7688 }
7689 , {{ 200, 200, 200, 100, 200}
7690 , { 200, 200, 200, 100, 200}
7691 , { 190, 190, 190, 100, 190}
7692 , { 100, 100, 100, 10, 100}
7693 , { 190, 190, 190, 100, 190}
7694 }
7695 , {{ 240, 240, 240, 190, 240}
7696 , { 240, 240, 240, 140, 240}
7697 , { 220, 220, 220, 190, 220}
7698 , { 240, 240, 240, 140, 240}
7699 , { 210, 210, 210, 170, 210}
7700 }
7701 , {{ 190, 190, 190, 100, 190}
7702 , { 100, 100, 100, 10, 100}
7703 , { 190, 190, 190, 100, 190}
7704 , { 80, 50, 50, 80, 50}
7705 , { 190, 190, 190, 100, 190}
7706 }
7707 , {{ 240, 240, 240, 170, 240}
7708 , { 240, 240, 240, 140, 240}
7709 , { 210, 210, 210, 170, 210}
7710 , { 240, 240, 240, 140, 240}
7711 , { 180, 180, 120, 20, 120}
7712 }
7713 }
7714 , {{ 240, 190, 240, 190, 210}
7715 , { 240, 190, 240, 190, 210}
7716 , { 220, 180, 220, 180, 190}
7717 , { 240, 190, 240, 190, 210}
7718 , { 210, 160, 210, 160, 180}
7719 }
7720 , {{ 200, 150, 200, 150, 170}
7721 , { 200, 150, 200, 150, 170}
7722 , { 190, 150, 190, 150, 160}
7723 , { 160, 60, 160, 60, 130}
7724 , { 190, 150, 190, 150, 160}
7725 }
7726 , {{ 240, 190, 240, 190, 210}
7727 , { 240, 190, 240, 190, 210}
7728 , { 220, 180, 220, 180, 190}
7729 , { 240, 190, 240, 190, 210}
7730 , { 210, 160, 210, 160, 180}
7731 }
7732 , {{ 190, 150, 190, 150, 160}
7733 , { 160, 60, 160, 60, 130}
7734 , { 190, 150, 190, 150, 160}
7735 , { 50, 0, 50, 0, 20}
7736 , { 190, 150, 190, 150, 160}
7737 }
7738 , {{ 240, 190, 240, 190, 210}
7739 , { 240, 190, 240, 190, 210}
7740 , { 210, 160, 210, 160, 180}
7741 , { 240, 190, 240, 190, 210}
7742 , { 120, 70, 120, 70, 90}
7743 }
7744 }
7745 , {{ 240, 180, 240, 150, 240}
7746 , { 240, 130, 240, 80, 240}
7747 , { 220, 180, 220, 70, 220}
7748 , { 240, 130, 240, 150, 240}
7749 , { 210, 160, 210, 90, 210}
7750 }
7751 , {{ 200, 90, 200, 80, 200}
7752 , { 200, 90, 200, 40, 200}
7753 , { 190, 90, 190, 40, 190}
7754 , { 100, 0, 100, 80, 100}
7755 , { 190, 90, 190, 40, 190}
7756 }
7757 , {{ 240, 180, 240, 80, 240}
7758 , { 240, 130, 240, 80, 240}
7759 , { 220, 180, 220, 70, 220}
7760 , { 240, 130, 240, 80, 240}
7761 , { 210, 160, 210, 50, 210}
7762 }
```

```
7763 , {{ 190,    90,   190,   150,   190} }  
7764 , { 100,     0,   100,    80,   100} }  
7765 , { 190,    90,   190,    40,   190} }  
7766 , { 150,    70,     50,   150,    50} }  
7767 , { 190,    90,   190,    40,   190} }  
7768 } }  
7769 , {{ 240,   160,   240,    90,   240} }  
7770 , { 240,   130,   240,    80,   240} }  
7771 , { 210,   160,   210,    50,   210} }  
7772 , { 240,   130,   240,    80,   240} }  
7773 , { 120,    10,   120,    90,   120} }  
7774 } }  
7775 } }  
7776 , {{{ 240,   190,   240,   190,   170} } }  
7777 , { 240,   190,   240,   190,   170} } }  
7778 , { 220,   180,   220,   180,   140} } }  
7779 , { 240,   190,   240,   190,   150} } }  
7780 , { 210,   160,   210,   160,   120} } }  
7781 } }  
7782 , {{ 200,   150,   200,   150,   170} } }  
7783 , { 200,   150,   200,   150,   170} } }  
7784 , { 190,   150,   190,   150,   110} } }  
7785 , { 160,    60,   160,    60,    20} } }  
7786 , { 190,   150,   190,   150,   110} } }  
7787 } }  
7788 , {{ 240,   190,   240,   190,   150} } }  
7789 , { 240,   190,   240,   190,   150} } }  
7790 , { 220,   180,   220,   180,   140} } }  
7791 , { 240,   190,   240,   190,   150} } }  
7792 , { 210,   160,   210,   160,   120} } }  
7793 } }  
7794 , {{ 190,   150,   190,   150,   110} } }  
7795 , { 160,    60,   160,    60,    20} } }  
7796 , { 190,   150,   190,   150,   110} } }  
7797 , { 90,     0,     50,     0,    90} } }  
7798 , { 190,   150,   190,   150,   110} } }  
7799 } }  
7800 , {{ 240,   190,   240,   190,   150} } }  
7801 , { 240,   190,   240,   190,   150} } }  
7802 , { 210,   160,   210,   160,   120} } }  
7803 , { 240,   190,   240,   190,   150} } }  
7804 , { 120,    70,   120,    70,    30} } }  
7805 } }  
7806 } }  
7807 } }  
7808 , {{{{ 210,   210,   210,   170,   210} } } } }  
7809 , { 210,   210,   210,   170,   210} } } } }  
7810 , { 190,   190,   190,   160,   190} } } } }  
7811 , { 180,   180,   180,   150,   180} } } } }  
7812 , { 190,   190,   190,   150,   190} } } } }  
7813 } } } }  
7814 , {{ 210,   210,   210,   170,   210} } } } }  
7815 , { 210,   210,   210,   170,   210} } } } }  
7816 , { 190,   190,   190,   140,   190} } } } }  
7817 , { 70,     10,    70,   -10,    40} } } } }  
7818 , { 190,   190,   190,   140,   190} } } } }  
7819 } } } }  
7820 , {{ 190,   190,   190,   150,   190} } } } }  
7821 , { 180,   180,   180,   140,   180} } } } }  
7822 , { 190,   190,   190,   150,   190} } } } }  
7823 , { 180,   180,   180,   140,   180} } } } }  
7824 , { 190,   190,   190,   150,   190} } } } }  
7825 } } } }  
7826 , {{ 190,   190,   190,   150,   190} } } } }  
7827 , { 130,    70,   130,    50,   100} } } } }  
7828 , { 190,   190,   190,   140,   190} } } } }  
7829 , { 150,    70,     50,   150,    90} } } } }  
7830 , { 190,   190,   190,   140,   190} } } } }  
7831 } } } }  
7832 , {{ 190,   190,   190,   160,   190} } } } }  
7833 , { 180,   180,   180,   140,   180} } } } }  
7834 , { 190,   190,   190,   160,   190} } } } }  
7835 , { 180,   180,   180,   140,   180} } } } }  
7836 , { 170,   170,   110,    90,   110} } } } }  
7837 } } } }  
7838 } } } }  
7839 , {{{{ 210,   210,   210,   160,   210} } } } } }  
7840 , { 210,   210,   210,   120,   210} } } } } }  
7841 , { 190,   190,   190,   160,   190} } } } } }  
7842 , { 180,   180,   180,    90,   180} } } } } }  
7843 , { 190,   190,   190,   150,   190} } } } } }  
7844 } } } }  
7845 , {{ 210,   210,   210,   120,   210} } } } } }  
7846 , { 210,   210,   210,   120,   210} } } } } }  
7847 , { 190,   190,   190,    90,   190} } } } } }  
7848 , { 10,     10,     10,   -80,    10} } } } } }  
7849 , { 190,   190,   190,    90,   190} } } } } }
```

```
7850    }
7851 ,{{ 190, 190, 190, 150, 190}
7852 ,{ 180, 180, 180, 90, 180}
7853 ,{ 190, 190, 190, 150, 190}
7854 ,{ 180, 180, 180, 90, 180}
7855 ,{ 190, 190, 190, 150, 190}
7856 }
7857 ,{{ 190, 190, 190, 90, 190}
7858 ,{ 70, 70, 70, -20, 70}
7859 ,{ 190, 190, 190, 90, 190}
7860 ,{ 80, 50, 50, 80, 50}
7861 ,{ 190, 190, 190, 90, 190}
7862 }
7863 ,{{ 190, 190, 190, 160, 190}
7864 ,{ 180, 180, 180, 90, 180}
7865 ,{ 190, 190, 190, 160, 190}
7866 ,{ 180, 180, 180, 90, 180}
7867 ,{ 170, 170, 110, 20, 110}
7868 }
7869 }
7870 ,{{{{ 210, 170, 210, 170, 180}
7871 ,{ 210, 170, 210, 170, 180}
7872 ,{ 190, 150, 190, 150, 160}
7873 ,{ 180, 140, 180, 140, 150}
7874 ,{ 190, 140, 190, 140, 160}
7875 }
7876 ,{{ 210, 170, 210, 170, 180}
7877 ,{ 210, 170, 210, 170, 180}
7878 ,{ 190, 140, 190, 140, 160}
7879 ,{ 70, -30, 70, -30, 40}
7880 ,{ 190, 140, 190, 140, 160}
7881 }
7882 ,{{ 190, 140, 190, 140, 160}
7883 ,{ 180, 140, 180, 140, 150}
7884 ,{ 190, 140, 190, 140, 160}
7885 ,{ 180, 140, 180, 140, 150}
7886 ,{ 190, 140, 190, 140, 160}
7887 }
7888 ,{{ 190, 140, 190, 140, 160}
7889 ,{ 130, 30, 130, 30, 100}
7890 ,{ 190, 140, 190, 140, 160}
7891 ,{ 50, 0, 50, 0, 20}
7892 ,{ 190, 140, 190, 140, 160}
7893 }
7894 ,{{ 190, 150, 190, 150, 160}
7895 ,{ 180, 140, 180, 140, 150}
7896 ,{ 190, 150, 190, 150, 160}
7897 ,{ 180, 140, 180, 140, 150}
7898 ,{ 110, 70, 110, 70, 80}
7899 }
7900 }
7901 ,{{{{ 210, 150, 210, 150, 210}
7902 ,{ 210, 110, 210, 60, 210}
7903 ,{ 190, 150, 190, 40, 190}
7904 ,{ 180, 80, 180, 150, 180}
7905 ,{ 190, 140, 190, 90, 190}
7906 }
7907 ,{{ 210, 110, 210, 60, 210}
7908 ,{ 210, 110, 210, 60, 210}
7909 ,{ 190, 80, 190, 30, 190}
7910 ,{ 10, -90, 10, -10, 10}
7911 ,{ 190, 80, 190, 30, 190}
7912 }
7913 ,{{ 190, 140, 190, 30, 190}
7914 ,{ 180, 80, 180, 30, 180}
7915 ,{ 190, 140, 190, 30, 190}
7916 ,{ 180, 80, 180, 30, 180}
7917 ,{ 190, 140, 190, 30, 190}
7918 }
7919 ,{{ 190, 80, 190, 150, 190}
7920 ,{ 70, -30, 70, 50, 70}
7921 ,{ 190, 80, 190, 30, 190}
7922 ,{ 150, 70, 50, 150, 50}
7923 ,{ 190, 80, 190, 30, 190}
7924 }
7925 ,{{ 190, 150, 190, 90, 190}
7926 ,{ 180, 80, 180, 30, 180}
7927 ,{ 190, 150, 190, 40, 190}
7928 ,{ 180, 80, 180, 30, 180}
7929 ,{ 110, 10, 110, 90, 110}
7930 }
7931 }
7932 ,{{{{ 210, 170, 210, 170, 190}
7933 ,{ 210, 170, 210, 170, 190}
7934 ,{ 190, 150, 190, 150, 110}
7935 ,{ 180, 140, 180, 140, 100}
7936 ,{ 190, 140, 190, 140, 100}
```

```
7937      }
7938 ,{{ 210,    170,    210,    170,    190}
7939 ,{ 210,    170,    210,    170,    190}
7940 ,{ 190,    140,    190,    140,   100}
7941 ,{ 70,    -30,     70,    -30,   -70}
7942 ,{ 190,    140,    190,    140,   100}
7943 }
7944 ,{{ 190,    140,    190,    140,   100}
7945 ,{ 180,    140,    180,    140,   100}
7946 ,{ 190,    140,    190,    140,   100}
7947 ,{ 180,    140,    180,    140,   100}
7948 ,{ 190,    140,    190,    140,   100}
7949 }
7950 ,{{ 190,    140,    190,    140,   100}
7951 ,{ 130,     30,    130,     30,   -10}
7952 ,{ 190,    140,    190,    140,   100}
7953 ,{ 90,      0,     50,      0,    90}
7954 ,{ 190,    140,    190,    140,   100}
7955 }
7956 ,{{ 190,    150,    190,    150,   110}
7957 ,{ 180,    140,    180,    140,   100}
7958 ,{ 190,    150,    190,    150,   110}
7959 ,{ 180,    140,    180,    140,   100}
7960 ,{ 110,     70,    110,     70,    30}
7961 }
7962 }
7963 }
7964 ,{{{ 370,    370,    340,    300,    340}
7965 ,{ 340,    340,    340,    300,    340}
7966 ,{ 310,    310,    310,    270,    310}
7967 ,{ 310,    310,    310,    280,    310}
7968 ,{ 370,    370,    310,    280,    310}
7969 }
7970 ,{{ 340,    340,    340,    300,    340}
7971 ,{ 340,    340,    340,    300,    340}
7972 ,{ 310,    310,    310,    260,    310}
7973 ,{ 290,    230,    290,    200,    260}
7974 ,{ 310,    310,    310,    260,    310}
7975 }
7976 ,{{ 310,    310,    310,    270,    310}
7977 ,{ 310,    310,    310,    260,    310}
7978 ,{ 310,    310,    310,    270,    310}
7979 ,{ 310,    310,    310,    260,    310}
7980 ,{ 310,    310,    310,    270,    310}
7981 }
7982 ,{{ 330,    310,    330,    280,    310}
7983 ,{ 330,    270,    330,    240,    300}
7984 ,{ 310,    310,    310,    260,    310}
7985 ,{ 280,    200,    180,    280,    220}
7986 ,{ 310,    310,    310,    260,    310}
7987 }
7988 ,{{ 370,    370,    310,    280,    310}
7989 ,{ 310,    310,    310,    260,    310}
7990 ,{ 310,    310,    310,    270,    310}
7991 ,{ 310,    310,    310,    260,    310}
7992 ,{ 370,    370,    310,    280,    310}
7993 }
7994 }
7995 ,{{{ 370,    370,    340,    270,    340}
7996 ,{ 340,    340,    340,    250,    340}
7997 ,{ 310,    310,    310,    270,    310}
7998 ,{ 310,    310,    310,    210,    310}
7999 ,{ 370,    370,    310,    270,    310}
8000 }
8001 ,{{ 340,    340,    340,    250,    340}
8002 ,{ 340,    340,    340,    250,    340}
8003 ,{ 310,    310,    310,    210,    310}
8004 ,{ 230,    230,    230,    130,    230}
8005 ,{ 310,    310,    310,    210,    310}
8006 }
8007 ,{{ 310,    310,    310,    270,    310}
8008 ,{ 310,    310,    310,    210,    310}
8009 ,{ 310,    310,    310,    270,    310}
8010 ,{ 310,    310,    310,    210,    310}
8011 ,{ 310,    310,    310,    270,    310}
8012 }
8013 ,{{ 310,    310,    310,    210,    310}
8014 ,{ 270,    270,    270,    170,    270}
8015 ,{ 310,    310,    310,    210,    310}
8016 ,{ 210,    180,    180,    210,    180}
8017 ,{ 310,    310,    310,    210,    310}
8018 }
8019 ,{{ 370,    370,    310,    270,    310}
8020 ,{ 310,    310,    310,    210,    310}
8021 ,{ 310,    310,    310,    270,    310}
8022 ,{ 310,    310,    310,    210,    310}
8023 ,{ 370,    370,    310,    210,    310}
```

```
8024      }
8025    }
8026 ,{{ 340, 300, 340, 300, 310}
8027 ,{ 340, 300, 340, 300, 310}
8028 ,{ 310, 260, 310, 260, 280}
8029 ,{ 310, 260, 310, 260, 280}
8030 ,{ 310, 260, 310, 260, 280}
8031   }
8032 ,{{ 340, 300, 340, 300, 310}
8033 ,{ 340, 300, 340, 300, 310}
8034 ,{ 310, 260, 310, 260, 280}
8035 ,{ 290, 180, 290, 180, 260}
8036 ,{ 310, 260, 310, 260, 280}
8037   }
8038 ,{{ 310, 260, 310, 260, 280}
8039 ,{ 310, 260, 310, 260, 280}
8040 ,{ 310, 260, 310, 260, 280}
8041 ,{ 310, 260, 310, 260, 280}
8042 ,{ 310, 260, 310, 260, 280}
8043   }
8044 ,{{ 330, 260, 330, 260, 300}
8045 ,{ 330, 220, 330, 220, 300}
8046 ,{ 310, 260, 310, 260, 280}
8047 ,{ 180, 130, 180, 130, 150}
8048 ,{ 310, 260, 310, 260, 280}
8049   }
8050 ,{{ 310, 260, 310, 260, 280}
8051 ,{ 310, 260, 310, 260, 280}
8052 ,{ 310, 260, 310, 260, 280}
8053 ,{ 310, 260, 310, 260, 280}
8054 ,{ 310, 260, 310, 260, 280}
8055   }
8056   }
8057 ,{{ 340, 260, 340, 280, 340}
8058 ,{ 340, 240, 340, 240, 340}
8059 ,{ 310, 260, 310, 150, 310}
8060 ,{ 310, 200, 310, 280, 310}
8061 ,{ 310, 260, 310, 280, 310}
8062   }
8063 ,{{ 340, 240, 340, 200, 340}
8064 ,{ 340, 240, 340, 190, 340}
8065 ,{ 310, 200, 310, 150, 310}
8066 ,{ 230, 120, 230, 200, 230}
8067 ,{ 310, 200, 310, 150, 310}
8068   }
8069 ,{{ 310, 260, 310, 150, 310}
8070 ,{ 310, 200, 310, 150, 310}
8071 ,{ 310, 260, 310, 150, 310}
8072 ,{ 310, 200, 310, 150, 310}
8073 ,{ 310, 260, 310, 150, 310}
8074   }
8075 ,{{ 310, 200, 310, 280, 310}
8076 ,{ 270, 160, 270, 240, 270}
8077 ,{ 310, 200, 310, 150, 310}
8078 ,{ 280, 200, 180, 280, 180}
8079 ,{ 310, 200, 310, 150, 310}
8080   }
8081 ,{{ 310, 260, 310, 280, 310}
8082 ,{ 310, 200, 310, 150, 310}
8083 ,{ 310, 260, 310, 150, 310}
8084 ,{ 310, 200, 310, 150, 310}
8085 ,{ 310, 200, 310, 280, 310}
8086   }
8087   }
8088 ,{{ 340, 300, 340, 300, 320}
8089 ,{ 340, 300, 340, 300, 320}
8090 ,{ 310, 260, 310, 260, 220}
8091 ,{ 310, 260, 310, 260, 220}
8092 ,{ 310, 260, 310, 260, 220}
8093   }
8094 ,{{ 340, 300, 340, 300, 320}
8095 ,{ 340, 300, 340, 300, 320}
8096 ,{ 310, 260, 310, 260, 220}
8097 ,{ 290, 180, 290, 180, 140}
8098 ,{ 310, 260, 310, 260, 220}
8099   }
8100 ,{{ 310, 260, 310, 260, 220}
8101 ,{ 310, 260, 310, 260, 220}
8102 ,{ 310, 260, 310, 260, 220}
8103 ,{ 310, 260, 310, 260, 220}
8104 ,{ 310, 260, 310, 260, 220}
8105   }
8106 ,{{ 330, 260, 330, 260, 220}
8107 ,{ 330, 220, 330, 220, 180}
8108 ,{ 310, 260, 310, 260, 220}
8109 ,{ 220, 130, 180, 130, 220}
8110 ,{ 310, 260, 310, 260, 220}
```

```
8111      }
8112 ,{{ 310, 260, 310, 260, 220}
8113 ,{ 310, 260, 310, 260, 220}
8114 ,{ 310, 260, 310, 260, 220}
8115 ,{ 310, 260, 310, 260, 220}
8116 ,{ 310, 260, 310, 260, 220}
8117 }
8118 }
8119 }
8120 ,{{{ 370, 340, 370, 280, 340}
8121 ,{ 370, 310, 370, 280, 340}
8122 ,{ 280, 280, 280, 240, 280}
8123 ,{ 280, 280, 280, 250, 280}
8124 ,{ 340, 340, 280, 250, 280}
8125 }
8126 ,{{ 280, 280, 280, 230, 280}
8127 ,{ 240, 240, 240, 200, 240}
8128 ,{ 280, 280, 280, 230, 280}
8129 ,{ 200, 140, 200, 120, 170}
8130 ,{ 280, 280, 280, 230, 280}
8131 }
8132 ,{{ 280, 280, 280, 240, 280}
8133 ,{ 280, 280, 280, 230, 280}
8134 ,{ 280, 280, 280, 240, 280}
8135 ,{ 280, 280, 280, 230, 280}
8136 ,{ 280, 280, 280, 240, 280}
8137 }
8138 ,{{ 370, 310, 370, 280, 340}
8139 ,{ 370, 310, 370, 280, 340}
8140 ,{ 280, 280, 280, 230, 280}
8141 ,{ 250, 170, 150, 250, 190}
8142 ,{ 280, 280, 280, 230, 280}
8143 }
8144 ,{{ 340, 340, 280, 250, 280}
8145 ,{ 280, 280, 280, 230, 280}
8146 ,{ 280, 280, 280, 240, 280}
8147 ,{ 280, 280, 280, 230, 280}
8148 ,{ 340, 340, 280, 250, 280}
8149 }
8150 }
8151 ,{{{ 340, 340, 310, 240, 310}
8152 ,{ 310, 310, 310, 210, 310}
8153 ,{ 280, 280, 280, 240, 280}
8154 ,{ 280, 280, 280, 180, 280}
8155 ,{ 340, 340, 280, 240, 280}
8156 }
8157 ,{{ 280, 280, 280, 180, 280}
8158 ,{ 240, 240, 240, 150, 240}
8159 ,{ 280, 280, 280, 180, 280}
8160 ,{ 140, 140, 140, 50, 140}
8161 ,{ 280, 280, 280, 180, 280}
8162 }
8163 ,{{ 280, 280, 280, 240, 280}
8164 ,{ 280, 280, 280, 180, 280}
8165 ,{ 280, 280, 280, 240, 280}
8166 ,{ 280, 280, 280, 180, 280}
8167 ,{ 280, 280, 280, 240, 280}
8168 }
8169 ,{{ 310, 310, 310, 210, 310}
8170 ,{ 310, 310, 310, 210, 310}
8171 ,{ 280, 280, 280, 180, 280}
8172 ,{ 180, 150, 150, 180, 150}
8173 ,{ 280, 280, 280, 180, 280}
8174 }
8175 ,{{ 340, 340, 280, 240, 280}
8176 ,{ 280, 280, 280, 180, 280}
8177 ,{ 280, 280, 280, 240, 280}
8178 ,{ 280, 280, 280, 180, 280}
8179 ,{ 340, 340, 280, 180, 280}
8180 }
8181 }
8182 ,{{{ 370, 260, 370, 260, 340}
8183 ,{ 370, 260, 370, 260, 340}
8184 ,{ 280, 230, 280, 230, 250}
8185 ,{ 280, 230, 280, 230, 250}
8186 ,{ 280, 230, 280, 230, 250}
8187 }
8188 ,{{ 280, 230, 280, 230, 250}
8189 ,{ 240, 200, 240, 200, 210}
8190 ,{ 280, 230, 280, 230, 250}
8191 ,{ 200, 100, 200, 100, 170}
8192 ,{ 280, 230, 280, 230, 250}
8193 }
8194 ,{{ 280, 230, 280, 230, 250}
8195 ,{ 280, 230, 280, 230, 250}
8196 ,{ 280, 230, 280, 230, 250}
8197 ,{ 280, 230, 280, 230, 250}
```

```
8198 , { 280, 230, 280, 230, 250}
8199 }
8200 , {{ 370, 260, 370, 260, 340}
8201 , { 370, 260, 370, 260, 340}
8202 , { 280, 230, 280, 230, 250}
8203 , { 150, 100, 150, 100, 120}
8204 , { 280, 230, 280, 230, 250}
8205 }
8206 , {{ 280, 230, 280, 230, 250}
8207 , { 280, 230, 280, 230, 250}
8208 , { 280, 230, 280, 230, 250}
8209 , { 280, 230, 280, 230, 250}
8210 , { 280, 230, 280, 230, 250}
8211 }
8212 }
8213 , {{{ 310, 230, 310, 280, 310}
8214 , { 310, 200, 310, 280, 310}
8215 , { 280, 230, 280, 120, 280}
8216 , { 280, 170, 280, 250, 280}
8217 , { 280, 230, 280, 250, 280}
8218 }
8219 , {{ 280, 170, 280, 120, 280}
8220 , { 240, 140, 240, 90, 240}
8221 , { 280, 170, 280, 120, 280}
8222 , { 140, 40, 140, 120, 140}
8223 , { 280, 170, 280, 120, 280}
8224 }
8225 , {{ 280, 230, 280, 120, 280}
8226 , { 280, 170, 280, 120, 280}
8227 , { 280, 230, 280, 120, 280}
8228 , { 280, 170, 280, 120, 280}
8229 , { 280, 230, 280, 120, 280}
8230 }
8231 , {{ 310, 200, 310, 280, 310}
8232 , { 310, 200, 310, 280, 310}
8233 , { 280, 170, 280, 120, 280}
8234 , { 250, 170, 150, 250, 150}
8235 , { 280, 170, 280, 120, 280}
8236 }
8237 , {{ 280, 230, 280, 250, 280}
8238 , { 280, 170, 280, 120, 280}
8239 , { 280, 230, 280, 120, 280}
8240 , { 280, 170, 280, 120, 280}
8241 , { 280, 170, 280, 250, 280}
8242 }
8243 }
8244 , {{{ 370, 260, 370, 260, 220}
8245 , { 370, 260, 370, 260, 220}
8246 , { 280, 230, 280, 230, 190}
8247 , { 280, 230, 280, 230, 190}
8248 , { 280, 230, 280, 230, 190}
8249 }
8250 , {{ 280, 230, 280, 230, 220}
8251 , { 240, 200, 240, 200, 220}
8252 , { 280, 230, 280, 230, 190}
8253 , { 200, 100, 200, 100, 60}
8254 , { 280, 230, 280, 230, 190}
8255 }
8256 , {{ 280, 230, 280, 230, 190}
8257 , { 280, 230, 280, 230, 190}
8258 , { 280, 230, 280, 230, 190}
8259 , { 280, 230, 280, 230, 190}
8260 , { 280, 230, 280, 230, 190}
8261 }
8262 , {{ 370, 260, 370, 260, 220}
8263 , { 370, 260, 370, 260, 220}
8264 , { 280, 230, 280, 230, 190}
8265 , { 190, 100, 150, 100, 190}
8266 , { 280, 230, 280, 230, 190}
8267 }
8268 , {{ 280, 230, 280, 230, 190}
8269 , { 280, 230, 280, 230, 190}
8270 , { 280, 230, 280, 230, 190}
8271 , { 280, 230, 280, 230, 190}
8272 , { 280, 230, 280, 230, 190}
8273 }
8274 }
8275 }
8276 , {{{{ 280, 280, 280, 230, 280}
8277 , { 280, 280, 280, 230, 280}
8278 , { 260, 260, 260, 220, 260}
8279 , { 260, 260, 260, 220, 260}
8280 , { 260, 260, 260, 220, 260}
8281 }
8282 , {{ 280, 280, 280, 230, 280}
8283 , { 280, 280, 280, 230, 280}
8284 , { 250, 250, 250, 210, 250}
```

```
8285 , { 210, 150, 210, 130, 180}
8286 , { 250, 250, 250, 210, 250}
8287 }
8288 , {{ 260, 260, 260, 220, 260}
8289 , { 260, 260, 260, 220, 260}
8290 , { 260, 260, 260, 220, 260}
8291 , { 260, 260, 260, 220, 260}
8292 , { 260, 260, 260, 220, 260}
8293 }
8294 , {{ 280, 250, 280, 210, 250}
8295 , { 280, 220, 280, 200, 250}
8296 , { 250, 250, 250, 210, 250}
8297 , { 210, 130, 100, 210, 150}
8298 , { 250, 250, 250, 210, 250}
8299 }
8300 , {{ 260, 260, 260, 220, 260}
8301 , { 260, 260, 260, 220, 260}
8302 , { 260, 260, 260, 220, 260}
8303 , { 260, 260, 260, 220, 260}
8304 , { 230, 230, 170, 140, 170}
8305 }
8306 }
8307 , {{ { 280, 280, 280, 220, 280}
8308 , { 280, 280, 280, 180, 280}
8309 , { 260, 260, 260, 220, 260}
8310 , { 260, 260, 260, 170, 260}
8311 , { 260, 260, 260, 220, 260}
8312 }
8313 , {{ { 280, 280, 280, 180, 280}
8314 , { 280, 280, 280, 180, 280}
8315 , { 250, 250, 250, 160, 250}
8316 , { 150, 150, 150, 60, 150}
8317 , { 250, 250, 250, 160, 250}
8318 }
8319 , {{ { 260, 260, 260, 220, 260}
8320 , { 260, 260, 260, 170, 260}
8321 , { 260, 260, 260, 220, 260}
8322 , { 260, 260, 260, 170, 260}
8323 , { 260, 260, 260, 220, 260}
8324 }
8325 , {{ { 250, 250, 250, 160, 250}
8326 , { 220, 220, 220, 130, 220}
8327 , { 250, 250, 250, 160, 250}
8328 , { 140, 100, 100, 140, 100}
8329 , { 250, 250, 250, 160, 250}
8330 }
8331 , {{ { 260, 260, 260, 220, 260}
8332 , { 260, 260, 260, 170, 260}
8333 , { 260, 260, 260, 220, 260}
8334 , { 260, 260, 260, 170, 260}
8335 , { 230, 230, 170, 70, 170}
8336 }
8337 }
8338 , {{ { { 280, 230, 280, 230, 250}
8339 , { 280, 230, 280, 230, 250}
8340 , { 260, 210, 260, 210, 230}
8341 , { 260, 220, 260, 220, 230}
8342 , { 260, 210, 260, 210, 230}
8343 }
8344 , {{ { 280, 230, 280, 230, 250}
8345 , { 280, 230, 280, 230, 250}
8346 , { 250, 210, 250, 210, 220}
8347 , { 210, 110, 210, 110, 180}
8348 , { 250, 210, 250, 210, 220}
8349 }
8350 , {{ { 260, 220, 260, 220, 230}
8351 , { 260, 220, 260, 220, 230}
8352 , { 260, 210, 260, 210, 230}
8353 , { 260, 220, 260, 220, 230}
8354 , { 260, 210, 260, 210, 230}
8355 }
8356 , {{ { 280, 210, 280, 210, 250}
8357 , { 280, 180, 280, 180, 250}
8358 , { 250, 210, 250, 210, 220}
8359 , { 100, 60, 100, 60, 70}
8360 , { 250, 210, 250, 210, 220}
8361 }
8362 , {{ { 260, 220, 260, 220, 230}
8363 , { 260, 220, 260, 220, 230}
8364 , { 260, 210, 260, 210, 230}
8365 , { 260, 220, 260, 220, 230}
8366 , { 170, 120, 170, 120, 140}
8367 }
8368 }
8369 , {{ { { 280, 210, 280, 210, 280}
8370 , { 280, 170, 280, 200, 280}
8371 , { 260, 210, 260, 100, 260}
```

```
8372 , { 260, 160, 260, 210, 260}
8373 , { 260, 210, 260, 140, 260}
8374 }
8375 , {{ 280, 170, 280, 130, 280}
8376 , { 280, 170, 280, 120, 280}
8377 , { 250, 150, 250, 100, 250}
8378 , { 150, 50, 150, 130, 150}
8379 , { 250, 150, 250, 100, 250}
8380 }
8381 , {{ 260, 210, 260, 110, 260}
8382 , { 260, 160, 260, 110, 260}
8383 , { 260, 210, 260, 100, 260}
8384 , { 260, 160, 260, 110, 260}
8385 , { 260, 210, 260, 100, 260}
8386 }
8387 , {{ 250, 150, 250, 210, 250}
8388 , { 220, 120, 220, 200, 220}
8389 , { 250, 150, 250, 100, 250}
8390 , { 210, 130, 100, 210, 100}
8391 , { 250, 150, 250, 100, 250}
8392 }
8393 , {{ 260, 210, 260, 140, 260}
8394 , { 260, 160, 260, 110, 260}
8395 , { 260, 210, 260, 100, 260}
8396 , { 260, 160, 260, 110, 260}
8397 , { 170, 60, 170, 140, 170}
8398 }
8399 }
8400 , {{{ 280, 230, 280, 230, 250}
8401 , { 280, 230, 280, 230, 250}
8402 , { 260, 210, 260, 210, 170}
8403 , { 260, 220, 260, 220, 180}
8404 , { 260, 210, 260, 210, 170}
8405 }
8406 , {{ 280, 230, 280, 230, 250}
8407 , { 280, 230, 280, 230, 250}
8408 , { 250, 210, 250, 210, 170}
8409 , { 210, 110, 210, 110, 70}
8410 , { 250, 210, 250, 210, 170}
8411 }
8412 , {{ 260, 220, 260, 220, 180}
8413 , { 260, 220, 260, 220, 180}
8414 , { 260, 210, 260, 210, 170}
8415 , { 260, 220, 260, 220, 180}
8416 , { 260, 210, 260, 210, 170}
8417 }
8418 , {{ 280, 210, 280, 210, 170}
8419 , { 280, 180, 280, 180, 140}
8420 , { 250, 210, 250, 210, 170}
8421 , { 150, 60, 100, 60, 150}
8422 , { 250, 210, 250, 210, 170}
8423 }
8424 , {{ 260, 220, 260, 220, 180}
8425 , { 260, 220, 260, 220, 180}
8426 , { 260, 210, 260, 210, 170}
8427 , { 260, 220, 260, 220, 180}
8428 , { 170, 120, 170, 120, 80}
8429 }
8430 }
8431 }
8432 , {{{{ 280, 280, 280, 240, 280}
8433 , { 280, 280, 280, 230, 280}
8434 , { 280, 280, 280, 240, 280}
8435 , { 280, 280, 280, 230, 280}
8436 , { 280, 280, 280, 240, 280}
8437 }
8438 , {{ 280, 280, 280, 230, 280}
8439 , { 280, 280, 280, 230, 280}
8440 , { 230, 230, 230, 190, 230}
8441 , { 230, 170, 230, 150, 200}
8442 , { 230, 230, 230, 190, 230}
8443 }
8444 , {{ 280, 280, 280, 240, 280}
8445 , { 280, 280, 280, 230, 280}
8446 , { 280, 280, 280, 240, 280}
8447 , { 280, 280, 280, 230, 280}
8448 , { 280, 280, 280, 240, 280}
8449 }
8450 , {{ 240, 230, 240, 230, 230}
8451 , { 240, 180, 240, 160, 210}
8452 , { 230, 230, 230, 190, 230}
8453 , { 230, 150, 120, 230, 170}
8454 , { 230, 230, 230, 190, 230}
8455 }
8456 , {{ 280, 280, 280, 230, 280}
8457 , { 280, 280, 280, 230, 280}
8458 , { 250, 250, 250, 210, 250}
```

```
8459 , { 280, 280, 280, 230, 280}
8460 , { 250, 250, 190, 170, 190}
8461 }
8462 }
8463 , {{ 280, 280, 280, 240, 280}
8464 , { 280, 280, 280, 180, 280}
8465 , { 280, 280, 280, 240, 280}
8466 , { 280, 280, 280, 180, 280}
8467 , { 280, 280, 280, 240, 280}
8468 }
8469 , {{ 280, 280, 280, 180, 280}
8470 , { 280, 280, 280, 180, 280}
8471 , { 230, 230, 230, 140, 230}
8472 , { 170, 170, 170, 80, 170}
8473 , { 230, 230, 230, 140, 230}
8474 }
8475 , {{ 280, 280, 280, 240, 280}
8476 , { 280, 280, 280, 180, 280}
8477 , { 280, 280, 280, 240, 280}
8478 , { 280, 280, 280, 180, 280}
8479 , { 280, 280, 280, 240, 280}
8480 }
8481 , {{ 230, 230, 230, 160, 230}
8482 , { 180, 180, 180, 90, 180}
8483 , { 230, 230, 230, 140, 230}
8484 , { 160, 120, 120, 160, 120}
8485 , { 230, 230, 230, 140, 230}
8486 }
8487 , {{ 280, 280, 280, 210, 280}
8488 , { 280, 280, 280, 180, 280}
8489 , { 250, 250, 250, 210, 250}
8490 , { 280, 280, 280, 180, 280}
8491 , { 250, 250, 190, 100, 190}
8492 }
8493 }
8494 , {{ 280, 230, 280, 230, 250}
8495 , { 280, 230, 280, 230, 250}
8496 , { 280, 230, 280, 230, 250}
8497 , { 280, 230, 280, 230, 250}
8498 , { 280, 230, 280, 230, 250}
8499 }
8500 , {{ 280, 230, 280, 230, 250}
8501 , { 280, 230, 280, 230, 250}
8502 , { 230, 190, 230, 190, 200}
8503 , { 230, 130, 230, 130, 200}
8504 , { 230, 190, 230, 190, 200}
8505 }
8506 , {{ 280, 230, 280, 230, 250}
8507 , { 280, 230, 280, 230, 250}
8508 , { 280, 230, 280, 230, 250}
8509 , { 280, 230, 280, 230, 250}
8510 , { 280, 230, 280, 230, 250}
8511 }
8512 , {{ 240, 190, 240, 190, 210}
8513 , { 240, 140, 240, 140, 210}
8514 , { 230, 190, 230, 190, 200}
8515 , { 120, 80, 120, 80, 90}
8516 , { 230, 190, 230, 190, 200}
8517 }
8518 , {{ 280, 230, 280, 230, 250}
8519 , { 280, 230, 280, 230, 250}
8520 , { 250, 200, 250, 200, 220}
8521 , { 280, 230, 280, 230, 250}
8522 , { 190, 150, 190, 150, 160}
8523 }
8524 }
8525 , {{ 280, 230, 280, 230, 280}
8526 , { 280, 170, 280, 160, 280}
8527 , { 280, 230, 280, 120, 280}
8528 , { 280, 170, 280, 230, 280}
8529 , { 280, 230, 280, 170, 280}
8530 }
8531 , {{ 280, 170, 280, 150, 280}
8532 , { 280, 170, 280, 120, 280}
8533 , { 230, 130, 230, 80, 230}
8534 , { 170, 70, 170, 150, 170}
8535 , { 230, 130, 230, 80, 230}
8536 }
8537 , {{ 280, 230, 280, 120, 280}
8538 , { 280, 170, 280, 120, 280}
8539 , { 280, 230, 280, 120, 280}
8540 , { 280, 170, 280, 120, 280}
8541 , { 280, 230, 280, 120, 280}
8542 }
8543 , {{ 230, 150, 230, 230, 230}
8544 , { 180, 80, 180, 160, 180}
8545 , { 230, 130, 230, 80, 230}
```

```
8546 , { 230, 150, 120, 230, 120}
8547 , { 230, 130, 230, 80, 230}
8548 }
8549 , {{ 280, 200, 280, 170, 280}
8550 , { 280, 170, 280, 120, 280}
8551 , { 250, 200, 250, 90, 250}
8552 , { 280, 170, 280, 120, 280}
8553 , { 190, 90, 190, 170, 190}
8554 }
8555 }
8556 , {{{ 280, 230, 280, 230, 250}
8557 , { 280, 230, 280, 230, 250}
8558 , { 280, 230, 280, 230, 190}
8559 , { 280, 230, 280, 230, 190}
8560 , { 280, 230, 280, 230, 190}
8561 }
8562 , {{ 280, 230, 280, 230, 250}
8563 , { 280, 230, 280, 230, 250}
8564 , { 230, 190, 230, 190, 150}
8565 , { 230, 130, 230, 130, 90}
8566 , { 230, 190, 230, 190, 150}
8567 }
8568 , {{ 280, 230, 280, 230, 190}
8569 , { 280, 230, 280, 230, 190}
8570 , { 280, 230, 280, 230, 190}
8571 , { 280, 230, 280, 230, 190}
8572 , { 280, 230, 280, 230, 190}
8573 }
8574 , {{ 240, 190, 240, 190, 170}
8575 , { 240, 140, 240, 140, 100}
8576 , { 230, 190, 230, 190, 150}
8577 , { 170, 80, 120, 80, 170}
8578 , { 230, 190, 230, 190, 150}
8579 }
8580 , {{ 280, 230, 280, 230, 190}
8581 , { 280, 230, 280, 230, 190}
8582 , { 250, 200, 250, 200, 160}
8583 , { 280, 230, 280, 230, 190}
8584 , { 190, 150, 190, 150, 110}
8585 }
8586 }
8587 }
8588 , {{{{ 370, 370, 370, 300, 340}
8589 , { 370, 340, 370, 300, 340}
8590 , { 310, 310, 310, 270, 310}
8591 , { 310, 310, 310, 280, 310}
8592 , { 370, 370, 310, 280, 310}
8593 }
8594 , {{ 340, 340, 340, 300, 340}
8595 , { 340, 340, 340, 300, 340}
8596 , { 310, 310, 310, 260, 310}
8597 , { 290, 230, 290, 200, 260}
8598 , { 310, 310, 310, 260, 310}
8599 }
8600 , {{ 310, 310, 310, 270, 310}
8601 , { 310, 310, 310, 260, 310}
8602 , { 310, 310, 310, 270, 310}
8603 , { 310, 310, 310, 260, 310}
8604 , { 310, 310, 310, 270, 310}
8605 }
8606 , {{ 370, 310, 370, 280, 340}
8607 , { 370, 310, 370, 280, 340}
8608 , { 310, 310, 310, 260, 310}
8609 , { 280, 200, 180, 280, 220}
8610 , { 310, 310, 310, 260, 310}
8611 }
8612 , {{ 370, 370, 310, 280, 310}
8613 , { 310, 310, 310, 260, 310}
8614 , { 310, 310, 310, 270, 310}
8615 , { 310, 310, 310, 260, 310}
8616 , { 370, 370, 310, 280, 310}
8617 }
8618 }
8619 , {{{{ 370, 370, 340, 270, 340}
8620 , { 340, 340, 340, 250, 340}
8621 , { 310, 310, 310, 270, 310}
8622 , { 310, 310, 310, 210, 310}
8623 , { 370, 370, 310, 270, 310}
8624 }
8625 , {{ 340, 340, 340, 250, 340}
8626 , { 340, 340, 340, 250, 340}
8627 , { 310, 310, 310, 210, 310}
8628 , { 230, 230, 230, 130, 230}
8629 , { 310, 310, 310, 210, 310}
8630 }
8631 , {{ 310, 310, 310, 270, 310}
8632 , { 310, 310, 310, 210, 310}
```

```
8633 , { 310, 310, 310, 270, 310}
8634 , { 310, 310, 310, 210, 310}
8635 , { 310, 310, 310, 270, 310}
8636 }
8637 , {{ 310, 310, 310, 210, 310}
8638 , { 310, 310, 310, 210, 310}
8639 , { 310, 310, 310, 210, 310}
8640 , { 210, 180, 180, 210, 180}
8641 , { 310, 310, 310, 210, 310}
8642 }
8643 , {{ 370, 370, 310, 270, 310}
8644 , { 310, 310, 310, 210, 310}
8645 , { 310, 310, 310, 270, 310}
8646 , { 310, 310, 310, 210, 310}
8647 , { 370, 370, 310, 210, 310}
8648 }
8649 }
8650 , {{{ 370, 300, 370, 300, 340}
8651 , { 370, 300, 370, 300, 340}
8652 , { 310, 260, 310, 260, 280}
8653 , { 310, 260, 310, 260, 280}
8654 , { 310, 260, 310, 260, 280}
8655 }
8656 , {{ 340, 300, 340, 300, 310}
8657 , { 340, 300, 340, 300, 310}
8658 , { 310, 260, 310, 260, 280}
8659 , { 290, 180, 290, 180, 260}
8660 , { 310, 260, 310, 260, 280}
8661 }
8662 , {{ 310, 260, 310, 260, 280}
8663 , { 310, 260, 310, 260, 280}
8664 , { 310, 260, 310, 260, 280}
8665 , { 310, 260, 310, 260, 280}
8666 , { 310, 260, 310, 260, 280}
8667 }
8668 , {{ 370, 260, 370, 260, 340}
8669 , { 370, 260, 370, 260, 340}
8670 , { 310, 260, 310, 260, 280}
8671 , { 180, 130, 180, 130, 150}
8672 , { 310, 260, 310, 260, 280}
8673 }
8674 , {{ 310, 260, 310, 260, 280}
8675 , { 310, 260, 310, 260, 280}
8676 , { 310, 260, 310, 260, 280}
8677 , { 310, 260, 310, 260, 280}
8678 , { 310, 260, 310, 260, 280}
8679 }
8680 }
8681 , {{{ 340, 260, 340, 280, 340}
8682 , { 340, 240, 340, 280, 340}
8683 , { 310, 260, 310, 150, 310}
8684 , { 310, 200, 310, 280, 310}
8685 , { 310, 260, 310, 280, 310}
8686 }
8687 , {{ 340, 240, 340, 200, 340}
8688 , { 340, 240, 340, 190, 340}
8689 , { 310, 200, 310, 150, 310}
8690 , { 230, 120, 230, 200, 230}
8691 , { 310, 200, 310, 150, 310}
8692 }
8693 , {{ 310, 260, 310, 150, 310}
8694 , { 310, 200, 310, 150, 310}
8695 , { 310, 260, 310, 150, 310}
8696 , { 310, 200, 310, 150, 310}
8697 , { 310, 260, 310, 150, 310}
8698 }
8699 , {{ 310, 200, 310, 280, 310}
8700 , { 310, 200, 310, 280, 310}
8701 , { 310, 200, 310, 150, 310}
8702 , { 280, 200, 180, 280, 180}
8703 , { 310, 200, 310, 150, 310}
8704 }
8705 , {{ 310, 260, 310, 280, 310}
8706 , { 310, 200, 310, 150, 310}
8707 , { 310, 260, 310, 150, 310}
8708 , { 310, 200, 310, 150, 310}
8709 , { 310, 200, 310, 280, 310}
8710 }
8711 }
8712 , {{{ 370, 300, 370, 300, 320}
8713 , { 370, 300, 370, 300, 320}
8714 , { 310, 260, 310, 260, 220}
8715 , { 310, 260, 310, 260, 220}
8716 , { 310, 260, 310, 260, 220}
8717 }
8718 , {{ 340, 300, 340, 300, 320}
8719 , { 340, 300, 340, 300, 320}
```

```
8720 , { 310, 260, 310, 260, 220}
8721 , { 290, 180, 290, 180, 140}
8722 , { 310, 260, 310, 260, 220}
8723 }
8724 , {{ 310, 260, 310, 260, 220}
8725 , { 310, 260, 310, 260, 220}
8726 , { 310, 260, 310, 260, 220}
8727 , { 310, 260, 310, 260, 220}
8728 , { 310, 260, 310, 260, 220}
8729 }
8730 , {{ 370, 260, 370, 260, 220}
8731 , { 370, 260, 370, 260, 220}
8732 , { 310, 260, 310, 260, 220}
8733 , { 220, 130, 180, 130, 220}
8734 , { 310, 260, 310, 260, 220}
8735 }
8736 , {{ 310, 260, 310, 260, 220}
8737 , { 310, 260, 310, 260, 220}
8738 , { 310, 260, 310, 260, 220}
8739 , { 310, 260, 310, 260, 220}
8740 , { 310, 260, 310, 260, 220}
8741 }
8742 }
8743 }
8744 }
8745 , {{{{ { INF, INF, INF, INF, INF }
8746 , { INF, INF, INF, INF, INF }
8747 , { INF, INF, INF, INF, INF }
8748 , { INF, INF, INF, INF, INF }
8749 , { INF, INF, INF, INF, INF }
8750 }
8751 , {{ { INF, INF, INF, INF, INF }
8752 , { INF, INF, INF, INF, INF }
8753 , { INF, INF, INF, INF, INF }
8754 , { INF, INF, INF, INF, INF }
8755 , { INF, INF, INF, INF, INF }
8756 }
8757 , {{ { INF, INF, INF, INF, INF }
8758 , { INF, INF, INF, INF, INF }
8759 , { INF, INF, INF, INF, INF }
8760 , { INF, INF, INF, INF, INF }
8761 , { INF, INF, INF, INF, INF }
8762 }
8763 , {{ { INF, INF, INF, INF, INF }
8764 , { INF, INF, INF, INF, INF }
8765 , { INF, INF, INF, INF, INF }
8766 , { INF, INF, INF, INF, INF }
8767 , { INF, INF, INF, INF, INF }
8768 }
8769 , {{ { INF, INF, INF, INF, INF }
8770 , { INF, INF, INF, INF, INF }
8771 , { INF, INF, INF, INF, INF }
8772 , { INF, INF, INF, INF, INF }
8773 , { INF, INF, INF, INF, INF }
8774 }
8775 }
8776 , {{{ { INF, INF, INF, INF, INF }
8777 , { INF, INF, INF, INF, INF }
8778 , { INF, INF, INF, INF, INF }
8779 , { INF, INF, INF, INF, INF }
8780 , { INF, INF, INF, INF, INF }
8781 }
8782 , {{ { INF, INF, INF, INF, INF }
8783 , { INF, INF, INF, INF, INF }
8784 , { INF, INF, INF, INF, INF }
8785 , { INF, INF, INF, INF, INF }
8786 , { INF, INF, INF, INF, INF }
8787 }
8788 , {{ { INF, INF, INF, INF, INF }
8789 , { INF, INF, INF, INF, INF }
8790 , { INF, INF, INF, INF, INF }
8791 , { INF, INF, INF, INF, INF }
8792 , { INF, INF, INF, INF, INF }
8793 }
8794 , {{ { INF, INF, INF, INF, INF }
8795 , { INF, INF, INF, INF, INF }
8796 , { INF, INF, INF, INF, INF }
8797 , { INF, INF, INF, INF, INF }
8798 , { INF, INF, INF, INF, INF }
8799 }
8800 , {{ { INF, INF, INF, INF, INF }
8801 , { INF, INF, INF, INF, INF }
8802 , { INF, INF, INF, INF, INF }
8803 , { INF, INF, INF, INF, INF }
8804 , { INF, INF, INF, INF, INF }
8805 }
8806 }
```

```
8807 ,{{{{ INF, INF, INF, INF, INF
8808 ,{ INF, INF, INF, INF, INF
8809 ,{ INF, INF, INF, INF, INF
8810 ,{ INF, INF, INF, INF, INF
8811 ,{ INF, INF, INF, INF, INF
8812 }
8813 ,{{ INF, INF, INF, INF, INF
8814 ,{ INF, INF, INF, INF, INF
8815 ,{ INF, INF, INF, INF, INF
8816 ,{ INF, INF, INF, INF, INF
8817 ,{ INF, INF, INF, INF, INF
8818 }
8819 ,{{ INF, INF, INF, INF, INF
8820 ,{ INF, INF, INF, INF, INF
8821 ,{ INF, INF, INF, INF, INF
8822 ,{ INF, INF, INF, INF, INF
8823 ,{ INF, INF, INF, INF, INF
8824 }
8825 ,{{ INF, INF, INF, INF, INF
8826 ,{ INF, INF, INF, INF, INF
8827 ,{ INF, INF, INF, INF, INF
8828 ,{ INF, INF, INF, INF, INF
8829 ,{ INF, INF, INF, INF, INF
8830 }
8831 ,{{ INF, INF, INF, INF, INF
8832 ,{ INF, INF, INF, INF, INF
8833 ,{ INF, INF, INF, INF, INF
8834 ,{ INF, INF, INF, INF, INF
8835 ,{ INF, INF, INF, INF, INF
8836 }
8837 }
8838 ,{{{ INF, INF, INF, INF, INF
8839 ,{ INF, INF, INF, INF, INF
8840 ,{ INF, INF, INF, INF, INF
8841 ,{ INF, INF, INF, INF, INF
8842 ,{ INF, INF, INF, INF, INF
8843 }
8844 ,{{ INF, INF, INF, INF, INF
8845 ,{ INF, INF, INF, INF, INF
8846 ,{ INF, INF, INF, INF, INF
8847 ,{ INF, INF, INF, INF, INF
8848 ,{ INF, INF, INF, INF, INF
8849 }
8850 ,{{ INF, INF, INF, INF, INF
8851 ,{ INF, INF, INF, INF, INF
8852 ,{ INF, INF, INF, INF, INF
8853 ,{ INF, INF, INF, INF, INF
8854 ,{ INF, INF, INF, INF, INF
8855 }
8856 ,{{ INF, INF, INF, INF, INF
8857 ,{ INF, INF, INF, INF, INF
8858 ,{ INF, INF, INF, INF, INF
8859 ,{ INF, INF, INF, INF, INF
8860 ,{ INF, INF, INF, INF, INF
8861 }
8862 ,{{ INF, INF, INF, INF, INF
8863 ,{ INF, INF, INF, INF, INF
8864 ,{ INF, INF, INF, INF, INF
8865 ,{ INF, INF, INF, INF, INF
8866 ,{ INF, INF, INF, INF, INF
8867 }
8868 }
8869 ,{{{ INF, INF, INF, INF, INF
8870 ,{ INF, INF, INF, INF, INF
8871 ,{ INF, INF, INF, INF, INF
8872 ,{ INF, INF, INF, INF, INF
8873 ,{ INF, INF, INF, INF, INF
8874 }
8875 ,{{ INF, INF, INF, INF, INF
8876 ,{ INF, INF, INF, INF, INF
8877 ,{ INF, INF, INF, INF, INF
8878 ,{ INF, INF, INF, INF, INF
8879 ,{ INF, INF, INF, INF, INF
8880 }
8881 ,{{ INF, INF, INF, INF, INF
8882 ,{ INF, INF, INF, INF, INF
8883 ,{ INF, INF, INF, INF, INF
8884 ,{ INF, INF, INF, INF, INF
8885 ,{ INF, INF, INF, INF, INF
8886 }
8887 ,{{ INF, INF, INF, INF, INF
8888 ,{ INF, INF, INF, INF, INF
8889 ,{ INF, INF, INF, INF, INF
8890 ,{ INF, INF, INF, INF, INF
8891 ,{ INF, INF, INF, INF, INF
8892 }
8893 ,{{ INF, INF, INF, INF, INF
```

```
8894 , { INF, INF, INF, INF, INF }
8895 , { INF, INF, INF, INF, INF }
8896 , { INF, INF, INF, INF, INF }
8897 , { INF, INF, INF, INF, INF }
8898 }
8899 }
8900 }
8901 ,{ {{ 310, 300, 270, 310, 290}
8902 ,{ 300, 300, 270, 270, 290}
8903 ,{ 310, 290, 250, 310, 250}
8904 ,{ 300, 300, 270, 270, 270}
8905 ,{ 300, 270, 240, 300, 240}
8906 }
8907 ,{ { 290, 270, 230, 230, 290}
8908 ,{ 290, 270, 230, 230, 290}
8909 ,{ 260, 260, 220, 220, 220}
8910 ,{ 190, 170, 190, 130, 190}
8911 ,{ 260, 260, 220, 220, 220}
8912 }
8913 ,{ { 310, 300, 270, 310, 270}
8914 ,{ 300, 300, 270, 270, 270}
8915 ,{ 310, 290, 250, 310, 250}
8916 ,{ 300, 300, 270, 270, 270}
8917 ,{ 300, 270, 240, 300, 240}
8918 }
8919 ,{ { 260, 260, 220, 220, 220}
8920 ,{ 190, 170, 190, 130, 190}
8921 ,{ 260, 260, 220, 220, 220}
8922 ,{ 210, 130, 80, 210, 210}
8923 ,{ 260, 260, 220, 220, 220}
8924 }
8925 ,{ { 300, 300, 270, 300, 270}
8926 ,{ 300, 300, 270, 270, 270}
8927 ,{ 300, 270, 240, 300, 240}
8928 ,{ 300, 300, 270, 270, 270}
8929 ,{ 240, 240, 150, 150, 150}
8930 }
8931 }
8932 ,{ {{ 310, 300, 270, 310, 270}
8933 ,{ 300, 300, 270, 270, 270}
8934 ,{ 310, 290, 250, 310, 250}
8935 ,{ 300, 300, 270, 270, 270}
8936 ,{ 300, 270, 240, 300, 240}
8937 }
8938 ,{ { 270, 270, 230, 230, 230}
8939 ,{ 270, 270, 230, 230, 230}
8940 ,{ 260, 260, 220, 220, 220}
8941 ,{ 170, 170, 130, 130, 130}
8942 ,{ 260, 260, 220, 220, 220}
8943 }
8944 ,{ { 310, 300, 270, 310, 270}
8945 ,{ 300, 300, 270, 270, 270}
8946 ,{ 310, 290, 250, 310, 250}
8947 ,{ 300, 300, 270, 270, 270}
8948 ,{ 300, 270, 240, 300, 240}
8949 }
8950 ,{ { 260, 260, 220, 220, 220}
8951 ,{ 170, 170, 130, 130, 130}
8952 ,{ 260, 260, 220, 220, 220}
8953 ,{ 210, 110, 80, 210, 80}
8954 ,{ 260, 260, 220, 220, 220}
8955 }
8956 ,{ { 300, 300, 270, 300, 270}
8957 ,{ 300, 300, 270, 270, 270}
8958 ,{ 300, 270, 240, 300, 240}
8959 ,{ 300, 300, 270, 270, 270}
8960 ,{ 240, 240, 150, 150, 150}
8961 }
8962 }
8963 ,{ {{ 270, 270, 270, 270, 270}
8964 ,{ 270, 270, 270, 270, 270}
8965 ,{ 250, 250, 250, 250, 250}
8966 ,{ 270, 270, 270, 270, 270}
8967 ,{ 240, 240, 240, 240, 240}
8968 }
8969 ,{ { 230, 230, 230, 230, 230}
8970 ,{ 230, 230, 230, 230, 230}
8971 ,{ 220, 220, 220, 220, 220}
8972 ,{ 190, 130, 190, 130, 190}
8973 ,{ 220, 220, 220, 220, 220}
8974 }
8975 ,{ { 270, 270, 270, 270, 270}
8976 ,{ 270, 270, 270, 270, 270}
8977 ,{ 250, 250, 250, 250, 250}
8978 ,{ 270, 270, 270, 270, 270}
8979 ,{ 240, 240, 240, 240, 240}
8980 }
```

```
8981 , {{ 220, 220, 220, 220, 220}}
8982 , { 190, 130, 190, 130, 190}
8983 , { 220, 220, 220, 220, 220}
8984 , { 80, 80, 80, 80, 80}
8985 , { 220, 220, 220, 220, 220}
8986 }
8987 , {{ 270, 270, 270, 270, 270}}
8988 , { 270, 270, 270, 270, 270}
8989 , { 240, 240, 240, 240, 240}
8990 , { 270, 270, 270, 270, 270}
8991 , { 150, 150, 150, 150, 150}
8992 }
8993 }
8994 , {{{ 270, 230, 270, 210, 270}}
8995 , { 270, 190, 270, 140, 270}
8996 , { 250, 230, 250, 120, 250}
8997 , { 270, 190, 270, 210, 270}
8998 , { 240, 220, 240, 150, 240}
8999 }
9000 , {{ 230, 150, 230, 130, 230}}
9001 , { 230, 150, 230, 100, 230}
9002 , { 220, 140, 220, 90, 220}
9003 , { 130, 50, 130, 130, 130}
9004 , { 220, 140, 220, 90, 220}
9005 }
9006 , {{ 270, 230, 270, 140, 270}}
9007 , { 270, 190, 270, 140, 270}
9008 , { 250, 230, 250, 120, 250}
9009 , { 270, 190, 270, 140, 270}
9010 , { 240, 220, 240, 110, 240}
9011 }
9012 , {{ 220, 140, 220, 210, 220}}
9013 , { 130, 50, 130, 130, 130}
9014 , { 220, 140, 220, 90, 220}
9015 , { 210, 130, 80, 210, 80}
9016 , { 220, 140, 220, 90, 220}
9017 }
9018 , {{ 270, 220, 270, 150, 270}}
9019 , { 270, 190, 270, 140, 270}
9020 , { 240, 220, 240, 110, 240}
9021 , { 270, 190, 270, 140, 270}
9022 , { 150, 70, 150, 150, 150}
9023 }
9024 }
9025 , {{{ 290, 270, 270, 270, 290}}
9026 , { 290, 270, 270, 270, 290}
9027 , { 250, 250, 250, 250, 250}
9028 , { 270, 270, 270, 270, 270}
9029 , { 240, 240, 240, 240, 240}
9030 }
9031 , {{ 290, 230, 230, 230, 290}}
9032 , { 290, 230, 230, 230, 290}
9033 , { 220, 220, 220, 220, 220}
9034 , { 190, 130, 190, 130, 130}
9035 , { 220, 220, 220, 220, 220}
9036 }
9037 , {{ 270, 270, 270, 270, 270}}
9038 , { 270, 270, 270, 270, 270}
9039 , { 250, 250, 250, 250, 250}
9040 , { 270, 270, 270, 270, 270}
9041 , { 240, 240, 240, 240, 240}
9042 }
9043 , {{ 220, 220, 220, 220, 220}}
9044 , { 190, 130, 190, 130, 130}
9045 , { 220, 220, 220, 220, 220}
9046 , { 210, 80, 80, 80, 210}
9047 , { 220, 220, 220, 220, 220}
9048 }
9049 , {{ 270, 270, 270, 270, 270}}
9050 , { 270, 270, 270, 270, 270}
9051 , { 240, 240, 240, 240, 240}
9052 , { 270, 270, 270, 270, 270}
9053 , { 150, 150, 150, 150, 150}
9054 }
9055 }
9056 }
9057 , {{{{ 300, 280, 240, 280, 300}}
9058 , { 300, 280, 240, 240, 300}
9059 , { 280, 260, 220, 280, 220}
9060 , { 250, 250, 210, 210, 210}
9061 , { 280, 250, 220, 280, 220}
9062 }
9063 , {{ 300, 280, 240, 240, 300}}
9064 , { 300, 280, 240, 240, 300}
9065 , { 250, 250, 220, 220, 220}
9066 , { 100, 70, 100, 40, 100}
9067 , { 250, 250, 220, 220, 220}
```

```
9068    }
9069 ,{{ 280, 250, 220, 280, 220}
9070 ,{ 250, 250, 210, 210, 210}
9071 ,{ 280, 250, 220, 280, 220}
9072 ,{ 250, 250, 210, 210, 210}
9073 ,{ 280, 250, 220, 280, 220}
9074 }
9075 ,{{ 250, 250, 220, 220, 220}
9076 ,{ 160, 140, 160, 100, 160}
9077 ,{ 250, 250, 220, 220, 220}
9078 ,{ 210, 130, 80, 210, 210}
9079 ,{ 250, 250, 220, 220, 220}
9080 }
9081 ,{{ 280, 260, 220, 280, 220}
9082 ,{ 250, 250, 210, 210, 210}
9083 ,{ 280, 260, 220, 280, 220}
9084 ,{ 250, 250, 210, 210, 210}
9085 ,{ 240, 240, 140, 140, 140}
9086 }
9087 }
9088 ,{{{ 280, 280, 240, 280, 240}
9089 ,{ 280, 280, 240, 240, 240}
9090 ,{ 280, 260, 220, 280, 220}
9091 ,{ 250, 250, 210, 210, 210}
9092 ,{ 280, 250, 220, 280, 220}
9093 }
9094 ,{{ 280, 280, 240, 240, 240}
9095 ,{ 280, 280, 240, 240, 240}
9096 ,{ 250, 250, 220, 220, 220}
9097 ,{ 70, 70, 40, 40, 40}
9098 ,{ 250, 250, 220, 220, 220}
9099 }
9100 ,{{ 280, 250, 220, 280, 220}
9101 ,{ 250, 250, 210, 210, 210}
9102 ,{ 280, 250, 220, 280, 220}
9103 ,{ 250, 250, 210, 210, 210}
9104 ,{ 280, 250, 220, 280, 220}
9105 }
9106 ,{{ 250, 250, 220, 220, 220}
9107 ,{ 140, 140, 100, 100, 100}
9108 ,{ 250, 250, 220, 220, 220}
9109 ,{ 210, 110, 80, 210, 80}
9110 ,{ 250, 250, 220, 220, 220}
9111 }
9112 ,{{ 280, 260, 220, 280, 220}
9113 ,{ 250, 250, 210, 210, 210}
9114 ,{ 280, 260, 220, 280, 220}
9115 ,{ 250, 250, 210, 210, 210}
9116 ,{ 240, 240, 140, 140, 140}
9117 }
9118 }
9119 ,{{{ 240, 240, 240, 240, 240}
9120 ,{ 240, 240, 240, 240, 240}
9121 ,{ 220, 220, 220, 220, 220}
9122 ,{ 210, 210, 210, 210, 210}
9123 ,{ 220, 220, 220, 220, 220}
9124 }
9125 ,{{ 240, 240, 240, 240, 240}
9126 ,{ 240, 240, 240, 240, 240}
9127 ,{ 220, 220, 220, 220, 220}
9128 ,{ 100, 40, 100, 40, 100}
9129 ,{ 220, 220, 220, 220, 220}
9130 }
9131 ,{{ 220, 220, 220, 220, 220}
9132 ,{ 210, 210, 210, 210, 210}
9133 ,{ 220, 220, 220, 220, 220}
9134 ,{ 210, 210, 210, 210, 210}
9135 ,{ 220, 220, 220, 220, 220}
9136 }
9137 ,{{ 220, 220, 220, 220, 220}
9138 ,{ 160, 100, 160, 100, 160}
9139 ,{ 220, 220, 220, 220, 220}
9140 ,{ 80, 80, 80, 80, 80}
9141 ,{ 220, 220, 220, 220, 220}
9142 }
9143 ,{{ 220, 220, 220, 220, 220}
9144 ,{ 210, 210, 210, 210, 210}
9145 ,{ 220, 220, 220, 220, 220}
9146 ,{ 210, 210, 210, 210, 210}
9147 ,{ 140, 140, 140, 140, 140}
9148 }
9149 }
9150 ,{{{ 240, 200, 240, 210, 240}
9151 ,{ 240, 160, 240, 110, 240}
9152 ,{ 220, 200, 220, 90, 220}
9153 ,{ 210, 130, 210, 210, 210}
9154 ,{ 220, 200, 220, 140, 220}
```

```
9155      }
9156 ,{{ 240,   160,    240,   110,    240}
9157 ,{ 240,   160,    240,   110,    240}
9158 ,{ 220,   140,    220,    90,    220}
9159 ,{   40,   -40,     40,     40,     40}
9160 ,{ 220,   140,    220,    90,    220}
9161      }
9162 ,{{ 220,   200,    220,    90,    220}
9163 ,{ 210,   130,    210,    80,    210}
9164 ,{ 220,   200,    220,    90,    220}
9165 ,{ 210,   130,    210,    80,    210}
9166 ,{ 220,   200,    220,    90,    220}
9167      }
9168 ,{{ 220,   140,    220,   210,    220}
9169 ,{ 100,    20,    100,   100,   100}
9170 ,{ 220,   140,    220,    90,    220}
9171 ,{ 210,   130,     80,   210,     80}
9172 ,{ 220,   140,    220,    90,    220}
9173      }
9174 ,{{ 220,   200,    220,   140,    220}
9175 ,{ 210,   130,    210,    80,    210}
9176 ,{ 220,   200,    220,    90,    220}
9177 ,{ 210,   130,    210,    80,    210}
9178 ,{ 140,    90,   140,   140,   140}
9179      }
9180      }
9181 ,{{{ 300,   240,    240,   240,   300}
9182 ,{ 300,   240,    240,   240,   300}
9183 ,{ 220,   220,    220,   220,   220}
9184 ,{ 210,   210,    210,   210,   210}
9185 ,{ 220,   220,    220,   220,   220}
9186      }
9187 ,{{ 300,   240,    240,   240,   300}
9188 ,{ 300,   240,    240,   240,   300}
9189 ,{ 220,   220,    220,   220,   220}
9190 ,{ 100,    40,   100,    40,    50}
9191 ,{ 220,   220,    220,   220,   220}
9192      }
9193 ,{{ 220,   220,    220,   220,   220}
9194 ,{ 210,   210,    210,   210,   210}
9195 ,{ 220,   220,    220,   220,   220}
9196 ,{ 210,   210,    210,   210,   210}
9197 ,{ 220,   220,    220,   220,   220}
9198      }
9199 ,{{ 220,   220,    220,   220,   220}
9200 ,{ 160,   100,   160,   100,   140}
9201 ,{ 220,   220,    220,   220,   220}
9202 ,{ 210,    80,    80,    80,   210}
9203 ,{ 220,   220,    220,   220,   220}
9204      }
9205 ,{{ 220,   220,    220,   220,   220}
9206 ,{ 210,   210,    210,   210,   210}
9207 ,{ 220,   220,    220,   220,   220}
9208 ,{ 210,   210,    210,   210,   210}
9209 ,{ 140,   140,   140,   140,   140}
9210      }
9211      }
9212      }
9213 ,{{{ 430,   430,    370,   400,   430}
9214 ,{ 430,   410,    370,   370,   430}
9215 ,{ 400,   370,    340,   400,   340}
9216 ,{ 370,   370,    340,   340,   340}
9217 ,{ 430,   430,    340,   400,   340}
9218      }
9219 ,{ 430,   410,    370,   370,   430}
9220 ,{ 430,   410,    370,   370,   430}
9221 ,{ 370,   370,    340,   340,   340}
9222 ,{ 320,   290,    320,   260,   320}
9223 ,{ 370,   370,    340,   340,   340}
9224      }
9225 ,{ 400,   370,    340,   400,   340}
9226 ,{ 370,   370,    340,   340,   340}
9227 ,{ 400,   370,    340,   400,   340}
9228 ,{ 370,   370,    340,   340,   340}
9229 ,{ 400,   370,    340,   400,   340}
9230      }
9231 ,{ 370,   370,    360,   340,   360}
9232 ,{ 360,   360,    360,   300,   360}
9233 ,{ 370,   370,    340,   340,   340}
9234 ,{ 340,   260,    210,   340,   340}
9235 ,{ 370,   370,    340,   340,   340}
9236      }
9237 ,{ 430,   430,    340,   400,   340}
9238 ,{ 370,   370,    340,   340,   340}
9239 ,{ 400,   370,    340,   400,   340}
9240 ,{ 370,   370,    340,   340,   340}
9241 ,{ 430,   430,    340,   340,   340}
```

```
9242      }
9243  }
9244 ,{{ 430, 430, 370, 400, 370}
9245 ,{ 410, 410, 370, 370, 370}
9246 ,{ 400, 370, 340, 400, 340}
9247 ,{ 370, 370, 340, 340, 340}
9248 ,{ 430, 430, 340, 400, 340}
9249 }
9250 ,{{ 410, 410, 370, 370, 370}
9251 ,{ 410, 410, 370, 370, 370}
9252 ,{ 370, 370, 340, 340, 340}
9253 ,{ 290, 290, 260, 260, 260}
9254 ,{ 370, 370, 340, 340, 340}
9255 }
9256 ,{{ 400, 370, 340, 400, 340}
9257 ,{ 370, 370, 340, 340, 340}
9258 ,{ 400, 370, 340, 400, 340}
9259 ,{ 370, 370, 340, 340, 340}
9260 ,{ 400, 370, 340, 400, 340}
9261 }
9262 ,{{ 370, 370, 340, 340, 340}
9263 ,{ 360, 360, 300, 300, 300}
9264 ,{ 370, 370, 340, 340, 340}
9265 ,{ 340, 240, 210, 340, 210}
9266 ,{ 370, 370, 340, 340, 340}
9267 }
9268 ,{{ 430, 430, 340, 400, 340}
9269 ,{ 370, 370, 340, 340, 340}
9270 ,{ 400, 370, 340, 400, 340}
9271 ,{ 370, 370, 340, 340, 340}
9272 ,{ 430, 430, 340, 340, 340}
9273 }
9274 }
9275 ,{{ 370, 370, 370, 370, 370}
9276 ,{ 370, 370, 370, 370, 370}
9277 ,{ 340, 340, 340, 340, 340}
9278 ,{ 340, 340, 340, 340, 340}
9279 ,{ 340, 340, 340, 340, 340}
9280 }
9281 ,{{ 370, 370, 370, 370, 370}
9282 ,{ 370, 370, 370, 370, 370}
9283 ,{ 340, 340, 340, 340, 340}
9284 ,{ 320, 260, 320, 260, 320}
9285 ,{ 340, 340, 340, 340, 340}
9286 }
9287 ,{{ 340, 340, 340, 340, 340}
9288 ,{ 340, 340, 340, 340, 340}
9289 ,{ 340, 340, 340, 340, 340}
9290 ,{ 340, 340, 340, 340, 340}
9291 ,{ 340, 340, 340, 340, 340}
9292 }
9293 ,{{ 360, 340, 360, 340, 360}
9294 ,{ 360, 300, 360, 300, 360}
9295 ,{ 340, 340, 340, 340, 340}
9296 ,{ 210, 210, 210, 210, 210}
9297 ,{ 340, 340, 340, 340, 340}
9298 }
9299 ,{{ 340, 340, 340, 340, 340}
9300 ,{ 340, 340, 340, 340, 340}
9301 ,{ 340, 340, 340, 340, 340}
9302 ,{ 340, 340, 340, 340, 340}
9303 ,{ 340, 340, 340, 340, 340}
9304 }
9305 }
9306 ,{{ 370, 320, 370, 340, 370}
9307 ,{ 370, 290, 370, 300, 370}
9308 ,{ 340, 320, 340, 210, 340}
9309 ,{ 340, 260, 340, 340, 340}
9310 ,{ 340, 320, 340, 340, 340}
9311 }
9312 ,{{ 370, 290, 370, 260, 370}
9313 ,{ 370, 290, 370, 240, 370}
9314 ,{ 340, 260, 340, 210, 340}
9315 ,{ 260, 180, 260, 260, 260}
9316 ,{ 340, 260, 340, 210, 340}
9317 }
9318 ,{{ 340, 320, 340, 210, 340}
9319 ,{ 340, 260, 340, 210, 340}
9320 ,{ 340, 320, 340, 210, 340}
9321 ,{ 340, 260, 340, 210, 340}
9322 ,{ 340, 320, 340, 210, 340}
9323 }
9324 ,{{ 340, 260, 340, 340, 340}
9325 ,{ 300, 220, 300, 300, 300}
9326 ,{ 340, 260, 340, 210, 340}
9327 ,{ 340, 260, 210, 340, 210}
9328 ,{ 340, 260, 340, 210, 340}
```

```
9329      }
9330 ,{{ 340,   320,   340,   340,   340}
9331 ,{ 340,   260,   340,   210,   340}
9332 ,{ 340,   320,   340,   210,   340}
9333 ,{ 340,   260,   340,   210,   340}
9334 ,{ 340,   260,   340,   340,   340}
9335     }
9336   }
9337 ,{{{ 430,   370,   370,   370,   430}
9338 ,{ 430,   370,   370,   370,   430}
9339 ,{ 340,   340,   340,   340,   340}
9340 ,{ 340,   340,   340,   340,   340}
9341 ,{ 340,   340,   340,   340,   340}
9342     }
9343 ,{{ 430,   370,   370,   370,   430}
9344 ,{ 430,   370,   370,   370,   430}
9345 ,{ 340,   340,   340,   340,   340}
9346 ,{ 320,   260,   320,   260,   260}
9347 ,{ 340,   340,   340,   340,   340}
9348     }
9349 ,{{ 340,   340,   340,   340,   340}
9350 ,{ 340,   340,   340,   340,   340}
9351 ,{ 340,   340,   340,   340,   340}
9352 ,{ 340,   340,   340,   340,   340}
9353 ,{ 340,   340,   340,   340,   340}
9354     }
9355 ,{{ 360,   340,   360,   340,   340}
9356 ,{ 360,   300,   360,   300,   300}
9357 ,{ 340,   340,   340,   340,   340}
9358 ,{ 340,   210,   210,   210,   340}
9359 ,{ 340,   340,   340,   340,   340}
9360     }
9361 ,{{ 340,   340,   340,   340,   340}
9362 ,{ 340,   340,   340,   340,   340}
9363 ,{ 340,   340,   340,   340,   340}
9364 ,{ 340,   340,   340,   340,   340}
9365 ,{ 340,   340,   340,   340,   340}
9366     }
9367   }
9368   }
9369 ,{{{ 400,   400,   400,   370,   400}
9370 ,{ 400,   370,   400,   360,   400}
9371 ,{ 370,   340,   310,   370,   310}
9372 ,{ 340,   340,   310,   310,   310}
9373 ,{ 400,   400,   310,   370,   310}
9374     }
9375 ,{{ 360,   360,   310,   360,   330}
9376 ,{ 360,   360,   270,   360,   330}
9377 ,{ 340,   340,   310,   310,   310}
9378 ,{ 230,   220,   230,   170,   230}
9379 ,{ 340,   340,   310,   310,   310}
9380     }
9381 ,{{ 370,   340,   310,   370,   310}
9382 ,{ 340,   340,   310,   310,   310}
9383 ,{ 370,   340,   310,   370,   310}
9384 ,{ 340,   340,   310,   310,   310}
9385 ,{ 370,   340,   310,   370,   310}
9386     }
9387 ,{{ 400,   370,   400,   340,   400}
9388 ,{ 400,   370,   400,   340,   400}
9389 ,{ 340,   340,   310,   310,   310}
9390 ,{ 310,   230,   180,   310,   310}
9391 ,{ 340,   340,   310,   310,   310}
9392     }
9393 ,{{ 400,   400,   310,   370,   310}
9394 ,{ 340,   340,   310,   310,   310}
9395 ,{ 370,   340,   310,   370,   310}
9396 ,{ 340,   340,   310,   310,   310}
9397 ,{ 400,   400,   310,   310,   310}
9398     }
9399   }
9400 ,{{{ 400,   400,   340,   370,   340}
9401 ,{ 370,   370,   340,   360,   340}
9402 ,{ 370,   340,   310,   370,   310}
9403 ,{ 340,   340,   310,   310,   310}
9404 ,{ 400,   400,   310,   370,   310}
9405     }
9406 ,{{ 360,   360,   310,   360,   310}
9407 ,{ 360,   360,   270,   360,   270}
9408 ,{ 340,   340,   310,   310,   310}
9409 ,{ 220,   220,   170,   170,   170}
9410 ,{ 340,   340,   310,   310,   310}
9411     }
9412 ,{{ 370,   340,   310,   370,   310}
9413 ,{ 340,   340,   310,   310,   310}
9414 ,{ 370,   340,   310,   370,   310}
9415 ,{ 340,   340,   310,   310,   310}
```

```
9416 , { 370, 340, 310, 370, 310}
9417 }
9418 , {{ 370, 370, 340, 340, 340}
9419 , { 370, 370, 340, 340, 340}
9420 , { 340, 340, 310, 310, 310}
9421 , { 310, 210, 180, 310, 180}
9422 , { 340, 340, 310, 310, 310}
9423 }
9424 , {{ 400, 400, 310, 370, 310}
9425 , { 340, 340, 310, 310, 310}
9426 , { 370, 340, 310, 370, 310}
9427 , { 340, 340, 310, 310, 310}
9428 , { 400, 400, 310, 310, 310}
9429 }
9430 }
9431 , {{{ 400, 340, 400, 340, 400}
9432 , { 400, 340, 400, 340, 400}
9433 , { 310, 310, 310, 310, 310}
9434 , { 310, 310, 310, 310, 310}
9435 , { 310, 310, 310, 310, 310}
9436 }
9437 , {{ 310, 310, 310, 310, 310}
9438 , { 270, 270, 270, 270, 270}
9439 , { 310, 310, 310, 310, 310}
9440 , { 230, 170, 230, 170, 230}
9441 , { 310, 310, 310, 310, 310}
9442 }
9443 , {{ 310, 310, 310, 310, 310}
9444 , { 310, 310, 310, 310, 310}
9445 , { 310, 310, 310, 310, 310}
9446 , { 310, 310, 310, 310, 310}
9447 , { 310, 310, 310, 310, 310}
9448 }
9449 , {{ 400, 340, 400, 340, 400}
9450 , { 400, 340, 400, 340, 400}
9451 , { 310, 310, 310, 310, 310}
9452 , { 180, 180, 180, 180, 180}
9453 , { 310, 310, 310, 310, 310}
9454 }
9455 , {{ 310, 310, 310, 310, 310}
9456 , { 310, 310, 310, 310, 310}
9457 , { 310, 310, 310, 310, 310}
9458 , { 310, 310, 310, 310, 310}
9459 , { 310, 310, 310, 310, 310}
9460 }
9461 }
9462 , {{{ 340, 290, 340, 340, 340}
9463 , { 340, 260, 340, 340, 340}
9464 , { 310, 290, 310, 180, 310}
9465 , { 310, 230, 310, 310, 310}
9466 , { 310, 290, 310, 310, 310}
9467 }
9468 , {{ 310, 230, 310, 180, 310}
9469 , { 270, 190, 270, 140, 270}
9470 , { 310, 230, 310, 180, 310}
9471 , { 170, 40, 170, 170, 170}
9472 , { 310, 230, 310, 180, 310}
9473 }
9474 , {{ 310, 290, 310, 180, 310}
9475 , { 310, 230, 310, 180, 310}
9476 , { 310, 290, 310, 180, 310}
9477 , { 310, 230, 310, 180, 310}
9478 , { 310, 290, 310, 180, 310}
9479 }
9480 , {{ 340, 260, 340, 340, 340}
9481 , { 340, 260, 340, 340, 340}
9482 , { 310, 230, 310, 180, 310}
9483 , { 310, 230, 180, 310, 180}
9484 , { 310, 230, 310, 180, 310}
9485 }
9486 , {{ 310, 290, 310, 310, 310}
9487 , { 310, 230, 310, 180, 310}
9488 , { 310, 290, 310, 180, 310}
9489 , { 310, 230, 310, 180, 310}
9490 , { 310, 230, 310, 310, 310}
9491 }
9492 }
9493 , {{{ 400, 340, 400, 340, 340}
9494 , { 400, 340, 400, 340, 340}
9495 , { 310, 310, 310, 310, 310}
9496 , { 310, 310, 310, 310, 310}
9497 , { 310, 310, 310, 310, 310}
9498 }
9499 , {{ 330, 310, 310, 310, 330}
9500 , { 330, 270, 270, 270, 330}
9501 , { 310, 310, 310, 310, 310}
9502 , { 230, 170, 230, 170, 170}
```

```
9503 , { 310, 310, 310, 310, 310}
9504 }
9505 , {{ 310, 310, 310, 310, 310}
9506 , { 310, 310, 310, 310, 310}
9507 , { 310, 310, 310, 310, 310}
9508 , { 310, 310, 310, 310, 310}
9509 , { 310, 310, 310, 310, 310}
9510 }
9511 , {{ 400, 340, 400, 340, 340}
9512 , { 400, 340, 400, 340, 340}
9513 , { 310, 310, 310, 310, 310}
9514 , { 310, 180, 180, 180, 310}
9515 , { 310, 310, 310, 310, 310}
9516 }
9517 , {{ 310, 310, 310, 310, 310}
9518 , { 310, 310, 310, 310, 310}
9519 , { 310, 310, 310, 310, 310}
9520 , { 310, 310, 310, 310, 310}
9521 , { 310, 310, 310, 310, 310}
9522 }
9523 }
9524 }
9525 , {{{{ 370, 340, 310, 350, 370}
9526 , { 370, 340, 310, 310, 370}
9527 , { 350, 320, 290, 350, 290}
9528 , { 330, 330, 290, 290, 290}
9529 , { 350, 320, 290, 350, 290}
9530 }
9531 , {{ 370, 340, 310, 310, 370}
9532 , { 370, 340, 310, 310, 370}
9533 , { 320, 320, 280, 280, 280}
9534 , { 240, 220, 240, 180, 240}
9535 , { 320, 320, 280, 280, 280}
9536 }
9537 , {{ 350, 330, 290, 350, 290}
9538 , { 330, 330, 290, 290, 290}
9539 , { 350, 320, 290, 350, 290}
9540 , { 330, 330, 290, 290, 290}
9541 , { 350, 320, 290, 350, 290}
9542 }
9543 , {{ 320, 320, 310, 280, 310}
9544 , { 310, 290, 310, 250, 310}
9545 , { 320, 320, 280, 280, 280}
9546 , { 260, 180, 130, 260, 260}
9547 , { 320, 320, 280, 280, 280}
9548 }
9549 , {{ 350, 330, 290, 350, 290}
9550 , { 330, 330, 290, 290, 290}
9551 , { 350, 320, 290, 350, 290}
9552 , { 330, 330, 290, 290, 290}
9553 , { 290, 290, 200, 200, 200}
9554 }
9555 }
9556 , {{{ 350, 340, 310, 350, 310}
9557 , { 340, 340, 310, 310, 310}
9558 , { 350, 320, 290, 350, 290}
9559 , { 330, 330, 290, 290, 290}
9560 , { 350, 320, 290, 350, 290}
9561 }
9562 , {{ 340, 340, 310, 310, 310}
9563 , { 340, 340, 310, 310, 310}
9564 , { 320, 320, 280, 280, 280}
9565 , { 220, 220, 180, 180, 180}
9566 , { 320, 320, 280, 280, 280}
9567 }
9568 , {{ 350, 330, 290, 350, 290}
9569 , { 330, 330, 290, 290, 290}
9570 , { 350, 320, 290, 350, 290}
9571 , { 330, 330, 290, 290, 290}
9572 , { 350, 320, 290, 350, 290}
9573 }
9574 , {{ 320, 320, 280, 280, 280}
9575 , { 290, 290, 250, 250, 250}
9576 , { 320, 320, 280, 280, 280}
9577 , { 260, 170, 130, 260, 130}
9578 , { 320, 320, 280, 280, 280}
9579 }
9580 , {{ 350, 330, 290, 350, 290}
9581 , { 330, 330, 290, 290, 290}
9582 , { 350, 320, 290, 350, 290}
9583 , { 330, 330, 290, 290, 290}
9584 , { 290, 290, 200, 200, 200}
9585 }
9586 }
9587 , {{{ 310, 310, 310, 310, 310}
9588 , { 310, 310, 310, 310, 310}
9589 , { 290, 290, 290, 290, 290}
```

```
9590 , { 290, 290, 290, 290, 290}
9591 , { 290, 290, 290, 290, 290}
9592 }
9593 , {{ 310, 310, 310, 310, 310}
9594 , { 310, 310, 310, 310, 310}
9595 , { 280, 280, 280, 280, 280}
9596 , { 240, 180, 240, 180, 240}
9597 , { 280, 280, 280, 280, 280}
9598 }
9599 , {{ 290, 290, 290, 290, 290}
9600 , { 290, 290, 290, 290, 290}
9601 , { 290, 290, 290, 290, 290}
9602 , { 290, 290, 290, 290, 290}
9603 , { 290, 290, 290, 290, 290}
9604 }
9605 , {{ 310, 280, 310, 280, 310}
9606 , { 310, 250, 310, 250, 310}
9607 , { 280, 280, 280, 280, 280}
9608 , { 130, 130, 130, 130, 130}
9609 , { 280, 280, 280, 280, 280}
9610 }
9611 , {{ 290, 290, 290, 290, 290}
9612 , { 290, 290, 290, 290, 290}
9613 , { 290, 290, 290, 290, 290}
9614 , { 290, 290, 290, 290, 290}
9615 , { 200, 200, 200, 200, 200}
9616 }
9617 }
9618 , {{ { 310, 270, 310, 260, 310}
9619 , { 310, 230, 310, 250, 310}
9620 , { 290, 270, 290, 160, 290}
9621 , { 290, 210, 290, 260, 290}
9622 , { 290, 270, 290, 200, 290}
9623 }
9624 , {{ 310, 230, 310, 180, 310}
9625 , { 310, 230, 310, 180, 310}
9626 , { 280, 200, 280, 150, 280}
9627 , { 180, 100, 180, 180, 180}
9628 , { 280, 200, 280, 150, 280}
9629 }
9630 , {{ { 290, 270, 290, 160, 290}
9631 , { 290, 210, 290, 160, 290}
9632 , { 290, 270, 290, 160, 290}
9633 , { 290, 210, 290, 160, 290}
9634 , { 290, 270, 290, 160, 290}
9635 }
9636 , {{ { 280, 200, 280, 260, 280}
9637 , { 250, 170, 250, 250, 250}
9638 , { 280, 200, 280, 150, 280}
9639 , { 260, 180, 130, 260, 130}
9640 , { 280, 200, 280, 150, 280}
9641 }
9642 , {{ { 290, 270, 290, 200, 290}
9643 , { 290, 210, 290, 160, 290}
9644 , { 290, 270, 290, 160, 290}
9645 , { 290, 210, 290, 160, 290}
9646 , { 200, 120, 200, 200, 200}
9647 }
9648 }
9649 , {{ { 370, 310, 310, 310, 370}
9650 , { 370, 310, 310, 310, 370}
9651 , { 290, 290, 290, 290, 290}
9652 , { 290, 290, 290, 290, 290}
9653 , { 290, 290, 290, 290, 290}
9654 }
9655 , {{ { 370, 310, 310, 310, 370}
9656 , { 370, 310, 310, 310, 370}
9657 , { 280, 280, 280, 280, 280}
9658 , { 240, 180, 240, 180, 180}
9659 , { 280, 280, 280, 280, 280}
9660 }
9661 , {{ { 290, 290, 290, 290, 290}
9662 , { 290, 290, 290, 290, 290}
9663 , { 290, 290, 290, 290, 290}
9664 , { 290, 290, 290, 290, 290}
9665 , { 290, 290, 290, 290, 290}
9666 }
9667 , {{ { 310, 280, 310, 280, 280}
9668 , { 310, 250, 310, 250, 250}
9669 , { 280, 280, 280, 280, 280}
9670 , { 260, 130, 130, 130, 260}
9671 , { 280, 280, 280, 280, 280}
9672 }
9673 , {{ { 290, 290, 290, 290, 290}
9674 , { 290, 290, 290, 290, 290}
9675 , { 290, 290, 290, 290, 290}
9676 , { 290, 290, 290, 290, 290}
```

```
9677 , { 200, 200, 200, 200, 200}
9678 }
9679 }
9680 }
9681 , {{ { 370, 340, 310, 370, 370}
9682 , { 370, 340, 310, 310, 370}
9683 , { 370, 340, 310, 370, 310}
9684 , { 340, 340, 310, 310, 310}
9685 , { 370, 340, 310, 370, 310}
9686 }
9687 , {{ 370, 340, 310, 310, 370}
9688 , { 370, 340, 310, 310, 370}
9689 , { 300, 300, 260, 260, 260}
9690 , { 260, 240, 260, 200, 260}
9691 , { 300, 300, 260, 260, 260}
9692 }
9693 , {{ 370, 340, 310, 370, 310}
9694 , { 340, 340, 310, 310, 310}
9695 , { 370, 340, 310, 370, 310}
9696 , { 340, 340, 310, 310, 310}
9697 , { 370, 340, 310, 370, 310}
9698 }
9699 , {{ 300, 300, 270, 280, 280}
9700 , { 270, 250, 270, 210, 270}
9701 , { 300, 300, 260, 260, 260}
9702 , { 280, 200, 150, 280, 280}
9703 , { 300, 300, 260, 260, 260}
9704 }
9705 , {{ 340, 340, 310, 340, 310}
9706 , { 340, 340, 310, 310, 310}
9707 , { 340, 310, 280, 340, 280}
9708 , { 340, 340, 310, 310, 310}
9709 , { 320, 320, 220, 220, 220}
9710 }
9711 }
9712 , {{ { 370, 340, 310, 370, 310}
9713 , { 340, 340, 310, 310, 310}
9714 , { 370, 340, 310, 370, 310}
9715 , { 340, 340, 310, 310, 310}
9716 , { 370, 340, 310, 370, 310}
9717 }
9718 , {{ 340, 340, 310, 310, 310}
9719 , { 340, 340, 310, 310, 310}
9720 , { 300, 300, 260, 260, 260}
9721 , { 240, 240, 200, 200, 200}
9722 , { 300, 300, 260, 260, 260}
9723 }
9724 , {{ 370, 340, 310, 370, 310}
9725 , { 340, 340, 310, 310, 310}
9726 , { 370, 340, 310, 370, 310}
9727 , { 340, 340, 310, 310, 310}
9728 , { 370, 340, 310, 370, 310}
9729 }
9730 , {{ 300, 300, 260, 280, 260}
9731 , { 250, 250, 210, 210, 210}
9732 , { 300, 300, 260, 260, 260}
9733 , { 280, 190, 150, 280, 150}
9734 , { 300, 300, 260, 260, 260}
9735 }
9736 , {{ 340, 340, 310, 340, 310}
9737 , { 340, 340, 310, 310, 310}
9738 , { 340, 310, 280, 340, 280}
9739 , { 340, 340, 310, 310, 310}
9740 , { 320, 320, 220, 220, 220}
9741 }
9742 }
9743 , {{ { 310, 310, 310, 310, 310}
9744 , { 310, 310, 310, 310, 310}
9745 , { 310, 310, 310, 310, 310}
9746 , { 310, 310, 310, 310, 310}
9747 , { 310, 310, 310, 310, 310}
9748 }
9749 , {{ { 310, 310, 310, 310, 310}
9750 , { 310, 310, 310, 310, 310}
9751 , { 260, 260, 260, 260, 260}
9752 , { 260, 200, 260, 200, 260}
9753 , { 260, 260, 260, 260, 260}
9754 }
9755 , {{ { 310, 310, 310, 310, 310}
9756 , { 310, 310, 310, 310, 310}
9757 , { 310, 310, 310, 310, 310}
9758 , { 310, 310, 310, 310, 310}
9759 , { 310, 310, 310, 310, 310}
9760 }
9761 , {{ 270, 260, 270, 260, 270}
9762 , { 270, 210, 270, 210, 270}
9763 , { 260, 260, 260, 260, 260}
```

```
9764 , { 150, 150, 150, 150, 150}
9765 , { 260, 260, 260, 260, 260}
9766 }
9767 , {{ 310, 310, 310, 310, 310}
9768 , { 310, 310, 310, 310, 310}
9769 , { 280, 280, 280, 280, 280}
9770 , { 310, 310, 310, 310, 310}
9771 , { 220, 220, 220, 220, 220}
9772 }
9773 }
9774 , {{{ 310, 290, 310, 280, 310}
9775 , { 310, 230, 310, 210, 310}
9776 , { 310, 290, 310, 180, 310}
9777 , { 310, 230, 310, 280, 310}
9778 , { 310, 290, 310, 220, 310}
9779 }
9780 , {{ 310, 230, 310, 200, 310}
9781 , { 310, 230, 310, 180, 310}
9782 , { 260, 180, 260, 130, 260}
9783 , { 200, 120, 200, 200, 200}
9784 , { 260, 180, 260, 130, 260}
9785 }
9786 , {{ 310, 290, 310, 180, 310}
9787 , { 310, 230, 310, 180, 310}
9788 , { 310, 290, 310, 180, 310}
9789 , { 310, 230, 310, 180, 310}
9790 , { 310, 290, 310, 180, 310}
9791 }
9792 , {{ 280, 200, 260, 280, 260}
9793 , { 210, 130, 210, 210, 210}
9794 , { 260, 180, 260, 130, 260}
9795 , { 280, 200, 150, 280, 150}
9796 , { 260, 180, 260, 130, 260}
9797 }
9798 , {{ 310, 260, 310, 220, 310}
9799 , { 310, 230, 310, 180, 310}
9800 , { 280, 260, 280, 150, 280}
9801 , { 310, 230, 310, 180, 310}
9802 , { 220, 140, 220, 220, 220}
9803 }
9804 }
9805 , {{{ 370, 310, 310, 310, 370}
9806 , { 370, 310, 310, 310, 370}
9807 , { 310, 310, 310, 310, 310}
9808 , { 310, 310, 310, 310, 310}
9809 , { 310, 310, 310, 310, 310}
9810 }
9811 , {{ 370, 310, 310, 310, 370}
9812 , { 370, 310, 310, 310, 370}
9813 , { 260, 260, 260, 260, 260}
9814 , { 260, 200, 260, 200, 200}
9815 , { 260, 260, 260, 260, 260}
9816 }
9817 , {{ 310, 310, 310, 310, 310}
9818 , { 310, 310, 310, 310, 310}
9819 , { 310, 310, 310, 310, 310}
9820 , { 310, 310, 310, 310, 310}
9821 , { 310, 310, 310, 310, 310}
9822 }
9823 , {{ 280, 260, 270, 260, 280}
9824 , { 270, 210, 270, 210, 210}
9825 , { 260, 260, 260, 260, 260}
9826 , { 280, 150, 150, 150, 280}
9827 , { 260, 260, 260, 260, 260}
9828 }
9829 , {{ 310, 310, 310, 310, 310}
9830 , { 310, 310, 310, 310, 310}
9831 , { 280, 280, 280, 280, 280}
9832 , { 310, 310, 310, 310, 310}
9833 , { 220, 220, 220, 220, 220}
9834 }
9835 }
9836 }
9837 , {{{{ 430, 430, 400, 400, 430}
9838 , { 430, 410, 400, 370, 430}
9839 , { 400, 370, 340, 400, 340}
9840 , { 370, 370, 340, 340, 340}
9841 , { 430, 430, 340, 400, 340}
9842 }
9843 , {{ 430, 410, 370, 370, 430}
9844 , { 430, 410, 370, 370, 430}
9845 , { 370, 370, 340, 340, 340}
9846 , { 320, 290, 320, 260, 320}
9847 , { 370, 370, 340, 340, 340}
9848 }
9849 , {{ 400, 370, 340, 400, 340}
9850 , { 370, 370, 340, 340, 340}
```

```
9851 , { 400, 370, 340, 400, 340}
9852 , { 370, 370, 340, 340, 340}
9853 , { 400, 370, 340, 400, 340}
9854 }
9855 , {{ 400, 370, 400, 340, 400}
9856 , { 400, 370, 400, 340, 400}
9857 , { 370, 370, 340, 340, 340}
9858 , { 340, 260, 210, 340, 340}
9859 , { 370, 370, 340, 340, 340}
9860 }
9861 , {{ 430, 430, 340, 400, 340}
9862 , { 370, 370, 340, 340, 340}
9863 , { 400, 370, 340, 400, 340}
9864 , { 370, 370, 340, 340, 340}
9865 , { 430, 430, 340, 340, 340}
9866 }
9867 }
9868 , {{{ 430, 430, 370, 400, 370}
9869 , { 410, 410, 370, 370, 370}
9870 , { 400, 370, 340, 400, 340}
9871 , { 370, 370, 340, 340, 340}
9872 , { 430, 430, 340, 400, 340}
9873 }
9874 , {{ 410, 410, 370, 370, 370}
9875 , { 410, 410, 370, 370, 370}
9876 , { 370, 370, 340, 340, 340}
9877 , { 290, 290, 260, 260, 260}
9878 , { 370, 370, 340, 340, 340}
9879 }
9880 , {{ 400, 370, 340, 400, 340}
9881 , { 370, 370, 340, 340, 340}
9882 , { 400, 370, 340, 400, 340}
9883 , { 370, 370, 340, 340, 340}
9884 , { 400, 370, 340, 400, 340}
9885 }
9886 , {{ 370, 370, 340, 340, 340}
9887 , { 370, 370, 340, 340, 340}
9888 , { 370, 370, 340, 340, 340}
9889 , { 340, 240, 210, 340, 210}
9890 , { 370, 370, 340, 340, 340}
9891 }
9892 , {{ 430, 430, 340, 400, 340}
9893 , { 370, 370, 340, 340, 340}
9894 , { 400, 370, 340, 400, 340}
9895 , { 370, 370, 340, 340, 340}
9896 , { 430, 430, 340, 340, 340}
9897 }
9898 }
9899 , {{{ 400, 370, 400, 370, 400}
9900 , { 400, 370, 400, 370, 400}
9901 , { 340, 340, 340, 340, 340}
9902 , { 340, 340, 340, 340, 340}
9903 , { 340, 340, 340, 340, 340}
9904 }
9905 , {{ 370, 370, 370, 370, 370}
9906 , { 370, 370, 370, 370, 370}
9907 , { 340, 340, 340, 340, 340}
9908 , { 320, 260, 320, 260, 320}
9909 , { 340, 340, 340, 340, 340}
9910 }
9911 , {{ 340, 340, 340, 340, 340}
9912 , { 340, 340, 340, 340, 340}
9913 , { 340, 340, 340, 340, 340}
9914 , { 340, 340, 340, 340, 340}
9915 , { 340, 340, 340, 340, 340}
9916 }
9917 , {{ 400, 340, 400, 340, 400}
9918 , { 400, 340, 400, 340, 400}
9919 , { 340, 340, 340, 340, 340}
9920 , { 210, 210, 210, 210, 210}
9921 , { 340, 340, 340, 340, 340}
9922 }
9923 , {{ 340, 340, 340, 340, 340}
9924 , { 340, 340, 340, 340, 340}
9925 , { 340, 340, 340, 340, 340}
9926 , { 340, 340, 340, 340, 340}
9927 , { 340, 340, 340, 340, 340}
9928 }
9929 }
9930 , {{{ 370, 320, 370, 340, 370}
9931 , { 370, 290, 370, 340, 370}
9932 , { 340, 320, 340, 210, 340}
9933 , { 340, 260, 340, 340, 340}
9934 , { 340, 320, 340, 340, 340}
9935 }
9936 , {{ 370, 290, 370, 260, 370}
9937 , { 370, 290, 370, 240, 370}
```

```

9938 , { 340, 260, 340, 210, 340}
9939 , { 260, 180, 260, 260, 260}
9940 , { 340, 260, 340, 210, 340}
9941 }
9942 , {{ 340, 320, 340, 210, 340}
9943 , { 340, 260, 340, 210, 340}
9944 , { 340, 320, 340, 210, 340}
9945 , { 340, 260, 340, 210, 340}
9946 , { 340, 320, 340, 210, 340}
9947 }
9948 , {{ 340, 260, 340, 340, 340}
9949 , { 340, 260, 340, 340, 340}
9950 , { 340, 260, 340, 210, 340}
9951 , { 340, 260, 210, 340, 210}
9952 , { 340, 260, 340, 210, 340}
9953 }
9954 , {{ 340, 320, 340, 340, 340}
9955 , { 340, 260, 340, 210, 340}
9956 , { 340, 320, 340, 210, 340}
9957 , { 340, 260, 340, 210, 340}
9958 , { 340, 260, 340, 340, 340}
9959 }
9960 }
9961 , {{{ 430, 370, 400, 370, 430}
9962 , { 430, 370, 400, 370, 430}
9963 , { 340, 340, 340, 340, 340}
9964 , { 340, 340, 340, 340, 340}
9965 , { 340, 340, 340, 340, 340}
9966 }
9967 , {{ 430, 370, 370, 370, 430}
9968 , { 430, 370, 370, 370, 430}
9969 , { 340, 340, 340, 340, 340}
9970 , { 320, 260, 320, 260, 260}
9971 , { 340, 340, 340, 340, 340}
9972 }
9973 , {{ 340, 340, 340, 340, 340}
9974 , { 340, 340, 340, 340, 340}
9975 , { 340, 340, 340, 340, 340}
9976 , { 340, 340, 340, 340, 340}
9977 , { 340, 340, 340, 340, 340}
9978 }
9979 , {{ 400, 340, 400, 340, 340}
9980 , { 400, 340, 400, 340, 340}
9981 , { 340, 340, 340, 340, 340}
9982 , { 340, 210, 210, 210, 340}
9983 , { 340, 340, 340, 340, 340}
9984 }
9985 , {{ 340, 340, 340, 340, 340}
9986 , { 340, 340, 340, 340, 340}
9987 , { 340, 340, 340, 340, 340}
9988 , { 340, 340, 340, 340, 340}
9989 , { 340, 340, 340, 340, 340}
9990 }
9991 }
9992 }
9993 }};

```

## 18.169 intl22dH.h

```

1 PUBLIC int int22_dH[NBPAIRS+1][NBPAIRS+1][5][5][5][5] =
2 {{{{{ INF, INF, INF, INF, INF
3 , { INF, INF, INF, INF, INF
4 , { INF, INF, INF, INF, INF
5 , { INF, INF, INF, INF, INF
6 , { INF, INF, INF, INF, INF
7 }
8 , {{ INF, INF, INF, INF, INF
9 , { INF, INF, INF, INF, INF
10 , { INF, INF, INF, INF, INF
11 , { INF, INF, INF, INF, INF
12 , { INF, INF, INF, INF, INF
13 }
14 , {{ INF, INF, INF, INF, INF
15 , { INF, INF, INF, INF, INF
16 , { INF, INF, INF, INF, INF
17 , { INF, INF, INF, INF, INF
18 , { INF, INF, INF, INF, INF
19 }
20 , {{ INF, INF, INF, INF, INF
21 , { INF, INF, INF, INF, INF
22 , { INF, INF, INF, INF, INF
23 , { INF, INF, INF, INF, INF
24 , { INF, INF, INF, INF, INF
25 }
26 , {{ INF, INF, INF, INF, INF

```

```
27 , { INF, INF, INF, INF, INF }
28 , { INF, INF, INF, INF, INF }
29 , { INF, INF, INF, INF, INF }
30 , { INF, INF, INF, INF, INF }
31 }
32 }
33 , { { { INF, INF, INF, INF, INF }
34 , { INF, INF, INF, INF, INF }
35 , { INF, INF, INF, INF, INF }
36 , { INF, INF, INF, INF, INF }
37 , { INF, INF, INF, INF, INF }
38 }
39 , { { { INF, INF, INF, INF, INF }
40 , { INF, INF, INF, INF, INF }
41 , { INF, INF, INF, INF, INF }
42 , { INF, INF, INF, INF, INF }
43 , { INF, INF, INF, INF, INF }
44 }
45 , { { { INF, INF, INF, INF, INF }
46 , { INF, INF, INF, INF, INF }
47 , { INF, INF, INF, INF, INF }
48 , { INF, INF, INF, INF, INF }
49 , { INF, INF, INF, INF, INF }
50 }
51 , { { { INF, INF, INF, INF, INF }
52 , { INF, INF, INF, INF, INF }
53 , { INF, INF, INF, INF, INF }
54 , { INF, INF, INF, INF, INF }
55 , { INF, INF, INF, INF, INF }
56 }
57 , { { { INF, INF, INF, INF, INF }
58 , { INF, INF, INF, INF, INF }
59 , { INF, INF, INF, INF, INF }
60 , { INF, INF, INF, INF, INF }
61 , { INF, INF, INF, INF, INF }
62 }
63 }
64 , { { { INF, INF, INF, INF, INF }
65 , { INF, INF, INF, INF, INF }
66 , { INF, INF, INF, INF, INF }
67 , { INF, INF, INF, INF, INF }
68 , { INF, INF, INF, INF, INF }
69 }
70 , { { { INF, INF, INF, INF, INF }
71 , { INF, INF, INF, INF, INF }
72 , { INF, INF, INF, INF, INF }
73 , { INF, INF, INF, INF, INF }
74 , { INF, INF, INF, INF, INF }
75 }
76 , { { { INF, INF, INF, INF, INF }
77 , { INF, INF, INF, INF, INF }
78 , { INF, INF, INF, INF, INF }
79 , { INF, INF, INF, INF, INF }
80 , { INF, INF, INF, INF, INF }
81 }
82 , { { { INF, INF, INF, INF, INF }
83 , { INF, INF, INF, INF, INF }
84 , { INF, INF, INF, INF, INF }
85 , { INF, INF, INF, INF, INF }
86 , { INF, INF, INF, INF, INF }
87 }
88 , { { { INF, INF, INF, INF, INF }
89 , { INF, INF, INF, INF, INF }
90 , { INF, INF, INF, INF, INF }
91 , { INF, INF, INF, INF, INF }
92 , { INF, INF, INF, INF, INF }
93 }
94 }
95 , { { { INF, INF, INF, INF, INF }
96 , { INF, INF, INF, INF, INF }
97 , { INF, INF, INF, INF, INF }
98 , { INF, INF, INF, INF, INF }
99 , { INF, INF, INF, INF, INF }
100 }
101 , { { { INF, INF, INF, INF, INF }
102 , { INF, INF, INF, INF, INF }
103 , { INF, INF, INF, INF, INF }
104 , { INF, INF, INF, INF, INF }
105 , { INF, INF, INF, INF, INF }
106 }
107 , { { { INF, INF, INF, INF, INF }
108 , { INF, INF, INF, INF, INF }
109 , { INF, INF, INF, INF, INF }
110 , { INF, INF, INF, INF, INF }
111 , { INF, INF, INF, INF, INF }
112 }
113 , { { { INF, INF, INF, INF, INF }
```

```
114 , { INF, INF, INF, INF, INF }
115 , { INF, INF, INF, INF, INF }
116 , { INF, INF, INF, INF, INF }
117 , { INF, INF, INF, INF, INF }
118 }
119 , {{ INF, INF, INF, INF, INF }
120 , { INF, INF, INF, INF, INF }
121 , { INF, INF, INF, INF, INF }
122 , { INF, INF, INF, INF, INF }
123 , { INF, INF, INF, INF, INF }
124 }
125 }
126 , {{{ INF, INF, INF, INF, INF }
127 , { INF, INF, INF, INF, INF }
128 , { INF, INF, INF, INF, INF }
129 , { INF, INF, INF, INF, INF }
130 , { INF, INF, INF, INF, INF }
131 }
132 , {{ INF, INF, INF, INF, INF }
133 , { INF, INF, INF, INF, INF }
134 , { INF, INF, INF, INF, INF }
135 , { INF, INF, INF, INF, INF }
136 , { INF, INF, INF, INF, INF }
137 }
138 , {{ INF, INF, INF, INF, INF }
139 , { INF, INF, INF, INF, INF }
140 , { INF, INF, INF, INF, INF }
141 , { INF, INF, INF, INF, INF }
142 , { INF, INF, INF, INF, INF }
143 }
144 , {{ INF, INF, INF, INF, INF }
145 , { INF, INF, INF, INF, INF }
146 , { INF, INF, INF, INF, INF }
147 , { INF, INF, INF, INF, INF }
148 , { INF, INF, INF, INF, INF }
149 }
150 , {{ INF, INF, INF, INF, INF }
151 , { INF, INF, INF, INF, INF }
152 , { INF, INF, INF, INF, INF }
153 , { INF, INF, INF, INF, INF }
154 , { INF, INF, INF, INF, INF }
155 }
156 }
157 }
158 , {{{ INF, INF, INF, INF, INF }
159 , { INF, INF, INF, INF, INF }
160 , { INF, INF, INF, INF, INF }
161 , { INF, INF, INF, INF, INF }
162 , { INF, INF, INF, INF, INF }
163 }
164 , {{ INF, INF, INF, INF, INF }
165 , { INF, INF, INF, INF, INF }
166 , { INF, INF, INF, INF, INF }
167 , { INF, INF, INF, INF, INF }
168 , { INF, INF, INF, INF, INF }
169 }
170 , {{ INF, INF, INF, INF, INF }
171 , { INF, INF, INF, INF, INF }
172 , { INF, INF, INF, INF, INF }
173 , { INF, INF, INF, INF, INF }
174 , { INF, INF, INF, INF, INF }
175 }
176 , {{ INF, INF, INF, INF, INF }
177 , { INF, INF, INF, INF, INF }
178 , { INF, INF, INF, INF, INF }
179 , { INF, INF, INF, INF, INF }
180 , { INF, INF, INF, INF, INF }
181 }
182 , {{ INF, INF, INF, INF, INF }
183 , { INF, INF, INF, INF, INF }
184 , { INF, INF, INF, INF, INF }
185 , { INF, INF, INF, INF, INF }
186 , { INF, INF, INF, INF, INF }
187 }
188 }
189 , {{{ INF, INF, INF, INF, INF }
190 , { INF, INF, INF, INF, INF }
191 , { INF, INF, INF, INF, INF }
192 , { INF, INF, INF, INF, INF }
193 , { INF, INF, INF, INF, INF }
194 }
195 , {{ INF, INF, INF, INF, INF }
196 , { INF, INF, INF, INF, INF }
197 , { INF, INF, INF, INF, INF }
198 , { INF, INF, INF, INF, INF }
199 , { INF, INF, INF, INF, INF }
200 }
```

```
201 , {{ INF, INF, INF, INF, INF }  
202 , { INF, INF, INF, INF, INF }  
203 , { INF, INF, INF, INF, INF }  
204 , { INF, INF, INF, INF, INF }  
205 , { INF, INF, INF, INF, INF }  
206 }  
207 , {{ INF, INF, INF, INF, INF }  
208 , { INF, INF, INF, INF, INF }  
209 , { INF, INF, INF, INF, INF }  
210 , { INF, INF, INF, INF, INF }  
211 , { INF, INF, INF, INF, INF }  
212 }  
213 , {{ INF, INF, INF, INF, INF }  
214 , { INF, INF, INF, INF, INF }  
215 , { INF, INF, INF, INF, INF }  
216 , { INF, INF, INF, INF, INF }  
217 , { INF, INF, INF, INF, INF }  
218 }  
219 }  
220 , {{{ INF, INF, INF, INF, INF }  
221 , { INF, INF, INF, INF, INF }  
222 , { INF, INF, INF, INF, INF }  
223 , { INF, INF, INF, INF, INF }  
224 , { INF, INF, INF, INF, INF }  
225 }  
226 , {{ INF, INF, INF, INF, INF }  
227 , { INF, INF, INF, INF, INF }  
228 , { INF, INF, INF, INF, INF }  
229 , { INF, INF, INF, INF, INF }  
230 , { INF, INF, INF, INF, INF }  
231 }  
232 , {{ INF, INF, INF, INF, INF }  
233 , { INF, INF, INF, INF, INF }  
234 , { INF, INF, INF, INF, INF }  
235 , { INF, INF, INF, INF, INF }  
236 , { INF, INF, INF, INF, INF }  
237 }  
238 , {{ INF, INF, INF, INF, INF }  
239 , { INF, INF, INF, INF, INF }  
240 , { INF, INF, INF, INF, INF }  
241 , { INF, INF, INF, INF, INF }  
242 , { INF, INF, INF, INF, INF }  
243 }  
244 , {{ INF, INF, INF, INF, INF }  
245 , { INF, INF, INF, INF, INF }  
246 , { INF, INF, INF, INF, INF }  
247 , { INF, INF, INF, INF, INF }  
248 , { INF, INF, INF, INF, INF }  
249 }  
250 }  
251 , {{{ INF, INF, INF, INF, INF }  
252 , { INF, INF, INF, INF, INF }  
253 , { INF, INF, INF, INF, INF }  
254 , { INF, INF, INF, INF, INF }  
255 , { INF, INF, INF, INF, INF }  
256 }  
257 , {{ INF, INF, INF, INF, INF }  
258 , { INF, INF, INF, INF, INF }  
259 , { INF, INF, INF, INF, INF }  
260 , { INF, INF, INF, INF, INF }  
261 , { INF, INF, INF, INF, INF }  
262 }  
263 , {{ INF, INF, INF, INF, INF }  
264 , { INF, INF, INF, INF, INF }  
265 , { INF, INF, INF, INF, INF }  
266 , { INF, INF, INF, INF, INF }  
267 , { INF, INF, INF, INF, INF }  
268 }  
269 , {{ INF, INF, INF, INF, INF }  
270 , { INF, INF, INF, INF, INF }  
271 , { INF, INF, INF, INF, INF }  
272 , { INF, INF, INF, INF, INF }  
273 , { INF, INF, INF, INF, INF }  
274 }  
275 , {{ INF, INF, INF, INF, INF }  
276 , { INF, INF, INF, INF, INF }  
277 , { INF, INF, INF, INF, INF }  
278 , { INF, INF, INF, INF, INF }  
279 , { INF, INF, INF, INF, INF }  
280 }  
281 }  
282 , {{{ INF, INF, INF, INF, INF }  
283 , { INF, INF, INF, INF, INF }  
284 , { INF, INF, INF, INF, INF }  
285 , { INF, INF, INF, INF, INF }  
286 , { INF, INF, INF, INF, INF }  
287 }
```

```
288 , {{ INF, INF, INF, INF, INF }
289 , { INF, INF, INF, INF, INF }
290 , { INF, INF, INF, INF, INF }
291 , { INF, INF, INF, INF, INF }
292 , { INF, INF, INF, INF, INF }
293 }
294 , {{ INF, INF, INF, INF, INF }
295 , { INF, INF, INF, INF, INF }
296 , { INF, INF, INF, INF, INF }
297 , { INF, INF, INF, INF, INF }
298 , { INF, INF, INF, INF, INF }
299 }
300 , {{ INF, INF, INF, INF, INF }
301 , { INF, INF, INF, INF, INF }
302 , { INF, INF, INF, INF, INF }
303 , { INF, INF, INF, INF, INF }
304 , { INF, INF, INF, INF, INF }
305 }
306 , {{ INF, INF, INF, INF, INF }
307 , { INF, INF, INF, INF, INF }
308 , { INF, INF, INF, INF, INF }
309 , { INF, INF, INF, INF, INF }
310 , { INF, INF, INF, INF, INF }
311 }
312 }
313 }
314 , {{{ INF, INF, INF, INF, INF }
315 , { INF, INF, INF, INF, INF }
316 , { INF, INF, INF, INF, INF }
317 , { INF, INF, INF, INF, INF }
318 , { INF, INF, INF, INF, INF }
319 }
320 , {{ INF, INF, INF, INF, INF }
321 , { INF, INF, INF, INF, INF }
322 , { INF, INF, INF, INF, INF }
323 , { INF, INF, INF, INF, INF }
324 , { INF, INF, INF, INF, INF }
325 }
326 , {{ INF, INF, INF, INF, INF }
327 , { INF, INF, INF, INF, INF }
328 , { INF, INF, INF, INF, INF }
329 , { INF, INF, INF, INF, INF }
330 , { INF, INF, INF, INF, INF }
331 }
332 , {{ INF, INF, INF, INF, INF }
333 , { INF, INF, INF, INF, INF }
334 , { INF, INF, INF, INF, INF }
335 , { INF, INF, INF, INF, INF }
336 , { INF, INF, INF, INF, INF }
337 }
338 , {{ INF, INF, INF, INF, INF }
339 , { INF, INF, INF, INF, INF }
340 , { INF, INF, INF, INF, INF }
341 , { INF, INF, INF, INF, INF }
342 , { INF, INF, INF, INF, INF }
343 }
344 }
345 , {{{ INF, INF, INF, INF, INF }
346 , { INF, INF, INF, INF, INF }
347 , { INF, INF, INF, INF, INF }
348 , { INF, INF, INF, INF, INF }
349 , { INF, INF, INF, INF, INF }
350 }
351 , {{ INF, INF, INF, INF, INF }
352 , { INF, INF, INF, INF, INF }
353 , { INF, INF, INF, INF, INF }
354 , { INF, INF, INF, INF, INF }
355 , { INF, INF, INF, INF, INF }
356 }
357 , {{ INF, INF, INF, INF, INF }
358 , { INF, INF, INF, INF, INF }
359 , { INF, INF, INF, INF, INF }
360 , { INF, INF, INF, INF, INF }
361 , { INF, INF, INF, INF, INF }
362 }
363 , {{ INF, INF, INF, INF, INF }
364 , { INF, INF, INF, INF, INF }
365 , { INF, INF, INF, INF, INF }
366 , { INF, INF, INF, INF, INF }
367 , { INF, INF, INF, INF, INF }
368 }
369 , {{ INF, INF, INF, INF, INF }
370 , { INF, INF, INF, INF, INF }
371 , { INF, INF, INF, INF, INF }
372 , { INF, INF, INF, INF, INF }
373 , { INF, INF, INF, INF, INF }
374 }
```

```
375      }
376 ,{{{, INF, INF, INF, INF, INF}
377 ,{, INF, INF, INF, INF, INF}
378 ,{, INF, INF, INF, INF, INF}
379 ,{, INF, INF, INF, INF, INF}
380 ,{, INF, INF, INF, INF, INF}
381 }
382 ,{{, INF, INF, INF, INF, INF}
383 ,{, INF, INF, INF, INF, INF}
384 ,{, INF, INF, INF, INF, INF}
385 ,{, INF, INF, INF, INF, INF}
386 ,{, INF, INF, INF, INF, INF}
387 }
388 ,{{, INF, INF, INF, INF, INF}
389 ,{, INF, INF, INF, INF, INF}
390 ,{, INF, INF, INF, INF, INF}
391 ,{, INF, INF, INF, INF, INF}
392 ,{, INF, INF, INF, INF, INF}
393 }
394 ,{{, INF, INF, INF, INF, INF}
395 ,{, INF, INF, INF, INF, INF}
396 ,{, INF, INF, INF, INF, INF}
397 ,{, INF, INF, INF, INF, INF}
398 ,{, INF, INF, INF, INF, INF}
399 }
400 ,{{, INF, INF, INF, INF, INF}
401 ,{, INF, INF, INF, INF, INF}
402 ,{, INF, INF, INF, INF, INF}
403 ,{, INF, INF, INF, INF, INF}
404 ,{, INF, INF, INF, INF, INF}
405 }
406 }
407 ,{{{, INF, INF, INF, INF, INF}
408 ,{, INF, INF, INF, INF, INF}
409 ,{, INF, INF, INF, INF, INF}
410 ,{, INF, INF, INF, INF, INF}
411 ,{, INF, INF, INF, INF, INF}
412 }
413 ,{{, INF, INF, INF, INF, INF}
414 ,{, INF, INF, INF, INF, INF}
415 ,{, INF, INF, INF, INF, INF}
416 ,{, INF, INF, INF, INF, INF}
417 ,{, INF, INF, INF, INF, INF}
418 }
419 ,{{, INF, INF, INF, INF, INF}
420 ,{, INF, INF, INF, INF, INF}
421 ,{, INF, INF, INF, INF, INF}
422 ,{, INF, INF, INF, INF, INF}
423 ,{, INF, INF, INF, INF, INF}
424 }
425 ,{{, INF, INF, INF, INF, INF}
426 ,{, INF, INF, INF, INF, INF}
427 ,{, INF, INF, INF, INF, INF}
428 ,{, INF, INF, INF, INF, INF}
429 ,{, INF, INF, INF, INF, INF}
430 }
431 ,{{, INF, INF, INF, INF, INF}
432 ,{, INF, INF, INF, INF, INF}
433 ,{, INF, INF, INF, INF, INF}
434 ,{, INF, INF, INF, INF, INF}
435 ,{, INF, INF, INF, INF, INF}
436 }
437 }
438 ,{{{, INF, INF, INF, INF, INF}
439 ,{, INF, INF, INF, INF, INF}
440 ,{, INF, INF, INF, INF, INF}
441 ,{, INF, INF, INF, INF, INF}
442 ,{, INF, INF, INF, INF, INF}
443 }
444 ,{{, INF, INF, INF, INF, INF}
445 ,{, INF, INF, INF, INF, INF}
446 ,{, INF, INF, INF, INF, INF}
447 ,{, INF, INF, INF, INF, INF}
448 ,{, INF, INF, INF, INF, INF}
449 }
450 ,{{, INF, INF, INF, INF, INF}
451 ,{, INF, INF, INF, INF, INF}
452 ,{, INF, INF, INF, INF, INF}
453 ,{, INF, INF, INF, INF, INF}
454 ,{, INF, INF, INF, INF, INF}
455 }
456 ,{{, INF, INF, INF, INF, INF}
457 ,{, INF, INF, INF, INF, INF}
458 ,{, INF, INF, INF, INF, INF}
459 ,{, INF, INF, INF, INF, INF}
460 ,{, INF, INF, INF, INF, INF}
461 }
```

```
462 , {{ INF, INF, INF, INF, INF }
463 , { INF, INF, INF, INF, INF }
464 , { INF, INF, INF, INF, INF }
465 , { INF, INF, INF, INF, INF }
466 , { INF, INF, INF, INF, INF }
467 }
468 }
469 }
470 , {{{ INF, INF, INF, INF, INF }
471 , { INF, INF, INF, INF, INF }
472 , { INF, INF, INF, INF, INF }
473 , { INF, INF, INF, INF, INF }
474 , { INF, INF, INF, INF, INF }
475 }
476 , {{ INF, INF, INF, INF, INF }
477 , { INF, INF, INF, INF, INF }
478 , { INF, INF, INF, INF, INF }
479 , { INF, INF, INF, INF, INF }
480 , { INF, INF, INF, INF, INF }
481 }
482 , {{ INF, INF, INF, INF, INF }
483 , { INF, INF, INF, INF, INF }
484 , { INF, INF, INF, INF, INF }
485 , { INF, INF, INF, INF, INF }
486 , { INF, INF, INF, INF, INF }
487 }
488 , {{ INF, INF, INF, INF, INF }
489 , { INF, INF, INF, INF, INF }
490 , { INF, INF, INF, INF, INF }
491 , { INF, INF, INF, INF, INF }
492 , { INF, INF, INF, INF, INF }
493 }
494 , {{ INF, INF, INF, INF, INF }
495 , { INF, INF, INF, INF, INF }
496 , { INF, INF, INF, INF, INF }
497 , { INF, INF, INF, INF, INF }
498 , { INF, INF, INF, INF, INF }
499 }
500 }
501 , {{{ INF, INF, INF, INF, INF }
502 , { INF, INF, INF, INF, INF }
503 , { INF, INF, INF, INF, INF }
504 , { INF, INF, INF, INF, INF }
505 , { INF, INF, INF, INF, INF }
506 }
507 , {{ INF, INF, INF, INF, INF }
508 , { INF, INF, INF, INF, INF }
509 , { INF, INF, INF, INF, INF }
510 , { INF, INF, INF, INF, INF }
511 , { INF, INF, INF, INF, INF }
512 }
513 , {{ INF, INF, INF, INF, INF }
514 , { INF, INF, INF, INF, INF }
515 , { INF, INF, INF, INF, INF }
516 , { INF, INF, INF, INF, INF }
517 , { INF, INF, INF, INF, INF }
518 }
519 , {{ INF, INF, INF, INF, INF }
520 , { INF, INF, INF, INF, INF }
521 , { INF, INF, INF, INF, INF }
522 , { INF, INF, INF, INF, INF }
523 , { INF, INF, INF, INF, INF }
524 }
525 , {{ INF, INF, INF, INF, INF }
526 , { INF, INF, INF, INF, INF }
527 , { INF, INF, INF, INF, INF }
528 , { INF, INF, INF, INF, INF }
529 , { INF, INF, INF, INF, INF }
530 }
531 }
532 , {{{ INF, INF, INF, INF, INF }
533 , { INF, INF, INF, INF, INF }
534 , { INF, INF, INF, INF, INF }
535 , { INF, INF, INF, INF, INF }
536 , { INF, INF, INF, INF, INF }
537 }
538 , {{ INF, INF, INF, INF, INF }
539 , { INF, INF, INF, INF, INF }
540 , { INF, INF, INF, INF, INF }
541 , { INF, INF, INF, INF, INF }
542 , { INF, INF, INF, INF, INF }
543 }
544 , {{ INF, INF, INF, INF, INF }
545 , { INF, INF, INF, INF, INF }
546 , { INF, INF, INF, INF, INF }
547 , { INF, INF, INF, INF, INF }
548 , { INF, INF, INF, INF, INF }
```

```
549      }
550 ,{ { INF, INF, INF, INF, INF}
551 ,{ INF, INF, INF, INF, INF}
552 ,{ INF, INF, INF, INF, INF}
553 ,{ INF, INF, INF, INF, INF}
554 ,{ INF, INF, INF, INF, INF}
555 }
556 ,{ { INF, INF, INF, INF, INF}
557 ,{ INF, INF, INF, INF, INF}
558 ,{ INF, INF, INF, INF, INF}
559 ,{ INF, INF, INF, INF, INF}
560 ,{ INF, INF, INF, INF, INF}
561 }
562 }
563 ,{ { { INF, INF, INF, INF, INF}
564 ,{ { INF, INF, INF, INF, INF}
565 ,{ { INF, INF, INF, INF, INF}
566 ,{ { INF, INF, INF, INF, INF}
567 ,{ { INF, INF, INF, INF, INF}
568 }
569 ,{ { { INF, INF, INF, INF, INF}
570 ,{ { { INF, INF, INF, INF, INF}
571 ,{ { { INF, INF, INF, INF, INF}
572 ,{ { { INF, INF, INF, INF, INF}
573 ,{ { { INF, INF, INF, INF, INF}
574 }
575 ,{ { { INF, INF, INF, INF, INF}
576 ,{ { { INF, INF, INF, INF, INF}
577 ,{ { { INF, INF, INF, INF, INF}
578 ,{ { { INF, INF, INF, INF, INF}
579 ,{ { { INF, INF, INF, INF, INF}
580 }
581 ,{ { { INF, INF, INF, INF, INF}
582 ,{ { { INF, INF, INF, INF, INF}
583 ,{ { { INF, INF, INF, INF, INF}
584 ,{ { { INF, INF, INF, INF, INF}
585 ,{ { { INF, INF, INF, INF, INF}
586 }
587 ,{ { { INF, INF, INF, INF, INF}
588 ,{ { { INF, INF, INF, INF, INF}
589 ,{ { { INF, INF, INF, INF, INF}
590 ,{ { { INF, INF, INF, INF, INF}
591 ,{ { { INF, INF, INF, INF, INF}
592 }
593 }
594 ,{ { { { INF, INF, INF, INF, INF}
595 ,{ { { { INF, INF, INF, INF, INF}
596 ,{ { { { INF, INF, INF, INF, INF}
597 ,{ { { { INF, INF, INF, INF, INF}
598 ,{ { { { INF, INF, INF, INF, INF}
599 }
600 ,{ { { { INF, INF, INF, INF, INF}
601 ,{ { { { INF, INF, INF, INF, INF}
602 ,{ { { { INF, INF, INF, INF, INF}
603 ,{ { { { INF, INF, INF, INF, INF}
604 ,{ { { { INF, INF, INF, INF, INF}
605 }
606 ,{ { { { INF, INF, INF, INF, INF}
607 ,{ { { { INF, INF, INF, INF, INF}
608 ,{ { { { INF, INF, INF, INF, INF}
609 ,{ { { { INF, INF, INF, INF, INF}
610 ,{ { { { INF, INF, INF, INF, INF}
611 }
612 ,{ { { { INF, INF, INF, INF, INF}
613 ,{ { { { INF, INF, INF, INF, INF}
614 ,{ { { { INF, INF, INF, INF, INF}
615 ,{ { { { INF, INF, INF, INF, INF}
616 ,{ { { { INF, INF, INF, INF, INF}
617 }
618 ,{ { { { INF, INF, INF, INF, INF}
619 ,{ { { { INF, INF, INF, INF, INF}
620 ,{ { { { INF, INF, INF, INF, INF}
621 ,{ { { { INF, INF, INF, INF, INF}
622 ,{ { { { INF, INF, INF, INF, INF}
623 }
624 }
625 }
626 ,{ { { { { INF, INF, INF, INF, INF}
627 ,{ { { { { INF, INF, INF, INF, INF}
628 ,{ { { { { INF, INF, INF, INF, INF}
629 ,{ { { { { INF, INF, INF, INF, INF}
630 ,{ { { { { INF, INF, INF, INF, INF}
631 }
632 ,{ { { { { INF, INF, INF, INF, INF}
633 ,{ { { { { INF, INF, INF, INF, INF}
634 ,{ { { { { INF, INF, INF, INF, INF}
635 ,{ { { { { INF, INF, INF, INF, INF}}
```

```
636 , { INF, INF, INF, INF, INF }
637 }
638 , {{ INF, INF, INF, INF, INF }
639 , { INF, INF, INF, INF, INF }
640 , { INF, INF, INF, INF, INF }
641 , { INF, INF, INF, INF, INF }
642 , { INF, INF, INF, INF, INF }
643 }
644 , {{ INF, INF, INF, INF, INF }
645 , { INF, INF, INF, INF, INF }
646 , { INF, INF, INF, INF, INF }
647 , { INF, INF, INF, INF, INF }
648 , { INF, INF, INF, INF, INF }
649 }
650 , {{ INF, INF, INF, INF, INF }
651 , { INF, INF, INF, INF, INF }
652 , { INF, INF, INF, INF, INF }
653 , { INF, INF, INF, INF, INF }
654 , { INF, INF, INF, INF, INF }
655 }
656 }
657 , {{{ INF, INF, INF, INF, INF }
658 , { INF, INF, INF, INF, INF }
659 , { INF, INF, INF, INF, INF }
660 , { INF, INF, INF, INF, INF }
661 , { INF, INF, INF, INF, INF }
662 }
663 , {{ INF, INF, INF, INF, INF }
664 , { INF, INF, INF, INF, INF }
665 , { INF, INF, INF, INF, INF }
666 , { INF, INF, INF, INF, INF }
667 , { INF, INF, INF, INF, INF }
668 }
669 , {{ INF, INF, INF, INF, INF }
670 , { INF, INF, INF, INF, INF }
671 , { INF, INF, INF, INF, INF }
672 , { INF, INF, INF, INF, INF }
673 , { INF, INF, INF, INF, INF }
674 }
675 , {{ INF, INF, INF, INF, INF }
676 , { INF, INF, INF, INF, INF }
677 , { INF, INF, INF, INF, INF }
678 , { INF, INF, INF, INF, INF }
679 , { INF, INF, INF, INF, INF }
680 }
681 , {{ INF, INF, INF, INF, INF }
682 , { INF, INF, INF, INF, INF }
683 , { INF, INF, INF, INF, INF }
684 , { INF, INF, INF, INF, INF }
685 , { INF, INF, INF, INF, INF }
686 }
687 }
688 , {{{ INF, INF, INF, INF, INF }
689 , { INF, INF, INF, INF, INF }
690 , { INF, INF, INF, INF, INF }
691 , { INF, INF, INF, INF, INF }
692 , { INF, INF, INF, INF, INF }
693 }
694 , {{ INF, INF, INF, INF, INF }
695 , { INF, INF, INF, INF, INF }
696 , { INF, INF, INF, INF, INF }
697 , { INF, INF, INF, INF, INF }
698 , { INF, INF, INF, INF, INF }
699 }
700 , {{ INF, INF, INF, INF, INF }
701 , { INF, INF, INF, INF, INF }
702 , { INF, INF, INF, INF, INF }
703 , { INF, INF, INF, INF, INF }
704 , { INF, INF, INF, INF, INF }
705 }
706 , {{ INF, INF, INF, INF, INF }
707 , { INF, INF, INF, INF, INF }
708 , { INF, INF, INF, INF, INF }
709 , { INF, INF, INF, INF, INF }
710 , { INF, INF, INF, INF, INF }
711 }
712 , {{ INF, INF, INF, INF, INF }
713 , { INF, INF, INF, INF, INF }
714 , { INF, INF, INF, INF, INF }
715 , { INF, INF, INF, INF, INF }
716 , { INF, INF, INF, INF, INF }
717 }
718 }
719 , {{{ INF, INF, INF, INF, INF }
720 , { INF, INF, INF, INF, INF }
721 , { INF, INF, INF, INF, INF }
722 , { INF, INF, INF, INF, INF }}
```

```
723 , { INF, INF, INF, INF, INF }
724 }
725 , { { INF, INF, INF, INF, INF }
726 , { INF, INF, INF, INF, INF }
727 , { INF, INF, INF, INF, INF }
728 , { INF, INF, INF, INF, INF }
729 , { INF, INF, INF, INF, INF }
730 }
731 , { { INF, INF, INF, INF, INF }
732 , { INF, INF, INF, INF, INF }
733 , { INF, INF, INF, INF, INF }
734 , { INF, INF, INF, INF, INF }
735 , { INF, INF, INF, INF, INF }
736 }
737 , { { INF, INF, INF, INF, INF }
738 , { INF, INF, INF, INF, INF }
739 , { INF, INF, INF, INF, INF }
740 , { INF, INF, INF, INF, INF }
741 , { INF, INF, INF, INF, INF }
742 }
743 , { { INF, INF, INF, INF, INF }
744 , { INF, INF, INF, INF, INF }
745 , { INF, INF, INF, INF, INF }
746 , { INF, INF, INF, INF, INF }
747 , { INF, INF, INF, INF, INF }
748 }
749 }
750 , { { { INF, INF, INF, INF, INF }
751 , { INF, INF, INF, INF, INF }
752 , { INF, INF, INF, INF, INF }
753 , { INF, INF, INF, INF, INF }
754 , { INF, INF, INF, INF, INF }
755 }
756 , { { INF, INF, INF, INF, INF }
757 , { INF, INF, INF, INF, INF }
758 , { INF, INF, INF, INF, INF }
759 , { INF, INF, INF, INF, INF }
760 , { INF, INF, INF, INF, INF }
761 }
762 , { { INF, INF, INF, INF, INF }
763 , { INF, INF, INF, INF, INF }
764 , { INF, INF, INF, INF, INF }
765 , { INF, INF, INF, INF, INF }
766 , { INF, INF, INF, INF, INF }
767 }
768 , { { INF, INF, INF, INF, INF }
769 , { INF, INF, INF, INF, INF }
770 , { INF, INF, INF, INF, INF }
771 , { INF, INF, INF, INF, INF }
772 , { INF, INF, INF, INF, INF }
773 }
774 , { { INF, INF, INF, INF, INF }
775 , { INF, INF, INF, INF, INF }
776 , { INF, INF, INF, INF, INF }
777 , { INF, INF, INF, INF, INF }
778 , { INF, INF, INF, INF, INF }
779 }
780 }
781 }
782 , { { { { INF, INF, INF, INF, INF }
783 , { INF, INF, INF, INF, INF }
784 , { INF, INF, INF, INF, INF }
785 , { INF, INF, INF, INF, INF }
786 , { INF, INF, INF, INF, INF }
787 }
788 , { { INF, INF, INF, INF, INF }
789 , { INF, INF, INF, INF, INF }
790 , { INF, INF, INF, INF, INF }
791 , { INF, INF, INF, INF, INF }
792 , { INF, INF, INF, INF, INF }
793 }
794 , { { INF, INF, INF, INF, INF }
795 , { INF, INF, INF, INF, INF }
796 , { INF, INF, INF, INF, INF }
797 , { INF, INF, INF, INF, INF }
798 , { INF, INF, INF, INF, INF }
799 }
800 , { { INF, INF, INF, INF, INF }
801 , { INF, INF, INF, INF, INF }
802 , { INF, INF, INF, INF, INF }
803 , { INF, INF, INF, INF, INF }
804 , { INF, INF, INF, INF, INF }
805 }
806 , { { INF, INF, INF, INF, INF }
807 , { INF, INF, INF, INF, INF }
808 , { INF, INF, INF, INF, INF }
809 , { INF, INF, INF, INF, INF }
```

```
810 , { INF, INF, INF, INF, INF }
811 }
812 }
813 , {{ INF, INF, INF, INF, INF }
814 , { INF, INF, INF, INF, INF }
815 , { INF, INF, INF, INF, INF }
816 , { INF, INF, INF, INF, INF }
817 , { INF, INF, INF, INF, INF }
818 }
819 , {{ INF, INF, INF, INF, INF }
820 , { INF, INF, INF, INF, INF }
821 , { INF, INF, INF, INF, INF }
822 , { INF, INF, INF, INF, INF }
823 , { INF, INF, INF, INF, INF }
824 }
825 , {{ INF, INF, INF, INF, INF }
826 , { INF, INF, INF, INF, INF }
827 , { INF, INF, INF, INF, INF }
828 , { INF, INF, INF, INF, INF }
829 , { INF, INF, INF, INF, INF }
830 }
831 , {{ INF, INF, INF, INF, INF }
832 , { INF, INF, INF, INF, INF }
833 , { INF, INF, INF, INF, INF }
834 , { INF, INF, INF, INF, INF }
835 , { INF, INF, INF, INF, INF }
836 }
837 , {{ INF, INF, INF, INF, INF }
838 , { INF, INF, INF, INF, INF }
839 , { INF, INF, INF, INF, INF }
840 , { INF, INF, INF, INF, INF }
841 , { INF, INF, INF, INF, INF }
842 }
843 }
844 , {{ INF, INF, INF, INF, INF }
845 , { INF, INF, INF, INF, INF }
846 , { INF, INF, INF, INF, INF }
847 , { INF, INF, INF, INF, INF }
848 , { INF, INF, INF, INF, INF }
849 }
850 , {{ INF, INF, INF, INF, INF }
851 , { INF, INF, INF, INF, INF }
852 , { INF, INF, INF, INF, INF }
853 , { INF, INF, INF, INF, INF }
854 , { INF, INF, INF, INF, INF }
855 }
856 , {{ INF, INF, INF, INF, INF }
857 , { INF, INF, INF, INF, INF }
858 , { INF, INF, INF, INF, INF }
859 , { INF, INF, INF, INF, INF }
860 , { INF, INF, INF, INF, INF }
861 }
862 , {{ INF, INF, INF, INF, INF }
863 , { INF, INF, INF, INF, INF }
864 , { INF, INF, INF, INF, INF }
865 , { INF, INF, INF, INF, INF }
866 , { INF, INF, INF, INF, INF }
867 }
868 , {{ INF, INF, INF, INF, INF }
869 , { INF, INF, INF, INF, INF }
870 , { INF, INF, INF, INF, INF }
871 , { INF, INF, INF, INF, INF }
872 , { INF, INF, INF, INF, INF }
873 }
874 }
875 , {{ INF, INF, INF, INF, INF }
876 , { INF, INF, INF, INF, INF }
877 , { INF, INF, INF, INF, INF }
878 , { INF, INF, INF, INF, INF }
879 , { INF, INF, INF, INF, INF }
880 }
881 , {{ INF, INF, INF, INF, INF }
882 , { INF, INF, INF, INF, INF }
883 , { INF, INF, INF, INF, INF }
884 , { INF, INF, INF, INF, INF }
885 , { INF, INF, INF, INF, INF }
886 }
887 , {{ INF, INF, INF, INF, INF }
888 , { INF, INF, INF, INF, INF }
889 , { INF, INF, INF, INF, INF }
890 , { INF, INF, INF, INF, INF }
891 , { INF, INF, INF, INF, INF }
892 }
893 , {{ INF, INF, INF, INF, INF }
894 , { INF, INF, INF, INF, INF }
895 , { INF, INF, INF, INF, INF }
896 , { INF, INF, INF, INF, INF }
```

```
897 , { INF, INF, INF, INF, INF }
898 }
899 , { { INF, INF, INF, INF, INF }
900 , { INF, INF, INF, INF, INF }
901 , { INF, INF, INF, INF, INF }
902 , { INF, INF, INF, INF, INF }
903 , { INF, INF, INF, INF, INF }
904 }
905 }
906 , { { { INF, INF, INF, INF, INF }
907 , { INF, INF, INF, INF, INF }
908 , { INF, INF, INF, INF, INF }
909 , { INF, INF, INF, INF, INF }
910 , { INF, INF, INF, INF, INF }
911 }
912 , { { { INF, INF, INF, INF, INF }
913 , { INF, INF, INF, INF, INF }
914 , { INF, INF, INF, INF, INF }
915 , { INF, INF, INF, INF, INF }
916 , { INF, INF, INF, INF, INF }
917 }
918 , { { { INF, INF, INF, INF, INF }
919 , { INF, INF, INF, INF, INF }
920 , { INF, INF, INF, INF, INF }
921 , { INF, INF, INF, INF, INF }
922 , { INF, INF, INF, INF, INF }
923 }
924 , { { { INF, INF, INF, INF, INF }
925 , { INF, INF, INF, INF, INF }
926 , { INF, INF, INF, INF, INF }
927 , { INF, INF, INF, INF, INF }
928 , { INF, INF, INF, INF, INF }
929 }
930 , { { { INF, INF, INF, INF, INF }
931 , { INF, INF, INF, INF, INF }
932 , { INF, INF, INF, INF, INF }
933 , { INF, INF, INF, INF, INF }
934 , { INF, INF, INF, INF, INF }
935 }
936 }
937 }
938 , { { { { INF, INF, INF, INF, INF }
939 , { INF, INF, INF, INF, INF }
940 , { INF, INF, INF, INF, INF }
941 , { INF, INF, INF, INF, INF }
942 , { INF, INF, INF, INF, INF }
943 }
944 , { { { { INF, INF, INF, INF, INF }
945 , { INF, INF, INF, INF, INF }
946 , { INF, INF, INF, INF, INF }
947 , { INF, INF, INF, INF, INF }
948 , { INF, INF, INF, INF, INF }
949 }
950 , { { { { INF, INF, INF, INF, INF }
951 , { INF, INF, INF, INF, INF }
952 , { INF, INF, INF, INF, INF }
953 , { INF, INF, INF, INF, INF }
954 , { INF, INF, INF, INF, INF }
955 }
956 , { { { { INF, INF, INF, INF, INF }
957 , { INF, INF, INF, INF, INF }
958 , { INF, INF, INF, INF, INF }
959 , { INF, INF, INF, INF, INF }
960 , { INF, INF, INF, INF, INF }
961 }
962 , { { { { INF, INF, INF, INF, INF }
963 , { INF, INF, INF, INF, INF }
964 , { INF, INF, INF, INF, INF }
965 , { INF, INF, INF, INF, INF }
966 , { INF, INF, INF, INF, INF }
967 }
968 }
969 , { { { { INF, INF, INF, INF, INF }
970 , { INF, INF, INF, INF, INF }
971 , { INF, INF, INF, INF, INF }
972 , { INF, INF, INF, INF, INF }
973 , { INF, INF, INF, INF, INF }
974 }
975 , { { { { INF, INF, INF, INF, INF }
976 , { INF, INF, INF, INF, INF }
977 , { INF, INF, INF, INF, INF }
978 , { INF, INF, INF, INF, INF }
979 , { INF, INF, INF, INF, INF }
980 }
981 , { { { { INF, INF, INF, INF, INF }
982 , { INF, INF, INF, INF, INF }
983 , { INF, INF, INF, INF, INF }
```

```
984 , { INF, INF, INF, INF, INF }
985 , { INF, INF, INF, INF, INF }
986 }
987 , { { INF, INF, INF, INF, INF }
988 , { INF, INF, INF, INF, INF }
989 , { INF, INF, INF, INF, INF }
990 , { INF, INF, INF, INF, INF }
991 , { INF, INF, INF, INF, INF }
992 }
993 , { { INF, INF, INF, INF, INF }
994 , { INF, INF, INF, INF, INF }
995 , { INF, INF, INF, INF, INF }
996 , { INF, INF, INF, INF, INF }
997 , { INF, INF, INF, INF, INF }
998 }
999 }
1000 , { { { INF, INF, INF, INF, INF }
1001 , { INF, INF, INF, INF, INF }
1002 , { INF, INF, INF, INF, INF }
1003 , { INF, INF, INF, INF, INF }
1004 , { INF, INF, INF, INF, INF }
1005 }
1006 , { { INF, INF, INF, INF, INF }
1007 , { INF, INF, INF, INF, INF }
1008 , { INF, INF, INF, INF, INF }
1009 , { INF, INF, INF, INF, INF }
1010 , { INF, INF, INF, INF, INF }
1011 }
1012 , { { INF, INF, INF, INF, INF }
1013 , { INF, INF, INF, INF, INF }
1014 , { INF, INF, INF, INF, INF }
1015 , { INF, INF, INF, INF, INF }
1016 , { INF, INF, INF, INF, INF }
1017 }
1018 , { { INF, INF, INF, INF, INF }
1019 , { INF, INF, INF, INF, INF }
1020 , { INF, INF, INF, INF, INF }
1021 , { INF, INF, INF, INF, INF }
1022 , { INF, INF, INF, INF, INF }
1023 }
1024 , { { INF, INF, INF, INF, INF }
1025 , { INF, INF, INF, INF, INF }
1026 , { INF, INF, INF, INF, INF }
1027 , { INF, INF, INF, INF, INF }
1028 , { INF, INF, INF, INF, INF }
1029 }
1030 }
1031 , { { { INF, INF, INF, INF, INF }
1032 , { INF, INF, INF, INF, INF }
1033 , { INF, INF, INF, INF, INF }
1034 , { INF, INF, INF, INF, INF }
1035 , { INF, INF, INF, INF, INF }
1036 }
1037 , { { INF, INF, INF, INF, INF }
1038 , { INF, INF, INF, INF, INF }
1039 , { INF, INF, INF, INF, INF }
1040 , { INF, INF, INF, INF, INF }
1041 , { INF, INF, INF, INF, INF }
1042 }
1043 , { { INF, INF, INF, INF, INF }
1044 , { INF, INF, INF, INF, INF }
1045 , { INF, INF, INF, INF, INF }
1046 , { INF, INF, INF, INF, INF }
1047 , { INF, INF, INF, INF, INF }
1048 }
1049 , { { INF, INF, INF, INF, INF }
1050 , { INF, INF, INF, INF, INF }
1051 , { INF, INF, INF, INF, INF }
1052 , { INF, INF, INF, INF, INF }
1053 , { INF, INF, INF, INF, INF }
1054 }
1055 , { { INF, INF, INF, INF, INF }
1056 , { INF, INF, INF, INF, INF }
1057 , { INF, INF, INF, INF, INF }
1058 , { INF, INF, INF, INF, INF }
1059 , { INF, INF, INF, INF, INF }
1060 }
1061 }
1062 , { { { INF, INF, INF, INF, INF }
1063 , { INF, INF, INF, INF, INF }
1064 , { INF, INF, INF, INF, INF }
1065 , { INF, INF, INF, INF, INF }
1066 , { INF, INF, INF, INF, INF }
1067 }
1068 , { { INF, INF, INF, INF, INF }
1069 , { INF, INF, INF, INF, INF }
1070 , { INF, INF, INF, INF, INF }
```

```
1071 , { INF, INF, INF, INF, INF }
1072 , { INF, INF, INF, INF, INF }
1073 }
1074 , {{ INF, INF, INF, INF, INF }
1075 , { INF, INF, INF, INF, INF }
1076 , { INF, INF, INF, INF, INF }
1077 , { INF, INF, INF, INF, INF }
1078 , { INF, INF, INF, INF, INF }
1079 }
1080 , {{ INF, INF, INF, INF, INF }
1081 , { INF, INF, INF, INF, INF }
1082 , { INF, INF, INF, INF, INF }
1083 , { INF, INF, INF, INF, INF }
1084 , { INF, INF, INF, INF, INF }
1085 }
1086 , {{ INF, INF, INF, INF, INF }
1087 , { INF, INF, INF, INF, INF }
1088 , { INF, INF, INF, INF, INF }
1089 , { INF, INF, INF, INF, INF }
1090 , { INF, INF, INF, INF, INF }
1091 }
1092 }
1093 }
1094 , {{{{ INF, INF, INF, INF, INF }
1095 , { INF, INF, INF, INF, INF }
1096 , { INF, INF, INF, INF, INF }
1097 , { INF, INF, INF, INF, INF }
1098 , { INF, INF, INF, INF, INF }
1099 }
1100 , {{ INF, INF, INF, INF, INF }
1101 , { INF, INF, INF, INF, INF }
1102 , { INF, INF, INF, INF, INF }
1103 , { INF, INF, INF, INF, INF }
1104 , { INF, INF, INF, INF, INF }
1105 }
1106 , {{ INF, INF, INF, INF, INF }
1107 , { INF, INF, INF, INF, INF }
1108 , { INF, INF, INF, INF, INF }
1109 , { INF, INF, INF, INF, INF }
1110 , { INF, INF, INF, INF, INF }
1111 }
1112 , {{ INF, INF, INF, INF, INF }
1113 , { INF, INF, INF, INF, INF }
1114 , { INF, INF, INF, INF, INF }
1115 , { INF, INF, INF, INF, INF }
1116 , { INF, INF, INF, INF, INF }
1117 }
1118 , {{ INF, INF, INF, INF, INF }
1119 , { INF, INF, INF, INF, INF }
1120 , { INF, INF, INF, INF, INF }
1121 , { INF, INF, INF, INF, INF }
1122 , { INF, INF, INF, INF, INF }
1123 }
1124 }
1125 , {{{{ INF, INF, INF, INF, INF }
1126 , { INF, INF, INF, INF, INF }
1127 , { INF, INF, INF, INF, INF }
1128 , { INF, INF, INF, INF, INF }
1129 , { INF, INF, INF, INF, INF }
1130 }
1131 , {{ INF, INF, INF, INF, INF }
1132 , { INF, INF, INF, INF, INF }
1133 , { INF, INF, INF, INF, INF }
1134 , { INF, INF, INF, INF, INF }
1135 , { INF, INF, INF, INF, INF }
1136 }
1137 , {{ INF, INF, INF, INF, INF }
1138 , { INF, INF, INF, INF, INF }
1139 , { INF, INF, INF, INF, INF }
1140 , { INF, INF, INF, INF, INF }
1141 , { INF, INF, INF, INF, INF }
1142 }
1143 , {{ INF, INF, INF, INF, INF }
1144 , { INF, INF, INF, INF, INF }
1145 , { INF, INF, INF, INF, INF }
1146 , { INF, INF, INF, INF, INF }
1147 , { INF, INF, INF, INF, INF }
1148 }
1149 , {{ INF, INF, INF, INF, INF }
1150 , { INF, INF, INF, INF, INF }
1151 , { INF, INF, INF, INF, INF }
1152 , { INF, INF, INF, INF, INF }
1153 , { INF, INF, INF, INF, INF }
1154 }
1155 }
1156 , {{{{ INF, INF, INF, INF, INF }
1157 , { INF, INF, INF, INF, INF }
```

```
1158 , { INF, INF, INF, INF, INF }
1159 , { INF, INF, INF, INF, INF }
1160 , { INF, INF, INF, INF, INF }
1161 }
1162 , {{ INF, INF, INF, INF, INF }
1163 , { INF, INF, INF, INF, INF }
1164 , { INF, INF, INF, INF, INF }
1165 , { INF, INF, INF, INF, INF }
1166 , { INF, INF, INF, INF, INF }
1167 }
1168 , {{ INF, INF, INF, INF, INF }
1169 , { INF, INF, INF, INF, INF }
1170 , { INF, INF, INF, INF, INF }
1171 , { INF, INF, INF, INF, INF }
1172 , { INF, INF, INF, INF, INF }
1173 }
1174 , {{ INF, INF, INF, INF, INF }
1175 , { INF, INF, INF, INF, INF }
1176 , { INF, INF, INF, INF, INF }
1177 , { INF, INF, INF, INF, INF }
1178 , { INF, INF, INF, INF, INF }
1179 }
1180 , {{ INF, INF, INF, INF, INF }
1181 , { INF, INF, INF, INF, INF }
1182 , { INF, INF, INF, INF, INF }
1183 , { INF, INF, INF, INF, INF }
1184 , { INF, INF, INF, INF, INF }
1185 }
1186 }
1187 , {{{ INF, INF, INF, INF, INF }
1188 , { INF, INF, INF, INF, INF }
1189 , { INF, INF, INF, INF, INF }
1190 , { INF, INF, INF, INF, INF }
1191 , { INF, INF, INF, INF, INF }
1192 }
1193 , {{ INF, INF, INF, INF, INF }
1194 , { INF, INF, INF, INF, INF }
1195 , { INF, INF, INF, INF, INF }
1196 , { INF, INF, INF, INF, INF }
1197 , { INF, INF, INF, INF, INF }
1198 }
1199 , {{ INF, INF, INF, INF, INF }
1200 , { INF, INF, INF, INF, INF }
1201 , { INF, INF, INF, INF, INF }
1202 , { INF, INF, INF, INF, INF }
1203 , { INF, INF, INF, INF, INF }
1204 }
1205 , {{ INF, INF, INF, INF, INF }
1206 , { INF, INF, INF, INF, INF }
1207 , { INF, INF, INF, INF, INF }
1208 , { INF, INF, INF, INF, INF }
1209 , { INF, INF, INF, INF, INF }
1210 }
1211 , {{ INF, INF, INF, INF, INF }
1212 , { INF, INF, INF, INF, INF }
1213 , { INF, INF, INF, INF, INF }
1214 , { INF, INF, INF, INF, INF }
1215 , { INF, INF, INF, INF, INF }
1216 }
1217 }
1218 , {{{ INF, INF, INF, INF, INF }
1219 , { INF, INF, INF, INF, INF }
1220 , { INF, INF, INF, INF, INF }
1221 , { INF, INF, INF, INF, INF }
1222 , { INF, INF, INF, INF, INF }
1223 }
1224 , {{ INF, INF, INF, INF, INF }
1225 , { INF, INF, INF, INF, INF }
1226 , { INF, INF, INF, INF, INF }
1227 , { INF, INF, INF, INF, INF }
1228 , { INF, INF, INF, INF, INF }
1229 }
1230 , {{ INF, INF, INF, INF, INF }
1231 , { INF, INF, INF, INF, INF }
1232 , { INF, INF, INF, INF, INF }
1233 , { INF, INF, INF, INF, INF }
1234 , { INF, INF, INF, INF, INF }
1235 }
1236 , {{ INF, INF, INF, INF, INF }
1237 , { INF, INF, INF, INF, INF }
1238 , { INF, INF, INF, INF, INF }
1239 , { INF, INF, INF, INF, INF }
1240 , { INF, INF, INF, INF, INF }
1241 }
1242 , {{ INF, INF, INF, INF, INF }
1243 , { INF, INF, INF, INF, INF }
1244 , { INF, INF, INF, INF, INF }
```

```
1245 , { INF, INF, INF, INF, INF }
1246 , { INF, INF, INF, INF, INF }
1247 }
1248 }
1249 }
1250 }
1251 , { { { { { INF, INF, INF, INF, INF }
1252 , { INF, INF, INF, INF, INF }
1253 , { INF, INF, INF, INF, INF }
1254 , { INF, INF, INF, INF, INF }
1255 , { INF, INF, INF, INF, INF }
1256 }
1257 , { { INF, INF, INF, INF, INF }
1258 , { INF, INF, INF, INF, INF }
1259 , { INF, INF, INF, INF, INF }
1260 , { INF, INF, INF, INF, INF }
1261 , { INF, INF, INF, INF, INF }
1262 }
1263 , { { INF, INF, INF, INF, INF }
1264 , { INF, INF, INF, INF, INF }
1265 , { INF, INF, INF, INF, INF }
1266 , { INF, INF, INF, INF, INF }
1267 , { INF, INF, INF, INF, INF }
1268 }
1269 , { { INF, INF, INF, INF, INF }
1270 , { INF, INF, INF, INF, INF }
1271 , { INF, INF, INF, INF, INF }
1272 , { INF, INF, INF, INF, INF }
1273 , { INF, INF, INF, INF, INF }
1274 }
1275 , { { INF, INF, INF, INF, INF }
1276 , { INF, INF, INF, INF, INF }
1277 , { INF, INF, INF, INF, INF }
1278 , { INF, INF, INF, INF, INF }
1279 , { INF, INF, INF, INF, INF }
1280 }
1281 }
1282 , { { { INF, INF, INF, INF, INF }
1283 , { INF, INF, INF, INF, INF }
1284 , { INF, INF, INF, INF, INF }
1285 , { INF, INF, INF, INF, INF }
1286 , { INF, INF, INF, INF, INF }
1287 }
1288 , { { INF, INF, INF, INF, INF }
1289 , { INF, INF, INF, INF, INF }
1290 , { INF, INF, INF, INF, INF }
1291 , { INF, INF, INF, INF, INF }
1292 , { INF, INF, INF, INF, INF }
1293 }
1294 , { { INF, INF, INF, INF, INF }
1295 , { INF, INF, INF, INF, INF }
1296 , { INF, INF, INF, INF, INF }
1297 , { INF, INF, INF, INF, INF }
1298 , { INF, INF, INF, INF, INF }
1299 }
1300 , { { INF, INF, INF, INF, INF }
1301 , { INF, INF, INF, INF, INF }
1302 , { INF, INF, INF, INF, INF }
1303 , { INF, INF, INF, INF, INF }
1304 , { INF, INF, INF, INF, INF }
1305 }
1306 , { { INF, INF, INF, INF, INF }
1307 , { INF, INF, INF, INF, INF }
1308 , { INF, INF, INF, INF, INF }
1309 , { INF, INF, INF, INF, INF }
1310 , { INF, INF, INF, INF, INF }
1311 }
1312 }
1313 , { { { INF, INF, INF, INF, INF }
1314 , { INF, INF, INF, INF, INF }
1315 , { INF, INF, INF, INF, INF }
1316 , { INF, INF, INF, INF, INF }
1317 , { INF, INF, INF, INF, INF }
1318 }
1319 , { { INF, INF, INF, INF, INF }
1320 , { INF, INF, INF, INF, INF }
1321 , { INF, INF, INF, INF, INF }
1322 , { INF, INF, INF, INF, INF }
1323 , { INF, INF, INF, INF, INF }
1324 }
1325 , { { INF, INF, INF, INF, INF }
1326 , { INF, INF, INF, INF, INF }
1327 , { INF, INF, INF, INF, INF }
1328 , { INF, INF, INF, INF, INF }
1329 , { INF, INF, INF, INF, INF }
1330 }
1331 , { { INF, INF, INF, INF, INF }
```

```
1332 , { INF, INF, INF, INF, INF }
1333 , { INF, INF, INF, INF, INF }
1334 , { INF, INF, INF, INF, INF }
1335 , { INF, INF, INF, INF, INF }
1336 }
1337 , {{ INF, INF, INF, INF, INF }
1338 , { INF, INF, INF, INF, INF }
1339 , { INF, INF, INF, INF, INF }
1340 , { INF, INF, INF, INF, INF }
1341 , { INF, INF, INF, INF, INF }
1342 }
1343 }
1344 , {{{ INF, INF, INF, INF, INF }
1345 , { INF, INF, INF, INF, INF }
1346 , { INF, INF, INF, INF, INF }
1347 , { INF, INF, INF, INF, INF }
1348 , { INF, INF, INF, INF, INF }
1349 }
1350 , {{ INF, INF, INF, INF, INF }
1351 , { INF, INF, INF, INF, INF }
1352 , { INF, INF, INF, INF, INF }
1353 , { INF, INF, INF, INF, INF }
1354 , { INF, INF, INF, INF, INF }
1355 }
1356 , {{ INF, INF, INF, INF, INF }
1357 , { INF, INF, INF, INF, INF }
1358 , { INF, INF, INF, INF, INF }
1359 , { INF, INF, INF, INF, INF }
1360 , { INF, INF, INF, INF, INF }
1361 }
1362 , {{ INF, INF, INF, INF, INF }
1363 , { INF, INF, INF, INF, INF }
1364 , { INF, INF, INF, INF, INF }
1365 , { INF, INF, INF, INF, INF }
1366 , { INF, INF, INF, INF, INF }
1367 }
1368 , {{ INF, INF, INF, INF, INF }
1369 , { INF, INF, INF, INF, INF }
1370 , { INF, INF, INF, INF, INF }
1371 , { INF, INF, INF, INF, INF }
1372 , { INF, INF, INF, INF, INF }
1373 }
1374 }
1375 , {{{ INF, INF, INF, INF, INF }
1376 , { INF, INF, INF, INF, INF }
1377 , { INF, INF, INF, INF, INF }
1378 , { INF, INF, INF, INF, INF }
1379 , { INF, INF, INF, INF, INF }
1380 }
1381 , {{ INF, INF, INF, INF, INF }
1382 , { INF, INF, INF, INF, INF }
1383 , { INF, INF, INF, INF, INF }
1384 , { INF, INF, INF, INF, INF }
1385 , { INF, INF, INF, INF, INF }
1386 }
1387 , {{ INF, INF, INF, INF, INF }
1388 , { INF, INF, INF, INF, INF }
1389 , { INF, INF, INF, INF, INF }
1390 , { INF, INF, INF, INF, INF }
1391 , { INF, INF, INF, INF, INF }
1392 }
1393 , {{ INF, INF, INF, INF, INF }
1394 , { INF, INF, INF, INF, INF }
1395 , { INF, INF, INF, INF, INF }
1396 , { INF, INF, INF, INF, INF }
1397 , { INF, INF, INF, INF, INF }
1398 }
1399 , {{ INF, INF, INF, INF, INF }
1400 , { INF, INF, INF, INF, INF }
1401 , { INF, INF, INF, INF, INF }
1402 , { INF, INF, INF, INF, INF }
1403 , { INF, INF, INF, INF, INF }
1404 }
1405 }
1406 }
1407 , {{{ 80, -120, 30, 80, 80 }
1408 , { 30, -310, -170, 30, -110 }
1409 , { 80, -230, -110, 80, -60 }
1410 , { 80, -120, 30, 30, 80 }
1411 , { -30, -340, -220, -30, -170 }
1412 }
1413 , {{ -120, -460, -290, -120, -230 }
1414 , { -120, -460, -310, -120, -260 }
1415 , { -430, -770, -620, -430, -570 }
1416 , { -230, -670, -290, -980, -230 }
1417 , { -430, -770, -620, -430, -570 }
1418 }
```

```
1419 , {{ 30, -290, -170, 30, -110} }
1420 , { 30, -310, -170, 30, -110}
1421 , { 20, -290, -170, 20, -120}
1422 , { 30, -310, -170, 30, -110}
1423 , { -30, -340, -220, -30, -170}
1424 }
1425 , {{ 80, -120, 30, -430, 80}
1426 , { -520, -960, -580, -1270, -520}
1427 , { -430, -770, -620, -430, -570}
1428 , { 80, -120, 30, -430, 80}
1429 , { -430, -770, -620, -430, -570}
1430 }
1431 , {{ 80, -230, -110, 80, -60}
1432 , { 30, -310, -170, 30, -110}
1433 , { 80, -230, -110, 80, -60}
1434 , { 30, -310, -170, 30, -110}
1435 , { -860, -860, -960, -1410, -900}
1436 }
1437 }
1438 , {{ { 30, -120, 30, -520, 30}
1439 , { -170, -310, -170, -810, -170}
1440 , { -110, -260, -110, -520, -110}
1441 , { 30, -120, 30, -810, 30}
1442 , { -220, -370, -220, -630, -220}
1443 }
1444 , {{ -310, -460, -310, -960, -310}
1445 , { -310, -460, -310, -960, -310}
1446 , { -620, -770, -620, -1270, -620}
1447 , { -530, -670, -530, -1170, -530}
1448 , { -620, -770, -620, -1270, -620}
1449 }
1450 , {{ -170, -310, -170, -580, -170}
1451 , { -170, -310, -170, -810, -170}
1452 , { -170, -320, -170, -580, -170}
1453 , { -170, -310, -170, -810, -170}
1454 , { -220, -370, -220, -630, -220}
1455 }
1456 , {{ { 30, -120, 30, -1270, 30}
1457 , { -810, -960, -810, -1460, -810}
1458 , { -620, -770, -620, -1270, -620}
1459 , { 30, -120, 30, -1870, 30}
1460 , { -620, -770, -620, -1270, -620}
1461 }
1462 , {{ -110, -260, -110, -520, -110}
1463 , { -170, -310, -170, -810, -170}
1464 , { -110, -260, -110, -520, -110}
1465 , { -170, -310, -170, -810, -170}
1466 , { -860, -860, -960, -1600, -960}
1467 }
1468 }
1469 , {{ { 80, -430, 20, -430, 80}
1470 , { -110, -620, -170, -620, -110}
1471 , { -60, -570, -120, -570, -60}
1472 , { 80, -430, 20, -430, 80}
1473 , { -170, -680, -230, -680, -170}
1474 }
1475 , {{ -230, -770, -290, -770, -230}
1476 , { -260, -770, -320, -770, -260}
1477 , { -570, -1080, -630, -1080, -570}
1478 , { -230, -980, -290, -980, -230}
1479 , { -570, -1080, -630, -1080, -570}
1480 }
1481 , {{ -110, -620, -170, -620, -110}
1482 , { -110, -620, -170, -620, -110}
1483 , { -120, -630, -180, -630, -120}
1484 , { -110, -620, -170, -620, -110}
1485 , { -170, -680, -230, -680, -170}
1486 }
1487 , {{ { 80, -430, 20, -430, 80}
1488 , { -520, -1270, -580, -1270, -520}
1489 , { -570, -1080, -630, -1080, -570}
1490 , { 80, -430, 20, -430, 80}
1491 , { -570, -1080, -630, -1080, -570}
1492 }
1493 , {{ -60, -570, -120, -570, -60}
1494 , { -110, -620, -170, -620, -110}
1495 , { -60, -570, -120, -570, -60}
1496 , { -110, -620, -170, -620, -110}
1497 , { -900, -1410, -960, -1410, -900}
1498 }
1499 }
1500 , {{ { 80, -230, 30, 80, 30}
1501 , { 30, -530, -170, 30, -170}
1502 , { 80, -230, -110, 80, -110}
1503 , { 30, -530, 30, 30, 30}
1504 , { -30, -340, -220, -30, -220}
1505 }
```

```

1506 ,{{ -120, -670, -310, -120, -310}
1507 ,{ -120, -670, -310, -120, -310}
1508 ,{ -430, -980, -620, -430, -620}
1509 ,{ -530, -890, -530, -1580, -530}
1510 ,{ -430, -980, -620, -430, -620}
1511 }
1512 ,{{ 30, -290, -170, 30, -170}
1513 ,{ 30, -530, -170, 30, -170}
1514 ,{ 20, -290, -170, 20, -170}
1515 ,{ 30, -530, -170, 30, -170}
1516 ,{ -30, -340, -220, -30, -220}
1517 }
1518 ,{{ 30, -980, 30, -430, 30}
1519 ,{ -810, -1170, -810, -1870, -810}
1520 ,{ -430, -980, -620, -430, -620}
1521 ,{ 30, -1580, 30, -2280, 30}
1522 ,{ -430, -980, -620, -430, -620}
1523 }
1524 ,{{ 80, -230, -110, 80, -110}
1525 ,{ 30, -530, -170, 30, -170}
1526 ,{ 80, -230, -110, 80, -110}
1527 ,{ 30, -530, -170, 30, -170}
1528 ,{ -960, -1320, -960, -2010, -960}
1529 }
1530 }
1531 ,{{ { -30, -430, -30, -430, -860}
1532 ,{ -220, -620, -220, -620, -860}
1533 ,{ -170, -570, -170, -570, -900}
1534 ,{ -30, -430, -30, -430, -960}
1535 ,{ -280, -680, -280, -680, -1010}
1536 }
1537 ,{{ { -340, -770, -340, -770, -860}
1538 ,{ -370, -770, -370, -770, -860}
1539 ,{ -680, -1080, -680, -1080, -1410}
1540 ,{ -340, -980, -340, -980, -1320}
1541 ,{ -680, -1080, -680, -1080, -1410}
1542 }
1543 ,{{ { -220, -620, -220, -620, -960}
1544 ,{ -220, -620, -220, -620, -960}
1545 ,{ -230, -630, -230, -630, -960}
1546 ,{ -220, -620, -220, -620, -960}
1547 ,{ -280, -680, -280, -680, -1010}
1548 }
1549 ,{{ { -30, -430, -30, -430, -1410}
1550 ,{ -630, -1270, -630, -1270, -1600}
1551 ,{ -680, -1080, -680, -1080, -1410}
1552 ,{ -30, -430, -30, -430, -2010}
1553 ,{ -680, -1080, -680, -1080, -1410}
1554 }
1555 ,{{ { -170, -570, -170, -570, -900}
1556 ,{ -220, -620, -220, -620, -960}
1557 ,{ -170, -570, -170, -570, -900}
1558 ,{ -220, -620, -220, -620, -960}
1559 ,{ -1010, -1410, -1010, -1410, -1750}
1560 }
1561 }
1562 }
1563 ,{{ {{ 540, 180, 30, 540, 180}
1564 ,{ 10, -580, -150, 10, -90}
1565 ,{ 540, -350, -600, 540, -540}
1566 ,{ 180, 180, 30, -320, 180}
1567 ,{ -90, -740, -90, -260, -540}
1568 }
1569 ,{{ { -90, -350, -150, -100, -90}
1570 ,{ -90, -580, -150, -200, -90}
1571 ,{ -100, -350, -600, -100, -540}
1572 ,{ -630, -1790, -630, -1790, -1040}
1573 ,{ -400, -740, -600, -400, -540}
1574 }
1575 ,{{ { 540, -660, -510, 540, -400}
1576 ,{ 10, -660, -510, 10, -400}
1577 ,{ 540, -940, -820, 540, -760}
1578 ,{ -320, -660, -510, -320, -460}
1579 ,{ -260, -940, -820, -260, -550}
1580 }
1581 ,{{ { 180, 180, 30, -400, 180}
1582 ,{ -500, -1070, -500, -1080, -570}
1583 ,{ -400, -740, -600, -400, -540}
1584 ,{ 180, 180, 30, -430, 180}
1585 ,{ -400, -740, -600, -400, -540}
1586 }
1587 ,{{ { -90, -660, -90, -210, -460}
1588 ,{ -320, -660, -510, -320, -460}
1589 ,{ -210, -1250, -1130, -210, -1070}
1590 ,{ -320, -660, -510, -320, -460}
1591 ,{ -90, -830, -90, -810, -800}
1592 }

```

```

1593 }
1594 ,{{ 540, 180, -90, 540, 30}
1595 ,{ 10, -580, -220, 10, -150}
1596 ,{ 540, -740, -600, 540, -600}
1597 ,{ 180, 180, -390, -1160, 30}
1598 ,{ -90, -740, -90, -810, -600}
1599 }
1600 ,{{ -100, -580, -220, -100, -150}
1601 ,{ -150, -580, -220, -970, -150}
1602 ,{ -100, -740, -600, -100, -600}
1603 ,{ -1340, -2010, -1650, -1980, -1340}
1604 ,{ -600, -740, -600, -1240, -600}
1605 }
1606 ,{{ 540, -660, -510, 540, -510}
1607 ,{ 10, -660, -1150, 10, -510}
1608 ,{ 540, -960, -820, 540, -820}
1609 ,{ -510, -660, -510, -1160, -510}
1610 ,{ -820, -960, -820, -1220, -820}
1611 }
1612 ,{{ 180, 180, -390, -1240, 30}
1613 ,{ -860, -1340, -860, -2450, -860}
1614 ,{ -600, -740, -600, -1240, -600}
1615 ,{ 180, 180, -390, -1870, 30}
1616 ,{ -600, -740, -600, -1240, -600}
1617 }
1618 ,{{ -90, -660, -90, -810, -510}
1619 ,{ -510, -660, -510, -1160, -510}
1620 ,{ -1130, -1270, -1130, -1530, -1130}
1621 ,{ -510, -660, -510, -1160, -510}
1622 ,{ -90, -1240, -90, -810, -800}
1623 }
1624 }
1625 ,{{ 180, -430, 20, -430, 180}
1626 ,{ -90, -600, -500, -600, -90}
1627 ,{ -540, -1050, -600, -1050, -540}
1628 ,{ 180, -430, 20, -430, 180}
1629 ,{ -540, -830, -600, -1050, -540}
1630 }
1631 ,{{ -90, -600, -600, -600, -90}
1632 ,{ -90, -600, -1070, -600, -90}
1633 ,{ -540, -1050, -600, -1050, -540}
1634 ,{ -630, -1790, -630, -1790, -1040}
1635 ,{ -540, -1050, -600, -1050, -540}
1636 }
1637 ,{{ -460, -970, -520, -970, -460}
1638 ,{ -460, -970, -750, -970, -460}
1639 ,{ -760, -1270, -820, -1270, -760}
1640 ,{ -460, -970, -520, -970, -460}
1641 ,{ -550, -1270, -820, -1270, -550}
1642 }
1643 ,{{ 180, -430, 20, -430, 180}
1644 ,{ -500, -1070, -500, -1320, -570}
1645 ,{ -540, -1050, -600, -1050, -540}
1646 ,{ 180, -430, 20, -430, 180}
1647 ,{ -540, -1050, -600, -1050, -540}
1648 }
1649 ,{{ -460, -830, -520, -970, -460}
1650 ,{ -460, -970, -520, -970, -460}
1651 ,{ -1070, -1580, -1130, -1580, -1070}
1652 ,{ -460, -970, -520, -970, -460}
1653 ,{ -830, -830, -1710, -1260, -1460}
1654 }
1655 }
1656 ,{{ 30, -350, 30, -200, 30}
1657 ,{ -150, -870, -150, -200, -150}
1658 ,{ -210, -350, -600, -210, -600}
1659 ,{ 30, -870, 30, -320, 30}
1660 ,{ -260, -940, -600, -260, -600}
1661 }
1662 ,{{ -150, -350, -150, -200, -150}
1663 ,{ -150, -1600, -150, -200, -150}
1664 ,{ -350, -350, -600, -440, -600}
1665 ,{ -1340, -3070, -1340, -2390, -1340}
1666 ,{ -400, -960, -600, -400, -600}
1667 }
1668 ,{{ -260, -870, -510, -260, -510}
1669 ,{ -320, -1110, -510, -320, -510}
1670 ,{ -620, -940, -820, -620, -820}
1671 ,{ -320, -870, -510, -320, -510}
1672 ,{ -260, -940, -820, -260, -820}
1673 }
1674 ,{{ 30, -960, 30, -400, 30}
1675 ,{ -860, -1880, -860, -1080, -860}
1676 ,{ -400, -960, -600, -400, -600}
1677 ,{ 30, -1370, 30, -2280, 30}
1678 ,{ -400, -960, -600, -400, -600}
1679 }

```

```

1680 ,{{ -210, -870, -510, -210, -510}
1681 ,{ -320, -870, -510, -320, -510}
1682 ,{ -210, -1250, -1130, -210, -1130}
1683 ,{ -320, -870, -510, -320, -510}
1684 ,{ -800, -1360, -800, -1550, -800}
1685 }
1686 }
1687 ,{{ -200, -430, -200, -430, -230}
1688 ,{ -200, -600, -200, -600, -400}
1689 ,{ -650, -1050, -650, -1050, -1390}
1690 ,{ -230, -430, -570, -430, -230}
1691 ,{ -650, -1050, -650, -1050, -1390}
1692 }
1693 ,{{ -200, -600, -200, -600, -1390}
1694 ,{ -200, -600, -200, -600, -1490}
1695 ,{ -650, -1050, -650, -1050, -1390}
1696 ,{ -1150, -1790, -1150, -1790, -1520}
1697 ,{ -650, -1050, -650, -1050, -1390}
1698 }
1699 ,{{ -400, -970, -570, -970, -400}
1700 ,{ -400, -970, -570, -970, -400}
1701 ,{ -870, -1270, -870, -1270, -1610}
1702 ,{ -570, -970, -570, -970, -1300}
1703 ,{ -870, -1270, -870, -1270, -1610}
1704 }
1705 ,{{ -230, -430, -650, -430, -230}
1706 ,{ -1300, -1320, -1750, -1320, -1300}
1707 ,{ -650, -1050, -650, -1050, -1390}
1708 ,{ -230, -430, -880, -430, -230}
1709 ,{ -650, -1050, -650, -1050, -1390}
1710 }
1711 ,{{ -570, -970, -570, -970, -1300}
1712 ,{ -570, -970, -570, -970, -1300}
1713 ,{ -1180, -1580, -1180, -1580, -1920}
1714 ,{ -570, -970, -570, -970, -1300}
1715 ,{ -860, -1260, -860, -1260, -2350}
1716 }
1717 }
1718 }
1719 ,{{{{ 240, 40, 190, -270, 240}
1720 ,{ -590, -1030, -650, -870, -590}
1721 ,{ -870, -1180, -1060, -870, -1010}
1722 ,{ 240, 40, 190, -270, 240}
1723 ,{ -870, -970, -1060, -870, -1010}
1724 }
1725 ,{{ -780, -1210, -840, -870, -780}
1726 ,{ -1050, -1370, -1240, -1050, -1190}
1727 ,{ -870, -1210, -1060, -870, -1010}
1728 ,{ -780, -1220, -840, -1530, -780}
1729 ,{ -870, -1210, -1060, -870, -1010}
1730 }
1731 ,{{ -870, -1180, -1060, -870, -1010}
1732 ,{ -870, -1210, -1060, -870, -1010}
1733 ,{ -870, -1180, -1060, -870, -1010}
1734 ,{ -870, -1210, -1060, -870, -1010}
1735 ,{ -870, -1180, -1060, -870, -1010}
1736 }
1737 ,{{ 240, 40, 190, -270, 240}
1738 ,{ -590, -1030, -650, -1340, -590}
1739 ,{ -870, -1210, -1060, -870, -1010}
1740 ,{ 240, 40, 190, -270, 240}
1741 ,{ -870, -1210, -1060, -870, -1010}
1742 }
1743 ,{{ -870, -970, -1060, -870, -1010}
1744 ,{ -870, -1210, -1060, -870, -1010}
1745 ,{ -870, -1180, -1060, -870, -1010}
1746 ,{ -870, -1210, -1060, -870, -1010}
1747 ,{ -970, -970, -1060, -1520, -1010}
1748 }
1749 }
1750 ,{{{ 190, 40, 190, -1470, 190}
1751 ,{ -890, -1030, -890, -1530, -890}
1752 ,{ -1060, -1210, -1060, -1470, -1060}
1753 ,{ 190, 40, 190, -1710, 190}
1754 ,{ -970, -970, -1060, -1470, -1060}
1755 }
1756 ,{{ -1060, -1210, -1060, -1710, -1060}
1757 ,{ -1240, -1370, -1240, -1890, -1240}
1758 ,{ -1060, -1210, -1060, -1710, -1060}
1759 ,{ -1080, -1220, -1080, -1720, -1080}
1760 ,{ -1060, -1210, -1060, -1710, -1060}
1761 }
1762 ,{{ -1060, -1210, -1060, -1470, -1060}
1763 ,{ -1060, -1210, -1060, -1710, -1060}
1764 ,{ -1060, -1210, -1060, -1470, -1060}
1765 ,{ -1060, -1210, -1060, -1710, -1060}
1766 ,{ -1060, -1210, -1060, -1470, -1060}

```

```

1767      }
1768 ,{{ 190,    40,   190, -1530,   190}
1769 ,{ -890, -1030, -890, -1530, -890}
1770 ,{ -1060, -1210, -1060, -1710, -1060}
1771 ,{   190,    40,   190, -1710,   190}
1772 ,{ -1060, -1210, -1060, -1710, -1060}
1773 }
1774 ,{{ -970, -970, -1060, -1470, -1060}
1775 ,{ -1060, -1210, -1060, -1710, -1060}
1776 ,{ -1060, -1210, -1060, -1470, -1060}
1777 ,{ -1060, -1210, -1060, -1710, -1060}
1778 ,{ -970, -970, -1060, -1710, -1060}
1779 }
1780 }
1781 ,{{{{ 240, -270,   180, -270,   240}
1782 ,{ -590, -1340, -650, -1340, -590}
1783 ,{ -1010, -1520, -1070, -1520, -1010}
1784 ,{   240, -270,   180, -270,   240}
1785 ,{ -1010, -1520, -1070, -1520, -1010}
1786 }
1787 ,{{{{ -780, -1520, -840, -1520, -780}
1788 ,{ -1190, -1700, -1250, -1700, -1190}
1789 ,{ -1010, -1520, -1070, -1520, -1010}
1790 ,{   -780, -1530, -840, -1530, -780}
1791 ,{ -1010, -1520, -1070, -1520, -1010}
1792 }
1793 ,{{{{ -1010, -1520, -1070, -1520, -1010}
1794 ,{ -1010, -1520, -1070, -1520, -1010}
1795 ,{ -1010, -1520, -1070, -1520, -1010}
1796 ,{ -1010, -1520, -1070, -1520, -1010}
1797 ,{ -1010, -1520, -1070, -1520, -1010}
1798 }
1799 ,{{{{ 240, -270,   180, -270,   240}
1800 ,{ -590, -1340, -650, -1340, -590}
1801 ,{ -1010, -1520, -1070, -1520, -1010}
1802 ,{   240, -270,   180, -270,   240}
1803 ,{ -1010, -1520, -1070, -1520, -1010}
1804 }
1805 ,{{{{ -1010, -1520, -1070, -1520, -1010}
1806 ,{ -1010, -1520, -1070, -1520, -1010}
1807 ,{ -1010, -1520, -1070, -1520, -1010}
1808 ,{ -1010, -1520, -1070, -1520, -1010}
1809 ,{ -1010, -1520, -1070, -1520, -1010}
1810 }
1811 }
1812 ,{{{{ 190, -1180,   190, -870,   190}
1813 ,{ -870, -1250, -890, -870, -890}
1814 ,{ -870, -1180, -1060, -870, -1060}
1815 ,{   190, -1420,   190, -870,   190}
1816 ,{ -870, -1180, -1060, -870, -1060}
1817 }
1818 ,{{{{ -870, -1420, -1060, -870, -1060}
1819 ,{ -1050, -1600, -1240, -1050, -1240}
1820 ,{ -870, -1420, -1060, -870, -1060}
1821 ,{ -1080, -1440, -1080, -2130, -1080}
1822 ,{ -870, -1420, -1060, -870, -1060}
1823 }
1824 ,{{{{ -870, -1180, -1060, -870, -1060}
1825 ,{ -870, -1420, -1060, -870, -1060}
1826 ,{ -870, -1180, -1060, -870, -1060}
1827 ,{ -870, -1420, -1060, -870, -1060}
1828 ,{ -870, -1180, -1060, -870, -1060}
1829 }
1830 ,{{{{ 190, -1250,   190, -870,   190}
1831 ,{ -890, -1250, -890, -1940, -890}
1832 ,{ -870, -1420, -1060, -870, -1060}
1833 ,{   190, -1420,   190, -2120,   190}
1834 ,{ -870, -1420, -1060, -870, -1060}
1835 }
1836 ,{{{{ -870, -1180, -1060, -870, -1060}
1837 ,{ -870, -1420, -1060, -870, -1060}
1838 ,{ -870, -1180, -1060, -870, -1060}
1839 ,{ -870, -1420, -1060, -870, -1060}
1840 ,{ -1060, -1420, -1060, -2120, -1060}
1841 }
1842 }
1843 ,{{{{ 130, -270,   130, -270, -1680}
1844 ,{ -700, -1340, -700, -1340, -1680}
1845 ,{ -1120, -1520, -1120, -1520, -1850}
1846 ,{   130, -270,   130, -270, -1850}
1847 ,{ -1120, -1520, -1120, -1520, -1850}
1848 }
1849 ,{{{{ -890, -1520, -890, -1520, -1790}
1850 ,{ -1300, -1700, -1300, -1700, -1790}
1851 ,{ -1120, -1520, -1120, -1520, -1850}
1852 ,{ -890, -1530, -890, -1530, -1870}
1853 ,{ -1120, -1520, -1120, -1520, -1850}

```

```

1854      }
1855 ,{{ -1120, -1520, -1120, -1520, -1850}
1856 ,{ -1120, -1520, -1120, -1520, -1850}
1857 ,{ -1120, -1520, -1120, -1520, -1850}
1858 ,{ -1120, -1520, -1120, -1520, -1850}
1859 ,{ -1120, -1520, -1120, -1520, -1850}
1860      }
1861 ,{{ 130, -270, 130, -270, -1680}
1862 ,{ -700, -1340, -700, -1340, -1680}
1863 ,{ -1120, -1520, -1120, -1520, -1850}
1864 ,{ 130, -270, 130, -270, -1850}
1865 ,{ -1120, -1520, -1120, -1520, -1850}
1866      }
1867 ,{{ -1120, -1520, -1120, -1520, -1850}
1868 ,{ -1120, -1520, -1120, -1520, -1850}
1869 ,{ -1120, -1520, -1120, -1520, -1850}
1870 ,{ -1120, -1520, -1120, -1520, -1850}
1871 ,{ -1120, -1520, -1120, -1520, -1850}
1872      }
1873      }
1874      }
1875 ,{{{{ 800, 600, 740, 290, 800}
1876 ,{ 200, -140, 0, 200, 50}
1877 ,{ -310, -630, -510, -310, -450}
1878 ,{ 800, 600, 740, 290, 800}
1879 ,{ -310, -410, -510, -310, -450}
1880      }
1881 ,{{ 200, -140, 0, 200, 50}
1882 ,{ 200, -140, 0, 200, 50}
1883 ,{ -310, -650, -510, -310, -450}
1884 ,{ -550, -990, -610, -1300, -550}
1885 ,{ -310, -650, -510, -310, -450}
1886      }
1887 ,{{ -310, -630, -510, -310, -450}
1888 ,{ -310, -650, -510, -310, -450}
1889 ,{ -310, -630, -510, -310, -450}
1890 ,{ -310, -650, -510, -310, -450}
1891 ,{ -310, -630, -510, -310, -450}
1892      }
1893 ,{{{{ 800, 600, 740, 290, 800}
1894 ,{ -720, -1160, -780, -1470, -720}
1895 ,{ -310, -650, -510, -310, -450}
1896 ,{ 800, 600, 740, 290, 800}
1897 ,{ -310, -650, -510, -310, -450}
1898      }
1899 ,{{ -310, -410, -510, -310, -450}
1900 ,{ -310, -650, -510, -310, -450}
1901 ,{ -310, -630, -510, -310, -450}
1902 ,{ -310, -650, -510, -310, -450}
1903 ,{ -410, -410, -510, -960, -450}
1904      }
1905      }
1906 ,{{{{ 740, 600, 740, -640, 740}
1907 ,{ 0, -140, 0, -640, 0}
1908 ,{ -510, -650, -510, -910, -510}
1909 ,{ 740, 600, 740, -1150, 740}
1910 ,{ -410, -410, -510, -910, -510}
1911      }
1912 ,{{ 0, -140, 0, -640, 0}
1913 ,{ 0, -140, 0, -640, 0}
1914 ,{ -510, -650, -510, -1150, -510}
1915 ,{ -850, -990, -850, -1490, -850}
1916 ,{ -510, -650, -510, -1150, -510}
1917      }
1918 ,{{ -510, -650, -510, -910, -510}
1919 ,{ -510, -650, -510, -1150, -510}
1920 ,{ -510, -650, -510, -910, -510}
1921 ,{ -510, -650, -510, -1150, -510}
1922 ,{ -510, -650, -510, -910, -510}
1923      }
1924 ,{{{{ 740, 600, 740, -1150, 740}
1925 ,{ -1020, -1160, -1020, -1660, -1020}
1926 ,{ -510, -650, -510, -1150, -510}
1927 ,{ 740, 600, 740, -1150, 740}
1928 ,{ -510, -650, -510, -1150, -510}
1929      }
1930 ,{{ -410, -410, -510, -910, -510}
1931 ,{ -510, -650, -510, -1150, -510}
1932 ,{ -510, -650, -510, -910, -510}
1933 ,{ -510, -650, -510, -1150, -510}
1934 ,{ -410, -410, -510, -1150, -510}
1935      }
1936      }
1937 ,{{{{ 800, 290, 740, 290, 800}
1938 ,{ 50, -450, 0, -450, 50}
1939 ,{ -450, -960, -510, -960, -450}
1940 ,{ 800, 290, 740, 290, 800}

```

```
1941 , { -450, -960, -510, -960, -450}
1942 }
1943 , { { 50, -450, 0, -450, 50}
1944 , { 50, -450, 0, -450, 50}
1945 , { -450, -960, -510, -960, -450}
1946 , { -550, -1300, -610, -1300, -550}
1947 , { -450, -960, -510, -960, -450}
1948 }
1949 , { { -450, -960, -510, -960, -450}
1950 , { -450, -960, -510, -960, -450}
1951 , { -450, -960, -510, -960, -450}
1952 , { -450, -960, -510, -960, -450}
1953 , { -450, -960, -510, -960, -450}
1954 }
1955 , { { 800, 290, 740, 290, 800}
1956 , { -720, -1470, -780, -1470, -720}
1957 , { -450, -960, -510, -960, -450}
1958 , { 800, 290, 740, 290, 800}
1959 , { -450, -960, -510, -960, -450}
1960 }
1961 , { { -450, -960, -510, -960, -450}
1962 , { -450, -960, -510, -960, -450}
1963 , { -450, -960, -510, -960, -450}
1964 , { -450, -960, -510, -960, -450}
1965 , { -450, -960, -510, -960, -450}
1966 }
1967 }
1968 , { { { 740, -360, 740, 200, 740}
1969 , { 200, -360, 0, 200, 0}
1970 , { -310, -630, -510, -310, -510}
1971 , { 740, -870, 740, -310, 740}
1972 , { -310, -630, -510, -310, -510}
1973 }
1974 , { { 200, -360, 0, 200, 0}
1975 , { 200, -360, 0, 200, 0}
1976 , { -310, -870, -510, -310, -510}
1977 , { -850, -1210, -850, -1900, -850}
1978 , { -310, -870, -510, -310, -510}
1979 }
1980 , { { -310, -630, -510, -310, -510}
1981 , { -310, -870, -510, -310, -510}
1982 , { -310, -630, -510, -310, -510}
1983 , { -310, -870, -510, -310, -510}
1984 , { -310, -630, -510, -310, -510}
1985 }
1986 , { { 740, -870, 740, -310, 740}
1987 , { -1020, -1380, -1020, -2070, -1020}
1988 , { -310, -870, -510, -310, -510}
1989 , { 740, -870, 740, -1560, 740}
1990 , { -310, -870, -510, -310, -510}
1991 }
1992 , { { -310, -630, -510, -310, -510}
1993 , { -310, -870, -510, -310, -510}
1994 , { -310, -630, -510, -310, -510}
1995 , { -310, -870, -510, -310, -510}
1996 , { -510, -870, -510, -1560, -510}
1997 }
1998 }
1999 , { { { 690, 290, 690, 290, -550}
2000 , { -50, -450, -50, -450, -550}
2001 , { -560, -960, -560, -960, -1300}
2002 , { 690, 290, 690, 290, -1300}
2003 , { -560, -960, -560, -960, -1300}
2004 }
2005 , { { -50, -450, -50, -450, -550}
2006 , { -50, -450, -50, -450, -550}
2007 , { -560, -960, -560, -960, -1300}
2008 , { -660, -1300, -660, -1300, -1640}
2009 , { -560, -960, -560, -960, -1300}
2010 }
2011 , { { -560, -960, -560, -960, -1300}
2012 , { -560, -960, -560, -960, -1300}
2013 , { -560, -960, -560, -960, -1300}
2014 , { -560, -960, -560, -960, -1300}
2015 , { -560, -960, -560, -960, -1300}
2016 }
2017 , { { 690, 290, 690, 290, -1300}
2018 , { -830, -1470, -830, -1470, -1810}
2019 , { -560, -960, -560, -960, -1300}
2020 , { 690, 290, 690, 290, -1300}
2021 , { -560, -960, -560, -960, -1300}
2022 }
2023 , { { -560, -960, -560, -960, -1300}
2024 , { -560, -960, -560, -960, -1300}
2025 , { -560, -960, -560, -960, -1300}
2026 , { -560, -960, -560, -960, -1300}
2027 , { -560, -960, -560, -960, -1300}
```

```
2028    }
2029    }
2030    }
2031 ,{{{{ 1170,    970,   1120,    780,   1170}
2032 ,{ 780,    440,    580,    780,    640}
2033 ,{ 480,   170,    280,    480,    340}
2034 ,{ 1170,    970,   1120,    660,   1170}
2035 ,{ 480,   170,    280,    480,    340}
2036    }
2037 ,{{ 780,    440,    580,    780,    640}
2038 ,{ 780,    440,    580,    780,    640}
2039 ,{ 470,   130,    270,    470,    330}
2040 ,{ -510,  -950,  -570, -1260,  -510}
2041 ,{ 470,   130,    270,    470,    330}
2042    }
2043 ,{{ 490,   170,    290,    490,    340}
2044 ,{ 490,   140,    290,    490,    340}
2045 ,{ 480,   170,    280,    480,    340}
2046 ,{ 490,   140,    290,    490,    340}
2047 ,{ 480,   170,    280,    480,    340}
2048    }
2049 ,{{ 1170,    970,   1120,    660,   1170}
2050 ,{ -330,  -770,  -390, -1080,  -330}
2051 ,{ 470,   130,    270,    470,    330}
2052 ,{ 1170,    970,   1120,    660,   1170}
2053 ,{ 470,   130,    270,    470,    330}
2054    }
2055 ,{{ 490,   170,    290,    490,    340}
2056 ,{ 490,   140,    290,    490,    340}
2057 ,{ 480,   170,    280,    480,    340}
2058 ,{ 490,   140,    290,    490,    340}
2059 ,{ -600,  -600,  -690, -1150,  -640}
2060    }
2061    }
2062 ,{{{{ 1120,    970,   1120,     -60,   1120}
2063 ,{ 580,    440,    580,     -60,    580}
2064 ,{ 280,   140,    280,    -120,    280}
2065 ,{ 1120,    970,   1120,    -350,   1120}
2066 ,{ 280,   140,    280,    -120,    280}
2067    }
2068 ,{{ 580,    440,    580,     -60,    580}
2069 ,{ 580,    440,    580,     -60,    580}
2070 ,{ 270,   130,    270,    -370,    270}
2071 ,{ -800,  -950,  -800,  -1450,  -800}
2072 ,{ 270,   130,    270,    -370,    270}
2073    }
2074 ,{{ 290,   140,    290,    -120,    290}
2075 ,{ 290,   140,    290,    -350,    290}
2076 ,{ 280,   140,    280,    -120,    280}
2077 ,{ 290,   140,    290,    -350,    290}
2078 ,{ 280,   140,    280,    -120,    280}
2079    }
2080 ,{{ 1120,    970,   1120,    -370,   1120}
2081 ,{ -620,  -770,  -620,  -1270,  -620}
2082 ,{ 270,   130,    270,    -370,    270}
2083 ,{ 1120,    970,   1120,    -780,   1120}
2084 ,{ 270,   130,    270,    -370,    270}
2085    }
2086 ,{{ 290,   140,    290,    -120,    290}
2087 ,{ 290,   140,    290,    -350,    290}
2088 ,{ 280,   140,    280,    -120,    280}
2089 ,{ 290,   140,    290,    -350,    290}
2090 ,{ -600,  -600,  -690, -1340,  -690}
2091    }
2092    }
2093 ,{{{{ 1170,    660,   1110,    660,   1170}
2094 ,{ 640,   130,    580,    130,    640}
2095 ,{ 340,  -170,    280,   -170,    340}
2096 ,{ 1170,    660,   1110,    660,   1170}
2097 ,{ 340,  -170,    280,   -170,    340}
2098    }
2099 ,{{ 640,   130,    580,    130,    640}
2100 ,{ 640,   130,    580,    130,    640}
2101 ,{ 330,  -180,    270,   -180,    330}
2102 ,{ -510, -1260,  -570, -1260,  -510}
2103 ,{ 330,  -180,    270,   -180,    330}
2104    }
2105 ,{{ 340,  -160,    280,   -160,    340}
2106 ,{ 340,  -160,    280,   -160,    340}
2107 ,{ 340,  -170,    280,   -170,    340}
2108 ,{ 340,  -160,    280,   -160,    340}
2109 ,{ 340,  -170,    280,   -170,    340}
2110    }
2111 ,{{ 1170,    660,   1110,    660,   1170}
2112 ,{ -330, -1080,  -390, -1080,  -330}
2113 ,{ 330,  -180,    270,   -180,    330}
2114 ,{ 1170,    660,   1110,    660,   1170}
```

```
2115 , { 330, -180, 270, -180, 330}
2116 }
2117 , {{ 340, -160, 280, -160, 340}
2118 , { 340, -160, 280, -160, 340}
2119 , { 340, -170, 280, -170, 340}
2120 , { 340, -160, 280, -160, 340}
2121 , { -640, -1150, -700, -1150, -640}
2122 }
2123 }
2124 , {{ { 1120, 220, 1120, 780, 1120}
2125 , { 780, 220, 580, 780, 580}
2126 , { 480, 170, 280, 480, 280}
2127 , { 1120, -70, 1120, 490, 1120}
2128 , { 480, 170, 280, 480, 280}
2129 }
2130 , {{ { 780, 220, 580, 780, 580}
2131 , { 780, 220, 580, 780, 580}
2132 , { 470, -80, 270, 470, 270}
2133 , { -800, -1160, -800, -1860, -800}
2134 , { 470, -80, 270, 470, 270}
2135 }
2136 , {{ { 490, 170, 290, 490, 290}
2137 , { 490, -70, 290, 490, 290}
2138 , { 480, 170, 280, 480, 280}
2139 , { 490, -70, 290, 490, 290}
2140 , { 480, 170, 280, 480, 280}
2141 }
2142 , {{ { 1120, -80, 1120, 470, 1120}
2143 , { -620, -980, -620, -1680, -620}
2144 , { 470, -80, 270, 470, 270}
2145 , { 1120, -490, 1120, -1190, 1120}
2146 , { 470, -80, 270, 470, 270}
2147 }
2148 , {{ { 490, 170, 290, 490, 290}
2149 , { 490, -70, 290, 490, 290}
2150 , { 480, 170, 280, 480, 280}
2151 , { 490, -70, 290, 490, 290}
2152 , { -690, -1050, -690, -1750, -690}
2153 }
2154 }
2155 , {{ { { 1060, 660, 1060, 660, 40}
2156 , { 530, 130, 530, 130, 40}
2157 , { 230, -170, 230, -170, -500}
2158 , { 1060, 660, 1060, 660, -500}
2159 , { 230, -170, 230, -170, -500}
2160 }
2161 , {{ { 530, 130, 530, 130, 40}
2162 , { 530, 130, 530, 130, 40}
2163 , { 220, -180, 220, -180, -510}
2164 , { -620, -1260, -620, -1260, -1590}
2165 , { 220, -180, 220, -180, -510}
2166 }
2167 , {{ { 230, -160, 230, -160, -500}
2168 , { 230, -160, 230, -160, -500}
2169 , { 230, -170, 230, -170, -500}
2170 , { 230, -160, 230, -160, -500}
2171 , { 230, -170, 230, -170, -500}
2172 }
2173 , {{ { 1060, 660, 1060, 660, -510}
2174 , { -440, -1080, -440, -1080, -1410}
2175 , { 220, -180, 220, -180, -510}
2176 , { 1060, 660, 1060, 660, -920}
2177 , { 220, -180, 220, -180, -510}
2178 }
2179 , {{ { 230, -160, 230, -160, -500}
2180 , { 230, -160, 230, -160, -500}
2181 , { 230, -170, 230, -170, -500}
2182 , { 230, -160, 230, -160, -500}
2183 , { -750, -1150, -750, -1150, -1480}
2184 }
2185 }
2186 }
2187 , {{{ { 1350, 1160, 1300, 850, 1350}
2188 , { 850, 500, 650, 850, 700}
2189 , { 720, 400, 520, 720, 570}
2190 , { 1350, 1160, 1300, 850, 1350}
2191 , { 590, 270, 390, 590, 440}
2192 }
2193 , {{ { 850, 500, 650, 850, 700}
2194 , { 850, 500, 650, 850, 700}
2195 , { 570, 220, 370, 570, 420}
2196 , { -460, -900, -520, -1210, -460}
2197 , { 570, 220, 370, 570, 420}
2198 }
2199 , {{ { 720, 400, 520, 720, 570}
2200 , { 720, 370, 520, 720, 570}
2201 , { 720, 400, 520, 720, 570}
```

```
2202 , { 720, 370, 520, 720, 570}
2203 , { 590, 270, 390, 590, 440}
2204 }
2205 , {{ 1350, 1160, 1300, 850, 1350}
2206 , { -760, -1200, -820, -1510, -760}
2207 , { 570, 220, 370, 570, 420}
2208 , { 1350, 1160, 1300, 850, 1350}
2209 , { 570, 220, 370, 570, 420}
2210 }
2211 , {{ 720, 370, 520, 720, 570}
2212 , { 720, 370, 520, 720, 570}
2213 , { 280, -40, 80, 280, 130}
2214 , { 720, 370, 520, 720, 570}
2215 , { -320, -320, -420, -870, -360}
2216 }
2217 }
2218 , {{{ 1300, 1160, 1300, 120, 1300}
2219 , { 650, 500, 650, 0, 650}
2220 , { 520, 370, 520, 120, 520}
2221 , { 1300, 1160, 1300, -120, 1300}
2222 , { 390, 240, 390, -10, 390}
2223 }
2224 , {{ 650, 500, 650, 0, 650}
2225 , { 650, 500, 650, 0, 650}
2226 , { 370, 220, 370, -270, 370}
2227 , { -750, -900, -750, -1400, -750}
2228 , { 370, 220, 370, -270, 370}
2229 }
2230 , {{ 520, 370, 520, 120, 520}
2231 , { 520, 370, 520, -120, 520}
2232 , { 520, 370, 520, 120, 520}
2233 , { 520, 370, 520, -120, 520}
2234 , { 390, 240, 390, -10, 390}
2235 }
2236 , {{ 1300, 1160, 1300, -270, 1300}
2237 , { -1050, -1200, -1050, -1700, -1050}
2238 , { 370, 220, 370, -270, 370}
2239 , { 1300, 1160, 1300, -590, 1300}
2240 , { 370, 220, 370, -270, 370}
2241 }
2242 , {{ 520, 370, 520, -120, 520}
2243 , { 520, 370, 520, -120, 520}
2244 , { 80, -60, 80, -320, 80}
2245 , { 520, 370, 520, -120, 520}
2246 , { -320, -320, -420, -1060, -420}
2247 }
2248 }
2249 , {{{ 1350, 850, 1290, 850, 1350}
2250 , { 700, 190, 640, 190, 700}
2251 , { 570, 60, 510, 60, 570}
2252 , { 1350, 850, 1290, 850, 1350}
2253 , { 440, -60, 380, -60, 440}
2254 }
2255 , {{ 700, 190, 640, 190, 700}
2256 , { 700, 190, 640, 190, 700}
2257 , { 420, -80, 360, -80, 420}
2258 , { -460, -1210, -520, -1210, -460}
2259 , { 420, -80, 360, -80, 420}
2260 }
2261 , {{ 570, 60, 510, 60, 570}
2262 , { 570, 60, 510, 60, 570}
2263 , { 570, 60, 510, 60, 570}
2264 , { 570, 60, 510, 60, 570}
2265 , { 440, -60, 380, -60, 440}
2266 }
2267 , {{ 1350, 850, 1290, 850, 1350}
2268 , { -760, -1510, -820, -1510, -760}
2269 , { 420, -80, 360, -80, 420}
2270 , { 1350, 850, 1290, 850, 1350}
2271 , { 420, -80, 360, -80, 420}
2272 }
2273 , {{ 570, 60, 510, 60, 570}
2274 , { 570, 60, 510, 60, 570}
2275 , { 130, -370, 70, -370, 130}
2276 , { 570, 60, 510, 60, 570}
2277 , { -360, -870, -420, -870, -360}
2278 }
2279 }
2280 , {{{ 1300, 400, 1300, 850, 1300}
2281 , { 850, 290, 650, 850, 650}
2282 , { 720, 400, 520, 720, 520}
2283 , { 1300, 160, 1300, 720, 1300}
2284 , { 590, 270, 390, 590, 390}
2285 }
2286 , {{ 850, 290, 650, 850, 650}
2287 , { 850, 290, 650, 850, 650}
2288 , { 570, 10, 370, 570, 370}
```

```
2289 , { -750, -1110, -750, -1810, -750}
2290 , { 570, 10, 370, 570, 370}
2291 }
2292 , {{ 720, 400, 520, 720, 520}
2293 , { 720, 160, 520, 720, 520}
2294 , { 720, 400, 520, 720, 520}
2295 , { 720, 160, 520, 720, 520}
2296 , { 590, 270, 390, 590, 390}
2297 }
2298 , {{ 1300, 10, 1300, 570, 1300}
2299 , { -1050, -1410, -1050, -2110, -1050}
2300 , { 570, 10, 370, 570, 370}
2301 , { 1300, -310, 1300, -1000, 1300}
2302 , { 570, 10, 370, 570, 370}
2303 }
2304 , {{ 720, 160, 520, 720, 520}
2305 , { 720, 160, 520, 720, 520}
2306 , { 280, -40, 80, 280, 80}
2307 , { 720, 160, 520, 720, 520}
2308 , { -420, -780, -420, -1470, -420}
2309 }
2310 }
2311 , {{{ 1250, 850, 1250, 850, 100}
2312 , { 590, 190, 590, 190, 100}
2313 , { 460, 60, 460, 60, -270}
2314 , { 1250, 850, 1250, 850, -270}
2315 , { 330, -60, 330, -60, -400}
2316 }
2317 , {{ 590, 190, 590, 190, 100}
2318 , { 590, 190, 590, 190, 100}
2319 , { 310, -80, 310, -80, -420}
2320 , { -570, -1210, -570, -1210, -1540}
2321 , { 310, -80, 310, -80, -420}
2322 }
2323 , {{ 460, 60, 460, 60, -270}
2324 , { 460, 60, 460, 60, -270}
2325 , { 460, 60, 460, 60, -270}
2326 , { 460, 60, 460, 60, -270}
2327 , { 330, -60, 330, -60, -400}
2328 }
2329 , {{ 1250, 850, 1250, 850, -420}
2330 , { -870, -1510, -870, -1510, -1840}
2331 , { 310, -80, 310, -80, -420}
2332 , { 1250, 850, 1250, 850, -740}
2333 , { 310, -80, 310, -80, -420}
2334 }
2335 , {{ 460, 60, 460, 60, -270}
2336 , { 460, 60, 460, 60, -270}
2337 , { 20, -370, 20, -370, -710}
2338 , { 460, 60, 460, 60, -270}
2339 , { -470, -870, -470, -870, -1210}
2340 }
2341 }
2342 }
2343 , {{{{ 1350, 1160, 1300, 850, 1350}
2344 , { 850, 500, 650, 850, 700}
2345 , { 720, 400, 520, 720, 570}
2346 , { 1350, 1160, 1300, 850, 1350}
2347 , { 590, 270, 390, 590, 440}
2348 }
2349 , {{ 850, 500, 650, 850, 700}
2350 , { 850, 500, 650, 850, 700}
2351 , { 570, 220, 370, 570, 420}
2352 , { -230, -670, -290, -980, -230}
2353 , { 570, 220, 370, 570, 420}
2354 }
2355 , {{ 720, 400, 520, 720, 570}
2356 , { 720, 370, 520, 720, 570}
2357 , { 720, 400, 520, 720, 570}
2358 , { 720, 370, 520, 720, 570}
2359 , { 590, 270, 390, 590, 440}
2360 }
2361 , {{ 1350, 1160, 1300, 850, 1350}
2362 , { -330, -770, -390, -1080, -330}
2363 , { 570, 220, 370, 570, 420}
2364 , { 1350, 1160, 1300, 850, 1350}
2365 , { 570, 220, 370, 570, 420}
2366 }
2367 , {{ 720, 370, 520, 720, 570}
2368 , { 720, 370, 520, 720, 570}
2369 , { 480, 170, 280, 480, 340}
2370 , { 720, 370, 520, 720, 570}
2371 , { -90, -320, -90, -810, -360}
2372 }
2373 }
2374 , {{{ 1300, 1160, 1300, 540, 1300}
2375 , { 650, 500, 650, 10, 650}
```

```
2376 ,{ 540, 370, 520, 540, 520}
2377 ,{ 1300, 1160, 1300, -120, 1300}
2378 ,{ 390, 240, 390, -10, 390}
2379 }
2380 ,{{ 650, 500, 650, 0, 650}
2381 ,{ 650, 500, 650, 0, 650}
2382 ,{ 370, 220, 370, -100, 370}
2383 ,{ -530, -670, -530, -1170, -530}
2384 ,{ 370, 220, 370, -270, 370}
2385 }
2386 ,{{ 540, 370, 520, 540, 520}
2387 ,{ 520, 370, 520, 10, 520}
2388 ,{ 540, 370, 520, 540, 520}
2389 ,{ 520, 370, 520, -120, 520}
2390 ,{ 390, 240, 390, -10, 390}
2391 }
2392 ,{{ 1300, 1160, 1300, -270, 1300}
2393 ,{ -620, -770, -620, -1270, -620}
2394 ,{ 370, 220, 370, -270, 370}
2395 ,{ 1300, 1160, 1300, -590, 1300}
2396 ,{ 370, 220, 370, -270, 370}
2397 }
2398 ,{{ 520, 370, 520, -120, 520}
2399 ,{ 520, 370, 520, -120, 520}
2400 ,{ 280, 140, 280, -120, 280}
2401 ,{ 520, 370, 520, -120, 520}
2402 ,{ -90, -320, -90, -810, -420}
2403 }
2404 }
2405 ,{{{ 1350, 850, 1290, 850, 1350}
2406 ,{ 700, 190, 640, 190, 700}
2407 ,{ 570, 60, 510, 60, 570}
2408 ,{ 1350, 850, 1290, 850, 1350}
2409 ,{ 440, -60, 380, -60, 440}
2410 }
2411 ,{{ 700, 190, 640, 190, 700}
2412 ,{ 700, 190, 640, 190, 700}
2413 ,{ 420, -80, 360, -80, 420}
2414 ,{ -230, -980, -290, -980, -230}
2415 ,{ 420, -80, 360, -80, 420}
2416 }
2417 ,{{ 570, 60, 510, 60, 570}
2418 ,{ 570, 60, 510, 60, 570}
2419 ,{ 570, 60, 510, 60, 570}
2420 ,{ 570, 60, 510, 60, 570}
2421 ,{ 440, -60, 380, -60, 440}
2422 }
2423 ,{{ 1350, 850, 1290, 850, 1350}
2424 ,{ -330, -1070, -390, -1080, -330}
2425 ,{ 420, -80, 360, -80, 420}
2426 ,{ 1350, 850, 1290, 850, 1350}
2427 ,{ 420, -80, 360, -80, 420}
2428 }
2429 ,{{ 570, 60, 510, 60, 570}
2430 ,{ 570, 60, 510, 60, 570}
2431 ,{ 340, -170, 280, -170, 340}
2432 ,{ 570, 60, 510, 60, 570}
2433 ,{ -360, -830, -420, -870, -360}
2434 }
2435 }
2436 ,{{{ 1300, 400, 1300, 850, 1300}
2437 ,{ 850, 290, 650, 850, 650}
2438 ,{ 720, 400, 520, 720, 520}
2439 ,{ 1300, 160, 1300, 720, 1300}
2440 ,{ 590, 270, 390, 590, 390}
2441 }
2442 ,{{ 850, 290, 650, 850, 650}
2443 ,{ 850, 290, 650, 850, 650}
2444 ,{ 570, 10, 370, 570, 370}
2445 ,{ -530, -890, -530, -1580, -530}
2446 ,{ 570, 10, 370, 570, 370}
2447 }
2448 ,{{ 720, 400, 520, 720, 520}
2449 ,{ 720, 160, 520, 720, 520}
2450 ,{ 720, 400, 520, 720, 520}
2451 ,{ 720, 160, 520, 720, 520}
2452 ,{ 590, 270, 390, 590, 390}
2453 }
2454 ,{{ 1300, 10, 1300, 570, 1300}
2455 ,{ -620, -980, -620, -1080, -620}
2456 ,{ 570, 10, 370, 570, 370}
2457 ,{ 1300, -310, 1300, -1000, 1300}
2458 ,{ 570, 10, 370, 570, 370}
2459 }
2460 ,{{ 720, 170, 520, 720, 520}
2461 ,{ 720, 160, 520, 720, 520}
2462 ,{ 480, 170, 280, 480, 280}}
```

```
2463 , { 720, 160, 520, 720, 520}
2464 , { -420, -780, -420, -1470, -420}
2465 }
2466 }
2467 , {{ { 1250, 850, 1250, 850, 100}
2468 , { 590, 190, 590, 190, 100}
2469 , { 460, 60, 460, 60, -270}
2470 , { 1250, 850, 1250, 850, -230}
2471 , { 330, -60, 330, -60, -400}
2472 }
2473 , {{ 590, 190, 590, 190, 100}
2474 , { 590, 190, 590, 190, 100}
2475 , { 310, -80, 310, -80, -420}
2476 , { -340, -980, -340, -980, -1320}
2477 , { 310, -80, 310, -80, -420}
2478 }
2479 , {{ 460, 60, 460, 60, -270}
2480 , { 460, 60, 460, 60, -270}
2481 , { 460, 60, 460, 60, -270}
2482 , { 460, 60, 460, 60, -270}
2483 , { 330, -60, 330, -60, -400}
2484 }
2485 , {{ 1250, 850, 1250, 850, -230}
2486 , { -440, -1080, -440, -1080, -1300}
2487 , { 310, -80, 310, -80, -420}
2488 , { 1250, 850, 1250, 850, -230}
2489 , { 310, -80, 310, -80, -420}
2490 }
2491 , {{ 460, 60, 460, 60, -270}
2492 , { 460, 60, 460, 60, -270}
2493 , { 230, -170, 230, -170, -500}
2494 , { 460, 60, 460, 60, -270}
2495 , { -470, -870, -470, -870, -1210}
2496 }
2497 }
2498 }
2499 }
2500 , {{{{ { INF, INF, INF, INF, INF }
2501 , { INF, INF, INF, INF, INF }
2502 , { INF, INF, INF, INF, INF }
2503 , { INF, INF, INF, INF, INF }
2504 , { INF, INF, INF, INF, INF }
2505 }
2506 , {{ INF, INF, INF, INF, INF }
2507 , { INF, INF, INF, INF, INF }
2508 , { INF, INF, INF, INF, INF }
2509 , { INF, INF, INF, INF, INF }
2510 , { INF, INF, INF, INF, INF }
2511 }
2512 , {{ INF, INF, INF, INF, INF }
2513 , { INF, INF, INF, INF, INF }
2514 , { INF, INF, INF, INF, INF }
2515 , { INF, INF, INF, INF, INF }
2516 , { INF, INF, INF, INF, INF }
2517 }
2518 , {{ INF, INF, INF, INF, INF }
2519 , { INF, INF, INF, INF, INF }
2520 , { INF, INF, INF, INF, INF }
2521 , { INF, INF, INF, INF, INF }
2522 , { INF, INF, INF, INF, INF }
2523 }
2524 , {{ INF, INF, INF, INF, INF }
2525 , { INF, INF, INF, INF, INF }
2526 , { INF, INF, INF, INF, INF }
2527 , { INF, INF, INF, INF, INF }
2528 , { INF, INF, INF, INF, INF }
2529 }
2530 }
2531 , {{{ { INF, INF, INF, INF, INF }
2532 , { INF, INF, INF, INF, INF }
2533 , { INF, INF, INF, INF, INF }
2534 , { INF, INF, INF, INF, INF }
2535 , { INF, INF, INF, INF, INF }
2536 }
2537 , {{ INF, INF, INF, INF, INF }
2538 , { INF, INF, INF, INF, INF }
2539 , { INF, INF, INF, INF, INF }
2540 , { INF, INF, INF, INF, INF }
2541 , { INF, INF, INF, INF, INF }
2542 }
2543 , {{ INF, INF, INF, INF, INF }
2544 , { INF, INF, INF, INF, INF }
2545 , { INF, INF, INF, INF, INF }
2546 , { INF, INF, INF, INF, INF }
2547 , { INF, INF, INF, INF, INF }
2548 }
2549 , {{ INF, INF, INF, INF, INF }
```

```
2550 , { INF, INF, INF, INF, INF }
2551 , { INF, INF, INF, INF, INF }
2552 , { INF, INF, INF, INF, INF }
2553 , { INF, INF, INF, INF, INF }
2554 }
2555 , {{ INF, INF, INF, INF, INF }
2556 , { INF, INF, INF, INF, INF }
2557 , { INF, INF, INF, INF, INF }
2558 , { INF, INF, INF, INF, INF }
2559 , { INF, INF, INF, INF, INF }
2560 }
2561 }
2562 , {{{ INF, INF, INF, INF, INF }
2563 , { INF, INF, INF, INF, INF }
2564 , { INF, INF, INF, INF, INF }
2565 , { INF, INF, INF, INF, INF }
2566 , { INF, INF, INF, INF, INF }
2567 }
2568 , {{ INF, INF, INF, INF, INF }
2569 , { INF, INF, INF, INF, INF }
2570 , { INF, INF, INF, INF, INF }
2571 , { INF, INF, INF, INF, INF }
2572 , { INF, INF, INF, INF, INF }
2573 }
2574 , {{ INF, INF, INF, INF, INF }
2575 , { INF, INF, INF, INF, INF }
2576 , { INF, INF, INF, INF, INF }
2577 , { INF, INF, INF, INF, INF }
2578 , { INF, INF, INF, INF, INF }
2579 }
2580 , {{ INF, INF, INF, INF, INF }
2581 , { INF, INF, INF, INF, INF }
2582 , { INF, INF, INF, INF, INF }
2583 , { INF, INF, INF, INF, INF }
2584 , { INF, INF, INF, INF, INF }
2585 }
2586 , {{ INF, INF, INF, INF, INF }
2587 , { INF, INF, INF, INF, INF }
2588 , { INF, INF, INF, INF, INF }
2589 , { INF, INF, INF, INF, INF }
2590 , { INF, INF, INF, INF, INF }
2591 }
2592 }
2593 , {{{ INF, INF, INF, INF, INF }
2594 , { INF, INF, INF, INF, INF }
2595 , { INF, INF, INF, INF, INF }
2596 , { INF, INF, INF, INF, INF }
2597 , { INF, INF, INF, INF, INF }
2598 }
2599 , {{ INF, INF, INF, INF, INF }
2600 , { INF, INF, INF, INF, INF }
2601 , { INF, INF, INF, INF, INF }
2602 , { INF, INF, INF, INF, INF }
2603 , { INF, INF, INF, INF, INF }
2604 }
2605 , {{ INF, INF, INF, INF, INF }
2606 , { INF, INF, INF, INF, INF }
2607 , { INF, INF, INF, INF, INF }
2608 , { INF, INF, INF, INF, INF }
2609 , { INF, INF, INF, INF, INF }
2610 }
2611 , {{ INF, INF, INF, INF, INF }
2612 , { INF, INF, INF, INF, INF }
2613 , { INF, INF, INF, INF, INF }
2614 , { INF, INF, INF, INF, INF }
2615 , { INF, INF, INF, INF, INF }
2616 }
2617 , {{ INF, INF, INF, INF, INF }
2618 , { INF, INF, INF, INF, INF }
2619 , { INF, INF, INF, INF, INF }
2620 , { INF, INF, INF, INF, INF }
2621 , { INF, INF, INF, INF, INF }
2622 }
2623 }
2624 , {{{ INF, INF, INF, INF, INF }
2625 , { INF, INF, INF, INF, INF }
2626 , { INF, INF, INF, INF, INF }
2627 , { INF, INF, INF, INF, INF }
2628 , { INF, INF, INF, INF, INF }
2629 }
2630 , {{ INF, INF, INF, INF, INF }
2631 , { INF, INF, INF, INF, INF }
2632 , { INF, INF, INF, INF, INF }
2633 , { INF, INF, INF, INF, INF }
2634 , { INF, INF, INF, INF, INF }
2635 }
2636 , {{ INF, INF, INF, INF, INF }
```

```

2637 , { INF, INF, INF, INF, INF }
2638 , { INF, INF, INF, INF, INF }
2639 , { INF, INF, INF, INF, INF }
2640 , { INF, INF, INF, INF, INF }
2641 }
2642 , {{ INF, INF, INF, INF, INF }
2643 , { INF, INF, INF, INF, INF }
2644 , { INF, INF, INF, INF, INF }
2645 , { INF, INF, INF, INF, INF }
2646 , { INF, INF, INF, INF, INF }
2647 }
2648 , {{ INF, INF, INF, INF, INF }
2649 , { INF, INF, INF, INF, INF }
2650 , { INF, INF, INF, INF, INF }
2651 , { INF, INF, INF, INF, INF }
2652 , { INF, INF, INF, INF, INF }
2653 }
2654 }
2655 }
2656 , {{{{ 540, -90, 540, 180, -90}
2657 , { 540, -100, 540, 180, -90}
2658 , { 180, -90, -460, 180, -460}
2659 , { 30, -150, -260, 30, -210}
2660 , { -200, -200, -400, -230, -570}
2661 }
2662 , {{ 180, -350, -660, 180, -660}
2663 , { 180, -580, -660, 180, -660}
2664 , { -430, -600, -970, -430, -830}
2665 , { -350, -350, -870, -960, -870}
2666 , { -430, -600, -970, -430, -970}
2667 }
2668 , {{ 30, -150, -510, 30, -90}
2669 , { -90, -220, -510, -390, -90}
2670 , { 20, -600, -520, 20, -520}
2671 , { 30, -150, -510, 30, -510}
2672 , { -200, -200, -570, -650, -570}
2673 }
2674 , {{ 540, -100, 540, -400, -210}
2675 , { 540, -100, 540, -1240, -810}
2676 , { -430, -600, -970, -430, -970}
2677 , { -200, -200, -260, -400, -210}
2678 , { -430, -600, -970, -430, -970}
2679 }
2680 , {{ 180, -90, -400, 180, -460}
2681 , { 30, -150, -510, 30, -510}
2682 , { 180, -90, -460, 180, -460}
2683 , { 30, -150, -510, 30, -510}
2684 , { -230, -1390, -400, -230, -1300}
2685 }
2686 }
2687 , {{{ 10, -90, 10, -500, -320}
2688 , { 10, -150, 10, -860, -510}
2689 , { -90, -90, -460, -500, -460}
2690 , { -150, -150, -320, -860, -320}
2691 , { -200, -200, -400, -1300, -570}
2692 }
2693 , {{ -580, -580, -660, -1070, -660}
2694 , { -580, -580, -660, -1340, -660}
2695 , { -600, -600, -970, -1070, -970}
2696 , { -870, -1600, -1110, -1880, -870}
2697 , { -600, -600, -970, -1320, -970}
2698 }
2699 , {{ -150, -150, -510, -500, -510}
2700 , { -220, -220, -1150, -860, -510}
2701 , { -500, -1070, -750, -500, -520}
2702 , { -150, -150, -510, -860, -510}
2703 , { -200, -200, -570, -1750, -570}
2704 }
2705 , {{ 10, -200, 10, -1080, -320}
2706 , { 10, -970, 10, -2450, -1160}
2707 , { -600, -600, -970, -1320, -970}
2708 , { -200, -200, -320, -1080, -320}
2709 , { -600, -600, -970, -1320, -970}
2710 }
2711 , {{ -90, -90, -400, -570, -460}
2712 , { -150, -150, -510, -860, -510}
2713 , { -90, -90, -460, -570, -460}
2714 , { -150, -150, -510, -860, -510}
2715 , { -400, -1490, -400, -1300, -1300}
2716 }
2717 }
2718 , {{{ 540, -100, 540, -400, -210}
2719 , { 540, -100, 540, -600, -1130}
2720 , { -540, -540, -760, -540, -1070}
2721 , { -210, -350, -620, -400, -210}
2722 , { -650, -650, -870, -650, -1180}
2723 }

```

```
2724 ,{{ -350, -350, -940, -740, -1250} }  
2725 ,{ -740, -740, -960, -740, -1270} }  
2726 ,{ -1050, -1050, -1270, -1050, -1580} }  
2727 ,{ -350, -350, -940, -960, -1250} }  
2728 ,{ -1050, -1050, -1270, -1050, -1580} }  
2729 }  
2730 ,{{ -600, -600, -820, -600, -1130} }  
2731 ,{ -600, -600, -820, -600, -1130} }  
2732 ,{ -600, -600, -820, -600, -1130} }  
2733 ,{ -600, -600, -820, -600, -1130} }  
2734 ,{ -650, -650, -870, -650, -1180} }  
2735 }  
2736 ,{{ 540, -100, 540, -400, -210} }  
2737 ,{ 540, -100, 540, -1240, -1530} }  
2738 ,{ -1050, -1050, -1270, -1050, -1580} }  
2739 ,{ -210, -440, -620, -400, -210} }  
2740 ,{ -1050, -1050, -1270, -1050, -1580} }  
2741 }  
2742 ,{{ -540, -540, -760, -540, -1070} }  
2743 ,{ -600, -600, -820, -600, -1130} }  
2744 ,{ -540, -540, -760, -540, -1070} }  
2745 ,{ -600, -600, -820, -600, -1130} }  
2746 ,{ -1390, -1390, -1610, -1390, -1920} }  
2747 }  
2748 }  
2749 ,{{ { 180, -630, -320, 180, -320} }  
2750 ,{ 180, -1340, -510, 180, -510} }  
2751 ,{ 180, -630, -460, 180, -460} }  
2752 ,{ 30, -1340, -320, 30, -320} }  
2753 ,{ -230, -1150, -570, -230, -570} }  
2754 }  
2755 ,{{ { 180, -1790, -660, 180, -660} }  
2756 ,{ 180, -2010, -660, 180, -660} }  
2757 ,{ -430, -1790, -970, -430, -970} }  
2758 ,{ -870, -3070, -870, -1370, -870} }  
2759 ,{ -430, -1790, -970, -430, -970} }  
2760 }  
2761 ,{{ { 30, -630, -510, 30, -510} }  
2762 ,{ -390, -1650, -510, -390, -510} }  
2763 ,{ 20, -630, -520, 20, -520} }  
2764 ,{ 30, -1340, -510, 30, -510} }  
2765 ,{ -570, -1150, -570, -880, -570} }  
2766 }  
2767 ,{{ { -320, -1790, -320, -430, -320} }  
2768 ,{ -1160, -1980, -1160, -1870, -1160} }  
2769 ,{ -430, -1790, -970, -430, -970} }  
2770 ,{ -320, -2390, -320, -2280, -320} }  
2771 ,{ -430, -1790, -970, -430, -970} }  
2772 }  
2773 ,{{ { 180, -1040, -460, 180, -460} }  
2774 ,{ 30, -1340, -510, 30, -510} }  
2775 ,{ 180, -1040, -460, 180, -460} }  
2776 ,{ 30, -1340, -510, 30, -510} }  
2777 ,{ -230, -1520, -1300, -230, -1300} }  
2778 }  
2779 }  
2780 ,{{ { -90, -400, -260, -400, -90} }  
2781 ,{ -90, -600, -820, -600, -90} }  
2782 ,{ -540, -540, -550, -540, -830} }  
2783 ,{ -260, -400, -260, -400, -800} }  
2784 ,{ -650, -650, -870, -650, -860} }  
2785 }  
2786 ,{{ { -740, -740, -940, -740, -830} }  
2787 ,{ -740, -740, -960, -740, -1240} }  
2788 ,{ -830, -1050, -1270, -1050, -830} }  
2789 ,{ -940, -960, -940, -960, -1360} }  
2790 ,{ -1050, -1050, -1270, -1050, -1260} }  
2791 }  
2792 ,{{ { -90, -600, -820, -600, -90} }  
2793 ,{ -90, -600, -820, -600, -90} }  
2794 ,{ -600, -600, -820, -600, -1710} }  
2795 ,{ -600, -600, -820, -600, -800} }  
2796 ,{ -650, -650, -870, -650, -860} }  
2797 }  
2798 ,{{ { -260, -400, -260, -400, -810} }  
2799 ,{ -810, -1240, -1220, -1240, -810} }  
2800 ,{ -1050, -1050, -1270, -1050, -1260} }  
2801 ,{ -260, -400, -260, -400, -1550} }  
2802 ,{ -1050, -1050, -1270, -1050, -1260} }  
2803 }  
2804 ,{{ { -540, -540, -550, -540, -800} }  
2805 ,{ -600, -600, -820, -600, -800} }  
2806 ,{ -540, -540, -550, -540, -1460} }  
2807 ,{ -600, -600, -820, -600, -800} }  
2808 ,{ -1390, -1390, -1610, -1390, -2350} }  
2809 }  
2810 }
```

```
2811    }
2812 ,{{{{ 50,   50, -320,    50, -320}
2813 ,{ 50, -130, -490,    50, -490}
2814 ,{ -400, -580, -940, -400, -940}
2815 ,{ 50,   50, -320, -320, -320}
2816 ,{ -400, -540, -940, -400, -940}
2817 }
2818 ,{{ 50, -130, -490,    50, -490}
2819 ,{ 50, -130, -490,    50, -490}
2820 ,{ -400, -580, -940, -400, -940}
2821 ,{ -1320, -1320, -1680, -1770, -1680}
2822 ,{ -400, -580, -940, -400, -940}
2823 }
2824 ,{{ -320, -490, -860, -320, -860}
2825 ,{ -320, -490, -860, -320, -860}
2826 ,{ -620, -800, -1160, -620, -1160}
2827 ,{ -320, -490, -860, -320, -860}
2828 ,{ -620, -800, -1160, -620, -1160}
2829 }
2830 ,{{ 50,   50, -320, -400, -320}
2831 ,{ -840, -840, -1210, -1290, -1210}
2832 ,{ -400, -580, -940, -400, -940}
2833 ,{ 50,   50, -320, -400, -320}
2834 ,{ -400, -580, -940, -400, -940}
2835 }
2836 ,{{ -320, -490, -860, -320, -860}
2837 ,{ -320, -490, -860, -320, -860}
2838 ,{ -930, -1110, -1470, -930, -1470}
2839 ,{ -320, -490, -860, -320, -860}
2840 ,{ -540, -540, -1150, -1230, -1150}
2841 }
2842 }
2843 ,{{{{ 50,   50, -320, -840, -320}
2844 ,{ -130, -130, -490, -840, -490}
2845 ,{ -580, -580, -940, -1270, -940}
2846 ,{ 50,   50, -320, -1210, -320}
2847 ,{ -540, -540, -940, -1270, -940}
2848 }
2849 ,{{ -130, -130, -490, -840, -490}
2850 ,{ -130, -130, -490, -840, -490}
2851 ,{ -580, -580, -940, -1290, -940}
2852 ,{ -1320, -1320, -1680, -2030, -1680}
2853 ,{ -580, -580, -940, -1290, -940}
2854 }
2855 ,{{ -490, -490, -860, -1210, -860}
2856 ,{ -490, -490, -860, -1210, -860}
2857 ,{ -800, -800, -1160, -1270, -1160}
2858 ,{ -490, -490, -860, -1210, -860}
2859 ,{ -800, -800, -1160, -1270, -1160}
2860 }
2861 ,{{ 50,   50, -320, -1290, -320}
2862 ,{ -840, -840, -1210, -1560, -1210}
2863 ,{ -580, -580, -940, -1290, -940}
2864 ,{ 50,   50, -320, -1920, -320}
2865 ,{ -580, -580, -940, -1290, -940}
2866 }
2867 ,{{ -490, -490, -860, -1210, -860}
2868 ,{ -490, -490, -860, -1210, -860}
2869 ,{ -1110, -1110, -1470, -1580, -1470}
2870 ,{ -490, -490, -860, -1210, -860}
2871 ,{ -540, -540, -1150, -1500, -1150}
2872 }
2873 }
2874 ,{{{{ -400, -400, -620, -400, -930}
2875 ,{ -580, -580, -800, -580, -1110}
2876 ,{ -1030, -1030, -1250, -1030, -1560}
2877 ,{ -400, -400, -620, -400, -930}
2878 ,{ -1030, -1030, -1250, -1030, -1560}
2879 }
2880 ,{{ -580, -580, -800, -580, -1110}
2881 ,{ -580, -580, -800, -580, -1110}
2882 ,{ -1030, -1030, -1250, -1030, -1560}
2883 ,{ -1750, -1770, -1750, -1770, -2060}
2884 ,{ -1030, -1030, -1250, -1030, -1560}
2885 }
2886 ,{{ -940, -940, -1160, -940, -1470}
2887 ,{ -940, -940, -1160, -940, -1470}
2888 ,{ -1250, -1250, -1470, -1250, -1780}
2889 ,{ -940, -940, -1160, -940, -1470}
2890 ,{ -1250, -1250, -1470, -1250, -1780}
2891 }
2892 ,{{ -400, -400, -620, -400, -930}
2893 ,{ -1270, -1290, -1270, -1290, -1580}
2894 ,{ -1030, -1030, -1250, -1030, -1560}
2895 ,{ -400, -400, -620, -400, -930}
2896 ,{ -1030, -1030, -1250, -1030, -1560}
2897 }
```

```

2898 ,{{ -940, -940, -1160, -940, -1470}
2899 ,{ -940, -940, -1160, -940, -1470}
2900 ,{ -1560, -1560, -1780, -1560, -2090}
2901 ,{ -940, -940, -1160, -940, -1470}
2902 ,{ -1230, -1230, -1450, -1230, -1760}
2903 }
2904 }
2905 ,{{ 50, -1320, -320, 50, -320}
2906 ,{ 50, -1320, -490, 50, -490}
2907 ,{ -400, -1750, -940, -400, -940}
2908 ,{ -320, -1680, -320, -320, -320}
2909 ,{ -400, -1750, -940, -400, -940}
2910 }
2911 ,{{ 50, -1320, -490, 50, -490}
2912 ,{ 50, -1320, -490, 50, -490}
2913 ,{ -400, -1770, -940, -400, -940}
2914 ,{ -1680, -2510, -1680, -2390, -1680}
2915 ,{ -400, -1770, -940, -400, -940}
2916 }
2917 ,{{ -320, -1680, -860, -320, -860}
2918 ,{ -320, -1680, -860, -320, -860}
2919 ,{ -620, -1750, -1160, -620, -1160}
2920 ,{ -320, -1680, -860, -320, -860}
2921 ,{ -620, -1750, -1160, -620, -1160}
2922 }
2923 ,{{ -320, -1770, -320, -400, -320}
2924 ,{ -1210, -2030, -1210, -1920, -1210}
2925 ,{ -400, -1770, -940, -400, -940}
2926 ,{ -320, -2390, -320, -2280, -320}
2927 ,{ -400, -1770, -940, -400, -940}
2928 }
2929 ,{{ -320, -1680, -860, -320, -860}
2930 ,{ -320, -1680, -860, -320, -860}
2931 ,{ -930, -2060, -1470, -930, -1470}
2932 ,{ -320, -1680, -860, -320, -860}
2933 ,{ -1150, -1970, -1150, -1860, -1150}
2934 }
2935 }
2936 ,{{ 400, -400, -620, -400, -540}
2937 ,{ -540, -580, -800, -580, -540}
2938 ,{ -1030, -1030, -1250, -1030, -1230}
2939 ,{ -400, -400, -620, -400, -1150}
2940 ,{ -1030, -1030, -1250, -1030, -1230}
2941 }
2942 ,{{ -540, -580, -800, -580, -540}
2943 ,{ -540, -580, -800, -580, -540}
2944 ,{ -1030, -1030, -1250, -1030, -1230}
2945 ,{ -1750, -1770, -1750, -1770, -1970}
2946 ,{ -1030, -1030, -1250, -1030, -1230}
2947 }
2948 ,{{ -940, -940, -1160, -940, -1150}
2949 ,{ -940, -940, -1160, -940, -1150}
2950 ,{ -1250, -1250, -1470, -1250, -1450}
2951 ,{ -940, -940, -1160, -940, -1150}
2952 ,{ -1250, -1250, -1470, -1250, -1450}
2953 }
2954 ,{{ -400, -400, -620, -400, -1230}
2955 ,{ -1270, -1290, -1270, -1290, -1500}
2956 ,{ -1030, -1030, -1250, -1030, -1230}
2957 ,{ -400, -400, -620, -400, -1860}
2958 ,{ -1030, -1030, -1250, -1030, -1230}
2959 }
2960 ,{{ -940, -940, -1160, -940, -1150}
2961 ,{ -940, -940, -1160, -940, -1150}
2962 ,{ -1560, -1560, -1780, -1560, -1760}
2963 ,{ -940, -940, -1160, -940, -1150}
2964 ,{ -1230, -1230, -1450, -1230, -1440}
2965 }
2966 }
2967 }
2968 ,{{{{ 210, 210, -160, -240, -160}
2969 ,{ -870, -870, -1230, -870, -1230}
2970 ,{ -870, -1040, -1410, -870, -1410}
2971 ,{ 210, 210, -160, -240, -160}
2972 ,{ -800, -800, -1410, -870, -1410}
2973 }
2974 ,{{ -870, -1040, -1410, -870, -1410}
2975 ,{ -1050, -1220, -1590, -1050, -1590}
2976 ,{ -870, -1040, -1410, -870, -1410}
2977 ,{ -1060, -1060, -1420, -1510, -1420}
2978 ,{ -870, -1040, -1410, -870, -1410}
2979 }
2980 ,{{ -870, -1040, -1410, -870, -1410}
2981 ,{ -870, -1040, -1410, -870, -1410}
2982 ,{ -870, -1040, -1410, -870, -1410}
2983 ,{ -870, -1040, -1410, -870, -1410}
2984 ,{ -870, -1040, -1410, -870, -1410}

```

```

2985      }
2986 ,{{ 210,   210, -160, -240, -160}
2987 ,{ -870, -870, -1230, -1320, -1230}
2988 ,{ -870, -1040, -1410, -870, -1410}
2989 ,{ 210,   210, -160, -240, -160}
2990 ,{ -870, -1040, -1410, -870, -1410}
2991 }
2992 ,{{ -800, -800, -1410, -870, -1410}
2993 ,{ -870, -1040, -1410, -870, -1410}
2994 ,{ -870, -1040, -1410, -870, -1410}
2995 ,{ -870, -1040, -1410, -870, -1410}
2996 ,{ -800, -800, -1410, -1490, -1410}
2997 }
2998 }
2999 ,{{{{ 210,   210, -160, -1520, -160}
3000 ,{ -870, -870, -1230, -1580, -1230}
3001 ,{ -1040, -1040, -1410, -1520, -1410}
3002 ,{ 210,   210, -160, -1760, -160}
3003 ,{ -800, -800, -1410, -1520, -1410}
3004 }
3005 ,{{{{ -1040, -1040, -1410, -1760, -1410}
3006 ,{ -1220, -1220, -1590, -1940, -1590}
3007 ,{ -1040, -1040, -1410, -1760, -1410}
3008 ,{ -1060, -1060, -1420, -1770, -1420}
3009 ,{ -1040, -1040, -1410, -1760, -1410}
3010 }
3011 ,{{{{ -1040, -1040, -1410, -1520, -1410}
3012 ,{ -1040, -1040, -1410, -1760, -1410}
3013 ,{ -1040, -1040, -1410, -1520, -1410}
3014 ,{ -1040, -1040, -1410, -1760, -1410}
3015 ,{ -1040, -1040, -1410, -1520, -1410}
3016 }
3017 ,{{{{ 210,   210, -160, -1580, -160}
3018 ,{ -870, -870, -1230, -1580, -1230}
3019 ,{ -1040, -1040, -1410, -1760, -1410}
3020 ,{ 210,   210, -160, -1760, -160}
3021 ,{ -1040, -1040, -1410, -1760, -1410}
3022 }
3023 ,{{{{ -800, -800, -1410, -1520, -1410}
3024 ,{ -1040, -1040, -1410, -1760, -1410}
3025 ,{ -1040, -1040, -1410, -1520, -1410}
3026 ,{ -1040, -1040, -1410, -1760, -1410}
3027 ,{ -800, -800, -1410, -1760, -1410}
3028 }
3029 }
3030 ,{{{{ -240, -240, -460, -240, -770}
3031 ,{ -1300, -1320, -1300, -1320, -1610}
3032 ,{ -1490, -1490, -1710, -1490, -2020}
3033 ,{ -240, -240, -460, -240, -770}
3034 ,{ -1490, -1490, -1710, -1490, -2020}
3035 }
3036 ,{{{{ -1490, -1490, -1490, -1490, -1800}
3037 ,{ -1670, -1670, -1890, -1670, -2200}
3038 ,{ -1490, -1490, -1710, -1490, -2020}
3039 ,{ -1490, -1510, -1490, -1510, -1800}
3040 ,{ -1490, -1490, -1710, -1490, -2020}
3041 }
3042 ,{{{{ -1490, -1490, -1710, -1490, -2020}
3043 ,{ -1490, -1490, -1710, -1490, -2020}
3044 ,{ -1490, -1490, -1710, -1490, -2020}
3045 ,{ -1490, -1490, -1710, -1490, -2020}
3046 ,{ -1490, -1490, -1710, -1490, -2020}
3047 }
3048 ,{{{{ -240, -240, -460, -240, -770}
3049 ,{ -1300, -1320, -1300, -1320, -1610}
3050 ,{ -1490, -1490, -1710, -1490, -2020}
3051 ,{ -240, -240, -460, -240, -770}
3052 ,{ -1490, -1490, -1710, -1490, -2020}
3053 }
3054 ,{{{{ -1490, -1490, -1710, -1490, -2020}
3055 ,{ -1490, -1490, -1710, -1490, -2020}
3056 ,{ -1490, -1490, -1710, -1490, -2020}
3057 ,{ -1490, -1490, -1710, -1490, -2020}
3058 ,{ -1490, -1490, -1710, -1490, -2020}
3059 }
3060 }
3061 ,{{{{ -160, -1990, -160, -870, -160}
3062 ,{ -870, -2060, -1230, -870, -1230}
3063 ,{ -870, -1990, -1410, -870, -1410}
3064 ,{ -160, -2230, -160, -870, -160}
3065 ,{ -870, -1990, -1410, -870, -1410}
3066 }
3067 ,{{{{ -870, -2230, -1410, -870, -1410}
3068 ,{ -1050, -2410, -1590, -1050, -1590}
3069 ,{ -870, -2230, -1410, -870, -1410}
3070 ,{ -1420, -2250, -1420, -2130, -1420}
3071 ,{ -870, -2230, -1410, -870, -1410}

```

```
3072      }
3073 ,{{ -870, -1990, -1410, -870, -1410}
3074 ,{ -870, -2230, -1410, -870, -1410}
3075 ,{ -870, -1990, -1410, -870, -1410}
3076 ,{ -870, -2230, -1410, -870, -1410}
3077 ,{ -870, -1990, -1410, -870, -1410}
3078 }
3079 ,{{ -160, -2060, -160, -870, -160}
3080 ,{ -1230, -2060, -1230, -1940, -1230}
3081 ,{ -870, -2230, -1410, -870, -1410}
3082 ,{ -160, -2230, -160, -2120, -160}
3083 ,{ -870, -2230, -1410, -870, -1410}
3084 }
3085 ,{{ -870, -1990, -1410, -870, -1410}
3086 ,{ -870, -2230, -1410, -870, -1410}
3087 ,{ -870, -1990, -1410, -870, -1410}
3088 ,{ -870, -2230, -1410, -870, -1410}
3089 ,{ -1410, -2230, -1410, -2120, -1410}
3090 }
3091 }
3092 ,{{{{ -240, -240, -460, -240, -1520}
3093 ,{ -1300, -1320, -1300, -1320, -1520}
3094 ,{ -1490, -1490, -1710, -1490, -1700}
3095 ,{ -240, -240, -460, -240, -1700}
3096 ,{ -1490, -1490, -1710, -1490, -1700}
3097 }
3098 ,{{ -1490, -1490, -1490, -1490, -1640}
3099 ,{ -1640, -1670, -1890, -1670, -1640}
3100 ,{ -1490, -1490, -1710, -1490, -1700}
3101 ,{ -1490, -1510, -1490, -1510, -1710}
3102 ,{ -1490, -1490, -1710, -1490, -1700}
3103 }
3104 ,{{ -1490, -1490, -1710, -1490, -1700}
3105 ,{ -1490, -1490, -1710, -1490, -1700}
3106 ,{ -1490, -1490, -1710, -1490, -1700}
3107 ,{ -1490, -1490, -1710, -1490, -1700}
3108 ,{ -1490, -1490, -1710, -1490, -1700}
3109 }
3110 ,{{ -240, -240, -460, -240, -1520}
3111 ,{ -1300, -1320, -1300, -1320, -1520}
3112 ,{ -1490, -1490, -1710, -1490, -1700}
3113 ,{ -240, -240, -460, -240, -1700}
3114 ,{ -1490, -1490, -1710, -1490, -1700}
3115 }
3116 ,{{{{ -1490, -1490, -1710, -1490, -1700}
3117 ,{ -1490, -1490, -1710, -1490, -1700}
3118 ,{ -1490, -1490, -1710, -1490, -1700}
3119 ,{ -1490, -1490, -1710, -1490, -1700}
3120 ,{ -1490, -1490, -1710, -1490, -1700}
3121 }
3122 }
3123 }
3124 ,{{{{ 760, 760, 400, 310, 400}
3125 ,{ 200, -430, -340, 200, -340}
3126 ,{ -310, -490, -850, -310, -850}
3127 ,{ 760, 760, 400, 310, 400}
3128 ,{ -250, -250, -850, -310, -850}
3129 }
3130 ,{{ 200, -430, -340, 200, -340}
3131 ,{ 200, -430, -340, 200, -340}
3132 ,{ -310, -490, -850, -310, -850}
3133 ,{ -830, -830, -1190, -1280, -1190}
3134 ,{ -310, -490, -850, -310, -850}
3135 }
3136 ,{{ -310, -490, -850, -310, -850}
3137 ,{ -310, -490, -850, -310, -850}
3138 ,{ -310, -490, -850, -310, -850}
3139 ,{ -310, -490, -850, -310, -850}
3140 ,{ -310, -490, -850, -310, -850}
3141 }
3142 ,{{ 760, 760, 400, 310, 400}
3143 ,{ -1000, -1000, -1360, -1450, -1360}
3144 ,{ -310, -490, -850, -310, -850}
3145 ,{ 760, 760, 400, 310, 400}
3146 ,{ -310, -490, -850, -310, -850}
3147 }
3148 ,{{ -250, -250, -850, -310, -850}
3149 ,{ -310, -490, -850, -310, -850}
3150 ,{ -310, -490, -850, -310, -850}
3151 ,{ -310, -490, -850, -310, -850}
3152 ,{ -250, -250, -850, -940, -850}
3153 }
3154 }
3155 ,{{{{ 760, 760, 400, -690, 400}
3156 ,{ -340, -490, -340, -690, -340}
3157 ,{ -490, -490, -850, -960, -850}
3158 ,{ 760, 760, 400, -1200, 400}
```

```

3159 , { -250, -250, -850, -960, -850}
3160 }
3161 , {{ -340, -490, -340, -690, -340}
3162 , { -340, -2040, -340, -690, -340}
3163 , { -490, -490, -850, -1200, -850}
3164 , { -830, -830, -1190, -1540, -1190}
3165 , { -490, -490, -850, -1200, -850}
3166 }
3167 , {{ -490, -490, -850, -960, -850}
3168 , { -490, -490, -850, -1200, -850}
3169 , { -490, -490, -850, -960, -850}
3170 , { -490, -490, -850, -1200, -850}
3171 , { -490, -490, -850, -960, -850}
3172 }
3173 , {{ 760, 760, 400, -1200, 400}
3174 , { -1000, -1000, -1360, -1710, -1360}
3175 , { -490, -490, -850, -1200, -850}
3176 , { 760, 760, 400, -1200, 400}
3177 , { -490, -490, -850, -1200, -850}
3178 }
3179 , {{ -250, -250, -850, -960, -850}
3180 , { -490, -490, -850, -1200, -850}
3181 , { -490, -490, -850, -960, -850}
3182 , { -490, -490, -850, -1200, -850}
3183 , { -250, -250, -850, -1200, -850}
3184 }
3185 }
3186 , {{ { 310, 310, 90, 310, -220}
3187 , { -430, -430, -650, -430, -960}
3188 , { -940, -940, -1160, -940, -1470}
3189 , { 310, 310, 90, 310, -220}
3190 , { -940, -940, -1160, -940, -1470}
3191 }
3192 , {{ -430, -430, -650, -430, -960}
3193 , { -430, -430, -650, -430, -960}
3194 , { -940, -940, -1160, -940, -1470}
3195 , { -1260, -1280, -1260, -1280, -1570}
3196 , { -940, -940, -1160, -940, -1470}
3197 }
3198 , {{ -940, -940, -1160, -940, -1470}
3199 , { -940, -940, -1160, -940, -1470}
3200 , { -940, -940, -1160, -940, -1470}
3201 , { -940, -940, -1160, -940, -1470}
3202 , { -940, -940, -1160, -940, -1470}
3203 }
3204 , {{ { 310, 310, 90, 310, -220}
3205 , { -1430, -1450, -1430, -1450, -1740}
3206 , { -940, -940, -1160, -940, -1470}
3207 , { 310, 310, 90, 310, -220}
3208 , { -940, -940, -1160, -940, -1470}
3209 }
3210 , {{ -940, -940, -1160, -940, -1470}
3211 , { -940, -940, -1160, -940, -1470}
3212 , { -940, -940, -1160, -940, -1470}
3213 , { -940, -940, -1160, -940, -1470}
3214 , { -940, -940, -1160, -940, -1470}
3215 }
3216 }
3217 , {{ { 400, -1170, 400, 200, 400}
3218 , { 200, -1170, -340, 200, -340}
3219 , { -310, -1440, -850, -310, -850}
3220 , { 400, -1680, 400, -310, 400}
3221 , { -310, -1440, -850, -310, -850}
3222 }
3223 , {{ 200, -1170, -340, 200, -340}
3224 , { 200, -1170, -340, 200, -340}
3225 , { -310, -1680, -850, -310, -850}
3226 , { -1190, -2020, -1190, -1900, -1190}
3227 , { -310, -1680, -850, -310, -850}
3228 }
3229 , {{ -310, -1440, -850, -310, -850}
3230 , { -310, -1680, -850, -310, -850}
3231 , { -310, -1440, -850, -310, -850}
3232 , { -310, -1680, -850, -310, -850}
3233 , { -310, -1440, -850, -310, -850}
3234 }
3235 , {{ { 400, -1680, 400, -310, 400}
3236 , { -1360, -2190, -1360, -2070, -1360}
3237 , { -310, -1680, -850, -310, -850}
3238 , { 400, -1680, 400, -1560, 400}
3239 , { -310, -1680, -850, -310, -850}
3240 }
3241 , {{ -310, -1440, -850, -310, -850}
3242 , { -310, -1680, -850, -310, -850}
3243 , { -310, -1440, -850, -310, -850}
3244 , { -310, -1680, -850, -310, -850}
3245 , { -850, -1680, -850, -1560, -850}

```

```
3246      }
3247  }
3248 ,{{{{ 310,    310,     90,    310,   -390}
3249 ,{ -390,   -430,   -650,   -430,   -390}
3250 ,{ -940,   -940,  -1160,   -940,  -1140}
3251 ,{  310,    310,     90,    310,  -1140}
3252 ,{ -940,   -940,  -1160,   -940,  -1140}
3253 }
3254 ,{{ -390,   -430,   -650,   -430,   -390}
3255 ,{ -390,   -430,   -650,   -430,   -390}
3256 ,{ -940,   -940,  -1160,   -940,  -1140}
3257 ,{ -1260,  -1280,  -1260,  -1280, -1480}
3258 ,{ -940,   -940,  -1160,   -940,  -1140}
3259 }
3260 ,{{ -940,   -940,  -1160,   -940, -1140}
3261 ,{ -940,   -940,  -1160,   -940, -1140}
3262 ,{ -940,   -940,  -1160,   -940, -1140}
3263 ,{ -940,   -940,  -1160,   -940, -1140}
3264 ,{ -940,   -940,  -1160,   -940, -1140}
3265 }
3266 ,{{{{ 310,    310,     90,    310,  -1140}
3267 ,{ -1430,  -1450,  -1430,  -1450, -1650}
3268 ,{ -940,   -940,  -1160,   -940, -1140}
3269 ,{  310,    310,     90,    310,  -1140}
3270 ,{ -940,   -940,  -1160,   -940, -1140}
3271 }
3272 ,{{ -940,   -940,  -1160,   -940, -1140}
3273 ,{ -940,   -940,  -1160,   -940, -1140}
3274 ,{ -940,   -940,  -1160,   -940, -1140}
3275 ,{ -940,   -940,  -1160,   -940, -1140}
3276 ,{ -940,   -940,  -1160,   -940, -1140}
3277 }
3278 }
3279 }
3280 ,{{{{ 1140,   1140,    770,    780,    770}
3281 ,{  780,    600,    240,    780,    240}
3282 ,{  480,    300,    -60,    480,    -60}
3283 ,{  1140,   1140,    770,    690,    770}
3284 ,{  480,    300,    -60,    480,    -60}
3285 }
3286 ,{{  780,    600,    240,    780,    240}
3287 ,{  780,    600,    240,    780,    240}
3288 ,{  470,    290,    -70,    470,    -70}
3289 ,{ -780,   -780,  -1150,  -1230, -1150}
3290 ,{  470,    290,    -70,    470,    -70}
3291 }
3292 ,{{  490,    310,    -50,    490,    -50}
3293 ,{  490,    310,    -50,    490,    -50}
3294 ,{  480,    300,    -60,    480,    -60}
3295 ,{  490,    310,    -50,    490,    -50}
3296 ,{  480,    300,    -60,    480,    -60}
3297 }
3298 ,{{  1140,   1140,    770,    690,    770}
3299 ,{ -600,   -600,  -970,  -1050, -970}
3300 ,{  470,    290,    -70,    470,    -70}
3301 ,{  1140,   1140,    770,    690,    770}
3302 ,{  470,    290,    -70,    470,    -70}
3303 }
3304 ,{{  490,    310,    -50,    490,    -50}
3305 ,{  490,    310,    -50,    490,    -50}
3306 ,{  480,    300,    -60,    480,    -60}
3307 ,{  490,    310,    -50,    490,    -50}
3308 ,{ -430,   -430,  -1040, -1120, -1040}
3309 }
3310 }
3311 ,{{{{ 1140,   1140,    770,   -110,    770}
3312 ,{  600,    600,    240,   -110,    240}
3313 ,{  300,    300,    -60,   -170,    -60}
3314 ,{  1140,   1140,    770,   -400,    770}
3315 ,{  300,    300,    -60,   -170,    -60}
3316 }
3317 ,{{  600,    600,    240,   -110,    240}
3318 ,{  600,    600,    240,   -110,    240}
3319 ,{  290,    290,    -70,   -420,    -70}
3320 ,{ -780,   -780,  -1150,  -1500, -1150}
3321 ,{  290,    290,    -70,   -420,    -70}
3322 }
3323 ,{{  310,    310,    -50,   -170,    -50}
3324 ,{  310,    310,    -50,   -400,    -50}
3325 ,{  300,    300,    -60,   -170,    -60}
3326 ,{  310,    310,    -50,   -400,    -50}
3327 ,{  300,    300,    -60,   -170,    -60}
3328 }
3329 ,{{  1140,   1140,    770,   -420,    770}
3330 ,{ -600,   -600,  -970, -1320, -970}
3331 ,{  290,    290,    -70,   -420,    -70}
3332 ,{  1140,   1140,    770,   -830,    770}
```

```

3333 , { 290,   290,    -70,   -420,    -70}
3334 }
3335 , {{ 310,   310,    -50,   -170,    -50}
3336 , { 310,   310,    -50,   -400,    -50}
3337 , { 300,   300,    -60,   -170,    -60}
3338 , { 310,   310,    -50,   -400,    -50}
3339 , { -430,  -430,  -1040,  -1390,  -1040}
3340 }
3341 }
3342 , {{ { 690,   690,    470,   690,   160}
3343 , { 150,   150,    -60,   150,  -370}
3344 , { -140,  -140,   -360,  -140,  -670}
3345 , { 690,   690,    470,   690,   160}
3346 , { -140,  -140,   -360,  -140,  -670}
3347 }
3348 , {{ { 150,   150,    -60,   150,  -370}
3349 , { 150,   150,    -60,   150,  -370}
3350 , { -150,  -150,   -370,  -150,  -680}
3351 , { -1210, -1230, -1210, -1230, -1520}
3352 , { -150,  -150,   -370,  -150,  -680}
3353 }
3354 , {{ { -140,  -140,   -360,  -140,  -670}
3355 , { -140,  -140,   -360,  -140,  -670}
3356 , { -140,  -140,   -360,  -140,  -670}
3357 , { -140,  -140,   -360,  -140,  -670}
3358 , { -140,  -140,   -360,  -140,  -670}
3359 }
3360 , {{ { 690,   690,    470,   690,   160}
3361 , { -1030, -1050, -1030, -1050, -1340}
3362 , { -150,  -150,   -370,  -150,  -680}
3363 , { 690,   690,    470,   690,   160}
3364 , { -150,  -150,   -370,  -150,  -680}
3365 }
3366 , {{ { -140,  -140,   -360,  -140,  -670}
3367 , { -140,  -140,   -360,  -140,  -670}
3368 , { -140,  -140,   -360,  -140,  -670}
3369 , { -140,  -140,   -360,  -140,  -670}
3370 , { -1120, -1120, -1340, -1120, -1650}
3371 }
3372 }
3373 , {{ { 780,  -580,    770,   780,   770}
3374 , { 780,  -580,    240,   780,   240}
3375 , { 480,  -640,    -60,   480,  -60}
3376 , { 770,  -880,    770,   490,   770}
3377 , { 480,  -640,    -60,   480,  -60}
3378 }
3379 , {{ { 780,  -580,    240,   780,   240}
3380 , { 780,  -580,    240,   780,   240}
3381 , { 470,  -890,    -70,   470,  -70}
3382 , { -1150, -1970, -1150, -1860, -1150}
3383 , { 470,  -890,    -70,   470,  -70}
3384 }
3385 , {{ { 490,  -640,    -50,   490,  -50}
3386 , { 490,  -880,    -50,   490,  -50}
3387 , { 480,  -640,    -60,   480,  -60}
3388 , { 490,  -880,    -50,   490,  -50}
3389 , { 480,  -640,    -60,   480,  -60}
3390 }
3391 , {{ { 770,  -890,    770,   470,   770}
3392 , { -970, -1790, -970, -1680, -970}
3393 , { 470,  -890,    -70,   470,  -70}
3394 , { 770, -1300,    770, -1190,   770}
3395 , { 470,  -890,    -70,   470,  -70}
3396 }
3397 , {{ { 490,  -640,    -50,   490,  -50}
3398 , { 490,  -880,    -50,   490,  -50}
3399 , { 480,  -640,    -60,   480,  -60}
3400 , { 490,  -880,    -50,   490,  -50}
3401 , { -1040, -1860, -1040, -1750, -1040}
3402 }
3403 }
3404 , {{ { 690,   690,    470,   690,   190}
3405 , { 190,   150,    -60,   150,   190}
3406 , { -140,  -140,   -360,  -140,  -350}
3407 , { 690,   690,    470,   690,  -340}
3408 , { -140,  -140,   -360,  -140,  -350}
3409 }
3410 , {{ { 190,   150,    -60,   150,   190}
3411 , { 190,   150,    -60,   150,   190}
3412 , { -150,  -150,   -370,  -150,  -360}
3413 , { -1210, -1230, -1210, -1230, -1440}
3414 , { -150,  -150,   -370,  -150,  -360}
3415 }
3416 , {{ { -140,  -140,   -360,  -140,  -340}
3417 , { -140,  -140,   -360,  -140,  -340}
3418 , { -140,  -140,   -360,  -140,  -350}
3419 , { -140,  -140,   -360,  -140,  -340}

```

```
3420 , { -140, -140, -360, -140, -350}
3421 }
3422 , {{ 690, 690, 470, 690, -360}
3423 , { -1030, -1050, -1030, -1050, -1260}
3424 , { -150, -150, -370, -150, -360}
3425 , { 690, 690, 470, 690, -770}
3426 , { -150, -150, -370, -150, -360}
3427 }
3428 , {{ -140, -140, -360, -140, -340}
3429 , { -140, -140, -360, -140, -340}
3430 , { -140, -140, -360, -140, -350}
3431 , { -140, -140, -360, -140, -340}
3432 , { -1120, -1120, -1340, -1120, -1330}
3433 }
3434 }
3435 }
3436 , {{{ 1320, 1320, 960, 870, 960}
3437 , { 850, 670, 300, 850, 300}
3438 , { 720, 540, 170, 720, 170}
3439 , { 1320, 1320, 960, 870, 960}
3440 , { 590, 410, 40, 590, 40}
3441 }
3442 , {{ 850, 670, 300, 850, 300}
3443 , { 850, 670, 300, 850, 300}
3444 , { 570, 390, 20, 570, 20}
3445 , { -730, -730, -1100, -1180, -1100}
3446 , { 570, 390, 20, 570, 20}
3447 }
3448 , {{ 720, 540, 170, 720, 170}
3449 , { 720, 540, 170, 720, 170}
3450 , { 720, 540, 170, 720, 170}
3451 , { 720, 540, 170, 720, 170}
3452 , { 590, 410, 40, 590, 40}
3453 }
3454 , {{ 1320, 1320, 960, 870, 960}
3455 , { -1030, -1030, -1400, -1480, -1400}
3456 , { 570, 390, 20, 570, 20}
3457 , { 1320, 1320, 960, 870, 960}
3458 , { 570, 390, 20, 570, 20}
3459 }
3460 , {{ 720, 540, 170, 720, 170}
3461 , { 720, 540, 170, 720, 170}
3462 , { 280, 100, -260, 280, -260}
3463 , { 720, 540, 170, 720, 170}
3464 , { -160, -160, -760, -850, -760}
3465 }
3466 }
3467 , {{{ 1320, 1320, 960, 70, 960}
3468 , { 670, 670, 300, -40, 300}
3469 , { 540, 540, 170, 70, 170}
3470 , { 1320, 1320, 960, -170, 960}
3471 , { 410, 410, 40, -60, 40}
3472 }
3473 , {{ 670, 670, 300, -40, 300}
3474 , { 670, 670, 300, -40, 300}
3475 , { 390, 390, 20, -320, 20}
3476 , { -730, -730, -1100, -1450, -1100}
3477 , { 390, 390, 20, -320, 20}
3478 }
3479 , {{ 540, 540, 170, 70, 170}
3480 , { 540, 540, 170, -170, 170}
3481 , { 540, 540, 170, 70, 170}
3482 , { 540, 540, 170, -170, 170}
3483 , { 410, 410, 40, -60, 40}
3484 }
3485 , {{ 1320, 1320, 960, -320, 960}
3486 , { -1030, -1030, -1400, -1750, -1400}
3487 , { 390, 390, 20, -320, 20}
3488 , { 1320, 1320, 960, -640, 960}
3489 , { 390, 390, 20, -320, 20}
3490 }
3491 , {{ 540, 540, 170, -170, 170}
3492 , { 540, 540, 170, -170, 170}
3493 , { 100, 100, -260, -370, -260}
3494 , { 540, 540, 170, -170, 170}
3495 , { -160, -160, -760, -1110, -760}
3496 }
3497 }
3498 , {{{ 870, 870, 650, 870, 340}
3499 , { 220, 220, 0, 220, -310}
3500 , { 90, 90, -130, 90, -440}
3501 , { 870, 870, 650, 870, 340}
3502 , { -40, -40, -260, -40, -570}
3503 }
3504 , {{ 220, 220, 0, 220, -310}
3505 , { 220, 220, 0, 220, -310}
3506 , { -60, -60, -280, -60, -590}
```

```
3507 , { -1160, -1180, -1160, -1180, -1470}
3508 , { -60, -60, -280, -60, -590}
3509 }
3510 , {{ 90, 90, -130, 90, -440}
3511 , { 90, 90, -130, 90, -440}
3512 , { 90, 90, -130, 90, -440}
3513 , { 90, 90, -130, 90, -440}
3514 , { -40, -40, -260, -40, -570}
3515 }
3516 , {{ 870, 870, 650, 870, 340}
3517 , { -1460, -1480, -1460, -1480, -1770}
3518 , { -60, -60, -280, -60, -590}
3519 , { 870, 870, 650, 870, 340}
3520 , { -60, -60, -280, -60, -590}
3521 }
3522 , {{ 90, 90, -130, 90, -440}
3523 , { 90, 90, -130, 90, -440}
3524 , { -350, -350, -570, -350, -880}
3525 , { 90, 90, -130, 90, -440}
3526 , { -850, -850, -1070, -850, -1380}
3527 }
3528 }
3529 , {{{ 960, -410, 960, 850, 960}
3530 , { 850, -520, 300, 850, 300}
3531 , { 720, -410, 170, 720, 170}
3532 , { 960, -650, 960, 720, 960}
3533 , { 590, -540, 40, 590, 40}
3534 }
3535 , {{ 850, -520, 300, 850, 300}
3536 , { 850, -520, 300, 850, 300}
3537 , { 570, -800, 20, 570, 20}
3538 , { -1100, -1920, -1100, -1810, -1100}
3539 , { 570, -800, 20, 570, 20}
3540 }
3541 , {{ 720, -410, 170, 720, 170}
3542 , { 720, -650, 170, 720, 170}
3543 , { 720, -410, 170, 720, 170}
3544 , { 720, -650, 170, 720, 170}
3545 , { 590, -540, 40, 590, 40}
3546 }
3547 , {{ 960, -800, 960, 570, 960}
3548 , { -1400, -2220, -1400, -2110, -1400}
3549 , { 570, -800, 20, 570, 20}
3550 , { 960, -1120, 960, -1000, 960}
3551 , { 570, -800, 20, 570, 20}
3552 }
3553 , {{ 720, -650, 170, 720, 170}
3554 , { 720, -650, 170, 720, 170}
3555 , { 280, -850, -260, 280, -260}
3556 , { 720, -650, 170, 720, 170}
3557 , { -760, -1590, -760, -1470, -760}
3558 }
3559 }
3560 , {{{ 870, 870, 650, 870, 250}
3561 , { 250, 220, 0, 220, 250}
3562 , { 90, 90, -130, 90, -110}
3563 , { 870, 870, 650, 870, -110}
3564 , { -40, -40, -260, -40, -240}
3565 }
3566 , {{ 250, 220, 0, 220, 250}
3567 , { 250, 220, 0, 220, 250}
3568 , { -60, -60, -280, -60, -260}
3569 , { -1160, -1180, -1160, -1180, -1390}
3570 , { -60, -60, -280, -60, -260}
3571 }
3572 , {{ 90, 90, -130, 90, -110}
3573 , { 90, 90, -130, 90, -110}
3574 , { 90, 90, -130, 90, -110}
3575 , { 90, 90, -130, 90, -110}
3576 , { -40, -40, -260, -40, -240}
3577 }
3578 , {{ 870, 870, 650, 870, -260}
3579 , { -1460, -1480, -1460, -1480, -1690}
3580 , { -60, -60, -280, -60, -260}
3581 , { 870, 870, 650, 870, -580}
3582 , { -60, -60, -280, -60, -260}
3583 }
3584 , {{ 90, 90, -130, 90, -110}
3585 , { 90, 90, -130, 90, -110}
3586 , { -350, -350, -570, -350, -550}
3587 , { 90, 90, -130, 90, -110}
3588 , { -850, -850, -1070, -850, -1050}
3589 }
3590 }
3591 }
3592 , {{{{ 1320, 1320, 960, 870, 960}
3593 , { 850, 670, 540, 850, 300}
```

```
3594 , { 720, 540, 170, 720, 170}
3595 , { 1320, 1320, 960, 870, 960}
3596 , { 590, 410, 40, 590, 40}
3597 }
3598 , {{ 850, 670, 300, 850, 300}
3599 , { 850, 670, 300, 850, 300}
3600 , { 570, 390, 20, 570, 20}
3601 , { -350, -350, -870, -960, -870}
3602 , { 570, 390, 20, 570, 20}
3603 }
3604 , {{ 720, 540, 170, 720, 170}
3605 , { 720, 540, 170, 720, 170}
3606 , { 720, 540, 170, 720, 170}
3607 , { 720, 540, 170, 720, 170}
3608 , { 590, 410, 40, 590, 40}
3609 }
3610 , {{ 1320, 1320, 960, 870, 960}
3611 , { 540, -100, 540, -1050, -810}
3612 , { 570, 390, 20, 570, 20}
3613 , { 1320, 1320, 960, 870, 960}
3614 , { 570, 390, 20, 570, 20}
3615 }
3616 , {{ 720, 540, 170, 720, 170}
3617 , { 720, 540, 170, 720, 170}
3618 , { 480, 300, -60, 480, -60}
3619 , { 720, 540, 170, 720, 170}
3620 , { -160, -160, -400, -230, -760}
3621 }
3622 }
3623 , {{{ 1320, 1320, 960, 70, 960}
3624 , { 670, 670, 300, -40, 300}
3625 , { 540, 540, 170, 70, 170}
3626 , { 1320, 1320, 960, -170, 960}
3627 , { 410, 410, 40, -60, 40}
3628 }
3629 , {{ 670, 670, 300, -40, 300}
3630 , { 670, 670, 300, -40, 300}
3631 , { 390, 390, 20, -320, 20}
3632 , { -730, -730, -1100, -1450, -870}
3633 , { 390, 390, 20, -320, 20}
3634 }
3635 , {{ 540, 540, 170, 70, 170}
3636 , { 540, 540, 170, -170, 170}
3637 , { 540, 540, 170, 70, 170}
3638 , { 540, 540, 170, -170, 170}
3639 , { 410, 410, 40, -60, 40}
3640 }
3641 , {{ 1320, 1320, 960, -320, 960}
3642 , { 10, -600, 10, -1320, -970}
3643 , { 390, 390, 20, -320, 20}
3644 , { 1320, 1320, 960, -640, 960}
3645 , { 390, 390, 20, -320, 20}
3646 }
3647 , {{ 540, 540, 170, -170, 170}
3648 , { 540, 540, 170, -170, 170}
3649 , { 300, 300, -60, -170, -60}
3650 , { 540, 540, 170, -170, 170}
3651 , { -160, -160, -400, -1110, -760}
3652 }
3653 }
3654 , {{{ 870, 870, 650, 870, 340}
3655 , { 540, 220, 540, 220, -310}
3656 , { 90, 90, -130, 90, -440}
3657 , { 870, 870, 650, 870, 340}
3658 , { -40, -40, -260, -40, -570}
3659 }
3660 , {{ 220, 220, 0, 220, -310}
3661 , { 220, 220, 0, 220, -310}
3662 , { -60, -60, -280, -60, -590}
3663 , { -350, -350, -940, -960, -1250}
3664 , { -60, -60, -280, -60, -590}
3665 }
3666 , {{ 90, 90, -130, 90, -440}
3667 , { 90, 90, -130, 90, -440}
3668 , { 90, 90, -130, 90, -440}
3669 , { 90, 90, -130, 90, -440}
3670 , { -40, -40, -260, -40, -570}
3671 }
3672 , {{ 870, 870, 650, 870, 340}
3673 , { 540, -100, 540, -1050, -1340}
3674 , { -60, -60, -280, -60, -590}
3675 , { 870, 870, 650, 870, 340}
3676 , { -60, -60, -280, -60, -590}
3677 }
3678 , {{ 90, 90, -130, 90, -440}
3679 , { 90, 90, -130, 90, -440}
3680 , { -140, -140, -360, -140, -670}
```

```

3681 , { 90, 90, -130, 90, -440}
3682 , { -850, -850, -1070, -850, -1380}
3683 }
3684 }
3685 , {{ { 960, -410, 960, 850, 960}
3686 , { 850, -520, 300, 850, 300}
3687 , { 720, -410, 170, 720, 170}
3688 , { 960, -650, 960, 720, 960}
3689 , { 590, -540, 40, 590, 40}
3690 }
3691 , {{ 850, -520, 300, 850, 300}
3692 , { 850, -520, 300, 850, 300}
3693 , { 570, -800, 20, 570, 20}
3694 , { -870, -1920, -870, -1370, -870}
3695 , { 570, -800, 20, 570, 20}
3696 }
3697 , {{ 720, -410, 170, 720, 170}
3698 , { 720, -650, 170, 720, 170}
3699 , { 720, -410, 170, 720, 170}
3700 , { 720, -650, 170, 720, 170}
3701 , { 590, -540, 40, 590, 40}
3702 }
3703 , {{ 960, -800, 960, 570, 960}
3704 , { -970, -1790, -970, -1680, -970}
3705 , { 570, -800, 20, 570, 20}
3706 , { 960, -1120, 960, -1000, 960}
3707 , { 570, -800, 20, 570, 20}
3708 }
3709 , {{ 720, -640, 170, 720, 170}
3710 , { 720, -650, 170, 720, 170}
3711 , { 480, -640, -60, 480, -60}
3712 , { 720, -650, 170, 720, 170}
3713 , { -230, -1520, -760, -230, -760}
3714 }
3715 }
3716 , {{ { 870, 870, 650, 870, 250}
3717 , { 250, 220, 0, 220, 250}
3718 , { 90, 90, -130, 90, -110}
3719 , { 870, 870, 650, 870, -110}
3720 , { -40, -40, -260, -40, -240}
3721 }
3722 , {{ 250, 220, 0, 220, 250}
3723 , { 250, 220, 0, 220, 250}
3724 , { -60, -60, -280, -60, -260}
3725 , { -940, -960, -940, -960, -1360}
3726 , { -60, -60, -280, -60, -260}
3727 }
3728 , {{ 90, 90, -130, 90, -90}
3729 , { 90, 90, -130, 90, -90}
3730 , { 90, 90, -130, 90, -110}
3731 , { 90, 90, -130, 90, -110}
3732 , { -40, -40, -260, -40, -240}
3733 }
3734 , {{ 870, 870, 650, 870, -260}
3735 , { -810, -1050, -1030, -1050, -810}
3736 , { -60, -60, -280, -60, -260}
3737 , { 870, 870, 650, 870, -580}
3738 , { -60, -60, -280, -60, -260}
3739 }
3740 , {{ 90, 90, -130, 90, -110}
3741 , { 90, 90, -130, 90, -110}
3742 , { -140, -140, -360, -140, -350}
3743 , { 90, 90, -130, 90, -110}
3744 , { -850, -850, -1070, -850, -1050}
3745 }
3746 }
3747 }
3748 }
3749 , {{{ { INF, INF, INF, INF, INF}
3750 , { INF, INF, INF, INF, INF}
3751 , { INF, INF, INF, INF, INF}
3752 , { INF, INF, INF, INF, INF}
3753 , { INF, INF, INF, INF, INF}
3754 }
3755 , {{ INF, INF, INF, INF, INF}
3756 , { INF, INF, INF, INF, INF}
3757 , { INF, INF, INF, INF, INF}
3758 , { INF, INF, INF, INF, INF}
3759 , { INF, INF, INF, INF, INF}
3760 }
3761 , {{ INF, INF, INF, INF, INF}
3762 , { INF, INF, INF, INF, INF}
3763 , { INF, INF, INF, INF, INF}
3764 , { INF, INF, INF, INF, INF}
3765 , { INF, INF, INF, INF, INF}
3766 }
3767 , {{ INF, INF, INF, INF, INF}

```

```
3768 , { INF, INF, INF, INF, INF }
3769 , { INF, INF, INF, INF, INF }
3770 , { INF, INF, INF, INF, INF }
3771 , { INF, INF, INF, INF, INF }
3772 }
3773 , {{ INF, INF, INF, INF, INF }
3774 , { INF, INF, INF, INF, INF }
3775 , { INF, INF, INF, INF, INF }
3776 , { INF, INF, INF, INF, INF }
3777 , { INF, INF, INF, INF, INF }
3778 }
3779 }
3780 , {{{ INF, INF, INF, INF, INF }
3781 , { INF, INF, INF, INF, INF }
3782 , { INF, INF, INF, INF, INF }
3783 , { INF, INF, INF, INF, INF }
3784 , { INF, INF, INF, INF, INF }
3785 }
3786 , {{ INF, INF, INF, INF, INF }
3787 , { INF, INF, INF, INF, INF }
3788 , { INF, INF, INF, INF, INF }
3789 , { INF, INF, INF, INF, INF }
3790 , { INF, INF, INF, INF, INF }
3791 }
3792 , {{ INF, INF, INF, INF, INF }
3793 , { INF, INF, INF, INF, INF }
3794 , { INF, INF, INF, INF, INF }
3795 , { INF, INF, INF, INF, INF }
3796 , { INF, INF, INF, INF, INF }
3797 }
3798 , {{ INF, INF, INF, INF, INF }
3799 , { INF, INF, INF, INF, INF }
3800 , { INF, INF, INF, INF, INF }
3801 , { INF, INF, INF, INF, INF }
3802 , { INF, INF, INF, INF, INF }
3803 }
3804 , {{ INF, INF, INF, INF, INF }
3805 , { INF, INF, INF, INF, INF }
3806 , { INF, INF, INF, INF, INF }
3807 , { INF, INF, INF, INF, INF }
3808 , { INF, INF, INF, INF, INF }
3809 }
3810 }
3811 , {{{ INF, INF, INF, INF, INF }
3812 , { INF, INF, INF, INF, INF }
3813 , { INF, INF, INF, INF, INF }
3814 , { INF, INF, INF, INF, INF }
3815 , { INF, INF, INF, INF, INF }
3816 }
3817 , {{ INF, INF, INF, INF, INF }
3818 , { INF, INF, INF, INF, INF }
3819 , { INF, INF, INF, INF, INF }
3820 , { INF, INF, INF, INF, INF }
3821 , { INF, INF, INF, INF, INF }
3822 }
3823 , {{ INF, INF, INF, INF, INF }
3824 , { INF, INF, INF, INF, INF }
3825 , { INF, INF, INF, INF, INF }
3826 , { INF, INF, INF, INF, INF }
3827 , { INF, INF, INF, INF, INF }
3828 }
3829 , {{ INF, INF, INF, INF, INF }
3830 , { INF, INF, INF, INF, INF }
3831 , { INF, INF, INF, INF, INF }
3832 , { INF, INF, INF, INF, INF }
3833 , { INF, INF, INF, INF, INF }
3834 }
3835 , {{ INF, INF, INF, INF, INF }
3836 , { INF, INF, INF, INF, INF }
3837 , { INF, INF, INF, INF, INF }
3838 , { INF, INF, INF, INF, INF }
3839 , { INF, INF, INF, INF, INF }
3840 }
3841 }
3842 , {{{ INF, INF, INF, INF, INF }
3843 , { INF, INF, INF, INF, INF }
3844 , { INF, INF, INF, INF, INF }
3845 , { INF, INF, INF, INF, INF }
3846 , { INF, INF, INF, INF, INF }
3847 }
3848 , {{ INF, INF, INF, INF, INF }
3849 , { INF, INF, INF, INF, INF }
3850 , { INF, INF, INF, INF, INF }
3851 , { INF, INF, INF, INF, INF }
3852 , { INF, INF, INF, INF, INF }
3853 }
3854 , {{ INF, INF, INF, INF, INF }
```

```

3855 , { INF, INF, INF, INF, INF }
3856 , { INF, INF, INF, INF, INF }
3857 , { INF, INF, INF, INF, INF }
3858 , { INF, INF, INF, INF, INF }
3859 }
3860 , {{ INF, INF, INF, INF, INF }
3861 , { INF, INF, INF, INF, INF }
3862 , { INF, INF, INF, INF, INF }
3863 , { INF, INF, INF, INF, INF }
3864 , { INF, INF, INF, INF, INF }
3865 }
3866 , {{ INF, INF, INF, INF, INF }
3867 , { INF, INF, INF, INF, INF }
3868 , { INF, INF, INF, INF, INF }
3869 , { INF, INF, INF, INF, INF }
3870 , { INF, INF, INF, INF, INF }
3871 }
3872 }
3873 , {{{ INF, INF, INF, INF, INF }
3874 , { INF, INF, INF, INF, INF }
3875 , { INF, INF, INF, INF, INF }
3876 , { INF, INF, INF, INF, INF }
3877 , { INF, INF, INF, INF, INF }
3878 }
3879 , {{ INF, INF, INF, INF, INF }
3880 , { INF, INF, INF, INF, INF }
3881 , { INF, INF, INF, INF, INF }
3882 , { INF, INF, INF, INF, INF }
3883 , { INF, INF, INF, INF, INF }
3884 }
3885 , {{ INF, INF, INF, INF, INF }
3886 , { INF, INF, INF, INF, INF }
3887 , { INF, INF, INF, INF, INF }
3888 , { INF, INF, INF, INF, INF }
3889 , { INF, INF, INF, INF, INF }
3890 }
3891 , {{ INF, INF, INF, INF, INF }
3892 , { INF, INF, INF, INF, INF }
3893 , { INF, INF, INF, INF, INF }
3894 , { INF, INF, INF, INF, INF }
3895 , { INF, INF, INF, INF, INF }
3896 }
3897 , {{ INF, INF, INF, INF, INF }
3898 , { INF, INF, INF, INF, INF }
3899 , { INF, INF, INF, INF, INF }
3900 , { INF, INF, INF, INF, INF }
3901 , { INF, INF, INF, INF, INF }
3902 }
3903 }
3904 }
3905 , {{{{ 240, -780, -870, 240, -870}
3906 , { 190, -1060, -1060, 190, -970}
3907 , { 240, -780, -1010, 240, -1010}
3908 , { 190, -870, -870, 190, -870}
3909 , { 130, -890, -1120, 130, -1120}
3910 }
3911 , {{ 40, -1210, -1180, 40, -970}
3912 , { 40, -1210, -1210, 40, -970}
3913 , { -270, -1520, -1520, -270, -1520}
3914 , { -1180, -1420, -1180, -1250, -1180}
3915 , { -270, -1520, -1520, -270, -1520}
3916 }
3917 , {{ 190, -840, -1060, 190, -1060}
3918 , { 190, -1060, -1060, 190, -1060}
3919 , { 180, -840, -1070, 180, -1070}
3920 , { 190, -1060, -1060, 190, -1060}
3921 , { 130, -890, -1120, 130, -1120}
3922 }
3923 , {{ -270, -870, -870, -270, -870}
3924 , { -1470, -1710, -1470, -1530, -1470}
3925 , { -270, -1520, -1520, -270, -1520}
3926 , { -870, -870, -870, -870, -870}
3927 , { -270, -1520, -1520, -270, -1520}
3928 }
3929 , {{ 240, -780, -1010, 240, -1010}
3930 , { 190, -1060, -1060, 190, -1060}
3931 , { 240, -780, -1010, 240, -1010}
3932 , { 190, -1060, -1060, 190, -1060}
3933 , { -1680, -1790, -1850, -1680, -1850}
3934 }
3935 }
3936 , {{{ -590, -1050, -870, -590, -870}
3937 , { -890, -1240, -1060, -890, -1060}
3938 , { -590, -1190, -1010, -590, -1010}
3939 , { -870, -1050, -870, -890, -870}
3940 , { -700, -1300, -1120, -700, -1120}
3941 }

```

```
3942 ,{{ -1030, -1370, -1210, -1030, -1210}}
3943 ,{ -1030, -1370, -1210, -1030, -1210}
3944 ,{ -1340, -1700, -1520, -1340, -1520}
3945 ,{ -1250, -1600, -1420, -1250, -1420}
3946 ,{ -1340, -1700, -1520, -1340, -1520}
3947 }
3948 ,{{ -650, -1240, -1060, -650, -1060}}
3949 ,{ -890, -1240, -1060, -890, -1060}
3950 ,{ -650, -1250, -1070, -650, -1070}
3951 ,{ -890, -1240, -1060, -890, -1060}
3952 ,{ -700, -1300, -1120, -700, -1120}
3953 }
3954 ,{{ -870, -1050, -870, -1340, -870}}
3955 ,{ -1530, -1890, -1710, -1530, -1710}
3956 ,{ -1340, -1700, -1520, -1340, -1520}
3957 ,{ -870, -1050, -870, -1940, -870}
3958 ,{ -1340, -1700, -1520, -1340, -1520}
3959 }
3960 ,{{ -590, -1190, -1010, -590, -1010}}
3961 ,{ -890, -1240, -1060, -890, -1060}
3962 ,{ -590, -1190, -1010, -590, -1010}
3963 ,{ -890, -1240, -1060, -890, -1060}
3964 ,{ -1680, -1790, -1850, -1680, -1850}
3965 }
3966 }
3967 ,{{ -870, -870, -870, -870, -870}}
3968 ,{ -1060, -1060, -1060, -1060, -1060}
3969 ,{ -1010, -1010, -1010, -1010, -1010}
3970 ,{ -870, -870, -870, -870, -870}
3971 ,{ -1120, -1120, -1120, -1120, -1120}
3972 }
3973 ,{{ -1180, -1210, -1180, -1210, -1180}}
3974 ,{ -1210, -1210, -1210, -1210, -1210}
3975 ,{ -1520, -1520, -1520, -1520, -1520}
3976 ,{ -1180, -1420, -1180, -1420, -1180}
3977 ,{ -1520, -1520, -1520, -1520, -1520}
3978 }
3979 ,{{ -1060, -1060, -1060, -1060, -1060}}
3980 ,{ -1060, -1060, -1060, -1060, -1060}
3981 ,{ -1070, -1070, -1070, -1070, -1070}
3982 ,{ -1060, -1060, -1060, -1060, -1060}
3983 ,{ -1120, -1120, -1120, -1120, -1120}
3984 }
3985 ,{{ -870, -870, -870, -870, -870}}
3986 ,{ -1470, -1710, -1470, -1710, -1470}
3987 ,{ -1520, -1520, -1520, -1520, -1520}
3988 ,{ -870, -870, -870, -870, -870}
3989 ,{ -1520, -1520, -1520, -1520, -1520}
3990 }
3991 ,{{ -1010, -1010, -1010, -1010, -1010}}
3992 ,{ -1060, -1060, -1060, -1060, -1060}
3993 ,{ -1010, -1010, -1010, -1010, -1010}
3994 ,{ -1060, -1060, -1060, -1060, -1060}
3995 ,{ -1850, -1850, -1850, -1850, -1850}
3996 }
3997 }
3998 ,{{ 240, -780, -870, 240, -870}}
3999 ,{ 190, -1080, -1060, 190, -1060}
4000 ,{ 240, -780, -1010, 240, -1010}
4001 ,{ 190, -1080, -870, 190, -870}
4002 ,{ 130, -890, -1120, 130, -1120}
4003 }
4004 ,{{ 40, -1220, -1210, 40, -1210}}
4005 ,{ 40, -1220, -1210, 40, -1210}
4006 ,{ -270, -1530, -1520, -270, -1520}
4007 ,{ -1420, -1440, -1420, -1420, -1420}
4008 ,{ -270, -1530, -1520, -270, -1520}
4009 }
4010 ,{{ 190, -840, -1060, 190, -1060}}
4011 ,{ 190, -1080, -1060, 190, -1060}
4012 ,{ 180, -840, -1070, 180, -1070}
4013 ,{ 190, -1080, -1060, 190, -1060}
4014 ,{ 130, -890, -1120, 130, -1120}
4015 }
4016 ,{{ -270, -1530, -870, -270, -870}}
4017 ,{ -1710, -1720, -1710, -1710, -1710}
4018 ,{ -270, -1530, -1520, -270, -1520}
4019 ,{ -870, -2130, -870, -2120, -870}
4020 ,{ -270, -1530, -1520, -270, -1520}
4021 }
4022 ,{{ 240, -780, -1010, 240, -1010}}
4023 ,{ 190, -1080, -1060, 190, -1060}
4024 ,{ 240, -780, -1010, 240, -1010}
4025 ,{ 190, -1080, -1060, 190, -1060}
4026 ,{ -1850, -1870, -1850, -1850, -1850}
4027 }
4028 }
```

```
4029 ,{{{-870, -870, -870, -870, -970}}
4030 ,{-970, -1060, -1060, -1060, -970}
4031 ,{-1010, -1010, -1010, -1010, -1010}
4032 ,{-870, -870, -870, -870, -1060}
4033 ,{-1120, -1120, -1120, -1120, -1120}
4034 }
4035 ,{{-970, -1210, -1180, -1210, -970}}
4036 ,{-970, -1210, -1210, -1210, -970}
4037 ,{-1520, -1520, -1520, -1520, -1520}
4038 ,{-1180, -1420, -1180, -1420, -1420}
4039 ,{-1520, -1520, -1520, -1520, -1520}
4040 }
4041 ,{{-1060, -1060, -1060, -1060, -1060}}
4042 ,{-1060, -1060, -1060, -1060, -1060}
4043 ,{-1070, -1070, -1070, -1070, -1070}
4044 ,{-1060, -1060, -1060, -1060, -1060}
4045 ,{-1120, -1120, -1120, -1120, -1120}
4046 }
4047 ,{{-870, -870, -870, -870, -1520}}
4048 ,{-1470, -1710, -1470, -1710, -1710}
4049 ,{-1520, -1520, -1520, -1520, -1520}
4050 ,{-870, -870, -870, -870, -2120}
4051 ,{-1520, -1520, -1520, -1520, -1520}
4052 }
4053 ,{{-1010, -1010, -1010, -1010, -1010}}
4054 ,{-1060, -1060, -1060, -1060, -1060}
4055 ,{-1010, -1010, -1010, -1010, -1010}
4056 ,{-1060, -1060, -1060, -1060, -1060}
4057 ,{-1850, -1850, -1850, -1850, -1850}
4058 }
4059 }
4060 }
4061 ,{{{{210, -870, -870, 210, -800}}
4062 ,{210, -1040, -1040, 210, -800}
4063 ,{-240, -1490, -1490, -240, -1490}
4064 ,{-160, -870, -870, -160, -870}
4065 ,{-240, -1490, -1490, -240, -1490}
4066 }
4067 ,{{210, -1040, -1040, 210, -800}}
4068 ,{210, -1040, -1040, 210, -800}
4069 ,{-240, -1490, -1490, -240, -1490}
4070 ,{-1990, -2230, -1990, -2060, -1990}
4071 ,{-240, -1490, -1490, -240, -1490}
4072 }
4073 ,{{-160, -1410, -1410, -160, -1410}}
4074 ,{-160, -1410, -1410, -160, -1410}
4075 ,{-460, -1490, -1710, -460, -1710}
4076 ,{-160, -1410, -1410, -160, -1410}
4077 ,{-460, -1490, -1710, -460, -1710}
4078 }
4079 ,{{-240, -870, -870, -240, -870}}
4080 ,{-1520, -1760, -1520, -1580, -1520}
4081 ,{-240, -1490, -1490, -240, -1490}
4082 ,{-870, -870, -870, -870, -870}
4083 ,{-240, -1490, -1490, -240, -1490}
4084 }
4085 ,{{-160, -1410, -1410, -160, -1410}}
4086 ,{-160, -1410, -1410, -160, -1410}
4087 ,{-770, -1800, -2020, -770, -2020}
4088 ,{-160, -1410, -1410, -160, -1410}
4089 ,{-1520, -1640, -1700, -1520, -1700}
4090 }
4091 }
4092 ,{{{{-870, -1050, -870, -870, -870}}
4093 ,{-870, -1220, -1040, -870, -1040}
4094 ,{-1300, -1670, -1490, -1300, -1490}
4095 ,{-870, -1050, -870, -1230, -870}
4096 ,{-1300, -1640, -1490, -1300, -1490}
4097 }
4098 ,{{-870, -1220, -1040, -870, -1040}}
4099 ,{-870, -1220, -1040, -870, -1040}
4100 ,{-1320, -1670, -1490, -1320, -1490}
4101 ,{-2060, -2410, -2230, -2060, -2230}
4102 ,{-1320, -1670, -1490, -1320, -1490}
4103 }
4104 ,{{-1230, -1590, -1410, -1230, -1410}}
4105 ,{-1230, -1590, -1410, -1230, -1410}
4106 ,{-1300, -1890, -1710, -1300, -1710}
4107 ,{-1230, -1590, -1410, -1230, -1410}
4108 ,{-1300, -1890, -1710, -1300, -1710}
4109 }
4110 ,{{-870, -1050, -870, -1320, -870}}
4111 ,{-1580, -1940, -1760, -1580, -1760}
4112 ,{-1320, -1670, -1490, -1320, -1490}
4113 ,{-870, -1050, -870, -1940, -870}
4114 ,{-1320, -1670, -1490, -1320, -1490}
4115 }
```

```
4116 ,{{ -1230, -1590, -1410, -1230, -1410} }  
4117 ,{ -1230, -1590, -1410, -1230, -1410} }  
4118 ,{ -1610, -2200, -2020, -1610, -2020} }  
4119 ,{ -1230, -1590, -1410, -1230, -1410} }  
4120 ,{ -1520, -1640, -1700, -1520, -1700} }  
4121 } }  
4122 } }  
4123 ,{{ -870, -870, -870, -870, -870} }  
4124 ,{ -1040, -1040, -1040, -1040, -1040} }  
4125 ,{ -1490, -1490, -1490, -1490, -1490} }  
4126 ,{ -870, -870, -870, -870, -870} }  
4127 ,{ -1490, -1490, -1490, -1490, -1490} }  
4128 } }  
4129 ,{{ -1040, -1040, -1040, -1040, -1040} }  
4130 ,{ -1040, -1040, -1040, -1040, -1040} }  
4131 ,{ -1490, -1490, -1490, -1490, -1490} }  
4132 ,{ -1990, -2230, -1990, -2230, -1990} }  
4133 ,{ -1490, -1490, -1490, -1490, -1490} }  
4134 } }  
4135 ,{{ -1410, -1410, -1410, -1410, -1410} }  
4136 ,{ -1410, -1410, -1410, -1410, -1410} }  
4137 ,{ -1710, -1710, -1710, -1710, -1710} }  
4138 ,{ -1410, -1410, -1410, -1410, -1410} }  
4139 ,{ -1710, -1710, -1710, -1710, -1710} }  
4140 } }  
4141 ,{{ -870, -870, -870, -870, -870} }  
4142 ,{ -1520, -1760, -1520, -1760, -1520} }  
4143 ,{ -1490, -1490, -1490, -1490, -1490} }  
4144 ,{ -870, -870, -870, -870, -870} }  
4145 ,{ -1490, -1490, -1490, -1490, -1490} }  
4146 } }  
4147 ,{{ -1410, -1410, -1410, -1410, -1410} }  
4148 ,{ -1410, -1410, -1410, -1410, -1410} }  
4149 ,{ -2020, -2020, -2020, -2020, -2020} }  
4150 ,{ -1410, -1410, -1410, -1410, -1410} }  
4151 ,{ -1700, -1700, -1700, -1700, -1700} }  
4152 } }  
4153 } }  
4154 ,{{ 210, -1060, -870, 210, -870} }  
4155 ,{ 210, -1060, -1040, 210, -1040} }  
4156 ,{ -240, -1490, -1490, -240, -1490} }  
4157 ,{ -160, -1420, -870, -160, -870} }  
4158 ,{ -240, -1490, -1490, -240, -1490} }  
4159 } }  
4160 ,{{ 210, -1060, -1040, 210, -1040} }  
4161 ,{ 210, -1060, -1040, 210, -1040} }  
4162 ,{ -240, -1510, -1490, -240, -1490} }  
4163 ,{ -2230, -2250, -2230, -2230, -2230} }  
4164 ,{ -240, -1510, -1490, -240, -1490} }  
4165 } }  
4166 ,{{ -160, -1420, -1410, -160, -1410} }  
4167 ,{ -160, -1420, -1410, -160, -1410} }  
4168 ,{ -460, -1490, -1710, -460, -1710} }  
4169 ,{ -160, -1420, -1410, -160, -1410} }  
4170 ,{ -460, -1490, -1710, -460, -1710} }  
4171 } }  
4172 ,{{ -240, -1510, -870, -240, -870} }  
4173 ,{ -1760, -1770, -1760, -1760, -1760} }  
4174 ,{ -240, -1510, -1490, -240, -1490} }  
4175 ,{ -870, -2130, -870, -2120, -870} }  
4176 ,{ -240, -1510, -1490, -240, -1490} }  
4177 } }  
4178 ,{{ -160, -1420, -1410, -160, -1410} }  
4179 ,{ -160, -1420, -1410, -160, -1410} }  
4180 ,{ -770, -1800, -2020, -770, -2020} }  
4181 ,{ -160, -1420, -1410, -160, -1410} }  
4182 ,{ -1700, -1710, -1700, -1700, -1700} }  
4183 } }  
4184 } }  
4185 ,{{ -800, -870, -870, -870, -800} }  
4186 ,{ -800, -1040, -1040, -1040, -800} }  
4187 ,{ -1490, -1490, -1490, -1490, -1490} }  
4188 ,{ -870, -870, -870, -870, -1410} }  
4189 ,{ -1490, -1490, -1490, -1490, -1490} }  
4190 } }  
4191 ,{{ -800, -1040, -1040, -1040, -800} }  
4192 ,{ -800, -1040, -1040, -1040, -800} }  
4193 ,{ -1490, -1490, -1490, -1490, -1490} }  
4194 ,{ -1990, -2230, -1990, -2230, -2230} }  
4195 ,{ -1490, -1490, -1490, -1490, -1490} }  
4196 } }  
4197 ,{{ -1410, -1410, -1410, -1410, -1410} }  
4198 ,{ -1410, -1410, -1410, -1410, -1410} }  
4199 ,{ -1710, -1710, -1710, -1710, -1710} }  
4200 ,{ -1410, -1410, -1410, -1410, -1410} }  
4201 ,{ -1710, -1710, -1710, -1710, -1710} }  
4202 }
```

```

4203 ,{{ -870, -870, -870, -870, -1490}
4204 ,{ -1520, -1760, -1520, -1760, -1760}
4205 ,{ -1490, -1490, -1490, -1490, -1490}
4206 ,{ -870, -870, -870, -870, -2120}
4207 ,{ -1490, -1490, -1490, -1490, -1490}
4208 }
4209 ,{{ -1410, -1410, -1410, -1410, -1410}
4210 ,{ -1410, -1410, -1410, -1410, -1410}
4211 ,{ -2020, -2020, -2020, -2020, -2020}
4212 ,{ -1410, -1410, -1410, -1410, -1410}
4213 ,{ -1700, -1700, -1700, -1700, -1700}
4214 }
4215 }
4216 }
4217 ,{{{ -710, -710, -710, -710, -710}
4218 ,{ -710, -1780, -1540, -710, -1540}
4219 ,{ -710, -1730, -1960, -710, -1960}
4220 ,{ -710, -710, -710, -710, -710}
4221 ,{ -710, -1730, -1960, -710, -1960}
4222 }
4223 ,{{ -710, -1960, -1730, -710, -1730}
4224 ,{ -890, -2140, -2140, -890, -1900}
4225 ,{ -710, -1960, -1960, -710, -1960}
4226 ,{ -1730, -1970, -1730, -1800, -1730}
4227 ,{ -710, -1960, -1960, -710, -1960}
4228 }
4229 ,{{ -710, -1730, -1960, -710, -1960}
4230 ,{ -710, -1960, -1960, -710, -1960}
4231 ,{ -710, -1730, -1960, -710, -1960}
4232 ,{ -710, -1960, -1960, -710, -1960}
4233 ,{ -710, -1730, -1960, -710, -1960}
4234 }
4235 ,{{ -710, -710, -710, -710, -710}
4236 ,{ -1540, -1780, -1540, -1610, -1540}
4237 ,{ -710, -1960, -1960, -710, -1960}
4238 ,{ -710, -710, -710, -710, -710}
4239 ,{ -710, -1960, -1960, -710, -1960}
4240 }
4241 ,{{ -710, -1730, -1960, -710, -1960}
4242 ,{ -710, -1960, -1960, -710, -1960}
4243 ,{ -710, -1730, -1960, -710, -1960}
4244 ,{ -710, -1960, -1960, -710, -1960}
4245 ,{ -1780, -1900, -1960, -1780, -1960}
4246 }
4247 }
4248 ,{{{ -710, -890, -710, -1540, -710}
4249 ,{ -1610, -1960, -1780, -1610, -1780}
4250 ,{ -1540, -2140, -1960, -1540, -1960}
4251 ,{ -710, -890, -710, -1780, -710}
4252 ,{ -1540, -1900, -1960, -1540, -1960}
4253 }
4254 ,{{ -1780, -2140, -1960, -1780, -1960}
4255 ,{ -1960, -2320, -2140, -1960, -2140}
4256 ,{ -1780, -2140, -1960, -1780, -1960}
4257 ,{ -1800, -2150, -1970, -1800, -1970}
4258 ,{ -1780, -2140, -1960, -1780, -1960}
4259 }
4260 ,{{ -1540, -2140, -1960, -1540, -1960}
4261 ,{ -1780, -2140, -1960, -1780, -1960}
4262 ,{ -1540, -2140, -1960, -1540, -1960}
4263 ,{ -1780, -2140, -1960, -1780, -1960}
4264 ,{ -1540, -2140, -1960, -1540, -1960}
4265 }
4266 ,{{ -710, -890, -710, -1610, -710}
4267 ,{ -1610, -1960, -1780, -1610, -1780}
4268 ,{ -1780, -2140, -1960, -1780, -1960}
4269 ,{ -710, -890, -710, -1780, -710}
4270 ,{ -1780, -2140, -1960, -1780, -1960}
4271 }
4272 ,{{ -1540, -1900, -1960, -1540, -1960}
4273 ,{ -1780, -2140, -1960, -1780, -1960}
4274 ,{ -1540, -2140, -1960, -1540, -1960}
4275 ,{ -1780, -2140, -1960, -1780, -1960}
4276 ,{ -1780, -1900, -1960, -1780, -1960}
4277 }
4278 }
4279 ,{{{ -710, -710, -710, -710, -710}
4280 ,{ -1540, -1780, -1540, -1780, -1540}
4281 ,{ -1960, -1960, -1960, -1960, -1960}
4282 ,{ -710, -710, -710, -710, -710}
4283 ,{ -1960, -1960, -1960, -1960, -1960}
4284 }
4285 ,{{ -1730, -1960, -1730, -1960, -1730}
4286 ,{ -2140, -2140, -2140, -2140, -2140}
4287 ,{ -1960, -1960, -1960, -1960, -1960}
4288 ,{ -1730, -1970, -1730, -1970, -1730}
4289 ,{ -1960, -1960, -1960, -1960, -1960}

```

```

4290      }
4291 ,{{ -1960, -1960, -1960, -1960, -1960}
4292 ,{ -1960, -1960, -1960, -1960, -1960}
4293 ,{ -1960, -1960, -1960, -1960, -1960}
4294 ,{ -1960, -1960, -1960, -1960, -1960}
4295 ,{ -1960, -1960, -1960, -1960, -1960}
4296     }
4297 ,{{ -710, -710, -710, -710, -710}
4298 ,{ -1540, -1780, -1540, -1780, -1540}
4299 ,{ -1960, -1960, -1960, -1960, -1960}
4300 ,{ -710, -710, -710, -710, -710}
4301 ,{ -1960, -1960, -1960, -1960, -1960}
4302     }
4303 ,{{ -1960, -1960, -1960, -1960, -1960}
4304 ,{ -1960, -1960, -1960, -1960, -1960}
4305 ,{ -1960, -1960, -1960, -1960, -1960}
4306 ,{ -1960, -1960, -1960, -1960, -1960}
4307 ,{ -1960, -1960, -1960, -1960, -1960}
4308     }
4309   }
4310 ,{{ { -710, -1730, -710, -710, -710}
4311 ,{ -710, -1800, -1780, -710, -1780}
4312 ,{ -710, -1730, -1960, -710, -1960}
4313 ,{ -710, -1970, -710, -710, -710}
4314 ,{ -710, -1730, -1960, -710, -1960}
4315     }
4316 ,{{ { -710, -1970, -1960, -710, -1960}
4317 ,{ -890, -2150, -2140, -890, -2140}
4318 ,{ -710, -1970, -1960, -710, -1960}
4319 ,{ -1970, -1990, -1970, -1970, -1970}
4320 ,{ -710, -1970, -1960, -710, -1960}
4321     }
4322 ,{{ { -710, -1730, -1960, -710, -1960}
4323 ,{ -710, -1970, -1960, -710, -1960}
4324 ,{ -710, -1730, -1960, -710, -1960}
4325 ,{ -710, -1970, -1960, -710, -1960}
4326 ,{ -710, -1730, -1960, -710, -1960}
4327     }
4328 ,{{ { -710, -1800, -710, -710, -710}
4329 ,{ -1780, -1800, -1780, -1780, -1780}
4330 ,{ -710, -1970, -1960, -710, -1960}
4331 ,{ -710, -1970, -710, -1960, -710}
4332 ,{ -710, -1970, -1960, -710, -1960}
4333     }
4334 ,{{ { -710, -1730, -1960, -710, -1960}
4335 ,{ -710, -1970, -1960, -710, -1960}
4336 ,{ -710, -1730, -1960, -710, -1960}
4337 ,{ -710, -1970, -1960, -710, -1960}
4338 ,{ -1960, -1970, -1960, -1960, -1960}
4339     }
4340   }
4341 ,{{ {{ -710, -710, -710, -710, -1780}
4342 ,{ -1540, -1780, -1540, -1780, -1780}
4343 ,{ -1960, -1960, -1960, -1960, -1960}
4344 ,{ -710, -710, -710, -710, -1960}
4345 ,{ -1960, -1960, -1960, -1960, -1960}
4346     }
4347 ,{{ { -1730, -1960, -1730, -1960, -1900}
4348 ,{ -1900, -2140, -2140, -2140, -1900}
4349 ,{ -1960, -1960, -1960, -1960, -1960}
4350 ,{ -1730, -1970, -1730, -1970, -1970}
4351 ,{ -1960, -1960, -1960, -1960, -1960}
4352     }
4353 ,{{ { -1960, -1960, -1960, -1960, -1960}
4354 ,{ -1960, -1960, -1960, -1960, -1960}
4355 ,{ -1960, -1960, -1960, -1960, -1960}
4356 ,{ -1960, -1960, -1960, -1960, -1960}
4357 ,{ -1960, -1960, -1960, -1960, -1960}
4358     }
4359 ,{{ { -710, -710, -710, -710, -1780}
4360 ,{ -1540, -1780, -1540, -1780, -1780}
4361 ,{ -1960, -1960, -1960, -1960, -1960}
4362 ,{ -710, -710, -710, -710, -1960}
4363 ,{ -1960, -1960, -1960, -1960, -1960}
4364     }
4365 ,{{ { -1960, -1960, -1960, -1960, -1960}
4366 ,{ -1960, -1960, -1960, -1960, -1960}
4367 ,{ -1960, -1960, -1960, -1960, -1960}
4368 ,{ -1960, -1960, -1960, -1960, -1960}
4369 ,{ -1960, -1960, -1960, -1960, -1960}
4370     }
4371   }
4372   }
4373 ,{{ {{ 360, -70, -150, 360, -150}
4374 ,{ 360, -70, -890, 360, -650}
4375 ,{ -150, -1180, -1400, -150, -1400}
4376 ,{ -150, -150, -150, -150, -150}

```

```

4377 , { -150, -1180, -1400, -150, -1400}
4378 }
4379 , {{ 360, -70, -890, 360, -650}
4380 , { 360, -70, -890, 360, -650}
4381 , { -150, -1400, -1400, -150, -1400}
4382 , { -1500, -1600, -1500, -1570, -1500}
4383 , { -150, -1400, -1400, -150, -1400}
4384 }
4385 , {{ -150, -1180, -1400, -150, -1400}
4386 , { -150, -1400, -1400, -150, -1400}
4387 , { -150, -1180, -1400, -150, -1400}
4388 , { -150, -1400, -1400, -150, -1400}
4389 , { -150, -1180, -1400, -150, -1400}
4390 }
4391 , {{ -150, -150, -150, -150, -150}
4392 , { -1670, -1910, -1670, -1740, -1670}
4393 , { -150, -1400, -1400, -150, -1400}
4394 , { -150, -150, -150, -150, -150}
4395 , { -150, -1400, -1400, -150, -1400}
4396 }
4397 , {{ -150, -1180, -1400, -150, -1400}
4398 , { -150, -1400, -1400, -150, -1400}
4399 , { -150, -1180, -1400, -150, -1400}
4400 , { -150, -1400, -1400, -150, -1400}
4401 , { -1230, -1340, -1400, -1230, -1400}
4402 }
4403 }
4404 , {{ -30, -70, -150, -30, -150}
4405 , { -30, -70, -890, -30, -890}
4406 , { -990, -1580, -1400, -990, -1400}
4407 , { -150, -330, -150, -1230, -150}
4408 , { -990, -1340, -1400, -990, -1400}
4409 }
4410 , {{ -30, -70, -890, -30, -890}
4411 , { -30, -70, -890, -30, -890}
4412 , { -1230, -1580, -1400, -1230, -1400}
4413 , { -1570, -1600, -1740, -1570, -1740}
4414 , { -1230, -1580, -1400, -1230, -1400}
4415 }
4416 , {{ -990, -1580, -1400, -990, -1400}
4417 , { -1230, -1580, -1400, -1230, -1400}
4418 , { -990, -1580, -1400, -990, -1400}
4419 , { -1230, -1580, -1400, -1230, -1400}
4420 , { -990, -1580, -1400, -990, -1400}
4421 }
4422 , {{ -150, -330, -150, -1230, -150}
4423 , { -1740, -2090, -1910, -1740, -1910}
4424 , { -1230, -1580, -1400, -1230, -1400}
4425 , { -150, -330, -150, -1230, -150}
4426 , { -1230, -1580, -1400, -1230, -1400}
4427 }
4428 , {{ -990, -1340, -1400, -990, -1400}
4429 , { -1230, -1580, -1400, -1230, -1400}
4430 , { -990, -1580, -1400, -990, -1400}
4431 , { -1230, -1580, -1400, -1230, -1400}
4432 , { -1230, -1340, -1400, -1230, -1400}
4433 }
4434 }
4435 , {{ -150, -150, -150, -150, -150}
4436 , { -890, -890, -890, -890, -890}
4437 , { -1400, -1400, -1400, -1400, -1400}
4438 , { -150, -150, -150, -150, -150}
4439 , { -1400, -1400, -1400, -1400, -1400}
4440 }
4441 , {{ -890, -890, -890, -890, -890}
4442 , { -890, -890, -890, -890, -890}
4443 , { -1400, -1400, -1400, -1400, -1400}
4444 , { -1500, -1740, -1500, -1740, -1500}
4445 , { -1400, -1400, -1400, -1400, -1400}
4446 }
4447 , {{ -1400, -1400, -1400, -1400, -1400}
4448 , { -1400, -1400, -1400, -1400, -1400}
4449 , { -1400, -1400, -1400, -1400, -1400}
4450 , { -1400, -1400, -1400, -1400, -1400}
4451 , { -1400, -1400, -1400, -1400, -1400}
4452 }
4453 , {{ -150, -150, -150, -150, -150}
4454 , { -1670, -1910, -1670, -1910, -1670}
4455 , { -1400, -1400, -1400, -1400, -1400}
4456 , { -150, -150, -150, -150, -150}
4457 , { -1400, -1400, -1400, -1400, -1400}
4458 }
4459 , {{ -1400, -1400, -1400, -1400, -1400}
4460 , { -1400, -1400, -1400, -1400, -1400}
4461 , { -1400, -1400, -1400, -1400, -1400}
4462 , { -1400, -1400, -1400, -1400, -1400}
4463 , { -1400, -1400, -1400, -1400, -1400}

```

```

4464      }
4465  }
4466 ,{{{{ 360, -910, -150, 360, -150}
4467 ,{ 360, -910, -890, 360, -890}
4468 ,{ -150, -1180, -1400, -150, -1400}
4469 ,{ -150, -1420, -150, -150, -150}
4470 ,{ -150, -1180, -1400, -150, -1400}
4471 }
4472 ,{{ 360, -910, -890, 360, -890}
4473 ,{ 360, -910, -890, 360, -890}
4474 ,{ -150, -1420, -1400, -150, -1400}
4475 ,{ -1740, -3040, -1740, -1740, -1740}
4476 ,{ -150, -1420, -1400, -150, -1400}
4477 }
4478 ,{{ -150, -1180, -1400, -150, -1400}
4479 ,{ -150, -1420, -1400, -150, -1400}
4480 ,{ -150, -1180, -1400, -150, -1400}
4481 ,{ -150, -1420, -1400, -150, -1400}
4482 ,{ -150, -1180, -1400, -150, -1400}
4483 }
4484 ,{{ -150, -1420, -150, -150, -150}
4485 ,{ -1910, -1930, -1910, -1910, -1910}
4486 ,{ -150, -1420, -1400, -150, -1400}
4487 ,{ -150, -1420, -150, -1400, -150}
4488 ,{ -150, -1420, -1400, -150, -1400}
4489 }
4490 ,{{ -150, -1180, -1400, -150, -1400}
4491 ,{ -150, -1420, -1400, -150, -1400}
4492 ,{ -150, -1180, -1400, -150, -1400}
4493 ,{ -150, -1420, -1400, -150, -1400}
4494 ,{ -1400, -1420, -1400, -1400, -1400}
4495 }
4496 }
4497 ,{{{{ -150, -150, -150, -150, -650}
4498 ,{ -650, -890, -890, -890, -650}
4499 ,{ -1400, -1400, -1400, -1400, -1400}
4500 ,{ -150, -150, -150, -150, -1400}
4501 ,{ -1400, -1400, -1400, -1400, -1400}
4502 }
4503 ,{{ -650, -890, -890, -890, -650}
4504 ,{ -650, -890, -890, -890, -650}
4505 ,{ -1400, -1400, -1400, -1400, -1400}
4506 ,{ -1500, -1740, -1500, -1740, -1740}
4507 ,{ -1400, -1400, -1400, -1400, -1400}
4508 }
4509 ,{{ -1400, -1400, -1400, -1400, -1400}
4510 ,{ -1400, -1400, -1400, -1400, -1400}
4511 ,{ -1400, -1400, -1400, -1400, -1400}
4512 ,{ -1400, -1400, -1400, -1400, -1400}
4513 ,{ -1400, -1400, -1400, -1400, -1400}
4514 }
4515 ,{{{{ -150, -150, -150, -150, -1400}
4516 ,{ -1670, -1910, -1670, -1910, -1910}
4517 ,{ -1400, -1400, -1400, -1400, -1400}
4518 ,{ -150, -150, -150, -150, -1400}
4519 ,{ -1400, -1400, -1400, -1400, -1400}
4520 }
4521 ,{{{{ -1400, -1400, -1400, -1400, -1400}
4522 ,{ -1400, -1400, -1400, -1400, -1400}
4523 ,{ -1400, -1400, -1400, -1400, -1400}
4524 ,{ -1400, -1400, -1400, -1400, -1400}
4525 ,{ -1400, -1400, -1400, -1400, -1400}
4526 }
4527 }
4528 }
4529 ,{{{{ 940, 220, 220, 940, 220}
4530 ,{ 940, -310, -310, 940, -70}
4531 ,{ 640, -380, -610, 640, -610}
4532 ,{ 650, 220, 220, 650, 220}
4533 ,{ 640, -380, -610, 640, -610}
4534 }
4535 ,{{ 940, -310, -310, 940, -70}
4536 ,{ 940, -310, -310, 940, -70}
4537 ,{ 630, -620, -620, 630, -620}
4538 ,{ -1460, -1700, -1460, -1520, -1460}
4539 ,{ 630, -620, -620, 630, -620}
4540 }
4541 ,{{ 650, -380, -600, 650, -600}
4542 ,{ 650, -600, -600, 650, -600}
4543 ,{ 640, -380, -610, 640, -610}
4544 ,{ 650, -600, -600, 650, -600}
4545 ,{ 640, -380, -610, 640, -610}
4546 }
4547 ,{{ 630, 220, 220, 630, 220}
4548 ,{ -1280, -1520, -1280, -1340, -1280}
4549 ,{ 630, -620, -620, 630, -620}
4550 ,{ 220, 220, 220, 220, 220}

```

```

4551 , { 630, -620, -620, 630, -620}
4552 }
4553 , { { 650, -380, -600, 650, -600}
4554 , { 650, -600, -600, 650, -600}
4555 , { 640, -380, -610, 640, -610}
4556 , { 650, -600, -600, 650, -600}
4557 , { -1410, -1530, -1590, -1410, -1590}
4558 }
4559 }
4560 , { { { 220, 40, 220, -130, 220}
4561 , { -130, -490, -310, -130, -310}
4562 , { -190, -790, -610, -190, -610}
4563 , { 220, 40, 220, -430, 220}
4564 , { -190, -790, -610, -190, -610}
4565 }
4566 , { { -130, -490, -310, -130, -310}
4567 , { -130, -490, -310, -130, -310}
4568 , { -440, -800, -620, -440, -620}
4569 , { -1520, -1880, -1700, -1520, -1700}
4570 , { -440, -800, -620, -440, -620}
4571 }
4572 , { { -190, -780, -600, -190, -600}
4573 , { -430, -780, -600, -430, -600}
4574 , { -190, -790, -610, -190, -610}
4575 , { -430, -780, -600, -430, -600}
4576 , { -190, -790, -610, -190, -610}
4577 }
4578 , { { 220, 40, 220, -440, 220}
4579 , { -1340, -1700, -1520, -1340, -1520}
4580 , { -440, -800, -620, -440, -620}
4581 , { 220, 40, 220, -850, 220}
4582 , { -440, -800, -620, -440, -620}
4583 }
4584 , { { -190, -780, -600, -190, -600}
4585 , { -430, -780, -600, -430, -600}
4586 , { -190, -790, -610, -190, -610}
4587 , { -430, -780, -600, -430, -600}
4588 , { -1410, -1530, -1590, -1410, -1590}
4589 }
4590 }
4591 , { { { 220, 220, 220, 220, 220}
4592 , { -310, -310, -310, -310, -310}
4593 , { -610, -610, -610, -610, -610}
4594 , { 220, 220, 220, 220, 220}
4595 , { -610, -610, -610, -610, -610}
4596 }
4597 , { { -310, -310, -310, -310, -310}
4598 , { -310, -310, -310, -310, -310}
4599 , { -620, -620, -620, -620, -620}
4600 , { -1460, -1700, -1460, -1700, -1460}
4601 , { -620, -620, -620, -620, -620}
4602 }
4603 , { { -600, -600, -600, -600, -600}
4604 , { -600, -600, -600, -600, -600}
4605 , { -610, -610, -610, -610, -610}
4606 , { -600, -600, -600, -600, -600}
4607 , { -610, -610, -610, -610, -610}
4608 }
4609 , { { 220, 220, 220, 220, 220}
4610 , { -1280, -1520, -1280, -1520, -1280}
4611 , { -620, -620, -620, -620, -620}
4612 , { 220, 220, 220, 220, 220}
4613 , { -620, -620, -620, -620, -620}
4614 }
4615 , { { -600, -600, -600, -600, -600}
4616 , { -600, -600, -600, -600, -600}
4617 , { -610, -610, -610, -610, -610}
4618 , { -600, -600, -600, -600, -600}
4619 , { -1590, -1590, -1590, -1590, -1590}
4620 }
4621 }
4622 , { { { 940, -320, 220, 940, 220}
4623 , { 940, -320, -310, 940, -310}
4624 , { 640, -380, -610, 640, -610}
4625 , { 650, -620, 220, 650, 220}
4626 , { 640, -380, -610, 640, -610}
4627 }
4628 , { { 940, -320, -310, 940, -310}
4629 , { 940, -320, -310, 940, -310}
4630 , { 630, -630, -620, 630, -620}
4631 , { -1700, -1710, -1700, -1700, -1700}
4632 , { 630, -630, -620, 630, -620}
4633 }
4634 , { { 650, -380, -600, 650, -600}
4635 , { 650, -620, -600, 650, -600}
4636 , { 640, -380, -610, 640, -610}
4637 , { 650, -620, -600, 650, -600}

```

```

4638 , { 640, -380, -610, 640, -610}
4639 }
4640 , {{ 630, -630, 220, 630, 220}
4641 , { -1520, -1530, -1520, -1520, -1520}
4642 , { 630, -630, -620, 630, -620}
4643 , { 220, -1040, 220, -1030, 220}
4644 , { 630, -630, -620, 630, -620}
4645 }
4646 , {{ 650, -380, -600, 650, -600}
4647 , { 650, -620, -600, 650, -600}
4648 , { 640, -380, -610, 640, -610}
4649 , { 650, -620, -600, 650, -600}
4650 , { -1590, -1600, -1590, -1590, -1590}
4651 }
4652 }
4653 , {{{ 220, 220, 220, 220, -70}
4654 , { -70, -310, -310, -310, -70}
4655 , { -610, -610, -610, -610, -610}
4656 , { 220, 220, 220, 220, -600}
4657 , { -610, -610, -610, -610, -610}
4658 }
4659 , {{ -70, -310, -310, -310, -70}
4660 , { -70, -310, -310, -310, -70}
4661 , { -620, -620, -620, -620, -620}
4662 , { -1460, -1700, -1460, -1700, -1700}
4663 , { -620, -620, -620, -620, -620}
4664 }
4665 , {{ -600, -600, -600, -600, -600}
4666 , { -600, -600, -600, -600, -600}
4667 , { -610, -610, -610, -610, -610}
4668 , { -600, -600, -600, -600, -600}
4669 , { -610, -610, -610, -610, -610}
4670 }
4671 , {{ 220, 220, 220, 220, -620}
4672 , { -1280, -1520, -1280, -1520, -1520}
4673 , { -620, -620, -620, -620, -620}
4674 , { 220, 220, 220, 220, -1030}
4675 , { -620, -620, -620, -620, -620}
4676 }
4677 , {{ -600, -600, -600, -600, -600}
4678 , { -600, -600, -600, -600, -600}
4679 , { -610, -610, -610, -610, -610}
4680 , { -600, -600, -600, -600, -600}
4681 , { -1590, -1590, -1590, -1590, -1590}
4682 }
4683 }
4684 }
4685 , {{{{ 1010, 410, 410, 1010, 410}
4686 , { 1010, -240, -240, 1010, 0}
4687 , { 880, -150, -370, 880, -370}
4688 , { 880, 410, 410, 880, 410}
4689 , { 750, -280, -500, 750, -500}
4690 }
4691 , {{ 1010, -240, -240, 1010, 0}
4692 , { 1010, -240, -240, 1010, 0}
4693 , { 730, -520, -520, 730, -520}
4694 , { -1410, -1650, -1410, -1470, -1410}
4695 , { 730, -520, -520, 730, -520}
4696 }
4697 , {{ 880, -150, -370, 880, -370}
4698 , { 880, -370, -370, 880, -370}
4699 , { 880, -150, -370, 880, -370}
4700 , { 880, -370, -370, 880, -370}
4701 , { 750, -280, -500, 750, -500}
4702 }
4703 , {{ 730, 410, 410, 730, 410}
4704 , { -1710, -1950, -1710, -1770, -1710}
4705 , { 730, -520, -520, 730, -520}
4706 , { 410, 410, 410, 410, 410}
4707 , { 730, -520, -520, 730, -520}
4708 }
4709 , {{ 880, -370, -370, 880, -370}
4710 , { 880, -370, -370, 880, -370}
4711 , { 440, -590, -810, 440, -810}
4712 , { 880, -370, -370, 880, -370}
4713 , { -1140, -1250, -1310, -1140, -1310}
4714 }
4715 }
4716 , {{{ 410, 230, 410, 40, 410}
4717 , { -70, -420, -240, -70, -240}
4718 , { 40, -550, -370, 40, -370}
4719 , { 410, 230, 410, -200, 410}
4720 , { -90, -680, -500, -90, -500}
4721 }
4722 , {{ -70, -420, -240, -70, -240}
4723 , { -70, -420, -240, -70, -240}
4724 , { -350, -700, -520, -350, -520}

```

```
4725 ,{ -1470, -1830, -1650, -1470, -1650}
4726 ,{ -350, -700, -520, -350, -520}
4727 }
4728 ,{{ 40, -550, -370, 40, -370}
4729 ,{ -200, -550, -370, -200, -370}
4730 ,{ 40, -550, -370, 40, -370}
4731 ,{ -200, -550, -370, -200, -370}
4732 ,{ -90, -680, -500, -90, -500}
4733 }
4734 ,{{ 410, 230, 410, -350, 410}
4735 ,{ -1770, -2130, -1950, -1770, -1950}
4736 ,{ -350, -700, -520, -350, -520}
4737 ,{ 410, 230, 410, -670, 410}
4738 ,{ -350, -700, -520, -350, -520}
4739 }
4740 ,{{ -200, -550, -370, -200, -370}
4741 ,{ -200, -550, -370, -200, -370}
4742 ,{ -400, -990, -810, -400, -810}
4743 ,{ -200, -550, -370, -200, -370}
4744 ,{ -1140, -1250, -1310, -1140, -1310}
4745 }
4746 }
4747 ,{{{{ 410, 410, 410, 410, 410}
4748 ,{ -240, -240, -240, -240, -240}
4749 ,{ -370, -370, -370, -370, -370}
4750 ,{ 410, 410, 410, 410, 410}
4751 ,{ -500, -500, -500, -500, -500}
4752 }
4753 ,{{ -240, -240, -240, -240, -240}
4754 ,{ -240, -240, -240, -240, -240}
4755 ,{ -520, -520, -520, -520, -520}
4756 ,{ -1410, -1650, -1410, -1650, -1410}
4757 ,{ -520, -520, -520, -520, -520}
4758 }
4759 ,{{ -370, -370, -370, -370, -370}
4760 ,{ -370, -370, -370, -370, -370}
4761 ,{ -370, -370, -370, -370, -370}
4762 ,{ -370, -370, -370, -370, -370}
4763 ,{ -500, -500, -500, -500, -500}
4764 }
4765 ,{{{{ 410, 410, 410, 410, 410}
4766 ,{ -1710, -1950, -1710, -1950, -1710}
4767 ,{ -520, -520, -520, -520, -520}
4768 ,{ 410, 410, 410, 410, 410}
4769 ,{ -520, -520, -520, -520, -520}
4770 }
4771 ,{{ -370, -370, -370, -370, -370}
4772 ,{ -370, -370, -370, -370, -370}
4773 ,{ -810, -810, -810, -810, -810}
4774 ,{ -370, -370, -370, -370, -370}
4775 ,{ -1310, -1310, -1310, -1310, -1310}
4776 }
4777 }
4778 ,{{{{ 1010, -150, 410, 1010, 410}
4779 ,{ 1010, -260, -240, 1010, -240}
4780 ,{ 880, -150, -370, 880, -370}
4781 ,{ 880, -390, 410, 880, 410}
4782 ,{ 750, -280, -500, 750, -500}
4783 }
4784 ,{{ 1010, -260, -240, 1010, -240}
4785 ,{ 1010, -260, -240, 1010, -240}
4786 ,{ 730, -540, -520, 730, -520}
4787 ,{ -1650, -1660, -1650, -1650, -1650}
4788 ,{ 730, -540, -520, 730, -520}
4789 }
4790 ,{{ 880, -150, -370, 880, -370}
4791 ,{ 880, -390, -370, 880, -370}
4792 ,{ 880, -150, -370, 880, -370}
4793 ,{ 880, -390, -370, 880, -370}
4794 ,{ 750, -280, -500, 750, -500}
4795 }
4796 ,{{ 730, -540, 410, 730, 410}
4797 ,{ -1950, -1960, -1950, -1950, -1950}
4798 ,{ 730, -540, -520, 730, -520}
4799 ,{ 410, -860, 410, -840, 410}
4800 ,{ 730, -540, -520, 730, -520}
4801 }
4802 ,{{ 880, -390, -370, 880, -370}
4803 ,{ 880, -390, -370, 880, -370}
4804 ,{ 440, -590, -810, 440, -810}
4805 ,{ 880, -390, -370, 880, -370}
4806 ,{ -1310, -1330, -1310, -1310, -1310}
4807 }
4808 }
4809 ,{{{{ 410, 410, 410, 410, 0}
4810 ,{ 0, -240, -240, -240, 0}
4811 ,{ -370, -370, -370, -370, -370}
```

```
4812 , { 410, 410, 410, 410, -370}
4813 , { -500, -500, -500, -500, -500}
4814 }
4815 , {{ 0, -240, -240, -240, 0}
4816 , { 0, -240, -240, -240, 0}
4817 , { -520, -520, -520, -520, -520}
4818 , { -1410, -1650, -1410, -1650, -1650}
4819 , { -520, -520, -520, -520, -520}
4820 }
4821 , {{ -370, -370, -370, -370, -370}
4822 , { -370, -370, -370, -370, -370}
4823 , { -370, -370, -370, -370, -370}
4824 , { -370, -370, -370, -370, -370}
4825 , { -500, -500, -500, -500, -500}
4826 }
4827 , {{ 410, 410, 410, 410, -520}
4828 , { -1710, -1950, -1710, -1950, -1950}
4829 , { -520, -520, -520, -520, -520}
4830 , { 410, 410, 410, 410, -840}
4831 , { -520, -520, -520, -520, -520}
4832 }
4833 , {{ -370, -370, -370, -370, -370}
4834 , { -370, -370, -370, -370, -370}
4835 , { -810, -810, -810, -810, -810}
4836 , { -370, -370, -370, -370, -370}
4837 , { -1310, -1310, -1310, -1310, -1310}
4838 }
4839 }
4840 }
4841 , {{{ 1010, 410, 410, 1010, 410}
4842 , { 1010, -70, -240, 1010, 0}
4843 , { 880, -150, -370, 880, -370}
4844 , { 880, 410, 410, 880, 410}
4845 , { 750, -280, -500, 750, -500}
4846 }
4847 , {{ 1010, -70, -240, 1010, 0}
4848 , { 1010, -70, -240, 1010, 0}
4849 , { 730, -520, -520, 730, -520}
4850 , { -1180, -1420, -1180, -1250, -1180}
4851 , { 730, -520, -520, 730, -520}
4852 }
4853 , {{ 880, -150, -370, 880, -370}
4854 , { 880, -370, -370, 880, -370}
4855 , { 880, -150, -370, 880, -370}
4856 , { 880, -370, -370, 880, -370}
4857 , { 750, -280, -500, 750, -500}
4858 }
4859 , {{ 730, 410, 410, 730, 410}
4860 , { -1280, -1520, -1280, -1340, -1280}
4861 , { 730, -520, -520, 730, -520}
4862 , { 410, 410, 410, 410, 410}
4863 , { 730, -520, -520, 730, -520}
4864 }
4865 , {{ 880, -370, -370, 880, -370}
4866 , { 880, -370, -370, 880, -370}
4867 , { 640, -380, -610, 640, -610}
4868 , { 880, -370, -370, 880, -370}
4869 , { -1140, -1250, -1310, -1140, -1310}
4870 }
4871 }
4872 , {{{ 410, 230, 410, 40, 410}
4873 , { -30, -70, -240, -30, -240}
4874 , { 40, -550, -370, 40, -370}
4875 , { 410, 230, 410, -200, 410}
4876 , { -90, -680, -500, -90, -500}
4877 }
4878 , {{ -30, -70, -240, -30, -240}
4879 , { -30, -70, -240, -30, -240}
4880 , { -350, -700, -520, -350, -520}
4881 , { -1250, -1600, -1420, -1250, -1420}
4882 , { -350, -700, -520, -350, -520}
4883 }
4884 , {{ 40, -550, -370, 40, -370}
4885 , { -200, -550, -370, -200, -370}
4886 , { 40, -550, -370, 40, -370}
4887 , { -200, -550, -370, -200, -370}
4888 , { -90, -680, -500, -90, -500}
4889 }
4890 , {{ 410, 230, 410, -350, 410}
4891 , { -1340, -1700, -1520, -1340, -1520}
4892 , { -350, -700, -520, -350, -520}
4893 , { 410, 230, 410, -670, 410}
4894 , { -350, -700, -520, -350, -520}
4895 }
4896 , {{ -190, -550, -370, -190, -370}
4897 , { -200, -550, -370, -200, -370}
4898 , { -190, -790, -610, -190, -610}
```

```

4899 , { -200, -550, -370, -200, -370}
4900 , { -1140, -1250, -1310, -1140, -1310}
4901 }
4902 }
4903 , {{ 410, 410, 410, 410, 410}
4904 , { -240, -240, -240, -240, -240}
4905 , { -370, -370, -370, -370, -370}
4906 , { 410, 410, 410, 410, 410}
4907 , { -500, -500, -500, -500, -500}
4908 }
4909 , {{ -240, -240, -240, -240, -240}
4910 , { -240, -240, -240, -240, -240}
4911 , { -520, -520, -520, -520, -520}
4912 , { -1180, -1420, -1180, -1420, -1180}
4913 , { -520, -520, -520, -520, -520}
4914 }
4915 , {{ -370, -370, -370, -370, -370}
4916 , { -370, -370, -370, -370, -370}
4917 , { -370, -370, -370, -370, -370}
4918 , { -370, -370, -370, -370, -370}
4919 , { -500, -500, -500, -500, -500}
4920 }
4921 , {{ 410, 410, 410, 410, 410}
4922 , { -1280, -1520, -1280, -1520, -1280}
4923 , { -520, -520, -520, -520, -520}
4924 , { 410, 410, 410, 410, 410}
4925 , { -520, -520, -520, -520, -520}
4926 }
4927 , {{ -370, -370, -370, -370, -370}
4928 , { -370, -370, -370, -370, -370}
4929 , { -610, -610, -610, -610, -610}
4930 , { -370, -370, -370, -370, -370}
4931 , { -1310, -1310, -1310, -1310, -1310}
4932 }
4933 }
4934 , {{ 1010, -150, 410, 1010, 410}
4935 , { 1010, -260, -240, 1010, -240}
4936 , { 880, -150, -370, 880, -370}
4937 , { 880, -390, 410, 880, 410}
4938 , { 750, -280, -500, 750, -500}
4939 }
4940 , {{ 1010, -260, -240, 1010, -240}
4941 , { 1010, -260, -240, 1010, -240}
4942 , { 730, -540, -520, 730, -520}
4943 , { -1420, -1440, -1420, -1420, -1420}
4944 , { 730, -540, -520, 730, -520}
4945 }
4946 , {{ 880, -150, -370, 880, -370}
4947 , { 880, -390, -370, 880, -370}
4948 , { 880, -150, -370, 880, -370}
4949 , { 880, -390, -370, 880, -370}
4950 , { 750, -280, -500, 750, -500}
4951 }
4952 , {{ 730, -540, 410, 730, 410}
4953 , { -1520, -1530, -1520, -1520, -1520}
4954 , { 730, -540, -520, 730, -520}
4955 , { 410, -860, 410, -840, 410}
4956 , { 730, -540, -520, 730, -520}
4957 }
4958 , {{ 880, -380, -370, 880, -370}
4959 , { 880, -390, -370, 880, -370}
4960 , { 640, -380, -610, 640, -610}
4961 , { 880, -390, -370, 880, -370}
4962 , { -1310, -1330, -1310, -1310, -1310}
4963 }
4964 }
4965 , {{ 410, 410, 410, 410, 0}
4966 , { 0, -240, -240, -240, 0}
4967 , { -370, -370, -370, -370, -370}
4968 , { 410, 410, 410, 410, -370}
4969 , { -500, -500, -500, -500, -500}
4970 }
4971 , {{ 0, -240, -240, -240, 0}
4972 , { 0, -240, -240, -240, 0}
4973 , { -520, -520, -520, -520, -520}
4974 , { -1180, -1420, -1180, -1420, -1420}
4975 , { -520, -520, -520, -520, -520}
4976 }
4977 , {{ -370, -370, -370, -370, -370}
4978 , { -370, -370, -370, -370, -370}
4979 , { -370, -370, -370, -370, -370}
4980 , { -370, -370, -370, -370, -370}
4981 , { -500, -500, -500, -500, -500}
4982 }
4983 , {{ 410, 410, 410, 410, -520}
4984 , { -1280, -1520, -1280, -1520, -1520}
4985 , { -520, -520, -520, -520, -520}

```

```
4986 , { 410, 410, 410, 410, -840}
4987 , { -520, -520, -520, -520, -520}
4988 }
4989 , {{ -370, -370, -370, -370, -370}
4990 , { -370, -370, -370, -370, -370}
4991 , { -610, -610, -610, -610, -610}
4992 , { -370, -370, -370, -370, -370}
4993 , { -1310, -1310, -1310, -1310, -1310}
4994 }
4995 }
4996 }
4997 }
4998 , {{{{ INF, INF, INF, INF, INF
4999 , { INF, INF, INF, INF, INF
5000 , { INF, INF, INF, INF, INF
5001 , { INF, INF, INF, INF, INF
5002 , { INF, INF, INF, INF, INF
5003 }
5004 , {{ INF, INF, INF, INF, INF
5005 , { INF, INF, INF, INF, INF
5006 , { INF, INF, INF, INF, INF
5007 , { INF, INF, INF, INF, INF
5008 , { INF, INF, INF, INF, INF
5009 }
5010 , {{ INF, INF, INF, INF, INF
5011 , { INF, INF, INF, INF, INF
5012 , { INF, INF, INF, INF, INF
5013 , { INF, INF, INF, INF, INF
5014 , { INF, INF, INF, INF, INF
5015 }
5016 , {{ INF, INF, INF, INF, INF
5017 , { INF, INF, INF, INF, INF
5018 , { INF, INF, INF, INF, INF
5019 , { INF, INF, INF, INF, INF
5020 , { INF, INF, INF, INF, INF
5021 }
5022 , {{ INF, INF, INF, INF, INF
5023 , { INF, INF, INF, INF, INF
5024 , { INF, INF, INF, INF, INF
5025 , { INF, INF, INF, INF, INF
5026 , { INF, INF, INF, INF, INF
5027 }
5028 }
5029 , {{{ INF, INF, INF, INF, INF
5030 , { INF, INF, INF, INF, INF
5031 , { INF, INF, INF, INF, INF
5032 , { INF, INF, INF, INF, INF
5033 , { INF, INF, INF, INF, INF
5034 }
5035 , {{ INF, INF, INF, INF, INF
5036 , { INF, INF, INF, INF, INF
5037 , { INF, INF, INF, INF, INF
5038 , { INF, INF, INF, INF, INF
5039 , { INF, INF, INF, INF, INF
5040 }
5041 , {{ INF, INF, INF, INF, INF
5042 , { INF, INF, INF, INF, INF
5043 , { INF, INF, INF, INF, INF
5044 , { INF, INF, INF, INF, INF
5045 , { INF, INF, INF, INF, INF
5046 }
5047 , {{ INF, INF, INF, INF, INF
5048 , { INF, INF, INF, INF, INF
5049 , { INF, INF, INF, INF, INF
5050 , { INF, INF, INF, INF, INF
5051 , { INF, INF, INF, INF, INF
5052 }
5053 , {{ INF, INF, INF, INF, INF
5054 , { INF, INF, INF, INF, INF
5055 , { INF, INF, INF, INF, INF
5056 , { INF, INF, INF, INF, INF
5057 , { INF, INF, INF, INF, INF
5058 }
5059 }
5060 , {{{ INF, INF, INF, INF, INF
5061 , { INF, INF, INF, INF, INF
5062 , { INF, INF, INF, INF, INF
5063 , { INF, INF, INF, INF, INF
5064 , { INF, INF, INF, INF, INF
5065 }
5066 , {{ INF, INF, INF, INF, INF
5067 , { INF, INF, INF, INF, INF
5068 , { INF, INF, INF, INF, INF
5069 , { INF, INF, INF, INF, INF
5070 , { INF, INF, INF, INF, INF
5071 }
5072 , {{ INF, INF, INF, INF, INF
```

```
5073 , { INF, INF, INF, INF, INF }
5074 , { INF, INF, INF, INF, INF }
5075 , { INF, INF, INF, INF, INF }
5076 , { INF, INF, INF, INF, INF }
5077 }
5078 , {{ INF, INF, INF, INF, INF }
5079 , { INF, INF, INF, INF, INF }
5080 , { INF, INF, INF, INF, INF }
5081 , { INF, INF, INF, INF, INF }
5082 , { INF, INF, INF, INF, INF }
5083 }
5084 , {{ INF, INF, INF, INF, INF }
5085 , { INF, INF, INF, INF, INF }
5086 , { INF, INF, INF, INF, INF }
5087 , { INF, INF, INF, INF, INF }
5088 , { INF, INF, INF, INF, INF }
5089 }
5090 }
5091 , {{{ INF, INF, INF, INF, INF }
5092 , { INF, INF, INF, INF, INF }
5093 , { INF, INF, INF, INF, INF }
5094 , { INF, INF, INF, INF, INF }
5095 , { INF, INF, INF, INF, INF }
5096 }
5097 , {{ INF, INF, INF, INF, INF }
5098 , { INF, INF, INF, INF, INF }
5099 , { INF, INF, INF, INF, INF }
5100 , { INF, INF, INF, INF, INF }
5101 , { INF, INF, INF, INF, INF }
5102 }
5103 , {{ INF, INF, INF, INF, INF }
5104 , { INF, INF, INF, INF, INF }
5105 , { INF, INF, INF, INF, INF }
5106 , { INF, INF, INF, INF, INF }
5107 , { INF, INF, INF, INF, INF }
5108 }
5109 , {{ INF, INF, INF, INF, INF }
5110 , { INF, INF, INF, INF, INF }
5111 , { INF, INF, INF, INF, INF }
5112 , { INF, INF, INF, INF, INF }
5113 , { INF, INF, INF, INF, INF }
5114 }
5115 , {{ INF, INF, INF, INF, INF }
5116 , { INF, INF, INF, INF, INF }
5117 , { INF, INF, INF, INF, INF }
5118 , { INF, INF, INF, INF, INF }
5119 , { INF, INF, INF, INF, INF }
5120 }
5121 }
5122 , {{{ INF, INF, INF, INF, INF }
5123 , { INF, INF, INF, INF, INF }
5124 , { INF, INF, INF, INF, INF }
5125 , { INF, INF, INF, INF, INF }
5126 , { INF, INF, INF, INF, INF }
5127 }
5128 , {{ INF, INF, INF, INF, INF }
5129 , { INF, INF, INF, INF, INF }
5130 , { INF, INF, INF, INF, INF }
5131 , { INF, INF, INF, INF, INF }
5132 , { INF, INF, INF, INF, INF }
5133 }
5134 , {{ INF, INF, INF, INF, INF }
5135 , { INF, INF, INF, INF, INF }
5136 , { INF, INF, INF, INF, INF }
5137 , { INF, INF, INF, INF, INF }
5138 , { INF, INF, INF, INF, INF }
5139 }
5140 , {{ INF, INF, INF, INF, INF }
5141 , { INF, INF, INF, INF, INF }
5142 , { INF, INF, INF, INF, INF }
5143 , { INF, INF, INF, INF, INF }
5144 , { INF, INF, INF, INF, INF }
5145 }
5146 , {{ INF, INF, INF, INF, INF }
5147 , { INF, INF, INF, INF, INF }
5148 , { INF, INF, INF, INF, INF }
5149 , { INF, INF, INF, INF, INF }
5150 , { INF, INF, INF, INF, INF }
5151 }
5152 }
5153 }
5154 , {{{ 800, 200, -310, 800, -310 }
5155 , { 740, 0, -510, 740, -410 }
5156 , { 800, 50, -450, 800, -450 }
5157 , { 740, 200, -310, 740, -310 }
5158 , { 690, -50, -560, 690, -560 }
5159 }
```

```
5160 , {{ 600, -140, -630, 600, -410} }
5161 , { 600, -140, -650, 600, -410}
5162 , { 290, -450, -960, 290, -960}
5163 , { -360, -360, -630, -870, -630}
5164 , { 290, -450, -960, 290, -960}
5165 }
5166 , {{ 740, 0, -510, 740, -510} }
5167 , { 740, 0, -510, 740, -510}
5168 , { 740, 0, -510, 740, -510}
5169 , { 740, 0, -510, 740, -510}
5170 , { 690, -50, -560, 690, -560}
5171 }
5172 , {{ 290, 200, -310, 290, -310} }
5173 , { -640, -640, -910, -1150, -910}
5174 , { 290, -450, -960, 290, -960}
5175 , { 200, 200, -310, -310, -310}
5176 , { 290, -450, -960, 290, -960}
5177 }
5178 , {{ 800, 50, -450, 800, -450} }
5179 , { 740, 0, -510, 740, -510}
5180 , { 800, 50, -450, 800, -450}
5181 , { 740, 0, -510, 740, -510}
5182 , { -550, -550, -1300, -1300, -1300}
5183 }
5184 }
5185 , {{ { 200, 200, -310, -720, -310} }
5186 , { 0, 0, -510, -1020, -510}
5187 , { 50, 50, -450, -720, -450}
5188 , { 200, 200, -310, -1020, -310}
5189 , { -50, -50, -560, -830, -560}
5190 }
5191 , {{ -140, -140, -650, -1160, -650} }
5192 , { -140, -140, -650, -1160, -650}
5193 , { -450, -450, -960, -1470, -960}
5194 , { -360, -360, -870, -1380, -870}
5195 , { -450, -450, -960, -1470, -960}
5196 }
5197 , {{ 0, 0, -510, -780, -510} }
5198 , { 0, 0, -510, -1020, -510}
5199 , { 0, 0, -510, -780, -510}
5200 , { 0, 0, -510, -1020, -510}
5201 , { -50, -50, -560, -830, -560}
5202 }
5203 , {{ { 200, 200, -310, -1470, -310} }
5204 , { -640, -640, -1150, -1660, -1150}
5205 , { -450, -450, -960, -1470, -960}
5206 , { 200, 200, -310, -2070, -310}
5207 , { -450, -450, -960, -1470, -960}
5208 }
5209 , {{ 50, 50, -450, -720, -450} }
5210 , { 0, 0, -510, -1020, -510}
5211 , { 50, 50, -450, -720, -450}
5212 , { 0, 0, -510, -1020, -510}
5213 , { -550, -550, -1300, -1810, -1300}
5214 }
5215 }
5216 , {{ { -310, -310, -310, -310, -310} }
5217 , { -510, -510, -510, -510, -510}
5218 , { -450, -450, -450, -450, -450}
5219 , { -310, -310, -310, -310, -310}
5220 , { -560, -560, -560, -560, -560}
5221 }
5222 , {{ -630, -650, -630, -650, -630} }
5223 , { -650, -650, -650, -650, -650}
5224 , { -960, -960, -960, -960, -960}
5225 , { -630, -870, -630, -870, -630}
5226 , { -960, -960, -960, -960, -960}
5227 }
5228 , {{ -510, -510, -510, -510, -510} }
5229 , { -510, -510, -510, -510, -510}
5230 , { -510, -510, -510, -510, -510}
5231 , { -510, -510, -510, -510, -510}
5232 , { -560, -560, -560, -560, -560}
5233 }
5234 , {{ -310, -310, -310, -310, -310} }
5235 , { -910, -1150, -910, -1150, -910}
5236 , { -960, -960, -960, -960, -960}
5237 , { -310, -310, -310, -310, -310}
5238 , { -960, -960, -960, -960, -960}
5239 }
5240 , {{ -450, -450, -450, -450, -450} }
5241 , { -510, -510, -510, -510, -510}
5242 , { -450, -450, -450, -450, -450}
5243 , { -510, -510, -510, -510, -510}
5244 , { -1300, -1300, -1300, -1300, -1300}
5245 }
5246 }
```

```
5247 ,{{{{ 800, -550, -310, 800, -310}
5248 ,{ 740, -850, -510, 740, -510}
5249 ,{ 800, -550, -450, 800, -450}
5250 ,{ 740, -850, -310, 740, -310}
5251 ,{ 690, -660, -560, 690, -560}
5252 }
5253 ,{{ 600, -990, -650, 600, -650}
5254 ,{ 600, -990, -650, 600, -650}
5255 ,{ 290, -1300, -960, 290, -960}
5256 ,{ -870, -1210, -870, -870, -870}
5257 ,{ 290, -1300, -960, 290, -960}
5258 }
5259 ,{{ 740, -610, -510, 740, -510}
5260 ,{ 740, -850, -510, 740, -510}
5261 ,{ 740, -610, -510, 740, -510}
5262 ,{ 740, -850, -510, 740, -510}
5263 ,{ 690, -660, -560, 690, -560}
5264 }
5265 ,{{ 290, -1300, -310, 290, -310}
5266 ,{ -1150, -1490, -1150, -1150, -1150}
5267 ,{ 290, -1300, -960, 290, -960}
5268 ,{ -310, -1900, -310, -1560, -310}
5269 ,{ 290, -1300, -960, 290, -960}
5270 }
5271 ,{{ 800, -550, -450, 800, -450}
5272 ,{ 740, -850, -510, 740, -510}
5273 ,{ 800, -550, -450, 800, -450}
5274 ,{ 740, -850, -510, 740, -510}
5275 ,{ -1300, -1640, -1300, -1300, -1300}
5276 }
5277 }
5278 ,{{{{ -310, -310, -310, -310, -410}
5279 ,{ -410, -510, -510, -510, -410}
5280 ,{ -450, -450, -450, -450, -450}
5281 ,{ -310, -310, -310, -310, -510}
5282 ,{ -560, -560, -560, -560, -560}
5283 }
5284 ,{{ -410, -650, -630, -650, -410}
5285 ,{ -410, -650, -650, -650, -410}
5286 ,{ -960, -960, -960, -960, -960}
5287 ,{ -630, -870, -630, -870, -870}
5288 ,{ -960, -960, -960, -960, -960}
5289 }
5290 ,{{ -510, -510, -510, -510, -510}
5291 ,{ -510, -510, -510, -510, -510}
5292 ,{ -510, -510, -510, -510, -510}
5293 ,{ -510, -510, -510, -510, -510}
5294 ,{ -560, -560, -560, -560, -560}
5295 }
5296 ,{{ -310, -310, -310, -310, -960}
5297 ,{ -910, -1150, -910, -1150, -1150}
5298 ,{ -960, -960, -960, -960, -960}
5299 ,{ -310, -310, -310, -310, -1560}
5300 ,{ -960, -960, -960, -960, -960}
5301 }
5302 ,{{ -450, -450, -450, -450, -450}
5303 ,{ -510, -510, -510, -510, -510}
5304 ,{ -450, -450, -450, -450, -450}
5305 ,{ -510, -510, -510, -510, -510}
5306 ,{ -1300, -1300, -1300, -1300, -1300}
5307 }
5308 }
5309 }
5310 ,{{{{ 760, 200, -310, 760, -250}
5311 ,{ 760, -340, -490, 760, -250}
5312 ,{ 310, -430, -940, 310, -940}
5313 ,{ 400, 200, -310, 400, -310}
5314 ,{ 310, -390, -940, 310, -940}
5315 }
5316 ,{{ 760, -430, -490, 760, -250}
5317 ,{ 760, -490, -490, 760, -250}
5318 ,{ 310, -430, -940, 310, -940}
5319 ,{ -1170, -1170, -1440, -1680, -1440}
5320 ,{ 310, -430, -940, 310, -940}
5321 }
5322 ,{{ 400, -340, -850, 400, -850}
5323 ,{ 400, -340, -850, 400, -850}
5324 ,{ 90, -650, -1160, 90, -1160}
5325 ,{ 400, -340, -850, 400, -850}
5326 ,{ 90, -650, -1160, 90, -1160}
5327 }
5328 ,{{ 310, 200, -310, 310, -310}
5329 ,{ -690, -690, -960, -1200, -960}
5330 ,{ 310, -430, -940, 310, -940}
5331 ,{ 200, 200, -310, -310, -310}
5332 ,{ 310, -430, -940, 310, -940}
5333 }
```

```
5334 , {{ 400, -340, -850, 400, -850} }  
5335 , { 400, -340, -850, 400, -850} }  
5336 , { -220, -960, -1470, -220, -1470} }  
5337 , { 400, -340, -850, 400, -850} }  
5338 , { -390, -390, -1140, -1140, -1140} }  
5339 }  
5340 }  
5341 , {{ 200, 200, -310, -1000, -310} }  
5342 , { -340, -340, -490, -1000, -490} }  
5343 , { -430, -430, -940, -1430, -940} }  
5344 , { 200, 200, -310, -1360, -310} }  
5345 , { -390, -390, -940, -1430, -940} }  
5346 }  
5347 , {{ -430, -430, -490, -1000, -490} }  
5348 , { -490, -2040, -490, -1000, -490} }  
5349 , { -430, -430, -940, -1450, -940} }  
5350 , { -1170, -1170, -1680, -2190, -1680} }  
5351 , { -430, -430, -940, -1450, -940} }  
5352 }  
5353 , {{ -340, -340, -850, -1360, -850} }  
5354 , { -340, -340, -850, -1360, -850} }  
5355 , { -650, -650, -1160, -1430, -1160} }  
5356 , { -340, -340, -850, -1360, -850} }  
5357 , { -650, -650, -1160, -1430, -1160} }  
5358 }  
5359 , {{ 200, 200, -310, -1450, -310} }  
5360 , { -690, -690, -1200, -1710, -1200} }  
5361 , { -430, -430, -940, -1450, -940} }  
5362 , { 200, 200, -310, -2070, -310} }  
5363 , { -430, -430, -940, -1450, -940} }  
5364 }  
5365 , {{ -340, -340, -850, -1360, -850} }  
5366 , { -340, -340, -850, -1360, -850} }  
5367 , { -960, -960, -1470, -1740, -1470} }  
5368 , { -340, -340, -850, -1360, -850} }  
5369 , { -390, -390, -1140, -1650, -1140} }  
5370 }  
5371 }  
5372 , {{ -310, -310, -310, -310, -310} }  
5373 , { -490, -490, -490, -490, -490} }  
5374 , { -940, -940, -940, -940, -940} }  
5375 , { -310, -310, -310, -310, -310} }  
5376 , { -940, -940, -940, -940, -940} }  
5377 }  
5378 , {{ -490, -490, -490, -490, -490} }  
5379 , { -490, -490, -490, -490, -490} }  
5380 , { -940, -940, -940, -940, -940} }  
5381 , { -1440, -1680, -1440, -1680, -1440} }  
5382 , { -940, -940, -940, -940, -940} }  
5383 }  
5384 , {{ -850, -850, -850, -850, -850} }  
5385 , { -850, -850, -850, -850, -850} }  
5386 , { -1160, -1160, -1160, -1160, -1160} }  
5387 , { -850, -850, -850, -850, -850} }  
5388 , { -1160, -1160, -1160, -1160, -1160} }  
5389 }  
5390 , {{ -310, -310, -310, -310, -310} }  
5391 , { -960, -1200, -960, -1200, -960} }  
5392 , { -940, -940, -940, -940, -940} }  
5393 , { -310, -310, -310, -310, -310} }  
5394 , { -940, -940, -940, -940, -940} }  
5395 }  
5396 , {{ -850, -850, -850, -850, -850} }  
5397 , { -850, -850, -850, -850, -850} }  
5398 , { -1470, -1470, -1470, -1470, -1470} }  
5399 , { -850, -850, -850, -850, -850} }  
5400 , { -1140, -1140, -1140, -1140, -1140} }  
5401 }  
5402 }  
5403 , {{ 760, -830, -310, 760, -310} }  
5404 , { 760, -830, -490, 760, -490} }  
5405 , { 310, -1260, -940, 310, -940} }  
5406 , { 400, -1190, -310, 400, -310} }  
5407 , { 310, -1260, -940, 310, -940} }  
5408 }  
5409 , {{ 760, -830, -490, 760, -490} }  
5410 , { 760, -830, -490, 760, -490} }  
5411 , { 310, -1280, -940, 310, -940} }  
5412 , { -1680, -2020, -1680, -1680, -1680} }  
5413 , { 310, -1280, -940, 310, -940} }  
5414 }  
5415 , {{ 400, -1190, -850, 400, -850} }  
5416 , { 400, -1190, -850, 400, -850} }  
5417 , { 90, -1260, -1160, 90, -1160} }  
5418 , { 400, -1190, -850, 400, -850} }  
5419 , { 90, -1260, -1160, 90, -1160} }  
5420 }
```

```
5421 ,{{ 310, -1280, -310, 310, -310}
5422 ,{ -1200, -1540, -1200, -1200, -1200}
5423 ,{ 310, -1280, -940, 310, -940}
5424 ,{ -310, -1900, -310, -1560, -310}
5425 ,{ 310, -1280, -940, 310, -940}
5426 }
5427 ,{{ 400, -1190, -850, 400, -850}
5428 ,{ 400, -1190, -850, 400, -850}
5429 ,{ -220, -1570, -1470, -220, -1470}
5430 ,{ 400, -1190, -850, 400, -850}
5431 ,{ -1140, -1480, -1140, -1140, -1140}
5432 }
5433 }
5434 ,{{ { -250, -310, -310, -310, -250}
5435 ,{ -250, -490, -490, -490, -250}
5436 ,{ -940, -940, -940, -940, -940}
5437 ,{ -310, -310, -310, -310, -850}
5438 ,{ -940, -940, -940, -940, -940}
5439 }
5440 ,{{ { -250, -490, -490, -490, -250}
5441 ,{ -250, -490, -490, -490, -250}
5442 ,{ -940, -940, -940, -940, -940}
5443 ,{ -1440, -1680, -1440, -1680, -1680}
5444 ,{ -940, -940, -940, -940, -940}
5445 }
5446 ,{{ { -850, -850, -850, -850, -850}
5447 ,{ -850, -850, -850, -850, -850}
5448 ,{ -1160, -1160, -1160, -1160, -1160}
5449 ,{ -850, -850, -850, -850, -850}
5450 ,{ -1160, -1160, -1160, -1160, -1160}
5451 }
5452 ,{{ { -310, -310, -310, -310, -940}
5453 ,{ -960, -1200, -960, -1200, -1200}
5454 ,{ -940, -940, -940, -940, -940}
5455 ,{ -310, -310, -310, -310, -1560}
5456 ,{ -940, -940, -940, -940, -940}
5457 }
5458 ,{{ { -850, -850, -850, -850, -850}
5459 ,{ -850, -850, -850, -850, -850}
5460 ,{ -1470, -1470, -1470, -1470, -1470}
5461 ,{ -850, -850, -850, -850, -850}
5462 ,{ -1140, -1140, -1140, -1140, -1140}
5463 }
5464 }
5465 }
5466 ,{{ {{ 360, 360, -150, -150, -150}
5467 ,{ -30, -30, -990, -150, -990}
5468 ,{ -150, -890, -1400, -150, -1400}
5469 ,{ 360, 360, -150, -150, -150}
5470 ,{ -150, -650, -1400, -150, -1400}
5471 }
5472 ,{{ { -70, -70, -1180, -150, -1180}
5473 ,{ -70, -70, -1580, -330, -1340}
5474 ,{ -150, -890, -1400, -150, -1400}
5475 ,{ -910, -910, -1180, -1420, -1180}
5476 ,{ -150, -890, -1400, -150, -1400}
5477 }
5478 ,{{ { -150, -890, -1400, -150, -1400}
5479 ,{ -150, -890, -1400, -150, -1400}
5480 ,{ -150, -890, -1400, -150, -1400}
5481 ,{ -150, -890, -1400, -150, -1400}
5482 ,{ -150, -890, -1400, -150, -1400}
5483 }
5484 ,{{ { 360, 360, -150, -150, -150}
5485 ,{ -30, -30, -990, -1230, -990}
5486 ,{ -150, -890, -1400, -150, -1400}
5487 ,{ 360, 360, -150, -150, -150}
5488 ,{ -150, -890, -1400, -150, -1400}
5489 }
5490 ,{{ { -150, -650, -1400, -150, -1400}
5491 ,{ -150, -890, -1400, -150, -1400}
5492 ,{ -150, -890, -1400, -150, -1400}
5493 ,{ -150, -890, -1400, -150, -1400}
5494 ,{ -650, -650, -1400, -1400, -1400}
5495 }
5496 }
5497 ,{{ {{ 360, 360, -150, -1670, -150}
5498 ,{ -30, -30, -1230, -1740, -1230}
5499 ,{ -890, -890, -1400, -1670, -1400}
5500 ,{ 360, 360, -150, -1910, -150}
5501 ,{ -650, -650, -1400, -1670, -1400}
5502 }
5503 ,{{ { -70, -70, -1400, -1910, -1400}
5504 ,{ -70, -70, -1580, -2090, -1580}
5505 ,{ -890, -890, -1400, -1910, -1400}
5506 ,{ -910, -910, -1420, -1930, -1420}
5507 ,{ -890, -890, -1400, -1910, -1400}
```

```
5508      }
5509 ,{{ -890, -890, -1400, -1670, -1400}
5510 ,{ -890, -890, -1400, -1910, -1400}
5511 ,{ -890, -890, -1400, -1670, -1400}
5512 ,{ -890, -890, -1400, -1910, -1400}
5513 ,{ -890, -890, -1400, -1670, -1400}
5514 }
5515 ,{{ 360, 360, -150, -1740, -150}
5516 ,{ -30, -30, -1230, -1740, -1230}
5517 ,{ -890, -890, -1400, -1910, -1400}
5518 ,{ 360, 360, -150, -1910, -150}
5519 ,{ -890, -890, -1400, -1910, -1400}
5520 }
5521 ,{{ -650, -650, -1400, -1670, -1400}
5522 ,{ -890, -890, -1400, -1910, -1400}
5523 ,{ -890, -890, -1400, -1670, -1400}
5524 ,{ -890, -890, -1400, -1910, -1400}
5525 ,{ -650, -650, -1400, -1910, -1400}
5526 }
5527 }
5528 ,{{ { -150, -150, -150, -150, -150}
5529 ,{ -990, -1230, -990, -1230, -990}
5530 ,{ -1400, -1400, -1400, -1400, -1400}
5531 ,{ -150, -150, -150, -150, -150}
5532 ,{ -1400, -1400, -1400, -1400, -1400}
5533 }
5534 ,{{ -1180, -1400, -1180, -1400, -1180}
5535 ,{ -1580, -1580, -1580, -1580, -1580}
5536 ,{ -1400, -1400, -1400, -1400, -1400}
5537 ,{ -1180, -1420, -1180, -1420, -1180}
5538 ,{ -1400, -1400, -1400, -1400, -1400}
5539 }
5540 ,{{ -1400, -1400, -1400, -1400, -1400}
5541 ,{ -1400, -1400, -1400, -1400, -1400}
5542 ,{ -1400, -1400, -1400, -1400, -1400}
5543 ,{ -1400, -1400, -1400, -1400, -1400}
5544 ,{ -1400, -1400, -1400, -1400, -1400}
5545 }
5546 ,{{ { -150, -150, -150, -150, -150}
5547 ,{ -990, -1230, -990, -1230, -990}
5548 ,{ -1400, -1400, -1400, -1400, -1400}
5549 ,{ -150, -150, -150, -150, -150}
5550 ,{ -1400, -1400, -1400, -1400, -1400}
5551 }
5552 ,{{ { -1400, -1400, -1400, -1400, -1400}
5553 ,{ -1400, -1400, -1400, -1400, -1400}
5554 ,{ -1400, -1400, -1400, -1400, -1400}
5555 ,{ -1400, -1400, -1400, -1400, -1400}
5556 ,{ -1400, -1400, -1400, -1400, -1400}
5557 }
5558 }
5559 ,{{ { { -150, -1500, -150, -150, -150}
5560 ,{ -150, -1570, -1230, -150, -1230}
5561 ,{ -150, -1500, -1400, -150, -1400}
5562 ,{ -150, -1740, -150, -150, -150}
5563 ,{ -150, -1500, -1400, -150, -1400}
5564 }
5565 ,{{ { -150, -1600, -1400, -150, -1400}
5566 ,{ -330, -1600, -1580, -330, -1580}
5567 ,{ -150, -1740, -1400, -150, -1400}
5568 ,{ -1420, -3040, -1420, -1420, -1420}
5569 ,{ -150, -1740, -1400, -150, -1400}
5570 }
5571 ,{{ { -150, -1500, -1400, -150, -1400}
5572 ,{ -150, -1740, -1400, -150, -1400}
5573 ,{ -150, -1500, -1400, -150, -1400}
5574 ,{ -150, -1740, -1400, -150, -1400}
5575 ,{ -150, -1500, -1400, -150, -1400}
5576 }
5577 ,{{ { -150, -1570, -150, -150, -150}
5578 ,{ -1230, -1570, -1230, -1230, -1230}
5579 ,{ -150, -1740, -1400, -150, -1400}
5580 ,{ -150, -1740, -150, -1400, -150}
5581 ,{ -150, -1740, -1400, -150, -1400}
5582 }
5583 ,{{ { -150, -1500, -1400, -150, -1400}
5584 ,{ -150, -1740, -1400, -150, -1400}
5585 ,{ -150, -1500, -1400, -150, -1400}
5586 ,{ -150, -1740, -1400, -150, -1400}
5587 ,{ -1400, -1740, -1400, -1400, -1400}
5588 }
5589 }
5590 ,{{ { { -150, -150, -150, -150, -1230}
5591 ,{ -990, -1230, -990, -1230, -1230}
5592 ,{ -1400, -1400, -1400, -1400, -1400}
5593 ,{ -150, -150, -150, -150, -1400}
5594 ,{ -1400, -1400, -1400, -1400, -1400}
```

```
5595      }
5596 ,{{ -1180, -1400, -1180, -1400, -1340}
5597 ,{ -1340, -1580, -1580, -1580, -1340}
5598 ,{ -1400, -1400, -1400, -1400, -1400}
5599 ,{ -1180, -1420, -1180, -1420, -1420}
5600 ,{ -1400, -1400, -1400, -1400, -1400}
5601      }
5602 ,{{ -1400, -1400, -1400, -1400, -1400}
5603 ,{ -1400, -1400, -1400, -1400, -1400}
5604 ,{ -1400, -1400, -1400, -1400, -1400}
5605 ,{ -1400, -1400, -1400, -1400, -1400}
5606 ,{ -1400, -1400, -1400, -1400, -1400}
5607      }
5608 ,{{ -150, -150, -150, -150, -1230}
5609 ,{ -990, -1230, -990, -1230, -1230}
5610 ,{ -1400, -1400, -1400, -1400, -1400}
5611 ,{ -150, -150, -150, -150, -1400}
5612 ,{ -1400, -1400, -1400, -1400, -1400}
5613      }
5614 ,{{ -1400, -1400, -1400, -1400, -1400}
5615 ,{ -1400, -1400, -1400, -1400, -1400}
5616 ,{ -1400, -1400, -1400, -1400, -1400}
5617 ,{ -1400, -1400, -1400, -1400, -1400}
5618 ,{ -1400, -1400, -1400, -1400, -1400}
5619      }
5620      }
5621      }
5622 ,{{{ 910, 910, 400, 910, 400}
5623 ,{ 910, 170, -340, 910, -100}
5624 ,{ 400, -340, -850, 400, -850}
5625 ,{ 910, 910, 400, 400, 400}
5626 ,{ 400, -100, -850, 400, -850}
5627      }
5628 ,{{ 910, 170, -340, 910, -100}
5629 ,{ 910, 170, -340, 910, -100}
5630 ,{ 400, -340, -850, 400, -850}
5631 ,{ -680, -680, -950, -1190, -950}
5632 ,{ 400, -340, -850, 400, -850}
5633      }
5634 ,{{ 400, -340, -850, 400, -850}
5635 ,{ 400, -340, -850, 400, -850}
5636 ,{ 400, -340, -850, 400, -850}
5637 ,{ 400, -340, -850, 400, -850}
5638 ,{ 400, -340, -850, 400, -850}
5639      }
5640 ,{{ 910, 910, 400, 400, 400}
5641 ,{ -850, -850, -1120, -1360, -1120}
5642 ,{ 400, -340, -850, 400, -850}
5643 ,{ 910, 910, 400, 400, 400}
5644 ,{ 400, -340, -850, 400, -850}
5645      }
5646 ,{{ 400, -100, -850, 400, -850}
5647 ,{ 400, -340, -850, 400, -850}
5648 ,{ 400, -340, -850, 400, -850}
5649 ,{ 400, -340, -850, 400, -850}
5650 ,{ -100, -100, -850, -850, -850}
5651      }
5652      }
5653 ,{{{ 910, 910, 400, -850, 400}
5654 ,{ 170, 170, -340, -850, -340}
5655 ,{ -340, -340, -850, -1120, -850}
5656 ,{ 910, 910, 400, -1360, 400}
5657 ,{ -100, -100, -850, -1120, -850}
5658      }
5659 ,{{ 170, 170, -340, -850, -340}
5660 ,{ 170, 170, -340, -850, -340}
5661 ,{ -340, -340, -850, -1360, -850}
5662 ,{ -680, -680, -1190, -1700, -1190}
5663 ,{ -340, -340, -850, -1360, -850}
5664      }
5665 ,{{ -340, -340, -850, -1120, -850}
5666 ,{ -340, -340, -850, -1360, -850}
5667 ,{ -340, -340, -850, -1120, -850}
5668 ,{ -340, -340, -850, -1360, -850}
5669 ,{ -340, -340, -850, -1120, -850}
5670      }
5671 ,{{ 910, 910, 400, -1360, 400}
5672 ,{ -850, -850, -1360, -1870, -1360}
5673 ,{ -340, -340, -850, -1360, -850}
5674 ,{ 910, 910, 400, -1360, 400}
5675 ,{ -340, -340, -850, -1360, -850}
5676      }
5677 ,{{ -100, -100, -850, -1120, -850}
5678 ,{ -340, -340, -850, -1360, -850}
5679 ,{ -340, -340, -850, -1120, -850}
5680 ,{ -340, -340, -850, -1360, -850}
5681 ,{ -100, -100, -850, -1360, -850}
```

```
5682      }
5683  }
5684 ,{{ 400,   400,   400,   400,   400}
5685 ,{ -340,  -340,  -340,  -340,  -340}
5686 ,{ -850,  -850,  -850,  -850,  -850}
5687 ,{  400,   400,   400,   400,   400}
5688 ,{ -850,  -850,  -850,  -850,  -850}
5689 }
5690 ,{{ -340,  -340,  -340,  -340,  -340}
5691 ,{ -340,  -340,  -340,  -340,  -340}
5692 ,{ -850,  -850,  -850,  -850,  -850}
5693 ,{ -950, -1190, -950, -1190, -950}
5694 ,{ -850,  -850,  -850,  -850,  -850}
5695 }
5696 ,{{ -850,  -850,  -850,  -850,  -850}
5697 ,{ -850,  -850,  -850,  -850,  -850}
5698 ,{ -850,  -850,  -850,  -850,  -850}
5699 ,{ -850,  -850,  -850,  -850,  -850}
5700 ,{ -850,  -850,  -850,  -850,  -850}
5701 }
5702 ,{{  400,   400,   400,   400,   400}
5703 ,{ -1120, -1360, -1120, -1360, -1120}
5704 ,{ -850,  -850,  -850,  -850,  -850}
5705 ,{  400,   400,   400,   400,   400}
5706 ,{ -850,  -850,  -850,  -850,  -850}
5707 }
5708 ,{{ -850,  -850,  -850,  -850,  -850}
5709 ,{ -850,  -850,  -850,  -850,  -850}
5710 ,{ -850,  -850,  -850,  -850,  -850}
5711 ,{ -850,  -850,  -850,  -850,  -850}
5712 ,{ -850,  -850,  -850,  -850,  -850}
5713 }
5714 }
5715 ,{{  910,  -680,   400,   910,   400}
5716 ,{  910,  -680,  -340,   910,  -340}
5717 ,{  400,  -950,  -850,   400,  -850}
5718 ,{  400, -1190,   400,   400,   400}
5719 ,{  400,  -950,  -850,   400,  -850}
5720 }
5721 ,{{  910,  -680,  -340,   910,  -340}
5722 ,{  910,  -680,  -340,   910,  -340}
5723 ,{  400, -1190,  -850,   400,  -850}
5724 ,{ -1190, -1530, -1190, -1190, -1190}
5725 ,{  400, -1190,  -850,   400,  -850}
5726 }
5727 ,{{  400,  -950,  -850,   400,  -850}
5728 ,{  400, -1190,  -850,   400,  -850}
5729 ,{  400,  -950,  -850,   400,  -850}
5730 ,{  400, -1190,  -850,   400,  -850}
5731 ,{  400,  -950,  -850,   400,  -850}
5732 }
5733 ,{{  400, -1190,   400,   400,   400}
5734 ,{ -1360, -1700, -1360, -1360, -1360}
5735 ,{  400, -1190,  -850,   400,  -850}
5736 ,{  400, -1190,   400,  -850,   400}
5737 ,{  400, -1190,  -850,   400,  -850}
5738 }
5739 ,{{  400,  -950,  -850,   400,  -850}
5740 ,{  400, -1190,  -850,   400,  -850}
5741 ,{  400,  -950,  -850,   400,  -850}
5742 ,{  400, -1190,  -850,   400,  -850}
5743 ,{ -850, -1190,  -850,  -850,  -850}
5744 }
5745 }
5746 ,{{  400,   400,   400,   400,  -100}
5747 ,{ -100,  -340,  -340,  -340,  -100}
5748 ,{ -850,  -850,  -850,  -850,  -850}
5749 ,{  400,   400,   400,   400,  -850}
5750 ,{ -850,  -850,  -850,  -850,  -850}
5751 }
5752 ,{{ -100,  -340,  -340,  -340,  -100}
5753 ,{ -100,  -340,  -340,  -340,  -100}
5754 ,{ -850,  -850,  -850,  -850,  -850}
5755 ,{ -950, -1190, -950, -1190, -1190}
5756 ,{ -850,  -850,  -850,  -850,  -850}
5757 }
5758 ,{{ -850,  -850,  -850,  -850,  -850}
5759 ,{ -850,  -850,  -850,  -850,  -850}
5760 ,{ -850,  -850,  -850,  -850,  -850}
5761 ,{ -850,  -850,  -850,  -850,  -850}
5762 ,{ -850,  -850,  -850,  -850,  -850}
5763 }
5764 ,{{  400,   400,   400,   400,  -850}
5765 ,{ -1120, -1360, -1120, -1360, -1360}
5766 ,{ -850,  -850,  -850,  -850,  -850}
5767 ,{  400,   400,   400,   400,  -850}
5768 ,{ -850,  -850,  -850,  -850,  -850}
```

```
5769      }
5770 ,{{ -850, -850, -850, -850, -850}
5771 ,{ -850, -850, -850, -850, -850}
5772 ,{ -850, -850, -850, -850, -850}
5773 ,{ -850, -850, -850, -850, -850}
5774 ,{ -850, -850, -850, -850, -850}
5775     }
5776   }
5777 }
5778 ,{{{ 1490, 1280, 780, 1490, 780}
5779 ,{ 1490, 750, 240, 1490, 480}
5780 ,{ 1200, 450, -50, 1200, -50}
5781 ,{ 1280, 1280, 780, 1200, 780}
5782 ,{ 1200, 450, -50, 1200, -50}
5783     }
5784 ,{{ 1490, 750, 240, 1490, 480}
5785 ,{ 1490, 750, 240, 1490, 480}
5786 ,{ 1190, 440, -60, 1190, -60}
5787 ,{ -630, -630, -900, -1140, -900}
5788 ,{ 1190, 440, -60, 1190, -60}
5789     }
5790 ,{{ 1200, 460, -50, 1200, -50}
5791 ,{ 1200, 460, -50, 1200, -50}
5792 ,{ 1200, 450, -50, 1200, -50}
5793 ,{ 1200, 460, -50, 1200, -50}
5794 ,{ 1200, 450, -50, 1200, -50}
5795     }
5796 ,{{ 1280, 1280, 780, 1190, 780}
5797 ,{ -450, -450, -720, -960, -720}
5798 ,{ 1190, 440, -60, 1190, -60}
5799 ,{ 1280, 1280, 780, 780, 780}
5800 ,{ 1190, 440, -60, 1190, -60}
5801     }
5802 ,{{ 1200, 460, -50, 1200, -50}
5803 ,{ 1200, 460, -50, 1200, -50}
5804 ,{ 1200, 450, -50, 1200, -50}
5805 ,{ 1200, 460, -50, 1200, -50}
5806 ,{ -280, -280, -1030, -1030, -1030}
5807     }
5808 }
5809 ,{{{ 1280, 1280, 780, -260, 780}
5810 ,{ 750, 750, 240, -260, 240}
5811 ,{ 450, 450, -50, -320, -50}
5812 ,{ 1280, 1280, 780, -560, 780}
5813 ,{ 450, 450, -50, -320, -50}
5814     }
5815 ,{{ 750, 750, 240, -260, 240}
5816 ,{ 750, 750, 240, -260, 240}
5817 ,{ 440, 440, -60, -570, -60}
5818 ,{ -630, -630, -1140, -1650, -1140}
5819 ,{ 440, 440, -60, -570, -60}
5820     }
5821 ,{{ 460, 460, -50, -320, -50}
5822 ,{ 460, 460, -50, -560, -50}
5823 ,{ 450, 450, -50, -320, -50}
5824 ,{ 460, 460, -50, -560, -50}
5825 ,{ 450, 450, -50, -320, -50}
5826     }
5827 ,{{ 1280, 1280, 780, -570, 780}
5828 ,{ -450, -450, -960, -1470, -960}
5829 ,{ 440, 440, -60, -570, -60}
5830 ,{ 1280, 1280, 780, -980, 780}
5831 ,{ 440, 440, -60, -570, -60}
5832     }
5833 ,{{ 460, 460, -50, -320, -50}
5834 ,{ 460, 460, -50, -560, -50}
5835 ,{ 450, 450, -50, -320, -50}
5836 ,{ 460, 460, -50, -560, -50}
5837 ,{ -280, -280, -1030, -1540, -1030}
5838     }
5839 }
5840 ,{{{ 780, 780, 780, 780, 780}
5841 ,{ 240, 240, 240, 240, 240}
5842 ,{ -50, -50, -50, -50, -50}
5843 ,{ 780, 780, 780, 780, 780}
5844 ,{ -50, -50, -50, -50, -50}
5845     }
5846 ,{{ 240, 240, 240, 240, 240}
5847 ,{ 240, 240, 240, 240, 240}
5848 ,{ -60, -60, -60, -60, -60}
5849 ,{ -900, -1140, -900, -1140, -900}
5850 ,{ -60, -60, -60, -60, -60}
5851     }
5852 ,{{ -50, -50, -50, -50, -50}
5853 ,{ -50, -50, -50, -50, -50}
5854 ,{ -50, -50, -50, -50, -50}
5855 ,{ -50, -50, -50, -50, -50}
```

```
5856 , { -50, -50, -50, -50, -50}
5857 }
5858 , {{ 780, 780, 780, 780, 780}
5859 , { -720, -960, -720, -960, -720}
5860 , { -60, -60, -60, -60, -60}
5861 , { 780, 780, 780, 780, 780}
5862 , { -60, -60, -60, -60, -60}
5863 }
5864 , {{ -50, -50, -50, -50, -50}
5865 , { -50, -50, -50, -50, -50}
5866 , { -50, -50, -50, -50, -50}
5867 , { -50, -50, -50, -50, -50}
5868 , { -1030, -1030, -1030, -1030, -1030}
5869 }
5870 }
5871 , {{{ 1490, -90, 780, 1490, 780}
5872 , { 1490, -90, 240, 1490, 240}
5873 , { 1200, -150, -50, 1200, -50}
5874 , { 1200, -390, 780, 1200, 780}
5875 , { 1200, -150, -50, 1200, -50}
5876 }
5877 , {{ 1490, -90, 240, 1490, 240}
5878 , { 1490, -90, 240, 1490, 240}
5879 , { 1190, -400, -60, 1190, -60}
5880 , { -1140, -1480, -1140, -1140, -1140}
5881 , { 1190, -400, -60, 1190, -60}
5882 }
5883 , {{ 1200, -150, -50, 1200, -50}
5884 , { 1200, -390, -50, 1200, -50}
5885 , { 1200, -150, -50, 1200, -50}
5886 , { 1200, -390, -50, 1200, -50}
5887 , { 1200, -150, -50, 1200, -50}
5888 }
5889 , {{ 1190, -400, 780, 1190, 780}
5890 , { -960, -1300, -960, -960, -960}
5891 , { 1190, -400, -60, 1190, -60}
5892 , { 780, -810, 780, -470, 780}
5893 , { 1190, -400, -60, 1190, -60}
5894 }
5895 , {{ 1200, -150, -50, 1200, -50}
5896 , { 1200, -390, -50, 1200, -50}
5897 , { 1200, -150, -50, 1200, -50}
5898 , { 1200, -390, -50, 1200, -50}
5899 , { -1030, -1370, -1030, -1030, -1030}
5900 }
5901 }
5902 , {{{ 780, 780, 780, 780, 480}
5903 , { 480, 240, 240, 240, 480}
5904 , { -50, -50, -50, -50, -50}
5905 , { 780, 780, 780, 780, -50}
5906 , { -50, -50, -50, -50, -50}
5907 }
5908 , {{ 480, 240, 240, 240, 480}
5909 , { 480, 240, 240, 240, 480}
5910 , { -60, -60, -60, -60, -60}
5911 , { -900, -1140, -900, -1140, -1140}
5912 , { -60, -60, -60, -60, -60}
5913 }
5914 , {{ -50, -50, -50, -50, -50}
5915 , { -50, -50, -50, -50, -50}
5916 , { -50, -50, -50, -50, -50}
5917 , { -50, -50, -50, -50, -50}
5918 , { -50, -50, -50, -50, -50}
5919 }
5920 , {{ 780, 780, 780, 780, -60}
5921 , { -720, -960, -720, -960, -960}
5922 , { -60, -60, -60, -60, -60}
5923 , { 780, 780, 780, 780, -470}
5924 , { -60, -60, -60, -60, -60}
5925 }
5926 , {{ -50, -50, -50, -50, -50}
5927 , { -50, -50, -50, -50, -50}
5928 , { -50, -50, -50, -50, -50}
5929 , { -50, -50, -50, -50, -50}
5930 , { -1030, -1030, -1030, -1030, -1030}
5931 }
5932 }
5933 }
5934 , {{{{ 1560, 1470, 960, 1560, 960}
5935 , { 1560, 820, 310, 1560, 550}
5936 , { 1430, 690, 180, 1430, 180}
5937 , { 1470, 1470, 960, 1430, 960}
5938 , { 1300, 560, 50, 1300, 50}
5939 }
5940 , {{ 1560, 820, 310, 1560, 550}
5941 , { 1560, 820, 310, 1560, 550}
5942 , { 1280, 540, 30, 1280, 30}
```

```
5943 , { -580, -580, -850, -1090, -850}
5944 , { 1280, 540, 30, 1280, 30}
5945 }
5946 , {{ 1430, 690, 180, 1430, 180}
5947 , { 1430, 690, 180, 1430, 180}
5948 , { 1430, 690, 180, 1430, 180}
5949 , { 1430, 690, 180, 1430, 180}
5950 , { 1300, 560, 50, 1300, 50}
5951 }
5952 , {{ 1470, 1470, 960, 1280, 960}
5953 , { -880, -880, -1150, -1390, -1150}
5954 , { 1280, 540, 30, 1280, 30}
5955 , { 1470, 1470, 960, 960, 960}
5956 , { 1280, 540, 30, 1280, 30}
5957 }
5958 , {{ 1430, 690, 180, 1430, 180}
5959 , { 1430, 690, 180, 1430, 180}
5960 , { 990, 250, -260, 990, -260}
5961 , { 1430, 690, 180, 1430, 180}
5962 , { -10, -10, -760, -760, -760}
5963 }
5964 }
5965 , {{ { 1470, 1470, 960, -90, 960}
5966 , { 820, 820, 310, -200, 310}
5967 , { 690, 690, 180, -90, 180}
5968 , { 1470, 1470, 960, -330, 960}
5969 , { 560, 560, 50, -220, 50}
5970 }
5971 , {{ 820, 820, 310, -200, 310}
5972 , { 820, 820, 310, -200, 310}
5973 , { 540, 540, 30, -480, 30}
5974 , { -580, -580, -1090, -1600, -1090}
5975 , { 540, 540, 30, -480, 30}
5976 }
5977 , {{ 690, 690, 180, -90, 180}
5978 , { 690, 690, 180, -330, 180}
5979 , { 690, 690, 180, -90, 180}
5980 , { 690, 690, 180, -330, 180}
5981 , { 560, 560, 50, -220, 50}
5982 }
5983 , {{ { 1470, 1470, 960, -480, 960}
5984 , { -880, -880, -1390, -1900, -1390}
5985 , { 540, 540, 30, -480, 30}
5986 , { 1470, 1470, 960, -800, 960}
5987 , { 540, 540, 30, -480, 30}
5988 }
5989 , {{ { 690, 690, 180, -330, 180}
5990 , { 690, 690, 180, -330, 180}
5991 , { 250, 250, -260, -530, -260}
5992 , { 690, 690, 180, -330, 180}
5993 , { -10, -10, -760, -1270, -760}
5994 }
5995 }
5996 , {{ { 960, 960, 960, 960, 960}
5997 , { 310, 310, 310, 310, 310}
5998 , { 180, 180, 180, 180, 180}
5999 , { 960, 960, 960, 960, 960}
6000 , { 50, 50, 50, 50, 50}
6001 }
6002 , {{ { 310, 310, 310, 310, 310}
6003 , { 310, 310, 310, 310, 310}
6004 , { 30, 30, 30, 30, 30}
6005 , { -850, -1090, -850, -1090, -850}
6006 , { 30, 30, 30, 30, 30}
6007 }
6008 , {{ { 180, 180, 180, 180, 180}
6009 , { 180, 180, 180, 180, 180}
6010 , { 180, 180, 180, 180, 180}
6011 , { 180, 180, 180, 180, 180}
6012 , { 50, 50, 50, 50, 50}
6013 }
6014 , {{ { 960, 960, 960, 960, 960}
6015 , { -1150, -1390, -1150, -1390, -1150}
6016 , { 30, 30, 30, 30, 30}
6017 , { 960, 960, 960, 960, 960}
6018 , { 30, 30, 30, 30, 30}
6019 }
6020 , {{ { 180, 180, 180, 180, 180}
6021 , { 180, 180, 180, 180, 180}
6022 , { -260, -260, -260, -260, -260}
6023 , { 180, 180, 180, 180, 180}
6024 , { -760, -760, -760, -760, -760}
6025 }
6026 }
6027 , {{ { 1560, 80, 960, 1560, 960}
6028 , { 1560, -30, 310, 1560, 310}
6029 , { 1430, 80, 180, 1430, 180}
```

```
6030 , { 1430, -160, 960, 1430, 960}
6031 , { 1300, -50, 50, 1300, 50}
6032 }
6033 , {{ 1560, -30, 310, 1560, 310}
6034 , { 1560, -30, 310, 1560, 310}
6035 , { 1280, -310, 30, 1280, 30}
6036 , { -1090, -1430, -1090, -1090, -1090}
6037 , { 1280, -310, 30, 1280, 30}
6038 }
6039 , {{ 1430, 80, 180, 1430, 180}
6040 , { 1430, -160, 180, 1430, 180}
6041 , { 1430, 80, 180, 1430, 180}
6042 , { 1430, -160, 180, 1430, 180}
6043 , { 1300, -50, 50, 1300, 50}
6044 }
6045 , {{ 1280, -310, 960, 1280, 960}
6046 , { -1390, -1730, -1390, -1390, -1390}
6047 , { 1280, -310, 30, 1280, 30}
6048 , { 960, -630, 960, -290, 960}
6049 , { 1280, -310, 30, 1280, 30}
6050 }
6051 , {{ 1430, -160, 180, 1430, 180}
6052 , { 1430, -160, 180, 1430, 180}
6053 , { 990, -360, -260, 990, -260}
6054 , { 1430, -160, 180, 1430, 180}
6055 , { -760, -1100, -760, -760, -760}
6056 }
6057 }
6058 , {{{ 960, 960, 960, 960, 550}
6059 , { 550, 310, 310, 310, 550}
6060 , { 180, 180, 180, 180, 180}
6061 , { 960, 960, 960, 960, 180}
6062 , { 50, 50, 50, 50, 50}
6063 }
6064 , {{ 550, 310, 310, 310, 550}
6065 , { 550, 310, 310, 310, 550}
6066 , { 30, 30, 30, 30, 30}
6067 , { -850, -1090, -850, -1090, -1090}
6068 , { 30, 30, 30, 30, 30}
6069 }
6070 , {{ 180, 180, 180, 180, 180}
6071 , { 180, 180, 180, 180, 180}
6072 , { 180, 180, 180, 180, 180}
6073 , { 180, 180, 180, 180, 180}
6074 , { 50, 50, 50, 50, 50}
6075 }
6076 , {{ 960, 960, 960, 960, 30}
6077 , { -1150, -1390, -1150, -1390, -1390}
6078 , { 30, 30, 30, 30, 30}
6079 , { 960, 960, 960, 960, -290}
6080 , { 30, 30, 30, 30, 30}
6081 }
6082 , {{ 180, 180, 180, 180, 180}
6083 , { 180, 180, 180, 180, 180}
6084 , { -260, -260, -260, -260, -260}
6085 , { 180, 180, 180, 180, 180}
6086 , { -760, -760, -760, -760, -760}
6087 }
6088 }
6089 }
6090 , {{{{ 1560, 1470, 960, 1560, 960}
6091 , { 1560, 820, 310, 1560, 550}
6092 , { 1430, 690, 180, 1430, 180}
6093 , { 1470, 1470, 960, 1430, 960}
6094 , { 1300, 560, 50, 1300, 50}
6095 }
6096 , {{ 1560, 820, 310, 1560, 550}
6097 , { 1560, 820, 310, 1560, 550}
6098 , { 1280, 540, 30, 1280, 30}
6099 , { -360, -360, -630, -870, -630}
6100 , { 1280, 540, 30, 1280, 30}
6101 }
6102 , {{ 1430, 690, 180, 1430, 180}
6103 , { 1430, 690, 180, 1430, 180}
6104 , { 1430, 690, 180, 1430, 180}
6105 , { 1430, 690, 180, 1430, 180}
6106 , { 1300, 560, 50, 1300, 50}
6107 }
6108 , {{ 1470, 1470, 960, 1280, 960}
6109 , { -30, -30, -720, -960, -720}
6110 , { 1280, 540, 30, 1280, 30}
6111 , { 1470, 1470, 960, 960, 960}
6112 , { 1280, 540, 30, 1280, 30}
6113 }
6114 , {{ 1430, 690, 180, 1430, 180}
6115 , { 1430, 690, 180, 1430, 180}
6116 , { 1200, 450, -50, 1200, -50}
```

```
6117 , { 1430,   690,   180,  1430,   180}
6118 , { -10,   -10,  -760,  -760,  -760}
6119 }
6120 }
6121 ,{{ 1470,  1470,   960,   -90,   960}
6122 , { 820,   820,   310,  -200,   310}
6123 , { 690,   690,   180,   -90,   180}
6124 , { 1470,  1470,   960,  -330,   960}
6125 , { 560,   560,    50,  -220,    50}
6126 }
6127 ,{{ 820,   820,   310,  -200,   310}
6128 , { 820,   820,   310,  -200,   310}
6129 , { 540,   540,    30,  -480,    30}
6130 , { -360,  -360,  -870, -1380,  -870}
6131 , { 540,   540,    30,  -480,    30}
6132 }
6133 ,{{ 690,   690,   180,   -90,   180}
6134 , { 690,   690,   180,  -330,   180}
6135 , { 690,   690,   180,   -90,   180}
6136 , { 690,   690,   180,  -330,   180}
6137 , { 560,   560,    50,  -220,    50}
6138 }
6139 ,{{ 1470,  1470,   960,  -480,   960}
6140 , { -30,   -30,  -960, -1470,  -960}
6141 , { 540,   540,    30,  -480,    30}
6142 , { 1470,  1470,   960,  -800,   960}
6143 , { 540,   540,    30,  -480,    30}
6144 }
6145 ,{{ 690,   690,   180,  -320,   180}
6146 , { 690,   690,   180,  -330,   180}
6147 , { 450,   450,    -50,  -320,   -50}
6148 , { 690,   690,   180,  -330,   180}
6149 , { -10,   -10,  -760, -1270,  -760}
6150 }
6151 }
6152 ,{{ 960,   960,   960,   960,   960}
6153 , { 310,   310,   310,   310,   310}
6154 , { 180,   180,   180,   180,   180}
6155 , { 960,   960,   960,   960,   960}
6156 , { 50,    50,    50,    50,    50}
6157 }
6158 ,{{ 310,   310,   310,   310,   310}
6159 , { 310,   310,   310,   310,   310}
6160 , { 30,    30,    30,    30,    30}
6161 , { -630,  -870,  -630,  -870,  -630}
6162 , { 30,    30,    30,    30,    30}
6163 }
6164 ,{{ 180,   180,   180,   180,   180}
6165 , { 180,   180,   180,   180,   180}
6166 , { 180,   180,   180,   180,   180}
6167 , { 180,   180,   180,   180,   180}
6168 , { 50,    50,    50,    50,    50}
6169 }
6170 ,{{ 960,   960,   960,   960,   960}
6171 , { -720,  -960,  -720,  -960,  -720}
6172 , { 30,    30,    30,    30,    30}
6173 , { 960,   960,   960,   960,   960}
6174 , { 30,    30,    30,    30,    30}
6175 }
6176 ,{{ 180,   180,   180,   180,   180}
6177 , { 180,   180,   180,   180,   180}
6178 , { -50,   -50,   -50,   -50,   -50}
6179 , { 180,   180,   180,   180,   180}
6180 , { -760,  -760,  -760,  -760,  -760}
6181 }
6182 }
6183 ,{{ 1560,    80,   960,  1560,   960}
6184 , { 1560,   -30,   310,  1560,   310}
6185 , { 1430,    80,   180,  1430,   180}
6186 , { 1430,  -160,   960,  1430,   960}
6187 , { 1300,   -50,    50,  1300,    50}
6188 }
6189 ,{{ 1560,   -30,   310,  1560,   310}
6190 , { 1560,   -30,   310,  1560,   310}
6191 , { 1280,  -310,    30,  1280,    30}
6192 , { -870, -1210,  -870,  -870,  -870}
6193 , { 1280,  -310,    30,  1280,    30}
6194 }
6195 ,{{ 1430,    80,   180,  1430,   180}
6196 , { 1430,  -160,   180,  1430,   180}
6197 , { 1430,    80,   180,  1430,   180}
6198 , { 1430,  -160,   180,  1430,   180}
6199 , { 1300,   -50,    50,  1300,    50}
6200 }
6201 ,{{ 1280,  -310,   960,  1280,   960}
6202 , { -960, -1300,  -960,  -960,  -960}
6203 , { 1280,  -310,    30,  1280,    30}
```

```
6204 , { 960, -630, 960, -290, 960}
6205 , { 1280, -310, 30, 1280, 30}
6206 }
6207 , {{ 1430, -150, 180, 1430, 180}
6208 , { 1430, -160, 180, 1430, 180}
6209 , { 1200, -150, -50, 1200, -50}
6210 , { 1430, -160, 180, 1430, 180}
6211 , { -760, -1100, -760, -760, -760}
6212 }
6213 }
6214 , {{{ 960, 960, 960, 960, 550}
6215 , { 550, 310, 310, 310, 550}
6216 , { 180, 180, 180, 180, 180}
6217 , { 960, 960, 960, 960, 180}
6218 , { 50, 50, 50, 50, 50}
6219 }
6220 , {{ 550, 310, 310, 310, 550}
6221 , { 550, 310, 310, 310, 550}
6222 , { 30, 30, 30, 30, 30}
6223 , { -630, -870, -630, -870, -870}
6224 , { 30, 30, 30, 30, 30}
6225 }
6226 , {{ 180, 180, 180, 180, 180}
6227 , { 180, 180, 180, 180, 180}
6228 , { 180, 180, 180, 180, 180}
6229 , { 180, 180, 180, 180, 180}
6230 , { 50, 50, 50, 50, 50}
6231 }
6232 , {{ 960, 960, 960, 960, 30}
6233 , { -720, -960, -720, -960, -960}
6234 , { 30, 30, 30, 30, 30}
6235 , { 960, 960, 960, 960, -290}
6236 , { 30, 30, 30, 30, 30}
6237 }
6238 , {{ 180, 180, 180, 180, 180}
6239 , { 180, 180, 180, 180, 180}
6240 , { -50, -50, -50, -50, -50}
6241 , { 180, 180, 180, 180, 180}
6242 , { -760, -760, -760, -760, -760}
6243 }
6244 }
6245 }
6246 }
6247 , {{{{{ INF, INF, INF, INF, INF
6248 , { INF, INF, INF, INF, INF
6249 , { INF, INF, INF, INF, INF
6250 , { INF, INF, INF, INF, INF
6251 , { INF, INF, INF, INF, INF
6252 }
6253 , {{ INF, INF, INF, INF, INF
6254 , { INF, INF, INF, INF, INF
6255 , { INF, INF, INF, INF, INF
6256 , { INF, INF, INF, INF, INF
6257 , { INF, INF, INF, INF, INF
6258 }
6259 , {{ INF, INF, INF, INF, INF
6260 , { INF, INF, INF, INF, INF
6261 , { INF, INF, INF, INF, INF
6262 , { INF, INF, INF, INF, INF
6263 , { INF, INF, INF, INF, INF
6264 }
6265 , {{ INF, INF, INF, INF, INF
6266 , { INF, INF, INF, INF, INF
6267 , { INF, INF, INF, INF, INF
6268 , { INF, INF, INF, INF, INF
6269 , { INF, INF, INF, INF, INF
6270 }
6271 , {{ INF, INF, INF, INF, INF
6272 , { INF, INF, INF, INF, INF
6273 , { INF, INF, INF, INF, INF
6274 , { INF, INF, INF, INF, INF
6275 , { INF, INF, INF, INF, INF
6276 }
6277 }
6278 , {{{ INF, INF, INF, INF, INF
6279 , { INF, INF, INF, INF, INF
6280 , { INF, INF, INF, INF, INF
6281 , { INF, INF, INF, INF, INF
6282 , { INF, INF, INF, INF, INF
6283 }
6284 , {{ INF, INF, INF, INF, INF
6285 , { INF, INF, INF, INF, INF
6286 , { INF, INF, INF, INF, INF
6287 , { INF, INF, INF, INF, INF
6288 , { INF, INF, INF, INF, INF
6289 }
6290 , {{ INF, INF, INF, INF, INF
```

```
6291 , { INF, INF, INF, INF, INF }
6292 , { INF, INF, INF, INF, INF }
6293 , { INF, INF, INF, INF, INF }
6294 , { INF, INF, INF, INF, INF }
6295 }
6296 , {{ INF, INF, INF, INF, INF }
6297 , { INF, INF, INF, INF, INF }
6298 , { INF, INF, INF, INF, INF }
6299 , { INF, INF, INF, INF, INF }
6300 , { INF, INF, INF, INF, INF }
6301 }
6302 , {{ INF, INF, INF, INF, INF }
6303 , { INF, INF, INF, INF, INF }
6304 , { INF, INF, INF, INF, INF }
6305 , { INF, INF, INF, INF, INF }
6306 , { INF, INF, INF, INF, INF }
6307 }
6308 }
6309 , {{{ INF, INF, INF, INF, INF }
6310 , { INF, INF, INF, INF, INF }
6311 , { INF, INF, INF, INF, INF }
6312 , { INF, INF, INF, INF, INF }
6313 , { INF, INF, INF, INF, INF }
6314 }
6315 , {{ INF, INF, INF, INF, INF }
6316 , { INF, INF, INF, INF, INF }
6317 , { INF, INF, INF, INF, INF }
6318 , { INF, INF, INF, INF, INF }
6319 , { INF, INF, INF, INF, INF }
6320 }
6321 , {{ INF, INF, INF, INF, INF }
6322 , { INF, INF, INF, INF, INF }
6323 , { INF, INF, INF, INF, INF }
6324 , { INF, INF, INF, INF, INF }
6325 , { INF, INF, INF, INF, INF }
6326 }
6327 , {{ INF, INF, INF, INF, INF }
6328 , { INF, INF, INF, INF, INF }
6329 , { INF, INF, INF, INF, INF }
6330 , { INF, INF, INF, INF, INF }
6331 , { INF, INF, INF, INF, INF }
6332 }
6333 , {{ INF, INF, INF, INF, INF }
6334 , { INF, INF, INF, INF, INF }
6335 , { INF, INF, INF, INF, INF }
6336 , { INF, INF, INF, INF, INF }
6337 , { INF, INF, INF, INF, INF }
6338 }
6339 }
6340 , {{{ INF, INF, INF, INF, INF }
6341 , { INF, INF, INF, INF, INF }
6342 , { INF, INF, INF, INF, INF }
6343 , { INF, INF, INF, INF, INF }
6344 , { INF, INF, INF, INF, INF }
6345 }
6346 , {{ INF, INF, INF, INF, INF }
6347 , { INF, INF, INF, INF, INF }
6348 , { INF, INF, INF, INF, INF }
6349 , { INF, INF, INF, INF, INF }
6350 , { INF, INF, INF, INF, INF }
6351 }
6352 , {{ INF, INF, INF, INF, INF }
6353 , { INF, INF, INF, INF, INF }
6354 , { INF, INF, INF, INF, INF }
6355 , { INF, INF, INF, INF, INF }
6356 , { INF, INF, INF, INF, INF }
6357 }
6358 , {{ INF, INF, INF, INF, INF }
6359 , { INF, INF, INF, INF, INF }
6360 , { INF, INF, INF, INF, INF }
6361 , { INF, INF, INF, INF, INF }
6362 , { INF, INF, INF, INF, INF }
6363 }
6364 , {{ INF, INF, INF, INF, INF }
6365 , { INF, INF, INF, INF, INF }
6366 , { INF, INF, INF, INF, INF }
6367 , { INF, INF, INF, INF, INF }
6368 , { INF, INF, INF, INF, INF }
6369 }
6370 }
6371 , {{{ INF, INF, INF, INF, INF }
6372 , { INF, INF, INF, INF, INF }
6373 , { INF, INF, INF, INF, INF }
6374 , { INF, INF, INF, INF, INF }
6375 , { INF, INF, INF, INF, INF }
6376 }
6377 , {{ INF, INF, INF, INF, INF }
```

```
6378 , { INF, INF, INF, INF, INF }
6379 , { INF, INF, INF, INF, INF }
6380 , { INF, INF, INF, INF, INF }
6381 , { INF, INF, INF, INF, INF }
6382 }
6383 , {{ INF, INF, INF, INF, INF }
6384 , { INF, INF, INF, INF, INF }
6385 , { INF, INF, INF, INF, INF }
6386 , { INF, INF, INF, INF, INF }
6387 , { INF, INF, INF, INF, INF }
6388 }
6389 , {{ INF, INF, INF, INF, INF }
6390 , { INF, INF, INF, INF, INF }
6391 , { INF, INF, INF, INF, INF }
6392 , { INF, INF, INF, INF, INF }
6393 , { INF, INF, INF, INF, INF }
6394 }
6395 , {{ INF, INF, INF, INF, INF }
6396 , { INF, INF, INF, INF, INF }
6397 , { INF, INF, INF, INF, INF }
6398 , { INF, INF, INF, INF, INF }
6399 , { INF, INF, INF, INF, INF }
6400 }
6401 }
6402 }
6403 , {{{ 1170, 780, 490, 1170, 490}
6404 , { 1120, 580, 290, 1120, 290}
6405 , { 1170, 640, 340, 1170, 340}
6406 , { 1120, 780, 490, 1120, 490}
6407 , { 1060, 530, 230, 1060, 230}
6408 }
6409 , {{ 970, 440, 170, 970, 170}
6410 , { 970, 440, 140, 970, 140}
6411 , { 660, 130, -160, 660, -160}
6412 , { 220, 220, 170, -80, 170}
6413 , { 660, 130, -160, 660, -160}
6414 }
6415 , {{ 1120, 580, 290, 1120, 290}
6416 , { 1120, 580, 290, 1120, 290}
6417 , { 1110, 580, 280, 1110, 280}
6418 , { 1120, 580, 290, 1120, 290}
6419 , { 1060, 530, 230, 1060, 230}
6420 }
6421 , {{ 780, 780, 490, 660, 490}
6422 , { -60, -60, -120, -370, -120}
6423 , { 660, 130, -160, 660, -160}
6424 , { 780, 780, 490, 470, 490}
6425 , { 660, 130, -160, 660, -160}
6426 }
6427 , {{ 1170, 640, 340, 1170, 340}
6428 , { 1120, 580, 290, 1120, 290}
6429 , { 1170, 640, 340, 1170, 340}
6430 , { 1120, 580, 290, 1120, 290}
6431 , { 40, 40, -500, -510, -500}
6432 }
6433 }
6434 , {{{ 780, 780, 490, -330, 490}
6435 , { 580, 580, 290, -620, 290}
6436 , { 640, 640, 340, -330, 340}
6437 , { 780, 780, 490, -620, 490}
6438 , { 530, 530, 230, -440, 230}
6439 }
6440 , {{ 440, 440, 140, -770, 140}
6441 , { 440, 440, 140, -770, 140}
6442 , { 130, 130, -160, -1080, -160}
6443 , { 220, 220, -70, -980, -70}
6444 , { 130, 130, -160, -1080, -160}
6445 }
6446 , {{ 580, 580, 290, -390, 290}
6447 , { 580, 580, 290, -620, 290}
6448 , { 580, 580, 280, -390, 280}
6449 , { 580, 580, 290, -620, 290}
6450 , { 530, 530, 230, -440, 230}
6451 }
6452 , {{ 780, 780, 490, -1080, 490}
6453 , { -60, -60, -350, -1270, -350}
6454 , { 130, 130, -160, -1080, -160}
6455 , { 780, 780, 490, -1680, 490}
6456 , { 130, 130, -160, -1080, -160}
6457 }
6458 , {{ 640, 640, 340, -330, 340}
6459 , { 580, 580, 290, -620, 290}
6460 , { 640, 640, 340, -330, 340}
6461 , { 580, 580, 290, -620, 290}
6462 , { 40, 40, -500, -1410, -500}
6463 }
6464 }
```

```
6465 ,{{{ 480,    470,    480,    470,    480}}
6466 ,{ 280,    270,    280,    270,    280}
6467 ,{ 340,    330,    340,    330,    340}
6468 ,{ 480,    470,    480,    470,    480}
6469 ,{ 230,    220,    230,    220,    230}
6470 }
6471 ,{{ 170,    130,    170,    130,    170}}
6472 ,{ 140,    130,    140,    130,    140}
6473 ,{ -170,   -180,   -170,   -180,   -170}
6474 ,{ 170,   -80,    170,   -80,    170}
6475 ,{ -170,   -180,   -170,   -180,   -170}
6476 }
6477 ,{{ 280,    270,    280,    270,    280}}
6478 ,{ 280,    270,    280,    270,    280}
6479 ,{ 280,    270,    280,    270,    280}
6480 ,{ 280,    270,    280,    270,    280}
6481 ,{ 230,    220,    230,    220,    230}
6482 }
6483 ,{{ 480,    470,    480,    470,    480}}
6484 ,{ -120,   -370,   -120,   -370,   -120}
6485 ,{ -170,   -180,   -170,   -180,   -170}
6486 ,{ 480,    470,    480,    470,    480}
6487 ,{ -170,   -180,   -170,   -180,   -170}
6488 }
6489 ,{{ 340,    330,    340,    330,    340}}
6490 ,{ 280,    270,    280,    270,    280}
6491 ,{ 340,    330,    340,    330,    340}
6492 ,{ 280,    270,    280,    270,    280}
6493 ,{ -500,   -510,   -500,   -510,   -500}
6494 }
6495 }
6496 ,{{{ 1170,   -510,    490,   1170,    490}}
6497 ,{ 1120,   -800,    290,   1120,    290}
6498 ,{ 1170,   -510,    340,   1170,    340}
6499 ,{ 1120,   -800,    490,   1120,    490}
6500 ,{ 1060,   -620,    230,   1060,    230}
6501 }
6502 ,{{ 970,   -950,    140,   970,    140}}
6503 ,{ 970,   -950,    140,   970,    140}
6504 ,{ 660,   -1260,   -160,   660,   -160}
6505 ,{ -70,   -1160,   -70,   -490,   -70}
6506 ,{ 660,   -1260,   -160,   660,   -160}
6507 }
6508 ,{{ 1120,   -570,    290,   1120,    290}}
6509 ,{ 1120,   -800,    290,   1120,    290}
6510 ,{ 1110,   -570,    280,   1110,    280}
6511 ,{ 1120,   -800,    290,   1120,    290}
6512 ,{ 1060,   -620,    230,   1060,    230}
6513 }
6514 ,{{ 660,   -1260,    490,   660,    490}}
6515 ,{ -350,   -1450,   -350,   -780,   -350}
6516 ,{ 660,   -1260,   -160,   660,   -160}
6517 ,{ 490,   -1860,    490,   -1190,   490}
6518 ,{ 660,   -1260,   -160,   660,   -160}
6519 }
6520 ,{{ 1170,   -510,    340,   1170,    340}}
6521 ,{ 1120,   -800,    290,   1120,    290}
6522 ,{ 1170,   -510,    340,   1170,    340}
6523 ,{ 1120,   -800,    290,   1120,    290}
6524 ,{ -500,   -1590,   -500,   -920,   -500}
6525 }
6526 }
6527 ,{{{ 480,    470,    480,    470,   -600}}
6528 ,{ 280,    270,    280,    270,   -600}
6529 ,{ 340,    330,    340,    330,   -640}
6530 ,{ 480,    470,    480,    470,   -690}
6531 ,{ 230,    220,    230,    220,   -750}
6532 }
6533 ,{{ 170,    130,    170,    130,   -600}}
6534 ,{ 140,    130,    140,    130,   -600}
6535 ,{ -170,   -180,   -170,   -180,  -1150}
6536 ,{ 170,   -80,    170,   -80,  -1050}
6537 ,{ -170,   -180,   -170,   -180,  -1150}
6538 }
6539 ,{{ 280,    270,    280,    270,   -690}}
6540 ,{ 280,    270,    280,    270,   -690}
6541 ,{ 280,    270,    280,    270,   -700}
6542 ,{ 280,    270,    280,    270,   -690}
6543 ,{ 230,    220,    230,    220,   -750}
6544 }
6545 ,{{ 480,    470,    480,    470,  -1150}}
6546 ,{ -120,   -370,   -120,   -370,  -1340}
6547 ,{ -170,   -180,   -170,   -180,  -1150}
6548 ,{ 480,    470,    480,    470,  -1750}
6549 ,{ -170,   -180,   -170,   -180,  -1150}
6550 }
6551 ,{{ 340,    330,    340,    330,   -640}}
```

```
6552 , { 280,   270,   280,   270,  -690}
6553 , { 340,   330,   340,   330,  -640}
6554 , { 280,   270,   280,   270,  -690}
6555 , { -500,  -510,  -500,  -510, -1480}
6556 }
6557 }
6558 }
6559 ,{ {{ 1140,   780,   490,  1140,   490}
6560 , { 1140,   600,   310,  1140,   310}
6561 , { 690,   150,  -140,   690,  -140}
6562 , { 780,   780,   490,   770,   490}
6563 , { 690,   190,  -140,   690,  -140}
6564 }
6565 ,{ { 1140,   600,   310,  1140,   310}
6566 , { 1140,   600,   310,  1140,   310}
6567 , { 690,   150,  -140,   690,  -140}
6568 , { -580,  -580,  -640,  -890,  -640}
6569 , { 690,   150,  -140,   690,  -140}
6570 }
6571 ,{ { 770,   240,   -50,   770,   -50}
6572 , { 770,   240,   -50,   770,   -50}
6573 , { 470,   -60,  -360,   470,  -360}
6574 , { 770,   240,   -50,   770,   -50}
6575 , { 470,   -60,  -360,   470,  -360}
6576 }
6577 ,{ { 780,   780,   490,   690,   490}
6578 , { -110,  -110,  -170,  -420,  -170}
6579 , { 690,   150,  -140,   690,  -140}
6580 , { 780,   780,   490,   470,   490}
6581 , { 690,   150,  -140,   690,  -140}
6582 }
6583 ,{ { 770,   240,   -50,   770,   -50}
6584 , { 770,   240,   -50,   770,   -50}
6585 , { 160,  -370,  -670,   160,  -670}
6586 , { 770,   240,   -50,   770,   -50}
6587 , { 190,   190,  -340,  -360,  -340}
6588 }
6589 }
6590 ,{ {{ 780,   780,   490,  -600,   490}
6591 , { 600,   600,   310,  -600,   310}
6592 , { 150,   150,  -140,  -1030,  -140}
6593 , { 780,   780,   490,  -970,   490}
6594 , { 190,   190,  -140,  -1030,  -140}
6595 }
6596 ,{ { 600,   600,   310,  -600,   310}
6597 , { 600,   600,   310,  -600,   310}
6598 , { 150,   150,  -140,  -1050,  -140}
6599 , { -580,  -580,  -880,  -1790,  -880}
6600 , { 150,   150,  -140,  -1050,  -140}
6601 }
6602 ,{ { 240,   240,   -50,  -970,   -50}
6603 , { 240,   240,   -50,  -970,   -50}
6604 , { -60,   -60,  -360,  -1030,  -360}
6605 , { 240,   240,   -50,  -970,   -50}
6606 , { -60,   -60,  -360,  -1030,  -360}
6607 }
6608 ,{ { 780,   780,   490,  -1050,   490}
6609 , { -110,  -110,  -400,  -1320,  -400}
6610 , { 150,   150,  -140,  -1050,  -140}
6611 , { 780,   780,   490,  -1680,   490}
6612 , { 150,   150,  -140,  -1050,  -140}
6613 }
6614 ,{ { 240,   240,   -50,  -970,   -50}
6615 , { 240,   240,   -50,  -970,   -50}
6616 , { -370,  -370,  -670,  -1340,  -670}
6617 , { 240,   240,   -50,  -970,   -50}
6618 , { 190,   190,  -340,  -1260,  -340}
6619 }
6620 }
6621 ,{ {{ 480,   470,   480,   470,   480}
6622 , { 300,   290,   300,   290,   300}
6623 , { -140,  -150,  -140,  -150,  -140}
6624 , { 480,   470,   480,   470,   480}
6625 , { -140,  -150,  -140,  -150,  -140}
6626 }
6627 ,{ { 300,   290,   300,   290,   300}
6628 , { 300,   290,   300,   290,   300}
6629 , { -140,  -150,  -140,  -150,  -140}
6630 , { -640,  -890,  -640,  -890,  -640}
6631 , { -140,  -150,  -140,  -150,  -140}
6632 }
6633 ,{ { -60,   -70,   -60,   -70,   -60}
6634 , { -60,   -70,   -60,   -70,   -60}
6635 , { -360,  -370,  -360,  -370,  -360}
6636 , { -60,   -70,   -60,   -70,   -60}
6637 , { -360,  -370,  -360,  -370,  -360}
6638 }
```

```

6639 , {{ 480,   470,   480,   470,   480}
6640 , { -170,  -420,  -170,  -420,  -170}
6641 , { -140,  -150,  -140,  -150,  -140}
6642 , { 480,   470,   480,   470,   480}
6643 , { -140,  -150,  -140,  -150,  -140}
6644 }
6645 , {{ -60,   -70,   -60,   -70,   -60}
6646 , { -60,   -70,   -60,   -70,   -60}
6647 , { -670,  -680,  -670,  -680,  -670}
6648 , { -60,   -70,   -60,   -70,   -60}
6649 , { -350,  -360,  -350,  -360,  -350}
6650 }
6651 }
6652 , {{ { 1140,  -780,   490,  1140,   490}
6653 , { 1140,  -780,   310,  1140,   310}
6654 , { 690,  -1210,  -140,   690,  -140}
6655 , { 770,  -1150,   490,  770,   490}
6656 , { 690,  -1210,  -140,   690,  -140}
6657 }
6658 , {{ { 1140,  -780,   310,  1140,   310}
6659 , { 1140,  -780,   310,  1140,   310}
6660 , { 690,  -1230,  -140,   690,  -140}
6661 , { -880,  -1970,  -880,  -1300,  -880}
6662 , { 690,  -1230,  -140,   690,  -140}
6663 }
6664 , {{ { 770,  -1150,   -50,   770,   -50}
6665 , { 770,  -1150,   -50,   770,   -50}
6666 , { 470,  -1210,  -360,   470,  -360}
6667 , { 770,  -1150,   -50,   770,   -50}
6668 , { 470,  -1210,  -360,   470,  -360}
6669 }
6670 , {{ { 690,  -1230,   490,   690,   490}
6671 , { -400,  -1500,  -400,  -830,  -400}
6672 , { 690,  -1230,  -140,   690,  -140}
6673 , { 490,  -1860,   490,  -1190,   490}
6674 , { 690,  -1230,  -140,   690,  -140}
6675 }
6676 , {{ { 770,  -1150,   -50,   770,   -50}
6677 , { 770,  -1150,   -50,   770,   -50}
6678 , { 160,  -1520,  -670,   160,  -670}
6679 , { 770,  -1150,   -50,   770,   -50}
6680 , { -340,  -1440,  -340,  -770,  -340}
6681 }
6682 }
6683 , {{ { { 480,   470,   480,   470,  -430}
6684 , { { 300,   290,   300,   290,  -430}
6685 , { { -140,  -150,  -140,  -150,  -1120}
6686 , { { 480,   470,   480,   470,  -1040}
6687 , { { -140,  -150,  -140,  -150,  -1120}
6688 }
6689 , {{ { { 300,   290,   300,   290,  -430}
6690 , { { 300,   290,   300,   290,  -430}
6691 , { { -140,  -150,  -140,  -150,  -1120}
6692 , { { -640,  -890,  -640,  -890,  -1860}
6693 , { { -140,  -150,  -140,  -150,  -1120}
6694 }
6695 , {{ { { -60,   -70,   -60,   -70,  -1040}
6696 , { { -60,   -70,   -60,   -70,  -1040}
6697 , { { -360,  -370,  -360,  -370,  -1340}
6698 , { { -60,   -70,   -60,   -70,  -1040}
6699 , { { -360,  -370,  -360,  -370,  -1340}
6700 }
6701 , {{ { { 480,   470,   480,   470,  -1120}
6702 , { { -170,  -420,  -170,  -420,  -1390}
6703 , { { -140,  -150,  -140,  -150,  -1120}
6704 , { { 480,   470,   480,   470,  -1750}
6705 , { { -140,  -150,  -140,  -150,  -1120}
6706 }
6707 , {{ { { -60,   -70,   -60,   -70,  -1040}
6708 , { { -60,   -70,   -60,   -70,  -1040}
6709 , { { -670,  -680,  -670,  -680,  -1650}
6710 , { { -60,   -70,   -60,   -70,  -1040}
6711 , { { -350,  -360,  -350,  -360,  -1330}
6712 }
6713 }
6714 }
6715 , {{{ { 940,   940,   650,   630,   650}
6716 , { { 220,  -130,  -190,   220,  -190}
6717 , { { 220,  -310,  -600,   220,  -600}
6718 , { { 940,   940,   650,   630,   650}
6719 , { { 220,   -70,  -600,   220,  -600}
6720 }
6721 , {{ { { 220,  -310,  -380,   220,  -380}
6722 , { { 40,  -490,  -780,   40,  -780}
6723 , { { 220,  -310,  -600,   220,  -600}
6724 , { { -320,  -320,  -380,  -630,  -380}
6725 , { { 220,  -310,  -600,   220,  -600}

```

```
6726      }
6727 ,{{ 220, -310, -600, 220, -600}
6728 ,{ 220, -310, -600, 220, -600}
6729 ,{ 220, -310, -600, 220, -600}
6730 ,{ 220, -310, -600, 220, -600}
6731 ,{ 220, -310, -600, 220, -600}
6732 }
6733 ,{{ 940, 940, 650, 630, 650}
6734 ,{ -130, -130, -190, -440, -190}
6735 ,{ 220, -310, -600, 220, -600}
6736 ,{ 940, 940, 650, 630, 650}
6737 ,{ 220, -310, -600, 220, -600}
6738 }
6739 ,{{ 220, -70, -600, 220, -600}
6740 ,{ 220, -310, -600, 220, -600}
6741 ,{ 220, -310, -600, 220, -600}
6742 ,{ 220, -310, -600, 220, -600}
6743 ,{ -70, -70, -600, -620, -600}
6744 }
6745 }
6746 ,{{{{ 940, 940, 650, -1280, 650}
6747 ,{ -130, -130, -430, -1340, -430}
6748 ,{ -310, -310, -600, -1280, -600}
6749 ,{ 940, 940, 650, -1520, 650}
6750 ,{ -70, -70, -600, -1280, -600}
6751 }
6752 ,{{ -310, -310, -600, -1520, -600}
6753 ,{ -490, -490, -780, -1700, -780}
6754 ,{ -310, -310, -600, -1520, -600}
6755 ,{ -320, -320, -620, -1530, -620}
6756 ,{ -310, -310, -600, -1520, -600}
6757 }
6758 ,{{ -310, -310, -600, -1280, -600}
6759 ,{ -310, -310, -600, -1520, -600}
6760 ,{ -310, -310, -600, -1280, -600}
6761 ,{ -310, -310, -600, -1520, -600}
6762 ,{ -310, -310, -600, -1280, -600}
6763 }
6764 ,{{ 940, 940, 650, -1340, 650}
6765 ,{ -130, -130, -430, -1340, -430}
6766 ,{ -310, -310, -600, -1520, -600}
6767 ,{ 940, 940, 650, -1520, 650}
6768 ,{ -310, -310, -600, -1520, -600}
6769 }
6770 ,{{ -70, -70, -600, -1280, -600}
6771 ,{ -310, -310, -600, -1520, -600}
6772 ,{ -310, -310, -600, -1280, -600}
6773 ,{ -310, -310, -600, -1520, -600}
6774 ,{ -70, -70, -600, -1520, -600}
6775 }
6776 }
6777 ,{{{{ 640, 630, 640, 630, 640}
6778 ,{ -190, -440, -190, -440, -190}
6779 ,{ -610, -620, -610, -620, -610}
6780 ,{ 640, 630, 640, 630, 640}
6781 ,{ -610, -620, -610, -620, -610}
6782 }
6783 ,{{ -380, -620, -380, -620, -380}
6784 ,{ -790, -800, -790, -800, -790}
6785 ,{ -610, -620, -610, -620, -610}
6786 ,{ -380, -630, -380, -630, -380}
6787 ,{ -610, -620, -610, -620, -610}
6788 }
6789 ,{{ -610, -620, -610, -620, -610}
6790 ,{ -610, -620, -610, -620, -610}
6791 ,{ -610, -620, -610, -620, -610}
6792 ,{ -610, -620, -610, -620, -610}
6793 ,{ -610, -620, -610, -620, -610}
6794 }
6795 ,{{ 640, 630, 640, 630, 640}
6796 ,{ -190, -440, -190, -440, -190}
6797 ,{ -610, -620, -610, -620, -610}
6798 ,{ 640, 630, 640, 630, 640}
6799 ,{ -610, -620, -610, -620, -610}
6800 }
6801 ,{{ -610, -620, -610, -620, -610}
6802 ,{ -610, -620, -610, -620, -610}
6803 ,{ -610, -620, -610, -620, -610}
6804 ,{ -610, -620, -610, -620, -610}
6805 ,{ -610, -620, -610, -620, -610}
6806 }
6807 }
6808 ,{{{{ 650, -1460, 650, 220, 650}
6809 ,{ 220, -1520, -430, 220, -430}
6810 ,{ 220, -1460, -600, 220, -600}
6811 ,{ 650, -1700, 650, 220, 650}
6812 ,{ 220, -1460, -600, 220, -600}
```

```

6813      }
6814 ,{{ 220, -1700, -600, 220, -600}
6815 ,{ 40, -1880, -780, 40, -780}
6816 ,{ 220, -1700, -600, 220, -600}
6817 ,{ -620, -1710, -620, -1040, -620}
6818 ,{ 220, -1700, -600, 220, -600}
6819 }
6820 ,{{ 220, -1460, -600, 220, -600}
6821 ,{ 220, -1700, -600, 220, -600}
6822 ,{ 220, -1460, -600, 220, -600}
6823 ,{ 220, -1700, -600, 220, -600}
6824 ,{ 220, -1460, -600, 220, -600}
6825 }
6826 ,{{ 650, -1520, 650, 220, 650}
6827 ,{ -430, -1520, -430, -850, -430}
6828 ,{ 220, -1700, -600, 220, -600}
6829 ,{ 650, -1700, 650, -1030, 650}
6830 ,{ 220, -1700, -600, 220, -600}
6831 }
6832 ,{{ 220, -1460, -600, 220, -600}
6833 ,{ 220, -1700, -600, 220, -600}
6834 ,{ 220, -1460, -600, 220, -600}
6835 ,{ 220, -1700, -600, 220, -600}
6836 ,{ -600, -1700, -600, -1030, -600}
6837 }
6838 }
6839 ,{{{{ 640, 630, 640, 630, -1410}
6840 ,{ -190, -440, -190, -440, -1410}
6841 ,{ -610, -620, -610, -620, -1590}
6842 ,{ 640, 630, 640, 630, -1590}
6843 ,{ -610, -620, -610, -620, -1590}
6844 }
6845 ,{{ -380, -620, -380, -620, -1530}
6846 ,{ -790, -800, -790, -800, -1530}
6847 ,{ -610, -620, -610, -620, -1590}
6848 ,{ -380, -630, -380, -630, -1600}
6849 ,{ -610, -620, -610, -620, -1590}
6850 }
6851 ,{{ -610, -620, -610, -620, -1590}
6852 ,{ -610, -620, -610, -620, -1590}
6853 ,{ -610, -620, -610, -620, -1590}
6854 ,{ -610, -620, -610, -620, -1590}
6855 ,{ -610, -620, -610, -620, -1590}
6856 }
6857 ,{{{{ 640, 630, 640, 630, -1410}
6858 ,{ -190, -440, -190, -440, -1410}
6859 ,{ -610, -620, -610, -620, -1590}
6860 ,{ 640, 630, 640, 630, -1590}
6861 ,{ -610, -620, -610, -620, -1590}
6862 }
6863 ,{{ -610, -620, -610, -620, -1590}
6864 ,{ -610, -620, -610, -620, -1590}
6865 ,{ -610, -620, -610, -620, -1590}
6866 ,{ -610, -620, -610, -620, -1590}
6867 ,{ -610, -620, -610, -620, -1590}
6868 }
6869 }
6870 }
6871 ,{{{{{{ 1490, 1490, 1200, 1280, 1200}
6872 ,{ 1280, 750, 460, 1280, 460}
6873 ,{ 780, 240, -50, 780, -50}
6874 ,{ 1490, 1490, 1200, 1190, 1200}
6875 ,{ 780, 480, -50, 780, -50}
6876 }
6877 ,{{ 1280, 750, 460, 1280, 460}
6878 ,{ 1280, 750, 460, 1280, 460}
6879 ,{ 780, 240, -50, 780, -50}
6880 ,{ -90, -90, -150, -400, -150}
6881 ,{ 780, 240, -50, 780, -50}
6882 }
6883 ,{{ 780, 240, -50, 780, -50}
6884 ,{ 780, 240, -50, 780, -50}
6885 ,{ 780, 240, -50, 780, -50}
6886 ,{ 780, 240, -50, 780, -50}
6887 ,{ 780, 240, -50, 780, -50}
6888 }
6889 ,{{{{ 1490, 1490, 1200, 1190, 1200}
6890 ,{ -260, -260, -320, -570, -320}
6891 ,{ 780, 240, -50, 780, -50}
6892 ,{ 1490, 1490, 1200, 1190, 1200}
6893 ,{ 780, 240, -50, 780, -50}
6894 }
6895 ,{{ 780, 480, -50, 780, -50}
6896 ,{ 780, 240, -50, 780, -50}
6897 ,{ 780, 240, -50, 780, -50}
6898 ,{ 780, 240, -50, 780, -50}
6899 ,{ 480, 480, -50, -60, -50}

```

```
6900      }
6901    }
6902 ,{{ 1490, 1490, 1200, -450, 1200}
6903 ,{ 750, 750, 460, -450, 460}
6904 ,{ 240, 240, -50, -720, -50}
6905 ,{ 1490, 1490, 1200, -960, 1200}
6906 ,{ 480, 480, -50, -720, -50}
6907 }
6908 ,{{ 750, 750, 460, -450, 460}
6909 ,{ 750, 750, 460, -450, 460}
6910 ,{ 240, 240, -50, -960, -50}
6911 ,{ -90, -90, -390, -1300, -390}
6912 ,{ 240, 240, -50, -960, -50}
6913 }
6914 ,{{ 240, 240, -50, -720, -50}
6915 ,{ 240, 240, -50, -960, -50}
6916 ,{ 240, 240, -50, -720, -50}
6917 ,{ 240, 240, -50, -960, -50}
6918 ,{ 240, 240, -50, -720, -50}
6919 }
6920 ,{{ 1490, 1490, 1200, -960, 1200}
6921 ,{ -260, -260, -560, -1470, -560}
6922 ,{ 240, 240, -50, -960, -50}
6923 ,{ 1490, 1490, 1200, -960, 1200}
6924 ,{ 240, 240, -50, -960, -50}
6925 }
6926 ,{{ 480, 480, -50, -720, -50}
6927 ,{ 240, 240, -50, -960, -50}
6928 ,{ 240, 240, -50, -720, -50}
6929 ,{ 240, 240, -50, -960, -50}
6930 ,{ 480, 480, -50, -960, -50}
6931 }
6932 }
6933 ,{{ 1200, 1190, 1200, 1190, 1200}
6934 ,{ 450, 440, 450, 440, 450}
6935 ,{ -50, -60, -50, -60, -50}
6936 ,{ 1200, 1190, 1200, 1190, 1200}
6937 ,{ -50, -60, -50, -60, -50}
6938 }
6939 ,{{ 450, 440, 450, 440, 450}
6940 ,{ 450, 440, 450, 440, 450}
6941 ,{ -50, -60, -50, -60, -50}
6942 ,{ -150, -400, -150, -400, -150}
6943 ,{ -50, -60, -50, -60, -50}
6944 }
6945 ,{{ -50, -60, -50, -60, -50}
6946 ,{ -50, -60, -50, -60, -50}
6947 ,{ -50, -60, -50, -60, -50}
6948 ,{ -50, -60, -50, -60, -50}
6949 ,{ -50, -60, -50, -60, -50}
6950 }
6951 ,{{ 1200, 1190, 1200, 1190, 1200}
6952 ,{ -320, -570, -320, -570, -320}
6953 ,{ -50, -60, -50, -60, -50}
6954 ,{ 1200, 1190, 1200, 1190, 1200}
6955 ,{ -50, -60, -50, -60, -50}
6956 }
6957 ,{{ -50, -60, -50, -60, -50}
6958 ,{ -50, -60, -50, -60, -50}
6959 ,{ -50, -60, -50, -60, -50}
6960 ,{ -50, -60, -50, -60, -50}
6961 ,{ -50, -60, -50, -60, -50}
6962 }
6963 }
6964 ,{{ 1280, -630, 1200, 1280, 1200}
6965 ,{ 1280, -630, 460, 1280, 460}
6966 ,{ 780, -900, -50, 780, -50}
6967 ,{ 1200, -1140, 1200, 780, 1200}
6968 ,{ 780, -900, -50, 780, -50}
6969 }
6970 ,{{ 1280, -630, 460, 1280, 460}
6971 ,{ 1280, -630, 460, 1280, 460}
6972 ,{ 780, -1140, -50, 780, -50}
6973 ,{ -390, -1480, -390, -810, -390}
6974 ,{ 780, -1140, -50, 780, -50}
6975 }
6976 ,{{ 780, -900, -50, 780, -50}
6977 ,{ 780, -1140, -50, 780, -50}
6978 ,{ 780, -900, -50, 780, -50}
6979 ,{ 780, -1140, -50, 780, -50}
6980 ,{ 780, -900, -50, 780, -50}
6981 }
6982 ,{{ 1200, -1140, 1200, 780, 1200}
6983 ,{ -560, -1650, -560, -980, -560}
6984 ,{ 780, -1140, -50, 780, -50}
6985 ,{ 1200, -1140, 1200, -470, 1200}
6986 ,{ 780, -1140, -50, 780, -50}
```

```
6987    }
6988 ,{{ 780, -900, -50, 780, -50}
6989 ,{ 780, -1140, -50, 780, -50}
6990 ,{ 780, -900, -50, 780, -50}
6991 ,{ 780, -1140, -50, 780, -50}
6992 ,{ -50, -1140, -50, -470, -50}
6993 }
6994 }
6995 ,{{ 1200, 1190, 1200, 1190, -280}
6996 ,{ 450, 440, 450, 440, -280}
6997 ,{ -50, -60, -50, -60, -1030}
6998 ,{ 1200, 1190, 1200, 1190, -1030}
6999 ,{ -50, -60, -50, -60, -1030}
7000 }
7001 ,{{ 450, 440, 450, 440, -280}
7002 ,{ 450, 440, 450, 440, -280}
7003 ,{ -50, -60, -50, -60, -1030}
7004 ,{ -150, -400, -150, -400, -1370}
7005 ,{ -50, -60, -50, -60, -1030}
7006 }
7007 ,{{ -50, -60, -50, -60, -1030}
7008 ,{ -50, -60, -50, -60, -1030}
7009 ,{ -50, -60, -50, -60, -1030}
7010 ,{ -50, -60, -50, -60, -1030}
7011 ,{ -50, -60, -50, -60, -1030}
7012 }
7013 ,{{ 1200, 1190, 1200, 1190, -1030}
7014 ,{ -320, -570, -320, -570, -1540}
7015 ,{ -50, -60, -50, -60, -1030}
7016 ,{ 1200, 1190, 1200, 1190, -1030}
7017 ,{ -50, -60, -50, -60, -1030}
7018 }
7019 ,{{ -50, -60, -50, -60, -1030}
7020 ,{ -50, -60, -50, -60, -1030}
7021 ,{ -50, -60, -50, -60, -1030}
7022 ,{ -50, -60, -50, -60, -1030}
7023 ,{ -50, -60, -50, -60, -1030}
7024 }
7025 }
7026 }
7027 ,{{{{ 1870, 1870, 1570, 1870, 1570}
7028 ,{ 1870, 1340, 1040, 1870, 1040}
7029 ,{ 1570, 1040, 740, 1570, 740}
7030 ,{ 1870, 1870, 1570, 1570, 1570}
7031 ,{ 1570, 1040, 740, 1570, 740}
7032 }
7033 ,{{ 1870, 1340, 1040, 1870, 1040}
7034 ,{ 1870, 1340, 1040, 1870, 1040}
7035 ,{ 1560, 1030, 730, 1560, 730}
7036 ,{ -50, -50, -110, -360, -110}
7037 ,{ 1560, 1030, 730, 1560, 730}
7038 }
7039 ,{{ 1570, 1040, 750, 1570, 750}
7040 ,{ 1570, 1040, 750, 1570, 750}
7041 ,{ 1570, 1040, 740, 1570, 740}
7042 ,{ 1570, 1040, 750, 1570, 750}
7043 ,{ 1570, 1040, 740, 1570, 740}
7044 }
7045 ,{{ 1870, 1870, 1570, 1560, 1570}
7046 ,{ 130, 130, 70, -180, 70}
7047 ,{ 1560, 1030, 730, 1560, 730}
7048 ,{ 1870, 1870, 1570, 1560, 1570}
7049 ,{ 1560, 1030, 730, 1560, 730}
7050 }
7051 ,{{ 1570, 1040, 750, 1570, 750}
7052 ,{ 1570, 1040, 750, 1570, 750}
7053 ,{ 1570, 1040, 740, 1570, 740}
7054 ,{ 1570, 1040, 750, 1570, 750}
7055 ,{ 300, 300, -230, -250, -230}
7056 }
7057 }
7058 ,{{{{ 1870, 1870, 1570, 130, 1570}
7059 ,{ 1340, 1340, 1040, 130, 1040}
7060 ,{ 1040, 1040, 740, 70, 740}
7061 ,{ 1870, 1870, 1570, -160, 1570}
7062 ,{ 1040, 1040, 740, 70, 740}
7063 }
7064 ,{{ 1340, 1340, 1040, 130, 1040}
7065 ,{ 1340, 1340, 1040, 130, 1040}
7066 ,{ 1030, 1030, 730, -180, 730}
7067 ,{ -50, -50, -340, -1260, -340}
7068 ,{ 1030, 1030, 730, -180, 730}
7069 }
7070 ,{{ 1040, 1040, 750, 70, 750}
7071 ,{ 1040, 1040, 750, -160, 750}
7072 ,{ 1040, 1040, 740, 70, 740}
7073 ,{ 1040, 1040, 750, -160, 750}
```

```
7074 , { 1040, 1040, 740, 70, 740}
7075 }
7076 , {{ 1870, 1870, 1570, -180, 1570}
7077 , { 130, 130, -160, -1080, -160}
7078 , { 1030, 1030, 730, -180, 730}
7079 , { 1870, 1870, 1570, -590, 1570}
7080 , { 1030, 1030, 730, -180, 730}
7081 }
7082 , {{ 1040, 1040, 750, 70, 750}
7083 , { 1040, 1040, 750, -160, 750}
7084 , { 1040, 1040, 740, 70, 740}
7085 , { 1040, 1040, 750, -160, 750}
7086 , { 300, 300, -230, -1150, -230}
7087 }
7088 }
7089 , {{{ 1570, 1560, 1570, 1560, 1570}
7090 , { 1040, 1030, 1040, 1030, 1040}
7091 , { 740, 730, 740, 730, 740}
7092 , { 1570, 1560, 1570, 1560, 1570}
7093 , { 740, 730, 740, 730, 740}
7094 }
7095 , {{ 1040, 1030, 1040, 1030, 1040}
7096 , { 1040, 1030, 1040, 1030, 1040}
7097 , { 730, 720, 730, 720, 730}
7098 , { -110, -360, -110, -360, -110}
7099 , { 730, 720, 730, 720, 730}
7100 }
7101 , {{ 740, 730, 740, 730, 740}
7102 , { 740, 730, 740, 730, 740}
7103 , { 740, 730, 740, 730, 740}
7104 , { 740, 730, 740, 730, 740}
7105 , { 740, 730, 740, 730, 740}
7106 }
7107 , {{ 1570, 1560, 1570, 1560, 1570}
7108 , { 70, -180, 70, -180, 70}
7109 , { 730, 720, 730, 720, 730}
7110 , { 1570, 1560, 1570, 1560, 1570}
7111 , { 730, 720, 730, 720, 730}
7112 }
7113 , {{ 740, 730, 740, 730, 740}
7114 , { 740, 730, 740, 730, 740}
7115 , { 740, 730, 740, 730, 740}
7116 , { 740, 730, 740, 730, 740}
7117 , { -240, -250, -240, -250, -240}
7118 }
7119 }
7120 , {{{ 1870, -50, 1570, 1870, 1570}
7121 , { 1870, -50, 1040, 1870, 1040}
7122 , { 1570, -110, 740, 1570, 740}
7123 , { 1570, -340, 1570, 1570, 1570}
7124 , { 1570, -110, 740, 1570, 740}
7125 }
7126 , {{ 1870, -50, 1040, 1870, 1040}
7127 , { 1870, -50, 1040, 1870, 1040}
7128 , { 1560, -360, 730, 1560, 730}
7129 , { -340, -1440, -340, -770, -340}
7130 , { 1560, -360, 730, 1560, 730}
7131 }
7132 , {{ 1570, -110, 750, 1570, 750}
7133 , { 1570, -340, 750, 1570, 750}
7134 , { 1570, -110, 740, 1570, 740}
7135 , { 1570, -340, 750, 1570, 750}
7136 , { 1570, -110, 740, 1570, 740}
7137 }
7138 , {{ 1570, -360, 1570, 1560, 1570}
7139 , { -160, -1260, -160, -590, -160}
7140 , { 1560, -360, 730, 1560, 730}
7141 , { 1570, -770, 1570, -100, 1570}
7142 , { 1560, -360, 730, 1560, 730}
7143 }
7144 , {{ 1570, -110, 750, 1570, 750}
7145 , { 1570, -340, 750, 1570, 750}
7146 , { 1570, -110, 740, 1570, 740}
7147 , { 1570, -340, 750, 1570, 750}
7148 , { -230, -1330, -230, -660, -230}
7149 }
7150 }
7151 , {{{ 1570, 1560, 1570, 1560, 300}
7152 , { 1040, 1030, 1040, 1030, 300}
7153 , { 740, 730, 740, 730, -240}
7154 , { 1570, 1560, 1570, 1560, -230}
7155 , { 740, 730, 740, 730, -240}
7156 }
7157 , {{ 1040, 1030, 1040, 1030, 300}
7158 , { 1040, 1030, 1040, 1030, 300}
7159 , { 730, 720, 730, 720, -250}
7160 , { -110, -360, -110, -360, -1330}
```

```
7161 , { 730,    720,    730,    720,   -250}
7162 }
7163 , {{ 740,    730,    740,    730,   -230}
7164 , { 740,    730,    740,    730,   -230}
7165 , { 740,    730,    740,    730,   -240}
7166 , { 740,    730,    740,    730,   -230}
7167 , { 740,    730,    740,    730,   -240}
7168 }
7169 , {{ 1570,   1560,   1570,   1560,   -250}
7170 , { 70,     -180,     70,     -180,  -1150}
7171 , { 730,    720,    730,    720,   -250}
7172 , { 1570,   1560,   1570,   1560,   -660}
7173 , { 730,    720,    730,    720,   -250}
7174 }
7175 , {{ 740,    730,    740,    730,   -230}
7176 , { 740,    730,    740,    730,   -230}
7177 , { 740,    730,    740,    730,   -240}
7178 , { 740,    730,    740,    730,   -230}
7179 , { -240,   -250,   -240,   -250,  -1220}
7180 }
7181 }
7182 }
7183 , {{{ 2050,   2050,   1760,   1930,   1760}
7184 , { 1930,   1400,   1110,   1930,   1110}
7185 , { 1800,   1270,   980,   1800,   980}
7186 , { 2050,   2050,   1760,   1800,   1760}
7187 , { 1670,   1140,   850,   1670,   850}
7188 }
7189 , {{ 1930,   1400,   1110,   1930,   1110}
7190 , { 1930,   1400,   1110,   1930,   1110}
7191 , { 1650,   1120,   830,   1650,   830}
7192 , { 0,      0,     -60,   -310,   -60}
7193 , { 1650,   1120,   830,   1650,   830}
7194 }
7195 , {{ 1800,   1270,   980,   1800,   980}
7196 , { 1800,   1270,   980,   1800,   980}
7197 , { 1800,   1270,   980,   1800,   980}
7198 , { 1800,   1270,   980,   1800,   980}
7199 , { 1670,   1140,   850,   1670,   850}
7200 }
7201 , {{ 2050,   2050,   1760,   1740,   1760}
7202 , { -300,   -300,   -360,   -610,   -360}
7203 , { 1650,   1120,   830,   1650,   830}
7204 , { 2050,   2050,   1760,   1740,   1760}
7205 , { 1650,   1120,   830,   1650,   830}
7206 }
7207 , {{ 1800,   1270,   980,   1800,   980}
7208 , { 1800,   1270,   980,   1800,   980}
7209 , { 1360,   830,   540,   1360,   540}
7210 , { 1800,   1270,   980,   1800,   980}
7211 , { 570,    570,     40,     20,     40}
7212 }
7213 }
7214 , {{{ 2050,   2050,   1760,   300,   1760}
7215 , { 1400,   1400,   1110,   190,   1110}
7216 , { 1270,   1270,   980,   300,   980}
7217 , { 2050,   2050,   1760,   60,   1760}
7218 , { 1140,   1140,   850,   180,   850}
7219 }
7220 , {{ 1400,   1400,   1110,   190,   1110}
7221 , { 1400,   1400,   1110,   190,   1110}
7222 , { 1120,   1120,   830,   -80,   830}
7223 , { 0,      0,     -290,  -1210,  -290}
7224 , { 1120,   1120,   830,   -80,   830}
7225 }
7226 , {{ 1270,   1270,   980,   300,   980}
7227 , { 1270,   1270,   980,   60,   980}
7228 , { 1270,   1270,   980,   300,   980}
7229 , { 1270,   1270,   980,   60,   980}
7230 , { 1140,   1140,   850,   180,   850}
7231 }
7232 , {{ 2050,   2050,   1760,   -80,   1760}
7233 , { -300,   -300,   -590,  -1510,  -590}
7234 , { 1120,   1120,   830,   -80,   830}
7235 , { 2050,   2050,   1760,   -400,   1760}
7236 , { 1120,   1120,   830,   -80,   830}
7237 }
7238 , {{ 1270,   1270,   980,     60,   980}
7239 , { 1270,   1270,   980,     60,   980}
7240 , { 830,    830,     540,   -130,   540}
7241 , { 1270,   1270,   980,     60,   980}
7242 , { 570,    570,     40,   -870,   40}
7243 }
7244 }
7245 , {{{ 1750,   1740,   1750,   1740,   1750}
7246 , { 1100,   1090,   1100,   1090,   1100}
7247 , { 970,    960,    970,    960,    970}
```

```
7248 ,{ 1750, 1740, 1750, 1740, 1750}
7249 ,{ 840, 830, 840, 830, 840}
7250 }
7251 ,{{ 1100, 1090, 1100, 1090, 1100}
7252 ,{ 1100, 1090, 1100, 1090, 1100}
7253 ,{ 820, 810, 820, 810, 820}
7254 ,{ -60, -310, -60, -310, -60}
7255 ,{ 820, 810, 820, 810, 820}
7256 }
7257 ,{{ 970, 960, 970, 960, 970}
7258 ,{ 970, 960, 970, 960, 970}
7259 ,{ 970, 960, 970, 960, 970}
7260 ,{ 970, 960, 970, 960, 970}
7261 ,{ 840, 830, 840, 830, 840}
7262 }
7263 ,{{ 1750, 1740, 1750, 1740, 1750}
7264 ,{ -360, -610, -360, -610, -360}
7265 ,{ 820, 810, 820, 810, 820}
7266 ,{ 1750, 1740, 1750, 1740, 1750}
7267 ,{ 820, 810, 820, 810, 820}
7268 }
7269 ,{{ 970, 960, 970, 960, 970}
7270 ,{ 970, 960, 970, 960, 970}
7271 ,{ 530, 520, 530, 520, 530}
7272 ,{ 970, 960, 970, 960, 970}
7273 ,{ 30, 20, 30, 20, 30}
7274 }
7275 }
7276 ,{{ { 1930, 130, 1760, 1930, 1760}
7277 ,{ 1930, 10, 1110, 1930, 1110}
7278 ,{ 1800, 130, 980, 1800, 980}
7279 ,{ 1800, -110, 1760, 1800, 1760}
7280 ,{ 1670, 0, 850, 1670, 850}
7281 }
7282 ,{{ { 1930, 10, 1110, 1930, 1110}
7283 ,{ 1930, 10, 1110, 1930, 1110}
7284 ,{ 1650, -260, 830, 1650, 830}
7285 ,{ -290, -1390, -290, -720, -290}
7286 ,{ 1650, -260, 830, 1650, 830}
7287 }
7288 ,{{ { 1800, 130, 980, 1800, 980}
7289 ,{ 1800, -110, 980, 1800, 980}
7290 ,{ 1800, 130, 980, 1800, 980}
7291 ,{ 1800, -110, 980, 1800, 980}
7292 ,{ 1670, 0, 850, 1670, 850}
7293 }
7294 ,{{ { 1760, -260, 1760, 1650, 1760}
7295 ,{ -590, -1690, -590, -1020, -590}
7296 ,{ 1650, -260, 830, 1650, 830}
7297 ,{ 1760, -580, 1760, 80, 1760}
7298 ,{ 1650, -260, 830, 1650, 830}
7299 }
7300 ,{{ { 1800, -110, 980, 1800, 980}
7301 ,{ 1800, -110, 980, 1800, 980}
7302 ,{ 1360, -310, 540, 1360, 540}
7303 ,{ 1800, -110, 980, 1800, 980}
7304 ,{ 40, -1050, 40, -380, 40}
7305 }
7306 }
7307 ,{{ { 1750, 1740, 1750, 1740, 360}
7308 ,{ 1100, 1090, 1100, 1090, 360}
7309 ,{ 970, 960, 970, 960, 0}
7310 ,{ 1750, 1740, 1750, 1740, 0}
7311 ,{ 840, 830, 840, 830, -130}
7312 }
7313 ,{{ { 1100, 1090, 1100, 1090, 360}
7314 ,{ 1100, 1090, 1100, 1090, 360}
7315 ,{ 820, 810, 820, 810, -150}
7316 ,{ -60, -310, -60, -310, -1280}
7317 ,{ 820, 810, 820, 810, -150}
7318 }
7319 ,{{ { 970, 960, 970, 960, 0}
7320 ,{ 970, 960, 970, 960, 0}
7321 ,{ 970, 960, 970, 960, 0}
7322 ,{ 970, 960, 970, 960, 0}
7323 ,{ 840, 830, 840, 830, -130}
7324 }
7325 ,{{ { 1750, 1740, 1750, 1740, -150}
7326 ,{ -360, -610, -360, -610, -1580}
7327 ,{ 820, 810, 820, 810, -150}
7328 ,{ 1750, 1740, 1750, 1740, -470}
7329 ,{ 820, 810, 820, 810, -150}
7330 }
7331 ,{{ { 970, 960, 970, 960, 0}
7332 ,{ 970, 960, 970, 960, 0}
7333 ,{ 530, 520, 530, 520, -440}
7334 ,{ 970, 960, 970, 960, 0}
```

```

7335 , { 30, 20, 30, 20, -940}
7336 }
7337 }
7338 }
7339 ,{{{{ 2050, 2050, 1760, 1930, 1760}
7340 ,{ 1930, 1400, 1110, 1930, 1110}
7341 ,{ 1800, 1270, 980, 1800, 980}
7342 ,{ 2050, 2050, 1760, 1800, 1760}
7343 ,{ 1670, 1140, 850, 1670, 850}
7344 }
7345 ,{{ 1930, 1400, 1110, 1930, 1110}
7346 ,{ 1930, 1400, 1110, 1930, 1110}
7347 ,{ 1650, 1120, 830, 1650, 830}
7348 ,{ 220, 220, 170, -80, 170}
7349 ,{ 1650, 1120, 830, 1650, 830}
7350 }
7351 ,{{ 1800, 1270, 980, 1800, 980}
7352 ,{ 1800, 1270, 980, 1800, 980}
7353 ,{ 1800, 1270, 980, 1800, 980}
7354 ,{ 1800, 1270, 980, 1800, 980}
7355 ,{ 1670, 1140, 850, 1670, 850}
7356 }
7357 ,{{ 2050, 2050, 1760, 1740, 1760}
7358 ,{ 130, 130, 70, -180, 70}
7359 ,{ 1650, 1120, 830, 1650, 830}
7360 ,{ 2050, 2050, 1760, 1740, 1760}
7361 ,{ 1650, 1120, 830, 1650, 830}
7362 }
7363 ,{{ 1800, 1270, 980, 1800, 980}
7364 ,{ 1800, 1270, 980, 1800, 980}
7365 ,{ 1570, 1040, 740, 1570, 740}
7366 ,{ 1800, 1270, 980, 1800, 980}
7367 ,{ 570, 570, 40, 20, 40}
7368 }
7369 }
7370 ,{{{{ 2050, 2050, 1760, 300, 1760}
7371 ,{ 1400, 1400, 1110, 190, 1110}
7372 ,{ 1270, 1270, 980, 300, 980}
7373 ,{ 2050, 2050, 1760, 60, 1760}
7374 ,{ 1140, 1140, 850, 180, 850}
7375 }
7376 ,{{ 1400, 1400, 1110, 190, 1110}
7377 ,{ 1400, 1400, 1110, 190, 1110}
7378 ,{ 1120, 1120, 830, -80, 830}
7379 ,{ 220, 220, -70, -980, -70}
7380 ,{ 1120, 1120, 830, -80, 830}
7381 }
7382 ,{{ 1270, 1270, 980, 300, 980}
7383 ,{ 1270, 1270, 980, 60, 980}
7384 ,{ 1270, 1270, 980, 300, 980}
7385 ,{ 1270, 1270, 980, 60, 980}
7386 ,{ 1140, 1140, 850, 180, 850}
7387 }
7388 ,{{ 2050, 2050, 1760, -80, 1760}
7389 ,{ 130, 130, -160, -1080, -160}
7390 ,{ 1120, 1120, 830, -80, 830}
7391 ,{ 2050, 2050, 1760, -400, 1760}
7392 ,{ 1120, 1120, 830, -80, 830}
7393 }
7394 ,{{ 1270, 1270, 980, 70, 980}
7395 ,{ 1270, 1270, 980, 60, 980}
7396 ,{ 1040, 1040, 740, 70, 740}
7397 ,{ 1270, 1270, 980, 60, 980}
7398 ,{ 570, 570, 40, -870, 40}
7399 }
7400 }
7401 ,{{{{ 1750, 1740, 1750, 1740, 1750}
7402 ,{ 1100, 1090, 1100, 1090, 1100}
7403 ,{ 970, 960, 970, 960, 970}
7404 ,{ 1750, 1740, 1750, 1740, 1750}
7405 ,{ 840, 830, 840, 830, 840}
7406 }
7407 ,{{ 1100, 1090, 1100, 1090, 1100}
7408 ,{ 1100, 1090, 1100, 1090, 1100}
7409 ,{ 820, 810, 820, 810, 820}
7410 ,{ 170, -80, 170, -80, 170}
7411 ,{ 820, 810, 820, 810, 820}
7412 }
7413 ,{{ 970, 960, 970, 960, 970}
7414 ,{ 970, 960, 970, 960, 970}
7415 ,{ 970, 960, 970, 960, 970}
7416 ,{ 970, 960, 970, 960, 970}
7417 ,{ 840, 830, 840, 830, 840}
7418 }
7419 ,{{ 1750, 1740, 1750, 1740, 1750}
7420 ,{ 70, -180, 70, -180, 70}
7421 ,{ 820, 810, 820, 810, 820}

```

```

7422 , { 1750, 1740, 1750, 1740, 1750}
7423 , { 820, 810, 820, 810, 820}
7424 }
7425 , {{ 970, 960, 970, 960, 970}
7426 , { 970, 960, 970, 960, 970}
7427 , { 740, 730, 740, 730, 740}
7428 , { 970, 960, 970, 960, 970}
7429 , { 30, 20, 30, 20, 30}
7430 }
7431 }
7432 , {{{ 1930, 130, 1760, 1930, 1760}
7433 , { 1930, 10, 1110, 1930, 1110}
7434 , { 1800, 130, 980, 1800, 980}
7435 , { 1800, -110, 1760, 1800, 1760}
7436 , { 1670, 0, 850, 1670, 850}
7437 }
7438 , {{ 1930, 10, 1110, 1930, 1110}
7439 , { 1930, 10, 1110, 1930, 1110}
7440 , { 1650, -260, 830, 1650, 830}
7441 , { -70, -1160, -70, -490, -70}
7442 , { 1650, -260, 830, 1650, 830}
7443 }
7444 , {{ 1800, 130, 980, 1800, 980}
7445 , { 1800, -110, 980, 1800, 980}
7446 , { 1800, 130, 980, 1800, 980}
7447 , { 1800, -110, 980, 1800, 980}
7448 , { 1670, 0, 850, 1670, 850}
7449 }
7450 , {{ 1760, -260, 1760, 1650, 1760}
7451 , { -160, -1260, -160, -590, -160}
7452 , { 1650, -260, 830, 1650, 830}
7453 , { 1760, -580, 1760, 80, 1760}
7454 , { 1650, -260, 830, 1650, 830}
7455 }
7456 , {{ 1800, -110, 980, 1800, 980}
7457 , { 1800, -110, 980, 1800, 980}
7458 , { 1570, -110, 740, 1570, 740}
7459 , { 1800, -110, 980, 1800, 980}
7460 , { 40, -1050, 40, -380, 40}
7461 }
7462 }
7463 , {{{ 1750, 1740, 1750, 1740, 360}
7464 , { 1100, 1090, 1100, 1090, 360}
7465 , { 970, 960, 970, 960, 0}
7466 , { 1750, 1740, 1750, 1740, 0}
7467 , { 840, 830, 840, 830, -130}
7468 }
7469 , {{ 1100, 1090, 1100, 1090, 360}
7470 , { 1100, 1090, 1100, 1090, 360}
7471 , { 820, 810, 820, 810, -150}
7472 , { 170, -80, 170, -80, -1050}
7473 , { 820, 810, 820, 810, -150}
7474 }
7475 , {{ 970, 960, 970, 960, 0}
7476 , { 970, 960, 970, 960, 0}
7477 , { 970, 960, 970, 960, 0}
7478 , { 970, 960, 970, 960, 0}
7479 , { 840, 830, 840, 830, -130}
7480 }
7481 , {{ 1750, 1740, 1750, 1740, -150}
7482 , { 70, -180, 70, -180, -1150}
7483 , { 820, 810, 820, 810, -150}
7484 , { 1750, 1740, 1750, 1740, -470}
7485 , { 820, 810, 820, 810, -150}
7486 }
7487 , {{ 970, 960, 970, 960, 0}
7488 , { 970, 960, 970, 960, 0}
7489 , { 740, 730, 740, 730, -240}
7490 , { 970, 960, 970, 960, 0}
7491 , { 30, 20, 30, 20, -940}
7492 }
7493 }
7494 }
7495 }
7496 , {{{{{ INF, INF, INF, INF, INF
7497 , { INF, INF, INF, INF, INF
7498 , { INF, INF, INF, INF, INF
7499 , { INF, INF, INF, INF, INF
7500 , { INF, INF, INF, INF, INF
7501 }
7502 , {{ INF, INF, INF, INF, INF
7503 , { INF, INF, INF, INF, INF
7504 , { INF, INF, INF, INF, INF
7505 , { INF, INF, INF, INF, INF
7506 , { INF, INF, INF, INF, INF
7507 }
7508 , { { INF, INF, INF, INF, INF

```

```
7509 , { INF, INF, INF, INF, INF }
7510 , { INF, INF, INF, INF, INF }
7511 , { INF, INF, INF, INF, INF }
7512 , { INF, INF, INF, INF, INF }
7513 }
7514 , {{ INF, INF, INF, INF, INF }
7515 , { INF, INF, INF, INF, INF }
7516 , { INF, INF, INF, INF, INF }
7517 , { INF, INF, INF, INF, INF }
7518 , { INF, INF, INF, INF, INF }
7519 }
7520 , {{ INF, INF, INF, INF, INF }
7521 , { INF, INF, INF, INF, INF }
7522 , { INF, INF, INF, INF, INF }
7523 , { INF, INF, INF, INF, INF }
7524 , { INF, INF, INF, INF, INF }
7525 }
7526 }
7527 , {{{ INF, INF, INF, INF, INF }
7528 , { INF, INF, INF, INF, INF }
7529 , { INF, INF, INF, INF, INF }
7530 , { INF, INF, INF, INF, INF }
7531 , { INF, INF, INF, INF, INF }
7532 }
7533 , {{ INF, INF, INF, INF, INF }
7534 , { INF, INF, INF, INF, INF }
7535 , { INF, INF, INF, INF, INF }
7536 , { INF, INF, INF, INF, INF }
7537 , { INF, INF, INF, INF, INF }
7538 }
7539 , {{ INF, INF, INF, INF, INF }
7540 , { INF, INF, INF, INF, INF }
7541 , { INF, INF, INF, INF, INF }
7542 , { INF, INF, INF, INF, INF }
7543 , { INF, INF, INF, INF, INF }
7544 }
7545 , {{ INF, INF, INF, INF, INF }
7546 , { INF, INF, INF, INF, INF }
7547 , { INF, INF, INF, INF, INF }
7548 , { INF, INF, INF, INF, INF }
7549 , { INF, INF, INF, INF, INF }
7550 }
7551 , {{ INF, INF, INF, INF, INF }
7552 , { INF, INF, INF, INF, INF }
7553 , { INF, INF, INF, INF, INF }
7554 , { INF, INF, INF, INF, INF }
7555 , { INF, INF, INF, INF, INF }
7556 }
7557 }
7558 , {{{ INF, INF, INF, INF, INF }
7559 , { INF, INF, INF, INF, INF }
7560 , { INF, INF, INF, INF, INF }
7561 , { INF, INF, INF, INF, INF }
7562 , { INF, INF, INF, INF, INF }
7563 }
7564 , {{ INF, INF, INF, INF, INF }
7565 , { INF, INF, INF, INF, INF }
7566 , { INF, INF, INF, INF, INF }
7567 , { INF, INF, INF, INF, INF }
7568 , { INF, INF, INF, INF, INF }
7569 }
7570 , {{ INF, INF, INF, INF, INF }
7571 , { INF, INF, INF, INF, INF }
7572 , { INF, INF, INF, INF, INF }
7573 , { INF, INF, INF, INF, INF }
7574 , { INF, INF, INF, INF, INF }
7575 }
7576 , {{ INF, INF, INF, INF, INF }
7577 , { INF, INF, INF, INF, INF }
7578 , { INF, INF, INF, INF, INF }
7579 , { INF, INF, INF, INF, INF }
7580 , { INF, INF, INF, INF, INF }
7581 }
7582 , {{ INF, INF, INF, INF, INF }
7583 , { INF, INF, INF, INF, INF }
7584 , { INF, INF, INF, INF, INF }
7585 , { INF, INF, INF, INF, INF }
7586 , { INF, INF, INF, INF, INF }
7587 }
7588 }
7589 , {{{ INF, INF, INF, INF, INF }
7590 , { INF, INF, INF, INF, INF }
7591 , { INF, INF, INF, INF, INF }
7592 , { INF, INF, INF, INF, INF }
7593 , { INF, INF, INF, INF, INF }
7594 }
7595 , {{ INF, INF, INF, INF, INF }
```

```
7596 , { INF, INF, INF, INF, INF }
7597 , { INF, INF, INF, INF, INF }
7598 , { INF, INF, INF, INF, INF }
7599 , { INF, INF, INF, INF, INF }
7600 }
7601 , {{ INF, INF, INF, INF, INF }
7602 , { INF, INF, INF, INF, INF }
7603 , { INF, INF, INF, INF, INF }
7604 , { INF, INF, INF, INF, INF }
7605 , { INF, INF, INF, INF, INF }
7606 }
7607 , {{ INF, INF, INF, INF, INF }
7608 , { INF, INF, INF, INF, INF }
7609 , { INF, INF, INF, INF, INF }
7610 , { INF, INF, INF, INF, INF }
7611 , { INF, INF, INF, INF, INF }
7612 }
7613 , {{ INF, INF, INF, INF, INF }
7614 , { INF, INF, INF, INF, INF }
7615 , { INF, INF, INF, INF, INF }
7616 , { INF, INF, INF, INF, INF }
7617 , { INF, INF, INF, INF, INF }
7618 }
7619 }
7620 , {{{ INF, INF, INF, INF, INF }
7621 , { INF, INF, INF, INF, INF }
7622 , { INF, INF, INF, INF, INF }
7623 , { INF, INF, INF, INF, INF }
7624 , { INF, INF, INF, INF, INF }
7625 }
7626 , {{ INF, INF, INF, INF, INF }
7627 , { INF, INF, INF, INF, INF }
7628 , { INF, INF, INF, INF, INF }
7629 , { INF, INF, INF, INF, INF }
7630 , { INF, INF, INF, INF, INF }
7631 }
7632 , {{ INF, INF, INF, INF, INF }
7633 , { INF, INF, INF, INF, INF }
7634 , { INF, INF, INF, INF, INF }
7635 , { INF, INF, INF, INF, INF }
7636 , { INF, INF, INF, INF, INF }
7637 }
7638 , {{ INF, INF, INF, INF, INF }
7639 , { INF, INF, INF, INF, INF }
7640 , { INF, INF, INF, INF, INF }
7641 , { INF, INF, INF, INF, INF }
7642 , { INF, INF, INF, INF, INF }
7643 }
7644 , {{ INF, INF, INF, INF, INF }
7645 , { INF, INF, INF, INF, INF }
7646 , { INF, INF, INF, INF, INF }
7647 , { INF, INF, INF, INF, INF }
7648 , { INF, INF, INF, INF, INF }
7649 }
7650 }
7651 }
7652 , {{{{ 1350, 850, 720, 1350, 720}
7653 , { 1300, 650, 520, 1300, 520}
7654 , { 1350, 700, 570, 1350, 570}
7655 , { 1300, 850, 720, 1300, 720}
7656 , { 1250, 590, 460, 1250, 460}
7657 }
7658 , {{ 1160, 500, 400, 1160, 370}
7659 , { 1160, 500, 370, 1160, 370}
7660 , { 850, 190, 60, 850, 60}
7661 , { 400, 290, 400, 10, 160}
7662 , { 850, 190, 60, 850, 60}
7663 }
7664 , {{ 1300, 650, 520, 1300, 520}
7665 , { 1300, 650, 520, 1300, 520}
7666 , { 1290, 640, 510, 1290, 510}
7667 , { 1300, 650, 520, 1300, 520}
7668 , { 1250, 590, 460, 1250, 460}
7669 }
7670 , {{ 850, 850, 720, 850, 720}
7671 , { 120, 0, 120, -270, -120}
7672 , { 850, 190, 60, 850, 60}
7673 , { 850, 850, 720, 570, 720}
7674 , { 850, 190, 60, 850, 60}
7675 }
7676 , {{ 1350, 700, 570, 1350, 570}
7677 , { 1300, 650, 520, 1300, 520}
7678 , { 1350, 700, 570, 1350, 570}
7679 , { 1300, 650, 520, 1300, 520}
7680 , { 100, 100, -270, -420, -270}
7681 }
7682 }
```

```
7683 ,{{{ 850,     850,     720,    -760,     720}}
7684 ,{ 650,     650,     520,   -1050,     520}
7685 ,{ 700,     700,     570,    -760,     570}
7686 ,{ 850,     850,     720,   -1050,     720}
7687 ,{ 590,     590,     460,   -870,     460}
7688 }
7689 ,{{ 500,     500,     370,   -1200,     370}
7690 ,{ 500,     500,     370,   -1200,     370}
7691 ,{ 190,     190,      60,   -1510,      60}
7692 ,{ 290,     290,     160,   -1410,     160}
7693 ,{ 190,     190,      60,   -1510,      60}
7694 }
7695 ,{{ 650,     650,     520,   -820,     520}
7696 ,{ 650,     650,     520,   -1050,     520}
7697 ,{ 640,     640,     510,   -820,     510}
7698 ,{ 650,     650,     520,   -1050,     520}
7699 ,{ 590,     590,     460,   -870,     460}
7700 }
7701 ,{{ 850,     850,     720,   -1510,     720}
7702 ,{ 0,       0,   -120,   -1700,   -120}
7703 ,{ 190,     190,      60,   -1510,      60}
7704 ,{ 850,     850,     720,   -2110,     720}
7705 ,{ 190,     190,      60,   -1510,      60}
7706 }
7707 ,{{ 700,     700,     570,   -760,     570}
7708 ,{ 650,     650,     520,   -1050,     520}
7709 ,{ 700,     700,     570,   -760,     570}
7710 ,{ 650,     650,     520,   -1050,     520}
7711 ,{ 100,     100,   -270,  -1840,   -270}
7712 }
7713 }
7714 ,{{{ 720,     570,     720,     570,     280}}
7715 ,{ 520,     370,     520,     370,      80}
7716 ,{ 570,     420,     570,     420,    130}
7717 ,{ 720,     570,     720,     570,    280}
7718 ,{ 460,     310,     460,     310,      20}
7719 }
7720 ,{{ 400,     220,     400,     220,    -40}
7721 ,{ 370,     220,     370,     220,    -60}
7722 ,{ 60,     -80,      60,     -80,   -370}
7723 ,{ 400,      10,     400,      10,    -40}
7724 ,{ 60,     -80,      60,     -80,   -370}
7725 }
7726 ,{{ 520,     370,     520,     370,      80}
7727 ,{ 520,     370,     520,     370,      80}
7728 ,{ 510,     360,     510,     360,      70}
7729 ,{ 520,     370,     520,     370,      80}
7730 ,{ 460,     310,     460,     310,      20}
7731 }
7732 ,{{ 720,     570,     720,     570,     280}
7733 ,{ 120,   -270,     120,   -270,   -320}
7734 ,{ 60,     -80,      60,     -80,   -370}
7735 ,{ 720,     570,     720,     570,    280}
7736 ,{ 60,     -80,      60,     -80,   -370}
7737 }
7738 ,{{ 570,     420,     570,     420,    130}
7739 ,{ 520,     370,     520,     370,      80}
7740 ,{ 570,     420,     570,     420,    130}
7741 ,{ 520,     370,     520,     370,      80}
7742 ,{ -270,   -420,   -270,   -420,   -710}
7743 }
7744 }
7745 ,{{{ 1350,   -460,     720,   1350,     720}}
7746 ,{ 1300,   -750,     520,   1300,     520}
7747 ,{ 1350,   -460,     570,   1350,     570}
7748 ,{ 1300,   -750,     720,   1300,     720}
7749 ,{ 1250,   -570,     460,   1250,     460}
7750 }
7751 ,{{ 1160,   -900,     370,   1160,     370}
7752 ,{ 1160,   -900,     370,   1160,     370}
7753 ,{ 850,   -1210,      60,     850,      60}
7754 ,{ 160,   -1110,     160,   -310,     160}
7755 ,{ 850,   -1210,      60,     850,      60}
7756 }
7757 ,{{ 1300,   -520,     520,   1300,     520}
7758 ,{ 1300,   -750,     520,   1300,     520}
7759 ,{ 1290,   -520,     510,   1290,     510}
7760 ,{ 1300,   -750,     520,   1300,     520}
7761 ,{ 1250,   -570,     460,   1250,     460}
7762 }
7763 ,{{ 850,   -1210,     720,     850,     720}
7764 ,{ -120,  -1400,   -120,   -590,   -120}
7765 ,{ 850,   -1210,      60,     850,      60}
7766 ,{ 720,   -1810,     720,   -1000,     720}
7767 ,{ 850,   -1210,      60,     850,      60}
7768 }
7769 ,{{ 1350,   -460,     570,   1350,     570}
```

```
7770 , { 1300, -750, 520, 1300, 520}
7771 , { 1350, -460, 570, 1350, 570}
7772 , { 1300, -750, 520, 1300, 520}
7773 , { -270, -1540, -270, -740, -270}
7774 }
7775 }
7776 , {{ { 590, 570, 590, 570, -320}
7777 , { 390, 370, 390, 370, -320}
7778 , { 440, 420, 440, 420, -360}
7779 , { 590, 570, 590, 570, -420}
7780 , { 330, 310, 330, 310, -470}
7781 }
7782 , {{ 270, 220, 270, 220, -320}
7783 , { 240, 220, 240, 220, -320}
7784 , { -60, -80, -60, -80, -870}
7785 , { 270, 10, 270, 10, -780}
7786 , { -60, -80, -60, -80, -870}
7787 }
7788 , {{ 390, 370, 390, 370, -420}
7789 , { 390, 370, 390, 370, -420}
7790 , { 380, 360, 380, 360, -420}
7791 , { 390, 370, 390, 370, -420}
7792 , { 330, 310, 330, 310, -470}
7793 }
7794 , {{ 590, 570, 590, 570, -870}
7795 , { -10, -270, -10, -270, -1060}
7796 , { -60, -80, -60, -80, -870}
7797 , { 590, 570, 590, 570, -1470}
7798 , { -60, -80, -60, -80, -870}
7799 }
7800 , {{ 440, 420, 440, 420, -360}
7801 , { 390, 370, 390, 370, -420}
7802 , { 440, 420, 440, 420, -360}
7803 , { 390, 370, 390, 370, -420}
7804 , { -400, -420, -400, -420, -1210}
7805 }
7806 }
7807 }
7808 , {{{ 1320, 850, 720, 1320, 720}
7809 , { 1320, 670, 540, 1320, 540}
7810 , { 870, 220, 90, 870, 90}
7811 , { 960, 850, 720, 960, 720}
7812 , { 870, 250, 90, 870, 90}
7813 }
7814 , {{ 1320, 670, 540, 1320, 540}
7815 , { 1320, 670, 540, 1320, 540}
7816 , { 870, 220, 90, 870, 90}
7817 , { -410, -520, -410, -800, -650}
7818 , { 870, 220, 90, 870, 90}
7819 }
7820 , {{ 960, 300, 170, 960, 170}
7821 , { 960, 300, 170, 960, 170}
7822 , { 650, 0, -130, 650, -130}
7823 , { 960, 300, 170, 960, 170}
7824 , { 650, 0, -130, 650, -130}
7825 }
7826 , {{ 870, 850, 720, 870, 720}
7827 , { 70, -40, 70, -320, -170}
7828 , { 870, 220, 90, 870, 90}
7829 , { 850, 850, 720, 570, 720}
7830 , { 870, 220, 90, 870, 90}
7831 }
7832 , {{ 960, 300, 170, 960, 170}
7833 , { 960, 300, 170, 960, 170}
7834 , { 340, -310, -440, 340, -440}
7835 , { 960, 300, 170, 960, 170}
7836 , { 250, 250, -110, -260, -110}
7837 }
7838 }
7839 , {{{ 850, 850, 720, -1030, 720}
7840 , { 670, 670, 540, -1030, 540}
7841 , { 220, 220, 90, -1460, 90}
7842 , { 850, 850, 720, -1400, 720}
7843 , { 250, 250, 90, -1460, 90}
7844 }
7845 , {{ 670, 670, 540, -1030, 540}
7846 , { 670, 670, 540, -1030, 540}
7847 , { 220, 220, 90, -1480, 90}
7848 , { -520, -520, -650, -2220, -650}
7849 , { 220, 220, 90, -1480, 90}
7850 }
7851 , {{ 300, 300, 170, -1400, 170}
7852 , { 300, 300, 170, -1400, 170}
7853 , { 0, 0, -130, -1460, -130}
7854 , { 300, 300, 170, -1400, 170}
7855 , { 0, 0, -130, -1460, -130}
7856 }
```

```
7857 , {{ 850,     850,     720, -1480,    720} }  
7858 , { -40,     -40,     -170, -1750,   -170} }  
7859 , { 220,     220,      90, -1480,     90} }  
7860 , { 850,     850,     720, -2110,    720} }  
7861 , { 220,     220,      90, -1480,     90} }  
7862 }  
7863 , {{ 300,     300,     170, -1400,    170} }  
7864 , { 300,     300,     170, -1400,    170} }  
7865 , { -310,    -310,    -440, -1770,   -440} }  
7866 , { 300,     300,     170, -1400,    170} }  
7867 , { 250,     250,    -110, -1690,   -110} }  
7868 }  
7869 }  
7870 , {{ { 720,     570,     720,     570,    280} }  
7871 , { 540,     390,     540,     390,   100} }  
7872 , { 90,     -60,      90,     -60,  -350} }  
7873 , { 720,     570,     720,     570,    280} }  
7874 , { 90,     -60,      90,     -60,  -350} }  
7875 }  
7876 , {{ { 540,     390,     540,     390,   100} }  
7877 , { 540,     390,     540,     390,   100} }  
7878 , { 90,     -60,      90,     -60,  -350} }  
7879 , { -410,    -800,    -410,    -800,  -850} }  
7880 , { 90,     -60,      90,     -60,  -350} }  
7881 }  
7882 , {{ { 170,      20,     170,      20,  -260} }  
7883 , { 170,      20,     170,      20,  -260} }  
7884 , { -130,    -280,    -130,    -280,  -570} }  
7885 , { 170,      20,     170,      20,  -260} }  
7886 , { -130,    -280,    -130,    -280,  -570} }  
7887 }  
7888 , {{ { 720,     570,     720,     570,    280} }  
7889 , { 70,     -320,     70,     -320,  -370} }  
7890 , { 90,     -60,      90,     -60,  -350} }  
7891 , { 720,     570,     720,     570,    280} }  
7892 , { 90,     -60,      90,     -60,  -350} }  
7893 }  
7894 , {{ { 170,      20,     170,      20,  -260} }  
7895 , { 170,      20,     170,      20,  -260} }  
7896 , { -440,    -590,    -440,    -590,  -880} }  
7897 , { 170,      20,     170,      20,  -260} }  
7898 , { -110,    -260,    -110,    -260,  -550} }  
7899 }  
7900 }  
7901 , {{ { 1320,    -730,     720,    1320,    720} }  
7902 , { 1320,    -730,     540,    1320,    540} }  
7903 , { 870,    -1160,     90,     870,     90} }  
7904 , { 960,    -1100,     720,     960,    720} }  
7905 , { 870,    -1160,     90,     870,     90} }  
7906 }  
7907 , {{ { 1320,    -730,     540,    1320,    540} }  
7908 , { 1320,    -730,     540,    1320,    540} }  
7909 , { 870,    -1180,     90,     870,     90} }  
7910 , { -650,   -1920,    -650,   -1120,   -650} }  
7911 , { 870,    -1180,     90,     870,     90} }  
7912 }  
7913 , {{ { 960,   -1100,     170,     960,    170} }  
7914 , { 960,   -1100,     170,     960,    170} }  
7915 , { 650,   -1160,    -130,     650,   -130} }  
7916 , { 960,   -1100,     170,     960,    170} }  
7917 , { 650,   -1160,    -130,     650,   -130} }  
7918 }  
7919 , {{ { 870,   -1180,     720,     870,    720} }  
7920 , { -170,  -1450,    -170,    -640,   -170} }  
7921 , { 870,   -1180,     90,     870,     90} }  
7922 , { 720,   -1810,     720,   -1000,    720} }  
7923 , { 870,   -1180,     90,     870,     90} }  
7924 }  
7925 , {{ { 960,   -1100,     170,     960,    170} }  
7926 , { 960,   -1100,     170,     960,    170} }  
7927 , { 340,   -1470,    -440,     340,   -440} }  
7928 , { 960,   -1100,     170,     960,    170} }  
7929 , { -110,  -1390,    -110,    -580,   -110} }  
7930 }  
7931 }  
7932 , {{ { 590,     570,     590,     570,   -160} }  
7933 , { 410,     390,     410,     390,  -160} }  
7934 , { -40,     -60,     -40,     -60,  -850} }  
7935 , { 590,     570,     590,     570,  -760} }  
7936 , { -40,     -60,     -40,     -60,  -850} }  
7937 }  
7938 , {{ { 410,     390,     410,     390,   -160} }  
7939 , { 410,     390,     410,     390,  -160} }  
7940 , { -40,     -60,     -40,     -60,  -850} }  
7941 , { -540,    -800,    -540,    -800, -1590} }  
7942 , { -40,     -60,     -40,     -60,  -850} }  
7943 }
```

```
7944 , {{ 40,    20,     40,     20,   -760} }  
7945 , { 40,    20,     40,     20,   -760} }  
7946 , { -260,  -280,   -260,   -280,  -1070} }  
7947 , { 40,    20,     40,     20,   -760} }  
7948 , { -260,  -280,   -260,   -280,  -1070} }  
7949 } }  
7950 , {{ 590,   570,   590,   570,  -850} }  
7951 , { -60,   -320,   -60,   -320,  -1110} }  
7952 , { -40,   -60,    -40,    -60,  -850} }  
7953 , { 590,   570,   590,   570,  -1470} }  
7954 , { -40,   -60,    -40,    -60,  -850} }  
7955 } }  
7956 , {{ 40,    20,     40,     20,   -760} }  
7957 , { 40,    20,     40,     20,   -760} }  
7958 , { -570,  -590,   -570,   -590,  -1380} }  
7959 , { 40,    20,     40,     20,   -760} }  
7960 , { -240,  -260,   -240,   -260,  -1050} }  
7961 } }  
7962 } }  
7963 } }  
7964 , {{{ 1010, 1010,   880,   730,   880} } }  
7965 , { 410,   -70,    40,    410,  -200} } }  
7966 , { 410,  -240,   -370,   410,  -370} } }  
7967 , { 1010, 1010,   880,   730,   880} } }  
7968 , { 410,     0,  -370,   410,  -370} } }  
7969 } }  
7970 , {{ 410,  -240,   -150,   410,  -370} } }  
7971 , { 230,  -420,   -550,   230,  -550} } }  
7972 , { 410,  -240,   -370,   410,  -370} } }  
7973 , { -150,  -260,   -150,   -540,  -390} } }  
7974 , { 410,  -240,   -370,   410,  -370} } }  
7975 } }  
7976 , {{ 410,  -240,   -370,   410,  -370} } }  
7977 , { 410,  -240,   -370,   410,  -370} } }  
7978 , { 410,  -240,   -370,   410,  -370} } }  
7979 , { 410,  -240,   -370,   410,  -370} } }  
7980 , { 410,  -240,   -370,   410,  -370} } }  
7981 } }  
7982 , {{ 1010, 1010,   880,   730,   880} } }  
7983 , { 40,    -70,    40,  -350,  -200} } }  
7984 , { 410,  -240,   -370,   410,  -370} } }  
7985 , { 1010, 1010,   880,   730,   880} } }  
7986 , { 410,  -240,   -370,   410,  -370} } }  
7987 } }  
7988 , {{ 410,     0,  -370,   410,  -370} } }  
7989 , { 410,  -240,   -370,   410,  -370} } }  
7990 , { 410,  -240,   -370,   410,  -370} } }  
7991 , { 410,  -240,   -370,   410,  -370} } }  
7992 , {     0,     0,  -370,  -520,  -370} } }  
7993 } }  
7994 } }  
7995 , {{{ 1010, 1010,   880,  -1710,   880} } } }  
7996 , { -70,   -70,   -200,  -1770,  -200} } }  
7997 , { -240,  -240,   -370,  -1710,  -370} } }  
7998 , { 1010, 1010,   880,  -1950,   880} } }  
7999 , {     0,     0,  -370,  -1710,  -370} } }  
8000 } }  
8001 , {{ -240,  -240,   -370,  -1950,  -370} } }  
8002 , { -420,  -420,   -550,  -2130,  -550} } }  
8003 , { -240,  -240,   -370,  -1950,  -370} } }  
8004 , { -260,  -260,   -390,  -1960,  -390} } }  
8005 , { -240,  -240,   -370,  -1950,  -370} } }  
8006 } }  
8007 , {{ -240,  -240,   -370,  -1710,  -370} } }  
8008 , { -240,  -240,   -370,  -1950,  -370} } }  
8009 , { -240,  -240,   -370,  -1710,  -370} } }  
8010 , { -240,  -240,   -370,  -1950,  -370} } }  
8011 , { -240,  -240,   -370,  -1710,  -370} } }  
8012 } }  
8013 , {{ 1010, 1010,   880,  -1770,   880} } } }  
8014 , { -70,   -70,   -200,  -1770,  -200} } }  
8015 , { -240,  -240,   -370,  -1950,  -370} } }  
8016 , { 1010, 1010,   880,  -1950,   880} } }  
8017 , { -240,  -240,   -370,  -1950,  -370} } }  
8018 } }  
8019 , {{     0,     0,  -370,  -1710,  -370} } } }  
8020 , { -240,  -240,   -370,  -1950,  -370} } }  
8021 , { -240,  -240,   -370,  -1710,  -370} } }  
8022 , { -240,  -240,   -370,  -1950,  -370} } }  
8023 , {     0,     0,  -370,  -1950,  -370} } }  
8024 } }  
8025 } }  
8026 , {{{ 880,    730,   880,    730,    440} } } }  
8027 , { 40,   -350,    40,  -350,  -400} } }  
8028 , { -370,  -520,   -370,  -520,  -810} } }  
8029 , { 880,    730,   880,    730,    440} } }  
8030 , { -370,  -520,   -370,  -520,  -810} } }
```

```

8031      }
8032 ,{{ -150, -520, -150, -520, -590}
8033 ,{ -550, -700, -550, -700, -990}
8034 ,{ -370, -520, -370, -520, -810}
8035 ,{ -150, -540, -150, -540, -590}
8036 ,{ -370, -520, -370, -520, -810}
8037      }
8038 ,{{ -370, -520, -370, -520, -810}
8039 ,{ -370, -520, -370, -520, -810}
8040 ,{ -370, -520, -370, -520, -810}
8041 ,{ -370, -520, -370, -520, -810}
8042 ,{ -370, -520, -370, -520, -810}
8043      }
8044 ,{{ 880, 730, 880, 730, 440}
8045 ,{ 40, -350, 40, -350, -400}
8046 ,{ -370, -520, -370, -520, -810}
8047 ,{ 880, 730, 880, 730, 440}
8048 ,{ -370, -520, -370, -520, -810}
8049      }
8050 ,{{ -370, -520, -370, -520, -810}
8051 ,{ -370, -520, -370, -520, -810}
8052 ,{ -370, -520, -370, -520, -810}
8053 ,{ -370, -520, -370, -520, -810}
8054 ,{ -370, -520, -370, -520, -810}
8055      }
8056      }
8057 ,{{{{ 880, -1410, 880, 410, 880}
8058 ,{ 410, -1470, -200, 410, -200}
8059 ,{ 410, -1410, -370, 410, -370}
8060 ,{ 880, -1650, 880, 410, 880}
8061 ,{ 410, -1410, -370, 410, -370}
8062      }
8063 ,{{ 410, -1650, -370, 410, -370}
8064 ,{ 230, -1830, -550, 230, -550}
8065 ,{ 410, -1650, -370, 410, -370}
8066 ,{ -390, -1660, -390, -860, -390}
8067 ,{ 410, -1650, -370, 410, -370}
8068      }
8069 ,{{ 410, -1410, -370, 410, -370}
8070 ,{ 410, -1650, -370, 410, -370}
8071 ,{ 410, -1410, -370, 410, -370}
8072 ,{ 410, -1650, -370, 410, -370}
8073 ,{ 410, -1410, -370, 410, -370}
8074      }
8075 ,{{{{ 880, -1470, 880, 410, 880}
8076 ,{ -200, -1470, -200, -670, -200}
8077 ,{ 410, -1650, -370, 410, -370}
8078 ,{ 880, -1650, 880, -840, 880}
8079 ,{ 410, -1650, -370, 410, -370}
8080      }
8081 ,{{{{ 410, -1410, -370, 410, -370}
8082 ,{ 410, -1650, -370, 410, -370}
8083 ,{ 410, -1410, -370, 410, -370}
8084 ,{ 410, -1650, -370, 410, -370}
8085 ,{ -370, -1650, -370, -840, -370}
8086      }
8087      }
8088 ,{{{{ 750, 730, 750, 730, -1140}
8089 ,{ -90, -350, -90, -350, -1140}
8090 ,{ -500, -520, -500, -520, -1310}
8091 ,{ 750, 730, 750, 730, -1310}
8092 ,{ -500, -520, -500, -520, -1310}
8093      }
8094 ,{{{{ -280, -520, -280, -520, -1250}
8095 ,{ -680, -700, -680, -700, -1250}
8096 ,{ -500, -520, -500, -520, -1310}
8097 ,{ -280, -540, -280, -540, -1330}
8098 ,{ -500, -520, -500, -520, -1310}
8099      }
8100 ,{{{{ -500, -520, -500, -520, -1310}
8101 ,{ -500, -520, -500, -520, -1310}
8102 ,{ -500, -520, -500, -520, -1310}
8103 ,{ -500, -520, -500, -520, -1310}
8104 ,{ -500, -520, -500, -520, -1310}
8105      }
8106 ,{{{{ 750, 730, 750, 730, -1140}
8107 ,{ -90, -350, -90, -350, -1140}
8108 ,{ -500, -520, -500, -520, -1310}
8109 ,{ 750, 730, 750, 730, -1310}
8110 ,{ -500, -520, -500, -520, -1310}
8111      }
8112 ,{{{{ -500, -520, -500, -520, -1310}
8113 ,{ -500, -520, -500, -520, -1310}
8114 ,{ -500, -520, -500, -520, -1310}
8115 ,{ -500, -520, -500, -520, -1310}
8116 ,{ -500, -520, -500, -520, -1310}
8117      }

```

```
8118    }
8119    }
8120 ,{{{{ 1560, 1560, 1430, 1470, 1430}
8121 ,{ 1470, 820, 690, 1470, 690}
8122 ,{ 960, 310, 180, 960, 180}
8123 ,{ 1560, 1560, 1430, 1280, 1430}
8124 ,{ 960, 550, 180, 960, 180}
8125 }
8126 ,{{ 1470, 820, 690, 1470, 690}
8127 ,{ 1470, 820, 690, 1470, 690}
8128 ,{ 960, 310, 180, 960, 180}
8129 ,{ 80, -30, 80, -310, -160}
8130 ,{ 960, 310, 180, 960, 180}
8131 }
8132 ,{{ 960, 310, 180, 960, 180}
8133 ,{ 960, 310, 180, 960, 180}
8134 ,{ 960, 310, 180, 960, 180}
8135 ,{ 960, 310, 180, 960, 180}
8136 ,{ 960, 310, 180, 960, 180}
8137 }
8138 ,{{ 1560, 1560, 1430, 1280, 1430}
8139 ,{ -90, -200, -90, -480, -330}
8140 ,{ 960, 310, 180, 960, 180}
8141 ,{ 1560, 1560, 1430, 1280, 1430}
8142 ,{ 960, 310, 180, 960, 180}
8143 }
8144 ,{{ 960, 550, 180, 960, 180}
8145 ,{ 960, 310, 180, 960, 180}
8146 ,{ 960, 310, 180, 960, 180}
8147 ,{ 960, 310, 180, 960, 180}
8148 ,{ 550, 550, 180, 30, 180}
8149 }
8150 }
8151 ,{{{{ 1560, 1560, 1430, -880, 1430}
8152 ,{ 820, 820, 690, -880, 690}
8153 ,{ 310, 310, 180, -1150, 180}
8154 ,{ 1560, 1560, 1430, -1390, 1430}
8155 ,{ 550, 550, 180, -1150, 180}
8156 }
8157 ,{{ 820, 820, 690, -880, 690}
8158 ,{ 820, 820, 690, -880, 690}
8159 ,{ 310, 310, 180, -1390, 180}
8160 ,{ -30, -30, -160, -1730, -160}
8161 ,{ 310, 310, 180, -1390, 180}
8162 }
8163 ,{{ 310, 310, 180, -1150, 180}
8164 ,{ 310, 310, 180, -1390, 180}
8165 ,{ 310, 310, 180, -1150, 180}
8166 ,{ 310, 310, 180, -1390, 180}
8167 ,{ 310, 310, 180, -1150, 180}
8168 }
8169 ,{{{{ 1560, 1560, 1430, -1390, 1430}
8170 ,{ -200, -200, -330, -1900, -330}
8171 ,{ 310, 310, 180, -1390, 180}
8172 ,{ 1560, 1560, 1430, -1390, 1430}
8173 ,{ 310, 310, 180, -1390, 180}
8174 }
8175 ,{{ 550, 550, 180, -1150, 180}
8176 ,{ 310, 310, 180, -1390, 180}
8177 ,{ 310, 310, 180, -1150, 180}
8178 ,{ 310, 310, 180, -1390, 180}
8179 ,{ 550, 550, 180, -1390, 180}
8180 }
8181 }
8182 ,{{{{ 1430, 1280, 1430, 1280, 990}
8183 ,{ 690, 540, 690, 540, 250}
8184 ,{ 180, 30, 180, 30, -260}
8185 ,{ 1430, 1280, 1430, 1280, 990}
8186 ,{ 180, 30, 180, 30, -260}
8187 }
8188 ,{{ 690, 540, 690, 540, 250}
8189 ,{ 690, 540, 690, 540, 250}
8190 ,{ 180, 30, 180, 30, -260}
8191 ,{ 80, -310, 80, -310, -360}
8192 ,{ 180, 30, 180, 30, -260}
8193 }
8194 ,{{ 180, 30, 180, 30, -260}
8195 ,{ 180, 30, 180, 30, -260}
8196 ,{ 180, 30, 180, 30, -260}
8197 ,{ 180, 30, 180, 30, -260}
8198 ,{ 180, 30, 180, 30, -260}
8199 }
8200 ,{{{{ 1430, 1280, 1430, 1280, 990}
8201 ,{ -90, -480, -90, -480, -530}
8202 ,{ 180, 30, 180, 30, -260}
8203 ,{ 1430, 1280, 1430, 1280, 990}
8204 ,{ 180, 30, 180, 30, -260}
```

```
8205      }
8206 ,{{ 180,    30,    180,    30,   -260}
8207 ,{ 180,    30,    180,    30,   -260}
8208 ,{ 180,    30,    180,    30,   -260}
8209 ,{ 180,    30,    180,    30,   -260}
8210 ,{ 180,    30,    180,    30,   -260}
8211     }
8212   }
8213 ,{{ 1470,  -580,  1430,  1470,  1430}
8214 ,{ 1470,  -580,   690,  1470,   690}
8215 ,{ 960,  -850,   180,   960,   180}
8216 ,{ 1430, -1090,  1430,   960,  1430}
8217 ,{ 960,  -850,   180,   960,   180}
8218   }
8219 ,{{ 1470,  -580,   690,  1470,   690}
8220 ,{ 1470,  -580,   690,  1470,   690}
8221 ,{ 960,  -1090,   180,   960,   180}
8222 ,{ -160,  -1430,  -160,  -630,  -160}
8223 ,{ 960,  -1090,   180,   960,   180}
8224   }
8225 ,{{ 960,  -850,   180,   960,   180}
8226 ,{ 960,  -1090,   180,   960,   180}
8227 ,{ 960,  -850,   180,   960,   180}
8228 ,{ 960,  -1090,   180,   960,   180}
8229 ,{ 960,  -850,   180,   960,   180}
8230   }
8231 ,{{ 1430, -1090,  1430,   960,  1430}
8232 ,{ -330,  -1600,  -330,  -800,  -330}
8233 ,{ 960,  -1090,   180,   960,   180}
8234 ,{ 1430, -1090,  1430,  -290,  1430}
8235 ,{ 960,  -1090,   180,   960,   180}
8236   }
8237 ,{{ 960,  -850,   180,   960,   180}
8238 ,{ 960,  -1090,   180,   960,   180}
8239 ,{ 960,  -850,   180,   960,   180}
8240 ,{ 960,  -1090,   180,   960,   180}
8241 ,{ 180,  -1090,   180,  -290,   180}
8242   }
8243   }
8244 ,{{ 1300,  1280,  1300,  1280,   -10}
8245 ,{ 560,    540,    560,    540,   -10}
8246 ,{ 50,     30,     50,     30,  -760}
8247 ,{ 1300,  1280,  1300,  1280,  -760}
8248 ,{ 50,     30,     50,     30,  -760}
8249   }
8250 ,{{ 560,    540,    560,    540,   -10}
8251 ,{ 560,    540,    560,    540,   -10}
8252 ,{ 50,     30,     50,     30,  -760}
8253 ,{ -50,   -310,   -50,   -310, -1100}
8254 ,{ 50,     30,     50,     30,  -760}
8255   }
8256 ,{{ 50,     30,     50,     30,  -760}
8257 ,{ 50,     30,     50,     30,  -760}
8258 ,{ 50,     30,     50,     30,  -760}
8259 ,{ 50,     30,     50,     30,  -760}
8260 ,{ 50,     30,     50,     30,  -760}
8261   }
8262 ,{{ 1300,  1280,  1300,  1280,  -760}
8263 ,{ -220,  -480,  -220,  -480, -1270}
8264 ,{ 50,     30,     50,     30,  -760}
8265 ,{ 1300,  1280,  1300,  1280,  -760}
8266 ,{ 50,     30,     50,     30,  -760}
8267   }
8268 ,{{ 50,     30,     50,     30,  -760}
8269 ,{ 50,     30,     50,     30,  -760}
8270 ,{ 50,     30,     50,     30,  -760}
8271 ,{ 50,     30,     50,     30,  -760}
8272 ,{ 50,     30,     50,     30,  -760}
8273   }
8274   }
8275   }
8276 ,{{{ 2050,  1930,  1800,  2050,  1800}
8277 ,{ 2050,  1400,  1270,  2050,  1270}
8278 ,{ 1750,  1100,   970,  1750,   970}
8279 ,{ 1930,  1930,  1800,  1760,  1800}
8280 ,{ 1750,  1100,   970,  1750,   970}
8281   }
8282 ,{{ 2050,  1400,  1270,  2050,  1270}
8283 ,{ 2050,  1400,  1270,  2050,  1270}
8284 ,{ 1740,  1090,   960,  1740,   960}
8285 ,{ 130,    10,     130,   -260,  -110}
8286 ,{ 1740,  1090,   960,  1740,   960}
8287   }
8288 ,{{ 1760,  1110,   980,  1760,   980}
8289 ,{ 1760,  1110,   980,  1760,   980}
8290 ,{ 1750,  1100,   970,  1750,   970}
8291 ,{ 1760,  1110,   980,  1760,   980}
```

```
8292 , { 1750, 1100, 970, 1750, 970}
8293 }
8294 , {{ 1930, 1930, 1800, 1740, 1800}
8295 , { 300, 190, 300, -80, 60}
8296 , { 1740, 1090, 960, 1740, 960}
8297 , { 1930, 1930, 1800, 1650, 1800}
8298 , { 1740, 1090, 960, 1740, 960}
8299 }
8300 , {{ 1760, 1110, 980, 1760, 980}
8301 , { 1760, 1110, 980, 1760, 980}
8302 , { 1750, 1100, 970, 1750, 970}
8303 , { 1760, 1110, 980, 1760, 980}
8304 , { 360, 360, 0, -150, 0}
8305 }
8306 }
8307 , {{{ 1930, 1930, 1800, -300, 1800}
8308 , { 1400, 1400, 1270, -300, 1270}
8309 , { 1100, 1100, 970, -360, 970}
8310 , { 1930, 1930, 1800, -590, 1800}
8311 , { 1100, 1100, 970, -360, 970}
8312 }
8313 , {{ 1400, 1400, 1270, -300, 1270}
8314 , { 1400, 1400, 1270, -300, 1270}
8315 , { 1090, 1090, 960, -610, 960}
8316 , { 10, 10, -110, -1690, -110}
8317 , { 1090, 1090, 960, -610, 960}
8318 }
8319 , {{ 1110, 1110, 980, -360, 980}
8320 , { 1110, 1110, 980, -590, 980}
8321 , { 1100, 1100, 970, -360, 970}
8322 , { 1110, 1110, 980, -590, 980}
8323 , { 1100, 1100, 970, -360, 970}
8324 }
8325 , {{ 1930, 1930, 1800, -610, 1800}
8326 , { 190, 190, 60, -1510, 60}
8327 , { 1090, 1090, 960, -610, 960}
8328 , { 1930, 1930, 1800, -1020, 1800}
8329 , { 1090, 1090, 960, -610, 960}
8330 }
8331 , {{ 1110, 1110, 980, -360, 980}
8332 , { 1110, 1110, 980, -590, 980}
8333 , { 1100, 1100, 970, -360, 970}
8334 , { 1110, 1110, 980, -590, 980}
8335 , { 360, 360, 0, -1580, 0}
8336 }
8337 }
8338 , {{{ 1800, 1650, 1800, 1650, 1360}
8339 , { 1270, 1120, 1270, 1120, 830}
8340 , { 970, 820, 970, 820, 530}
8341 , { 1800, 1650, 1800, 1650, 1360}
8342 , { 970, 820, 970, 820, 530}
8343 }
8344 , {{ 1270, 1120, 1270, 1120, 830}
8345 , { 1270, 1120, 1270, 1120, 830}
8346 , { 960, 810, 960, 810, 520}
8347 , { 130, -260, 130, -260, -310}
8348 , { 960, 810, 960, 810, 520}
8349 }
8350 , {{ 980, 830, 980, 830, 540}
8351 , { 980, 830, 980, 830, 540}
8352 , { 970, 820, 970, 820, 530}
8353 , { 980, 830, 980, 830, 540}
8354 , { 970, 820, 970, 820, 530}
8355 }
8356 , {{ 1800, 1650, 1800, 1650, 1360}
8357 , { 300, -80, 300, -80, -130}
8358 , { 960, 810, 960, 810, 520}
8359 , { 1800, 1650, 1800, 1650, 1360}
8360 , { 960, 810, 960, 810, 520}
8361 }
8362 , {{ 980, 830, 980, 830, 540}
8363 , { 980, 830, 980, 830, 540}
8364 , { 970, 820, 970, 820, 530}
8365 , { 980, 830, 980, 830, 540}
8366 , { 0, -150, 0, -150, -440}
8367 }
8368 }
8369 , {{{ 2050, 0, 1800, 2050, 1800}
8370 , { 2050, 0, 1270, 2050, 1270}
8371 , { 1750, -60, 970, 1750, 970}
8372 , { 1800, -290, 1800, 1760, 1800}
8373 , { 1750, -60, 970, 1750, 970}
8374 }
8375 , {{ 2050, 0, 1270, 2050, 1270}
8376 , { 2050, 0, 1270, 2050, 1270}
8377 , { 1740, -310, 960, 1740, 960}
8378 , { -110, -1390, -110, -580, -110}
```

```
8379 , { 1740, -310, 960, 1740, 960}
8380 }
8381 , {{ 1760, -60, 980, 1760, 980}
8382 , { 1760, -290, 980, 1760, 980}
8383 , { 1750, -60, 970, 1750, 970}
8384 , { 1760, -290, 980, 1760, 980}
8385 , { 1750, -60, 970, 1750, 970}
8386 }
8387 , {{ 1800, -310, 1800, 1740, 1800}
8388 , { 60, -1210, 60, -400, 60}
8389 , { 1740, -310, 960, 1740, 960}
8390 , { 1800, -720, 1800, 80, 1800}
8391 , { 1740, -310, 960, 1740, 960}
8392 }
8393 , {{ 1760, -60, 980, 1760, 980}
8394 , { 1760, -290, 980, 1760, 980}
8395 , { 1750, -60, 970, 1750, 970}
8396 , { 1760, -290, 980, 1760, 980}
8397 , { 0, -1280, 0, -470, 0}
8398 }
8399 }
8400 , {{{ 1670, 1650, 1670, 1650, 570}
8401 , { 1140, 1120, 1140, 1120, 570}
8402 , { 840, 820, 840, 820, 30}
8403 , { 1670, 1650, 1670, 1650, 40}
8404 , { 840, 820, 840, 820, 30}
8405 }
8406 , {{ 1140, 1120, 1140, 1120, 570}
8407 , { 1140, 1120, 1140, 1120, 570}
8408 , { 830, 810, 830, 810, 20}
8409 , { 0, -260, 0, -260, -1050}
8410 , { 830, 810, 830, 810, 20}
8411 }
8412 , {{ 850, 830, 850, 830, 40}
8413 , { 850, 830, 850, 830, 40}
8414 , { 840, 820, 840, 820, 30}
8415 , { 850, 830, 850, 830, 40}
8416 , { 840, 820, 840, 820, 30}
8417 }
8418 , {{ 1670, 1650, 1670, 1650, 20}
8419 , { 180, -80, 180, -80, -870}
8420 , { 830, 810, 830, 810, 20}
8421 , { 1670, 1650, 1670, 1650, -380}
8422 , { 830, 810, 830, 810, 20}
8423 }
8424 , {{ 850, 830, 850, 830, 40}
8425 , { 850, 830, 850, 830, 40}
8426 , { 840, 820, 840, 820, 30}
8427 , { 850, 830, 850, 830, 40}
8428 , { -130, -150, -130, -150, -940}
8429 }
8430 }
8431 }
8432 , {{{{ 2120, 2120, 1990, 2120, 1990}
8433 , { 2120, 1470, 1340, 2120, 1340}
8434 , { 1990, 1340, 1210, 1990, 1210}
8435 , { 2120, 2120, 1990, 1990, 1990}
8436 , { 1860, 1210, 1080, 1860, 1080}
8437 }
8438 , {{ 2120, 1470, 1340, 2120, 1340}
8439 , { 2120, 1470, 1340, 2120, 1340}
8440 , { 1840, 1190, 1060, 1840, 1060}
8441 , { 180, 60, 180, -210, -60}
8442 , { 1840, 1190, 1060, 1840, 1060}
8443 }
8444 , {{ 1990, 1340, 1210, 1990, 1210}
8445 , { 1990, 1340, 1210, 1990, 1210}
8446 , { 1990, 1340, 1210, 1990, 1210}
8447 , { 1990, 1340, 1210, 1990, 1210}
8448 , { 1860, 1210, 1080, 1860, 1080}
8449 }
8450 , {{ 2120, 2120, 1990, 1840, 1990}
8451 , { -120, -230, -120, -510, -360}
8452 , { 1840, 1190, 1060, 1840, 1060}
8453 , { 2120, 2120, 1990, 1840, 1990}
8454 , { 1840, 1190, 1060, 1840, 1060}
8455 }
8456 , {{ 1990, 1340, 1210, 1990, 1210}
8457 , { 1990, 1340, 1210, 1990, 1210}
8458 , { 1550, 900, 770, 1550, 770}
8459 , { 1990, 1340, 1210, 1990, 1210}
8460 , { 640, 640, 270, 120, 270}
8461 }
8462 }
8463 , {{{ 2120, 2120, 1990, -120, 1990}
8464 , { 1470, 1470, 1340, -230, 1340}
8465 , { 1340, 1340, 1210, -120, 1210}
```

```
8466 , { 2120, 2120, 1990, -360, 1990}
8467 , { 1210, 1210, 1080, -250, 1080}
8468 }
8469 , { { 1470, 1470, 1340, -230, 1340}
8470 , { 1470, 1470, 1340, -230, 1340}
8471 , { 1190, 1190, 1060, -510, 1060}
8472 , { 60, 60, -60, -1640, -60}
8473 , { 1190, 1190, 1060, -510, 1060}
8474 }
8475 , { { 1340, 1340, 1210, -120, 1210}
8476 , { 1340, 1340, 1210, -360, 1210}
8477 , { 1340, 1340, 1210, -120, 1210}
8478 , { 1340, 1340, 1210, -360, 1210}
8479 , { 1210, 1210, 1080, -250, 1080}
8480 }
8481 , { { 2120, 2120, 1990, -510, 1990}
8482 , { -230, -230, -360, -1940, -360}
8483 , { 1190, 1190, 1060, -510, 1060}
8484 , { 2120, 2120, 1990, -830, 1990}
8485 , { 1190, 1190, 1060, -510, 1060}
8486 }
8487 , { { 1340, 1340, 1210, -360, 1210}
8488 , { 1340, 1340, 1210, -360, 1210}
8489 , { 900, 900, 770, -560, 770}
8490 , { 1340, 1340, 1210, -360, 1210}
8491 , { 640, 640, 270, -1300, 270}
8492 }
8493 }
8494 , { { { 1990, 1840, 1990, 1840, 1550}
8495 , { 1340, 1190, 1340, 1190, 900}
8496 , { 1210, 1060, 1210, 1060, 770}
8497 , { 1990, 1840, 1990, 1840, 1550}
8498 , { 1080, 930, 1080, 930, 640}
8499 }
8500 , { { 1340, 1190, 1340, 1190, 900}
8501 , { 1340, 1190, 1340, 1190, 900}
8502 , { 1060, 910, 1060, 910, 620}
8503 , { 180, -210, 180, -210, -260}
8504 , { 1060, 910, 1060, 910, 620}
8505 }
8506 , { { 1210, 1060, 1210, 1060, 770}
8507 , { 1210, 1060, 1210, 1060, 770}
8508 , { 1210, 1060, 1210, 1060, 770}
8509 , { 1210, 1060, 1210, 1060, 770}
8510 , { 1080, 930, 1080, 930, 640}
8511 }
8512 , { { 1990, 1840, 1990, 1840, 1550}
8513 , { -120, -510, -120, -510, -560}
8514 , { 1060, 910, 1060, 910, 620}
8515 , { 1990, 1840, 1990, 1840, 1550}
8516 , { 1060, 910, 1060, 910, 620}
8517 }
8518 , { { 1210, 1060, 1210, 1060, 770}
8519 , { 1210, 1060, 1210, 1060, 770}
8520 , { 770, 620, 770, 620, 330}
8521 , { 1210, 1060, 1210, 1060, 770}
8522 , { 270, 120, 270, 120, -170}
8523 }
8524 }
8525 , { { { 2120, 180, 1990, 2120, 1990}
8526 , { 2120, 60, 1340, 2120, 1340}
8527 , { 1990, 180, 1210, 1990, 1210}
8528 , { 1990, -60, 1990, 1990, 1990}
8529 , { 1860, 50, 1080, 1860, 1080}
8530 }
8531 , { { 2120, 60, 1340, 2120, 1340}
8532 , { 2120, 60, 1340, 2120, 1340}
8533 , { 1840, -210, 1060, 1840, 1060}
8534 , { -60, -1340, -60, -530, -60}
8535 , { 1840, -210, 1060, 1840, 1060}
8536 }
8537 , { { 1990, 180, 1210, 1990, 1210}
8538 , { 1990, -60, 1210, 1990, 1210}
8539 , { 1990, 180, 1210, 1990, 1210}
8540 , { 1990, -60, 1210, 1990, 1210}
8541 , { 1860, 50, 1080, 1860, 1080}
8542 }
8543 , { { 1990, -210, 1990, 1840, 1990}
8544 , { -360, -1640, -360, -830, -360}
8545 , { 1840, -210, 1060, 1840, 1060}
8546 , { 1990, -530, 1990, 270, 1990}
8547 , { 1840, -210, 1060, 1840, 1060}
8548 }
8549 , { { 1990, -60, 1210, 1990, 1210}
8550 , { 1990, -60, 1210, 1990, 1210}
8551 , { 1550, -260, 770, 1550, 770}
8552 , { 1990, -60, 1210, 1990, 1210}
```

```
8553 , { 270, -1000,   270,  -200,   270}
8554 }
8555 }
8556 ,{{ { 1860, 1840, 1860, 1840,  640}
8557 ,{ 1210, 1190, 1210, 1190,  640}
8558 ,{ 1080, 1060, 1080, 1060, 270}
8559 ,{ 1860, 1840, 1860, 1840, 270}
8560 ,{ 950, 930, 950, 930, 140}
8561 }
8562 ,{{ 1210, 1190, 1210, 1190, 640}
8563 ,{ 1210, 1190, 1210, 1190, 640}
8564 ,{ 930, 910, 930, 910, 120}
8565 ,{ 50, -210, 50, -210, -1000}
8566 ,{ 930, 910, 930, 910, 120}
8567 }
8568 ,{{ 1080, 1060, 1080, 1060, 270}
8569 ,{ 1080, 1060, 1080, 1060, 270}
8570 ,{ 1080, 1060, 1080, 1060, 270}
8571 ,{ 1080, 1060, 1080, 1060, 270}
8572 ,{ 950, 930, 950, 930, 140}
8573 }
8574 ,{{ 1860, 1840, 1860, 1840, 120}
8575 ,{ -250, -510, -250, -510, -1300}
8576 ,{ 930, 910, 930, 910, 120}
8577 ,{ 1860, 1840, 1860, 1840, -200}
8578 ,{ 930, 910, 930, 910, 120}
8579 }
8580 ,{{ 1080, 1060, 1080, 1060, 270}
8581 ,{ 1080, 1060, 1080, 1060, 270}
8582 ,{ 640, 620, 640, 620, -170}
8583 ,{ 1080, 1060, 1080, 1060, 270}
8584 ,{ 140, 120, 140, 120, -670}
8585 }
8586 }
8587 }
8588 ,{{{ { 2120, 2120, 1990, 2120, 1990}
8589 ,{ 2120, 1470, 1340, 2120, 1340}
8590 ,{ 1990, 1340, 1210, 1990, 1210}
8591 ,{ 2120, 2120, 1990, 1990, 1990}
8592 ,{ 1860, 1210, 1080, 1860, 1080}
8593 }
8594 ,{{ 2120, 1470, 1340, 2120, 1340}
8595 ,{ 2120, 1470, 1340, 2120, 1340}
8596 ,{ 1840, 1190, 1060, 1840, 1060}
8597 ,{ 400, 290, 400, 10, 160}
8598 ,{ 1840, 1190, 1060, 1840, 1060}
8599 }
8600 ,{{ 1990, 1340, 1210, 1990, 1210}
8601 ,{ 1990, 1340, 1210, 1990, 1210}
8602 ,{ 1990, 1340, 1210, 1990, 1210}
8603 ,{ 1990, 1340, 1210, 1990, 1210}
8604 ,{ 1860, 1210, 1080, 1860, 1080}
8605 }
8606 ,{{ 2120, 2120, 1990, 1840, 1990}
8607 ,{ 300, 190, 300, -80, 60}
8608 ,{ 1840, 1190, 1060, 1840, 1060}
8609 ,{ 2120, 2120, 1990, 1840, 1990}
8610 ,{ 1840, 1190, 1060, 1840, 1060}
8611 }
8612 ,{{ 1990, 1340, 1210, 1990, 1210}
8613 ,{ 1990, 1340, 1210, 1990, 1210}
8614 ,{ 1750, 1100, 970, 1750, 970}
8615 ,{ 1990, 1340, 1210, 1990, 1210}
8616 ,{ 640, 640, 270, 120, 270}
8617 }
8618 }
8619 ,{{{ { 2120, 2120, 1990, -120, 1990}
8620 ,{ 1470, 1470, 1340, -230, 1340}
8621 ,{ 1340, 1340, 1210, -120, 1210}
8622 ,{ 2120, 2120, 1990, -360, 1990}
8623 ,{ 1210, 1210, 1080, -250, 1080}
8624 }
8625 ,{{ 1470, 1470, 1340, -230, 1340}
8626 ,{ 1470, 1470, 1340, -230, 1340}
8627 ,{ 1190, 1190, 1060, -510, 1060}
8628 ,{ 290, 290, 160, -1410, 160}
8629 ,{ 1190, 1190, 1060, -510, 1060}
8630 }
8631 ,{{ 1340, 1340, 1210, -120, 1210}
8632 ,{ 1340, 1340, 1210, -360, 1210}
8633 ,{ 1340, 1340, 1210, -120, 1210}
8634 ,{ 1340, 1340, 1210, -360, 1210}
8635 ,{ 1210, 1210, 1080, -250, 1080}
8636 }
8637 ,{{ 2120, 2120, 1990, -510, 1990}
8638 ,{ 190, 190, 60, -1510, 60}
8639 ,{ 1190, 1190, 1060, -510, 1060}
```

```
8640 , { 2120, 2120, 1990, -830, 1990}
8641 , { 1190, 1190, 1060, -510, 1060}
8642 }
8643 , {{ 1340, 1340, 1210, -360, 1210}
8644 , { 1340, 1340, 1210, -360, 1210}
8645 , { 1100, 1100, 970, -360, 970}
8646 , { 1340, 1340, 1210, -360, 1210}
8647 , { 640, 640, 270, -1300, 270}
8648 }
8649 }
8650 , {{{ 1990, 1840, 1990, 1840, 1550}
8651 , { 1340, 1190, 1340, 1190, 900}
8652 , { 1210, 1060, 1210, 1060, 770}
8653 , { 1990, 1840, 1990, 1840, 1550}
8654 , { 1080, 930, 1080, 930, 640}
8655 }
8656 , {{ 1340, 1190, 1340, 1190, 900}
8657 , { 1340, 1190, 1340, 1190, 900}
8658 , { 1060, 910, 1060, 910, 620}
8659 , { 400, 10, 400, 10, -40}
8660 , { 1060, 910, 1060, 910, 620}
8661 }
8662 , {{ 1210, 1060, 1210, 1060, 770}
8663 , { 1210, 1060, 1210, 1060, 770}
8664 , { 1210, 1060, 1210, 1060, 770}
8665 , { 1210, 1060, 1210, 1060, 770}
8666 , { 1080, 930, 1080, 930, 640}
8667 }
8668 , {{ 1990, 1840, 1990, 1840, 1550}
8669 , { 300, -80, 300, -80, -130}
8670 , { 1060, 910, 1060, 910, 620}
8671 , { 1990, 1840, 1990, 1840, 1550}
8672 , { 1060, 910, 1060, 910, 620}
8673 }
8674 , {{ 1210, 1060, 1210, 1060, 770}
8675 , { 1210, 1060, 1210, 1060, 770}
8676 , { 970, 820, 970, 820, 530}
8677 , { 1210, 1060, 1210, 1060, 770}
8678 , { 270, 120, 270, 120, -170}
8679 }
8680 }
8681 , {{{ 2120, 180, 1990, 2120, 1990}
8682 , { 2120, 60, 1340, 2120, 1340}
8683 , { 1990, 180, 1210, 1990, 1210}
8684 , { 1990, -60, 1990, 1990, 1990}
8685 , { 1860, 50, 1080, 1860, 1080}
8686 }
8687 , {{ 2120, 60, 1340, 2120, 1340}
8688 , { 2120, 60, 1340, 2120, 1340}
8689 , { 1840, -210, 1060, 1840, 1060}
8690 , { 160, -1110, 160, -310, 160}
8691 , { 1840, -210, 1060, 1840, 1060}
8692 }
8693 , {{ 1990, 180, 1210, 1990, 1210}
8694 , { 1990, -60, 1210, 1990, 1210}
8695 , { 1990, 180, 1210, 1990, 1210}
8696 , { 1990, -60, 1210, 1990, 1210}
8697 , { 1860, 50, 1080, 1860, 1080}
8698 }
8699 , {{ 1990, -210, 1990, 1840, 1990}
8700 , { 60, -1210, 60, -400, 60}
8701 , { 1840, -210, 1060, 1840, 1060}
8702 , { 1990, -530, 1990, 270, 1990}
8703 , { 1840, -210, 1060, 1840, 1060}
8704 }
8705 , {{ 1990, -60, 1210, 1990, 1210}
8706 , { 1990, -60, 1210, 1990, 1210}
8707 , { 1750, -60, 970, 1750, 970}
8708 , { 1990, -60, 1210, 1990, 1210}
8709 , { 270, -1000, 270, -200, 270}
8710 }
8711 }
8712 , {{{ 1860, 1840, 1860, 1840, 640}
8713 , { 1210, 1190, 1210, 1190, 640}
8714 , { 1080, 1060, 1080, 1060, 270}
8715 , { 1860, 1840, 1860, 1840, 270}
8716 , { 950, 930, 950, 930, 140}
8717 }
8718 , {{ 1210, 1190, 1210, 1190, 640}
8719 , { 1210, 1190, 1210, 1190, 640}
8720 , { 930, 910, 930, 910, 120}
8721 , { 270, 10, 270, 10, -780}
8722 , { 930, 910, 930, 910, 120}
8723 }
8724 , {{ 1080, 1060, 1080, 1060, 270}
8725 , { 1080, 1060, 1080, 1060, 270}
8726 , { 1080, 1060, 1080, 1060, 270}
```

```
8727 , { 1080, 1060, 1080, 1060, 270}
8728 , { 950, 930, 950, 930, 140}
8729 }
8730 , {{ 1860, 1840, 1860, 1840, 120}
8731 , { 180, -80, 180, -80, -870}
8732 , { 930, 910, 930, 910, 120}
8733 , { 1860, 1840, 1860, 1840, -200}
8734 , { 930, 910, 930, 910, 120}
8735 }
8736 , {{ 1080, 1060, 1080, 1060, 270}
8737 , { 1080, 1060, 1080, 1060, 270}
8738 , { 840, 820, 840, 820, 30}
8739 , { 1080, 1060, 1080, 1060, 270}
8740 , { 140, 120, 140, 120, -670}
8741 }
8742 }
8743 }
8744 }
8745 , {{{{ INF, INF, INF, INF, INF
8746 , { INF, INF, INF, INF, INF
8747 , { INF, INF, INF, INF, INF
8748 , { INF, INF, INF, INF, INF
8749 , { INF, INF, INF, INF, INF
8750 }
8751 , {{ INF, INF, INF, INF, INF
8752 , { INF, INF, INF, INF, INF
8753 , { INF, INF, INF, INF, INF
8754 , { INF, INF, INF, INF, INF
8755 , { INF, INF, INF, INF, INF
8756 }
8757 , {{ INF, INF, INF, INF, INF
8758 , { INF, INF, INF, INF, INF
8759 , { INF, INF, INF, INF, INF
8760 , { INF, INF, INF, INF, INF
8761 , { INF, INF, INF, INF, INF
8762 }
8763 , {{ INF, INF, INF, INF, INF
8764 , { INF, INF, INF, INF, INF
8765 , { INF, INF, INF, INF, INF
8766 , { INF, INF, INF, INF, INF
8767 , { INF, INF, INF, INF, INF
8768 }
8769 , {{ INF, INF, INF, INF, INF
8770 , { INF, INF, INF, INF, INF
8771 , { INF, INF, INF, INF, INF
8772 , { INF, INF, INF, INF, INF
8773 , { INF, INF, INF, INF, INF
8774 }
8775 }
8776 , {{{ INF, INF, INF, INF, INF
8777 , { INF, INF, INF, INF, INF
8778 , { INF, INF, INF, INF, INF
8779 , { INF, INF, INF, INF, INF
8780 , { INF, INF, INF, INF, INF
8781 }
8782 , {{ INF, INF, INF, INF, INF
8783 , { INF, INF, INF, INF, INF
8784 , { INF, INF, INF, INF, INF
8785 , { INF, INF, INF, INF, INF
8786 , { INF, INF, INF, INF, INF
8787 }
8788 , {{ INF, INF, INF, INF, INF
8789 , { INF, INF, INF, INF, INF
8790 , { INF, INF, INF, INF, INF
8791 , { INF, INF, INF, INF, INF
8792 , { INF, INF, INF, INF, INF
8793 }
8794 , {{ INF, INF, INF, INF, INF
8795 , { INF, INF, INF, INF, INF
8796 , { INF, INF, INF, INF, INF
8797 , { INF, INF, INF, INF, INF
8798 , { INF, INF, INF, INF, INF
8799 }
8800 , {{ INF, INF, INF, INF, INF
8801 , { INF, INF, INF, INF, INF
8802 , { INF, INF, INF, INF, INF
8803 , { INF, INF, INF, INF, INF
8804 , { INF, INF, INF, INF, INF
8805 }
8806 }
8807 , {{{ INF, INF, INF, INF, INF
8808 , { INF, INF, INF, INF, INF
8809 , { INF, INF, INF, INF, INF
8810 , { INF, INF, INF, INF, INF
8811 , { INF, INF, INF, INF, INF
8812 }
8813 , {{ INF, INF, INF, INF, INF
```

```
8814 , { INF, INF, INF, INF, INF }
8815 , { INF, INF, INF, INF, INF }
8816 , { INF, INF, INF, INF, INF }
8817 , { INF, INF, INF, INF, INF }
8818 }
8819 , {{ INF, INF, INF, INF, INF }
8820 , { INF, INF, INF, INF, INF }
8821 , { INF, INF, INF, INF, INF }
8822 , { INF, INF, INF, INF, INF }
8823 , { INF, INF, INF, INF, INF }
8824 }
8825 , {{ INF, INF, INF, INF, INF }
8826 , { INF, INF, INF, INF, INF }
8827 , { INF, INF, INF, INF, INF }
8828 , { INF, INF, INF, INF, INF }
8829 , { INF, INF, INF, INF, INF }
8830 }
8831 , {{ INF, INF, INF, INF, INF }
8832 , { INF, INF, INF, INF, INF }
8833 , { INF, INF, INF, INF, INF }
8834 , { INF, INF, INF, INF, INF }
8835 , { INF, INF, INF, INF, INF }
8836 }
8837 }
8838 , {{{ INF, INF, INF, INF, INF }
8839 , { INF, INF, INF, INF, INF }
8840 , { INF, INF, INF, INF, INF }
8841 , { INF, INF, INF, INF, INF }
8842 , { INF, INF, INF, INF, INF }
8843 }
8844 , {{ INF, INF, INF, INF, INF }
8845 , { INF, INF, INF, INF, INF }
8846 , { INF, INF, INF, INF, INF }
8847 , { INF, INF, INF, INF, INF }
8848 , { INF, INF, INF, INF, INF }
8849 }
8850 , {{ INF, INF, INF, INF, INF }
8851 , { INF, INF, INF, INF, INF }
8852 , { INF, INF, INF, INF, INF }
8853 , { INF, INF, INF, INF, INF }
8854 , { INF, INF, INF, INF, INF }
8855 }
8856 , {{ INF, INF, INF, INF, INF }
8857 , { INF, INF, INF, INF, INF }
8858 , { INF, INF, INF, INF, INF }
8859 , { INF, INF, INF, INF, INF }
8860 , { INF, INF, INF, INF, INF }
8861 }
8862 , {{ INF, INF, INF, INF, INF }
8863 , { INF, INF, INF, INF, INF }
8864 , { INF, INF, INF, INF, INF }
8865 , { INF, INF, INF, INF, INF }
8866 , { INF, INF, INF, INF, INF }
8867 }
8868 }
8869 , {{{ INF, INF, INF, INF, INF }
8870 , { INF, INF, INF, INF, INF }
8871 , { INF, INF, INF, INF, INF }
8872 , { INF, INF, INF, INF, INF }
8873 , { INF, INF, INF, INF, INF }
8874 }
8875 , {{ INF, INF, INF, INF, INF }
8876 , { INF, INF, INF, INF, INF }
8877 , { INF, INF, INF, INF, INF }
8878 , { INF, INF, INF, INF, INF }
8879 , { INF, INF, INF, INF, INF }
8880 }
8881 , {{ INF, INF, INF, INF, INF }
8882 , { INF, INF, INF, INF, INF }
8883 , { INF, INF, INF, INF, INF }
8884 , { INF, INF, INF, INF, INF }
8885 , { INF, INF, INF, INF, INF }
8886 }
8887 , {{ INF, INF, INF, INF, INF }
8888 , { INF, INF, INF, INF, INF }
8889 , { INF, INF, INF, INF, INF }
8890 , { INF, INF, INF, INF, INF }
8891 , { INF, INF, INF, INF, INF }
8892 }
8893 , {{ INF, INF, INF, INF, INF }
8894 , { INF, INF, INF, INF, INF }
8895 , { INF, INF, INF, INF, INF }
8896 , { INF, INF, INF, INF, INF }
8897 , { INF, INF, INF, INF, INF }
8898 }
8899 }
8900 }
```

```
8901 ,{{{{ 1350,    850,    720,   1350,    720}}
8902 ,{ 1300,    650,    540,   1300,    520}
8903 ,{ 1350,    700,    570,   1350,    570}
8904 ,{ 1300,    850,    720,   1300,    720}
8905 ,{ 1250,    590,    460,   1250,    460}
8906 }
8907 ,{{ 1160,    500,    400,   1160,    370}
8908 ,{ 1160,    500,    370,   1160,    370}
8909 ,{  850,    190,     60,    850,     60}
8910 ,{   400,    290,    400,     10,   170}
8911 ,{   850,    190,     60,    850,     60}
8912 }
8913 ,{{ 1300,    650,    520,   1300,    520}
8914 ,{ 1300,    650,    520,   1300,    520}
8915 ,{ 1290,    640,    510,   1290,    510}
8916 ,{ 1300,    650,    520,   1300,    520}
8917 ,{ 1250,    590,    460,   1250,    460}
8918 }
8919 ,{{  850,    850,    720,    850,    720}
8920 ,{   540,      0,    540,   -270,   -120}
8921 ,{   850,    190,     60,    850,     60}
8922 ,{   850,    850,    720,    570,    720}
8923 ,{   850,    190,     60,    850,     60}
8924 }
8925 ,{{ 1350,    700,    570,   1350,    570}
8926 ,{ 1300,    650,    520,   1300,    520}
8927 ,{ 1350,    700,    570,   1350,    570}
8928 ,{ 1300,    650,    520,   1300,    520}
8929 ,{   100,    100,   -270,   -230,   -270}
8930 }
8931 }
8932 ,{{{{  850,    850,    720,   -330,    720}}
8933 ,{   650,    650,    520,   -620,    520}
8934 ,{   700,    700,    570,   -330,    570}
8935 ,{   850,    850,    720,   -620,    720}
8936 ,{   590,    590,    460,   -440,    460}
8937 }
8938 ,{{  500,    500,    370,   -770,    370}
8939 ,{   500,    500,    370,   -770,    370}
8940 ,{   190,    190,     60,   -1070,     60}
8941 ,{   290,    290,    160,   -980,    160}
8942 ,{   190,    190,     60,   -1080,     60}
8943 }
8944 ,{{  650,    650,    520,   -390,    520}
8945 ,{   650,    650,    520,   -620,    520}
8946 ,{   640,    640,    510,   -390,    510}
8947 ,{   650,    650,    520,   -620,    520}
8948 ,{   590,    590,    460,   -440,    460}
8949 }
8950 ,{{  850,    850,    720,   -1080,    720}
8951 ,{   10,      0,     10,   -1270,   -120}
8952 ,{   190,    190,     60,   -1080,     60}
8953 ,{   850,    850,    720,   -1080,    720}
8954 ,{   190,    190,     60,   -1080,     60}
8955 }
8956 ,{{  700,    700,    570,   -330,    570}
8957 ,{   650,    650,    520,   -620,    520}
8958 ,{   700,    700,    570,   -330,    570}
8959 ,{   650,    650,    520,   -620,    520}
8960 ,{   100,    100,   -270,   -1300,   -270}
8961 }
8962 }
8963 ,{{{{  720,    570,    720,    570,    480}}
8964 ,{   540,    370,    540,    370,    280}
8965 ,{   570,    420,    570,    420,    340}
8966 ,{   720,    570,    720,    570,    480}
8967 ,{   460,    310,    460,    310,    230}
8968 }
8969 ,{{  400,    220,    400,    220,    170}
8970 ,{   370,    220,    370,    220,    140}
8971 ,{   60,    -80,     60,    -80,   -170}
8972 ,{   400,     10,    400,     10,   170}
8973 ,{   60,    -80,     60,    -80,   -170}
8974 }
8975 ,{{  520,    370,    520,    370,    280}
8976 ,{   520,    370,    520,    370,    280}
8977 ,{   510,    360,    510,    360,    280}
8978 ,{   520,    370,    520,    370,    280}
8979 ,{   460,    310,    460,    310,    230}
8980 }
8981 ,{{  720,    570,    720,    570,    480}
8982 ,{   540,   -100,    540,   -270,   -120}
8983 ,{   60,    -80,     60,    -80,   -170}
8984 ,{   720,    570,    720,    570,    480}
8985 ,{   60,    -80,     60,    -80,   -170}
8986 }
8987 ,{{  570,    420,    570,    420,    340}}
```

```
8988 , { 520, 370, 520, 370, 280}
8989 , { 570, 420, 570, 420, 340}
8990 , { 520, 370, 520, 370, 280}
8991 , { -270, -420, -270, -420, -500}
8992 }
8993 }
8994 , {{ { 1350, -230, 720, 1350, 720}
8995 , { 1300, -530, 520, 1300, 520}
8996 , { 1350, -230, 570, 1350, 570}
8997 , { 1300, -530, 720, 1300, 720}
8998 , { 1250, -340, 460, 1250, 460}
8999 }
9000 , {{ { 1160, -670, 370, 1160, 370}
9001 , { 1160, -670, 370, 1160, 370}
9002 , { 850, -980, 60, 850, 60}
9003 , { 160, -890, 160, -310, 160}
9004 , { 850, -980, 60, 850, 60}
9005 }
9006 , {{ { 1300, -290, 520, 1300, 520}
9007 , { 1300, -530, 520, 1300, 520}
9008 , { 1290, -290, 510, 1290, 510}
9009 , { 1300, -530, 520, 1300, 520}
9010 , { 1250, -340, 460, 1250, 460}
9011 }
9012 , {{ { 850, -980, 720, 850, 720}
9013 , { -120, -1170, -120, -590, -120}
9014 , { 850, -980, 60, 850, 60}
9015 , { 720, -1580, 720, -1000, 720}
9016 , { 850, -980, 60, 850, 60}
9017 }
9018 , {{ { 1350, -230, 570, 1350, 570}
9019 , { 1300, -530, 520, 1300, 520}
9020 , { 1350, -230, 570, 1350, 570}
9021 , { 1300, -530, 520, 1300, 520}
9022 , { -230, -1320, -270, -230, -270}
9023 }
9024 }
9025 , {{{ { 590, 570, 590, 570, -90}
9026 , { 390, 370, 390, 370, -90}
9027 , { 440, 420, 440, 420, -360}
9028 , { 590, 570, 590, 570, -420}
9029 , { 330, 310, 330, 310, -470}
9030 }
9031 , {{ { 270, 220, 270, 220, -320}
9032 , { 240, 220, 240, 220, -320}
9033 , { -60, -80, -60, -80, -830}
9034 , { 270, 10, 270, 10, -780}
9035 , { -60, -80, -60, -80, -870}
9036 }
9037 , {{ { 390, 370, 390, 370, -90}
9038 , { 390, 370, 390, 370, -90}
9039 , { 380, 360, 380, 360, -420}
9040 , { 390, 370, 390, 370, -420}
9041 , { 330, 310, 330, 310, -470}
9042 }
9043 , {{ { 590, 570, 590, 570, -810}
9044 , { -10, -270, -10, -270, -810}
9045 , { -60, -80, -60, -80, -870}
9046 , { 590, 570, 590, 570, -1470}
9047 , { -60, -80, -60, -80, -870}
9048 }
9049 , {{ { 440, 420, 440, 420, -360}
9050 , { 390, 370, 390, 370, -420}
9051 , { 440, 420, 440, 420, -360}
9052 , { 390, 370, 390, 370, -420}
9053 , { -400, -420, -400, -420, -1210}
9054 }
9055 }
9056 }
9057 , {{{ { 1320, 850, 720, 1320, 720}
9058 , { 1320, 670, 540, 1320, 540}
9059 , { 870, 220, 90, 870, 90}
9060 , { 960, 850, 720, 960, 720}
9061 , { 870, 250, 90, 870, 90}
9062 }
9063 , {{ { 1320, 670, 540, 1320, 540}
9064 , { 1320, 670, 540, 1320, 540}
9065 , { 870, 220, 90, 870, 90}
9066 , { -410, -520, -410, -800, -640}
9067 , { 870, 220, 90, 870, 90}
9068 }
9069 , {{ { 960, 300, 170, 960, 170}
9070 , { 960, 300, 170, 960, 170}
9071 , { 650, 0, -130, 650, -130}
9072 , { 960, 300, 170, 960, 170}
9073 , { 650, 0, -130, 650, -130}
9074 }
```

```
9075 , {{ 870, 850, 720, 870, 720} }
9076 , { 70, -40, 70, -320, -170}
9077 , { 870, 220, 90, 870, 90}
9078 , { 850, 850, 720, 570, 720}
9079 , { 870, 220, 90, 870, 90}
9080 }
9081 , {{ 960, 300, 170, 960, 170} }
9082 , { 960, 300, 170, 960, 170}
9083 , { 340, -310, -440, 340, -440}
9084 , { 960, 300, 170, 960, 170}
9085 , { 250, 250, -90, -260, -110}
9086 }
9087 }
9088 , {{ { 850, 850, 720, 540, 720} }
9089 , { 670, 670, 540, 10, 540}
9090 , { 540, 220, 90, 540, 90}
9091 , { 850, 850, 720, -970, 720}
9092 , { 250, 250, 90, -810, 90}
9093 }
9094 , {{ { 670, 670, 540, -100, 540} }
9095 , { 670, 670, 540, -600, 540}
9096 , { 220, 220, 90, -100, 90}
9097 , { -520, -520, -650, -1790, -650}
9098 , { 220, 220, 90, -1050, 90}
9099 }
9100 , {{ { 540, 300, 170, 540, 170} }
9101 , { 300, 300, 170, 10, 170}
9102 , { 540, 0, -130, 540, -130}
9103 , { 300, 300, 170, -970, 170}
9104 , { 0, 0, -130, -1030, -130}
9105 }
9106 , {{ { 850, 850, 720, -1050, 720} }
9107 , { -40, -40, -170, -1320, -170}
9108 , { 220, 220, 90, -1050, 90}
9109 , { 850, 850, 720, -1680, 720}
9110 , { 220, 220, 90, -1050, 90}
9111 }
9112 , {{ { 300, 300, 170, -810, 170} }
9113 , { 300, 300, 170, -970, 170}
9114 , { -310, -310, -440, -1340, -440}
9115 , { 300, 300, 170, -970, 170}
9116 , { 250, 250, -90, -810, -110}
9117 }
9118 }
9119 , {{ { { 720, 570, 720, 570, 480} }
9120 , { 540, 390, 540, 390, 300}
9121 , { 90, -60, 90, -60, -140}
9122 , { 720, 570, 720, 570, 480}
9123 , { 90, -60, 90, -60, -140}
9124 }
9125 , {{ { 540, 390, 540, 390, 300} }
9126 , { 540, 390, 540, 390, 300}
9127 , { 90, -60, 90, -60, -140}
9128 , { -410, -800, -410, -800, -640}
9129 , { 90, -60, 90, -60, -140}
9130 }
9131 , {{ { 170, 20, 170, 20, -60} }
9132 , { 170, 20, 170, 20, -60}
9133 , { -130, -280, -130, -280, -360}
9134 , { 170, 20, 170, 20, -60}
9135 , { -130, -280, -130, -280, -360}
9136 }
9137 , {{ { 720, 570, 720, 570, 480} }
9138 , { 70, -320, 70, -320, -170}
9139 , { 90, -60, 90, -60, -140}
9140 , { 720, 570, 720, 570, 480}
9141 , { 90, -60, 90, -60, -140}
9142 }
9143 , {{ { 170, 20, 170, 20, -60} }
9144 , { 170, 20, 170, 20, -60}
9145 , { -440, -590, -440, -590, -670}
9146 , { 170, 20, 170, 20, -60}
9147 , { -110, -260, -110, -260, -350}
9148 }
9149 }
9150 , {{ { { 1320, -350, 720, 1320, 720} }
9151 , { 1320, -730, 540, 1320, 540}
9152 , { 870, -350, 90, 870, 90}
9153 , { 960, -870, 720, 960, 720}
9154 , { 870, -940, 90, 870, 90}
9155 }
9156 , {{ { { 1320, -350, 540, 1320, 540} }
9157 , { 1320, -730, 540, 1320, 540}
9158 , { 870, -350, 90, 870, 90}
9159 , { -650, -1920, -650, -1120, -650}
9160 , { 870, -960, 90, 870, 90}
9161 }
```

```
9162 , {{ 960, -870, 170, 960, 170} }
9163 , { 960, -1100, 170, 960, 170}
9164 , { 650, -940, -130, 650, -130}
9165 , { 960, -870, 170, 960, 170}
9166 , { 650, -940, -130, 650, -130}
9167 }
9168 , {{ 870, -960, 720, 870, 720} }
9169 , { -170, -1450, -170, -640, -170}
9170 , { 870, -960, 90, 870, 90}
9171 , { 720, -1370, 720, -1000, 720}
9172 , { 870, -960, 90, 870, 90}
9173 }
9174 , {{ 960, -870, 170, 960, 170} }
9175 , { 960, -870, 170, 960, 170}
9176 , { 340, -1250, -440, 340, -440}
9177 , { 960, -870, 170, 960, 170}
9178 , { -110, -1360, -110, -580, -110}
9179 }
9180 }
9181 , {{{ 590, 570, 590, 570, -160} }
9182 , { 410, 390, 410, 390, -160}
9183 , { -40, -60, -40, -60, -850}
9184 , { 590, 570, 590, 570, -230}
9185 , { -40, -60, -40, -60, -850}
9186 }
9187 , {{ 410, 390, 410, 390, -160} }
9188 , { 410, 390, 410, 390, -160}
9189 , { -40, -60, -40, -60, -850}
9190 , { -540, -800, -540, -800, -1520}
9191 , { -40, -60, -40, -60, -850}
9192 }
9193 , {{ 40, 20, 40, 20, -400} }
9194 , { 40, 20, 40, 20, -400}
9195 , { -260, -280, -260, -280, -1070}
9196 , { 40, 20, 40, 20, -760}
9197 , { -260, -280, -260, -280, -1070}
9198 }
9199 , {{ 590, 570, 590, 570, -230} }
9200 , { -60, -320, -60, -320, -1110}
9201 , { -40, -60, -40, -60, -850}
9202 , { 590, 570, 590, 570, -230}
9203 , { -40, -60, -40, -60, -850}
9204 }
9205 , {{ 40, 20, 40, 20, -760} }
9206 , { 40, 20, 40, 20, -760}
9207 , { -570, -590, -570, -590, -1380}
9208 , { 40, 20, 40, 20, -760}
9209 , { -240, -260, -240, -260, -1050}
9210 }
9211 }
9212 }
9213 , {{{{ 1010, 1010, 880, 730, 880} }
9214 , { 410, -30, 40, 410, -190}
9215 , { 410, -240, -370, 410, -370}
9216 , { 1010, 1010, 880, 730, 880}
9217 , { 410, 0, -370, 410, -370}
9218 }
9219 , {{ 410, -70, -150, 410, -370} }
9220 , { 230, -70, -550, 230, -550}
9221 , { 410, -240, -370, 410, -370}
9222 , { -150, -260, -150, -540, -380}
9223 , { 410, -240, -370, 410, -370}
9224 }
9225 , {{ 410, -240, -370, 410, -370} }
9226 , { 410, -240, -370, 410, -370}
9227 , { 410, -240, -370, 410, -370}
9228 , { 410, -240, -370, 410, -370}
9229 , { 410, -240, -370, 410, -370}
9230 }
9231 , {{ 1010, 1010, 880, 730, 880} }
9232 , { 40, -30, 40, -350, -190}
9233 , { 410, -240, -370, 410, -370}
9234 , { 1010, 1010, 880, 730, 880}
9235 , { 410, -240, -370, 410, -370}
9236 }
9237 , {{ 410, 0, -370, 410, -370} }
9238 , { 410, -240, -370, 410, -370}
9239 , { 410, -240, -370, 410, -370}
9240 , { 410, -240, -370, 410, -370}
9241 , { 0, 0, -370, -520, -370}
9242 }
9243 }
9244 , {{{{ 1010, 1010, 880, -1280, 880} }
9245 , { -30, -30, -200, -1340, -200}
9246 , { -240, -240, -370, -1280, -370}
9247 , { 1010, 1010, 880, -1520, 880}
9248 , { 0, 0, -370, -1280, -370}
```

```
9249      }
9250 ,{{ -70, -70, -370, -1520, -370}
9251 ,{ -70, -70, -550, -1700, -550}
9252 ,{ -240, -240, -370, -1520, -370}
9253 ,{ -260, -260, -390, -1530, -390}
9254 ,{ -240, -240, -370, -1520, -370}
9255 }
9256 ,{{ -240, -240, -370, -1280, -370}
9257 ,{ -240, -240, -370, -1520, -370}
9258 ,{ -240, -240, -370, -1280, -370}
9259 ,{ -240, -240, -370, -1520, -370}
9260 ,{ -240, -240, -370, -1280, -370}
9261 }
9262 ,{{ 1010, 1010, 880, -1340, 880}
9263 ,{ -30, -30, -200, -1340, -200}
9264 ,{ -240, -240, -370, -1520, -370}
9265 ,{ 1010, 1010, 880, -1520, 880}
9266 ,{ -240, -240, -370, -1520, -370}
9267 }
9268 ,{{ 0, 0, -370, -1280, -370}
9269 ,{ -240, -240, -370, -1520, -370}
9270 ,{ -240, -240, -370, -1280, -370}
9271 ,{ -240, -240, -370, -1520, -370}
9272 ,{ 0, 0, -370, -1520, -370}
9273 }
9274 }
9275 ,{{{{ 880, 730, 880, 730, 640}
9276 ,{ 40, -350, 40, -350, -190}
9277 ,{ -370, -520, -370, -520, -610}
9278 ,{ 880, 730, 880, 730, 640}
9279 ,{ -370, -520, -370, -520, -610}
9280 }
9281 ,{{{{ -150, -520, -150, -520, -380}
9282 ,{ -550, -700, -550, -700, -790}
9283 ,{ -370, -520, -370, -520, -610}
9284 ,{ -150, -540, -150, -540, -380}
9285 ,{ -370, -520, -370, -520, -610}
9286 }
9287 ,{{{{ -370, -520, -370, -520, -610}
9288 ,{ -370, -520, -370, -520, -610}
9289 ,{ -370, -520, -370, -520, -610}
9290 ,{ -370, -520, -370, -520, -610}
9291 ,{ -370, -520, -370, -520, -610}
9292 }
9293 ,{{{{ 880, 730, 880, 730, 640}
9294 ,{ 40, -350, 40, -350, -190}
9295 ,{ -370, -520, -370, -520, -610}
9296 ,{ 880, 730, 880, 730, 640}
9297 ,{ -370, -520, -370, -520, -610}
9298 }
9299 ,{{{{ -370, -520, -370, -520, -610}
9300 ,{ -370, -520, -370, -520, -610}
9301 ,{ -370, -520, -370, -520, -610}
9302 ,{ -370, -520, -370, -520, -610}
9303 ,{ -370, -520, -370, -520, -610}
9304 }
9305 }
9306 ,{{{{ 880, -1180, 880, 410, 880}
9307 ,{ 410, -1250, -200, 410, -200}
9308 ,{ 410, -1180, -370, 410, -370}
9309 ,{ 880, -1420, 880, 410, 880}
9310 ,{ 410, -1180, -370, 410, -370}
9311 }
9312 ,{{{{ 410, -1420, -370, 410, -370}
9313 ,{ 230, -1600, -550, 230, -550}
9314 ,{ 410, -1420, -370, 410, -370}
9315 ,{ -390, -1440, -390, -860, -390}
9316 ,{ 410, -1420, -370, 410, -370}
9317 }
9318 ,{{{{ 410, -1180, -370, 410, -370}
9319 ,{ 410, -1420, -370, 410, -370}
9320 ,{ 410, -1180, -370, 410, -370}
9321 ,{ 410, -1420, -370, 410, -370}
9322 ,{ 410, -1180, -370, 410, -370}
9323 }
9324 ,{{{{ 880, -1250, 880, 410, 880}
9325 ,{ -200, -1250, -200, -670, -200}
9326 ,{ 410, -1420, -370, 410, -370}
9327 ,{ 880, -1420, 880, -840, 880}
9328 ,{ 410, -1420, -370, 410, -370}
9329 }
9330 ,{{{{ 410, -1180, -370, 410, -370}
9331 ,{ 410, -1420, -370, 410, -370}
9332 ,{ 410, -1180, -370, 410, -370}
9333 ,{ 410, -1420, -370, 410, -370}
9334 ,{ -370, -1420, -370, -840, -370}
9335 }
```

```
9336      }
9337 ,{{ { 750,    730,    750,    730, -1140}
9338 ,{ -90,   -350,   -90,   -350, -1140}
9339 ,{ -500,   -520,   -500,   -520, -1310}
9340 ,{ 750,    730,    750,    730, -1310}
9341 ,{ -500,   -520,   -500,   -520, -1310}
9342      }
9343 ,{{ -280,   -520,   -280,   -520, -1250}
9344 ,{ -680,   -700,   -680,   -700, -1250}
9345 ,{ -500,   -520,   -500,   -520, -1310}
9346 ,{ -280,   -540,   -280,   -540, -1330}
9347 ,{ -500,   -520,   -500,   -520, -1310}
9348      }
9349 ,{{ -500,   -520,   -500,   -520, -1310}
9350 ,{ -500,   -520,   -500,   -520, -1310}
9351 ,{ -500,   -520,   -500,   -520, -1310}
9352 ,{ -500,   -520,   -500,   -520, -1310}
9353 ,{ -500,   -520,   -500,   -520, -1310}
9354      }
9355 ,{{ { 750,    730,    750,    730, -1140}
9356 ,{ -90,   -350,   -90,   -350, -1140}
9357 ,{ -500,   -520,   -500,   -520, -1310}
9358 ,{ 750,    730,    750,    730, -1310}
9359 ,{ -500,   -520,   -500,   -520, -1310}
9360      }
9361 ,{{ { -500,   -520,   -500,   -520, -1310}
9362 ,{ -500,   -520,   -500,   -520, -1310}
9363 ,{ -500,   -520,   -500,   -520, -1310}
9364 ,{ -500,   -520,   -500,   -520, -1310}
9365 ,{ -500,   -520,   -500,   -520, -1310}
9366      }
9367      }
9368      }
9369 ,{{ {{ { 1560,   1560,   1430,   1470,  1430}
9370 ,{ 1470,   820,    690,   1470,  690}
9371 ,{ 960,    310,    180,    960,  180}
9372 ,{ 1560,   1560,   1430,   1280, 1430}
9373 ,{ 960,    550,    180,    960,  180}
9374      }
9375 ,{{ { 1470,   820,    690,   1470,  690}
9376 ,{ 1470,   820,    690,   1470,  690}
9377 ,{ 960,    310,    180,    960,  180}
9378 ,{ 80,     -30,     80,    -310, -150}
9379 ,{ 960,    310,    180,    960,  180}
9380      }
9381 ,{{ { 960,    310,    180,    960,  180}
9382 ,{ 960,    310,    180,    960,  180}
9383 ,{ 960,    310,    180,    960,  180}
9384 ,{ 960,    310,    180,    960,  180}
9385 ,{ 960,    310,    180,    960,  180}
9386      }
9387 ,{{ { 1560,   1560,   1430,   1280, 1430}
9388 ,{ -90,   -200,   -90,   -480, -320}
9389 ,{ 960,    310,    180,    960,  180}
9390 ,{ 1560,   1560,   1430,   1280, 1430}
9391 ,{ 960,    310,    180,    960,  180}
9392      }
9393 ,{{ { 960,    550,    180,    960,  180}
9394 ,{ 960,    310,    180,    960,  180}
9395 ,{ 960,    310,    180,    960,  180}
9396 ,{ 960,    310,    180,    960,  180}
9397 ,{ 550,    550,    180,     30,  180}
9398      }
9399      }
9400 ,{{ {{ { 1560,   1560,   1430,   -30,  1430}
9401 ,{ 820,    820,    690,   -30,  690}
9402 ,{ 310,    310,    180,   -720, 180}
9403 ,{ 1560,   1560,   1430,   -960, 1430}
9404 ,{ 550,    550,    180,   -720, 180}
9405      }
9406 ,{{ { 820,    820,    690,   -30,  690}
9407 ,{ 820,    820,    690,   -30,  690}
9408 ,{ 310,    310,    180,   -960, 180}
9409 ,{ -30,   -30,   -160,  -1300, -160}
9410 ,{ 310,    310,    180,   -960, 180}
9411      }
9412 ,{{ { 310,    310,    180,   -720, 180}
9413 ,{ 310,    310,    180,   -960, 180}
9414 ,{ 310,    310,    180,   -720, 180}
9415 ,{ 310,    310,    180,   -960, 180}
9416 ,{ 310,    310,    180,   -720, 180}
9417      }
9418 ,{{ { 1560,   1560,   1430,   -960, 1430}
9419 ,{ -200,  -200,  -330,  -1470, -330}
9420 ,{ 310,    310,    180,   -960, 180}
9421 ,{ 1560,   1560,   1430,   -960, 1430}
9422 ,{ 310,    310,    180,   -960, 180}
```

```

9423    }
9424 ,{{ 550,   550,   180,  -720,   180}
9425 ,{ 310,   310,   180,  -960,   180}
9426 ,{ 310,   310,   180,  -720,   180}
9427 ,{ 310,   310,   180,  -960,   180}
9428 ,{ 550,   550,   180,  -960,   180}
9429 }
9430 }
9431 ,{{ 1430,  1280,  1430,  1280,  1200}
9432 ,{ 690,   540,   690,   540,   450}
9433 ,{ 180,    30,   180,    30,  -50}
9434 ,{ 1430,  1280,  1430,  1280,  1200}
9435 ,{ 180,    30,   180,    30,  -50}
9436 }
9437 ,{{ 690,   540,   690,   540,   450}
9438 ,{ 690,   540,   690,   540,   450}
9439 ,{ 180,    30,   180,    30,  -50}
9440 ,{ 80,   -310,   80,   -310,  -150}
9441 ,{ 180,    30,   180,    30,  -50}
9442 }
9443 ,{{ 180,    30,   180,    30,  -50}
9444 ,{ 180,    30,   180,    30,  -50}
9445 ,{ 180,    30,   180,    30,  -50}
9446 ,{ 180,    30,   180,    30,  -50}
9447 ,{ 180,    30,   180,    30,  -50}
9448 }
9449 ,{{ 1430,  1280,  1430,  1280,  1200}
9450 ,{ -90,  -480,  -90,  -480,  -320}
9451 ,{ 180,    30,   180,    30,  -50}
9452 ,{ 1430,  1280,  1430,  1280,  1200}
9453 ,{ 180,    30,   180,    30,  -50}
9454 }
9455 ,{{ 180,    30,   180,    30,  -50}
9456 ,{ 180,    30,   180,    30,  -50}
9457 ,{ 180,    30,   180,    30,  -50}
9458 ,{ 180,    30,   180,    30,  -50}
9459 ,{ 180,    30,   180,    30,  -50}
9460 }
9461 }
9462 ,{{ 1470,  -360,  1430,  1470,  1430}
9463 ,{ 1470,  -360,   690,  1470,   690}
9464 ,{ 960,  -630,   180,   960,   180}
9465 ,{ 1430,  -870,  1430,   960,  1430}
9466 ,{ 960,  -630,   180,   960,   180}
9467 }
9468 ,{{ 1470,  -360,   690,  1470,   690}
9469 ,{ 1470,  -360,   690,  1470,   690}
9470 ,{ 960,  -870,   180,   960,   180}
9471 ,{ -160, -1210,  -160,  -630,  -160}
9472 ,{ 960,  -870,   180,   960,   180}
9473 }
9474 ,{{ 960,  -630,   180,   960,   180}
9475 ,{ 960,  -870,   180,   960,   180}
9476 ,{ 960,  -630,   180,   960,   180}
9477 ,{ 960,  -870,   180,   960,   180}
9478 ,{ 960,  -630,   180,   960,   180}
9479 }
9480 ,{{ 1430,  -870,  1430,   960,  1430}
9481 ,{ -330, -1380,  -330,  -800,  -330}
9482 ,{ 960,  -870,   180,   960,   180}
9483 ,{ 1430,  -870,  1430,  -290,  1430}
9484 ,{ 960,  -870,   180,   960,   180}
9485 }
9486 ,{{ 960,  -630,   180,   960,   180}
9487 ,{ 960,  -870,   180,   960,   180}
9488 ,{ 960,  -630,   180,   960,   180}
9489 ,{ 960,  -870,   180,   960,   180}
9490 ,{ 180,  -870,   180,  -290,   180}
9491 }
9492 }
9493 ,{{ 1300,  1280,  1300,  1280,   -10}
9494 ,{ 560,   540,   560,   540,  -10}
9495 ,{ 50,    30,    50,    30,  -760}
9496 ,{ 1300,  1280,  1300,  1280,  -760}
9497 ,{ 50,    30,    50,    30,  -760}
9498 }
9499 ,{{ 560,   540,   560,   540,  -10}
9500 ,{ 560,   540,   560,   540,  -10}
9501 ,{ 50,    30,    50,    30,  -760}
9502 ,{ -50,  -310,  -50,  -310, -1100}
9503 ,{ 50,    30,    50,    30,  -760}
9504 }
9505 ,{{ 50,    30,    50,    30,  -760}
9506 ,{ 50,    30,    50,    30,  -760}
9507 ,{ 50,    30,    50,    30,  -760}
9508 ,{ 50,    30,    50,    30,  -760}
9509 ,{ 50,    30,    50,    30,  -760}

```

```
9510      }
9511 ,{{ 1300, 1280, 1300, 1280, -760}
9512 ,{ -220, -480, -220, -480, -1270}
9513 ,{ 50, 30, 50, 30, -760}
9514 ,{ 1300, 1280, 1300, 1280, -760}
9515 ,{ 50, 30, 50, 30, -760}
9516 }
9517 ,{{ 50, 30, 50, 30, -760}
9518 ,{ 50, 30, 50, 30, -760}
9519 ,{ 50, 30, 50, 30, -760}
9520 ,{ 50, 30, 50, 30, -760}
9521 ,{ 50, 30, 50, 30, -760}
9522 }
9523 }
9524 }
9525 ,{{{{ 2050, 1930, 1800, 2050, 1800}
9526 ,{ 2050, 1400, 1270, 2050, 1270}
9527 ,{ 1750, 1100, 970, 1750, 970}
9528 ,{ 1930, 1930, 1800, 1760, 1800}
9529 ,{ 1750, 1100, 970, 1750, 970}
9530 }
9531 ,{{ 2050, 1400, 1270, 2050, 1270}
9532 ,{ 2050, 1400, 1270, 2050, 1270}
9533 ,{ 1740, 1090, 960, 1740, 960}
9534 ,{ 130, 10, 130, -260, -110}
9535 ,{ 1740, 1090, 960, 1740, 960}
9536 }
9537 ,{{ 1760, 1110, 980, 1760, 980}
9538 ,{ 1760, 1110, 980, 1760, 980}
9539 ,{ 1750, 1100, 970, 1750, 970}
9540 ,{ 1760, 1110, 980, 1760, 980}
9541 ,{ 1750, 1100, 970, 1750, 970}
9542 }
9543 ,{{ 1930, 1930, 1800, 1740, 1800}
9544 ,{ 300, 190, 300, -80, 70}
9545 ,{ 1740, 1090, 960, 1740, 960}
9546 ,{ 1930, 1930, 1800, 1650, 1800}
9547 ,{ 1740, 1090, 960, 1740, 960}
9548 }
9549 ,{{ 1760, 1110, 980, 1760, 980}
9550 ,{ 1760, 1110, 980, 1760, 980}
9551 ,{ 1750, 1100, 970, 1750, 970}
9552 ,{ 1760, 1110, 980, 1760, 980}
9553 ,{ 360, 360, 0, -150, 0}
9554 }
9555 }
9556 ,{{{{ 1930, 1930, 1800, 130, 1800}
9557 ,{ 1400, 1400, 1270, 130, 1270}
9558 ,{ 1100, 1100, 970, 70, 970}
9559 ,{ 1930, 1930, 1800, -160, 1800}
9560 ,{ 1100, 1100, 970, 70, 970}
9561 }
9562 ,{{ 1400, 1400, 1270, 130, 1270}
9563 ,{ 1400, 1400, 1270, 130, 1270}
9564 ,{ 1090, 1090, 960, -180, 960}
9565 ,{ 10, 10, -110, -1260, -110}
9566 ,{ 1090, 1090, 960, -180, 960}
9567 }
9568 ,{{ 1110, 1110, 980, 70, 980}
9569 ,{ 1110, 1110, 980, -160, 980}
9570 ,{ 1100, 1100, 970, 70, 970}
9571 ,{ 1110, 1110, 980, -160, 980}
9572 ,{ 1100, 1100, 970, 70, 970}
9573 }
9574 ,{{ 1930, 1930, 1800, -180, 1800}
9575 ,{ 190, 190, 60, -1080, 60}
9576 ,{ 1090, 1090, 960, -180, 960}
9577 ,{ 1930, 1930, 1800, -590, 1800}
9578 ,{ 1090, 1090, 960, -180, 960}
9579 }
9580 ,{{ 1110, 1110, 980, 70, 980}
9581 ,{ 1110, 1110, 980, -160, 980}
9582 ,{ 1100, 1100, 970, 70, 970}
9583 ,{ 1110, 1110, 980, -160, 980}
9584 ,{ 360, 360, 0, -1150, 0}
9585 }
9586 }
9587 ,{{{{ 1800, 1650, 1800, 1650, 1570}
9588 ,{ 1270, 1120, 1270, 1120, 1040}
9589 ,{ 970, 820, 970, 820, 740}
9590 ,{ 1800, 1650, 1800, 1650, 1570}
9591 ,{ 970, 820, 970, 820, 740}
9592 }
9593 ,{{ 1270, 1120, 1270, 1120, 1040}
9594 ,{ 1270, 1120, 1270, 1120, 1040}
9595 ,{ 960, 810, 960, 810, 730}
9596 ,{ 130, -260, 130, -260, -110}
```

```

9597 , { 960,   810,   960,   810,   730}
9598 }
9599 , {{ 980,   830,   980,   830,   740}
9600 , { 980,   830,   980,   830,   740}
9601 , { 970,   820,   970,   820,   740}
9602 , { 980,   830,   980,   830,   740}
9603 , { 970,   820,   970,   820,   740}
9604 }
9605 , {{ 1800,  1650,  1800,  1650,  1570}
9606 , { 300,   -80,    300,   -80,    70}
9607 , { 960,   810,   960,   810,   730}
9608 , { 1800,  1650,  1800,  1650,  1570}
9609 , { 960,   810,   960,   810,   730}
9610 }
9611 , {{ 980,   830,   980,   830,   740}
9612 , { 980,   830,   980,   830,   740}
9613 , { 970,   820,   970,   820,   740}
9614 , { 980,   830,   980,   830,   740}
9615 , { 0,   -150,    0,   -150,  -240}
9616 }
9617 }
9618 , {{ { 2050,  220,   1800,  2050,  1800}
9619 , { 2050,  220,   1270,  2050,  1270}
9620 , { 1750,  170,   970,   1750,  970}
9621 , { 1800,  -70,   1800,  1760,  1800}
9622 , { 1750,  170,   970,   1750,  970}
9623 }
9624 , {{ 2050,  220,   1270,  2050,  1270}
9625 , { 2050,  220,   1270,  2050,  1270}
9626 , { 1740,  -80,   960,   1740,  960}
9627 , { -110,  -1160,  -110,  -580,  -110}
9628 , { 1740,  -80,   960,   1740,  960}
9629 }
9630 , {{ 1760,  170,   980,   1760,  980}
9631 , { 1760,  -70,   980,   1760,  980}
9632 , { 1750,  170,   970,   1750,  970}
9633 , { 1760,  -70,   980,   1760,  980}
9634 , { 1750,  170,   970,   1750,  970}
9635 }
9636 , {{ { 1800,  -80,   1800,  1740,  1800}
9637 , { 60,   -980,    60,   -400,    60}
9638 , { 1740,  -80,   960,   1740,  960}
9639 , { 1800,  -490,   1800,   80,   1800}
9640 , { 1740,  -80,   960,   1740,  960}
9641 }
9642 , {{ { 1760,  170,   980,   1760,  980}
9643 , { 1760,  -70,   980,   1760,  980}
9644 , { 1750,  170,   970,   1750,  970}
9645 , { 1760,  -70,   980,   1760,  980}
9646 , { 0,   -1050,    0,   -470,    0}
9647 }
9648 }
9649 , {{ { 1670,  1650,  1670,  1650,  570}
9650 , { 1140,  1120,  1140,  1120,  570}
9651 , { 840,   820,   840,   820,   30}
9652 , { 1670,  1650,  1670,  1650,   40}
9653 , { 840,   820,   840,   820,   30}
9654 }
9655 , {{ { 1140,  1120,  1140,  1120,  570}
9656 , { 1140,  1120,  1140,  1120,  570}
9657 , { 830,   810,   830,   810,   20}
9658 , { 0,   -260,    0,   -260,  -1050}
9659 , { 830,   810,   830,   810,   20}
9660 }
9661 , {{ { 850,   830,   850,   830,   40}
9662 , { 850,   830,   850,   830,   40}
9663 , { 840,   820,   840,   820,   30}
9664 , { 850,   830,   850,   830,   40}
9665 , { 840,   820,   840,   820,   30}
9666 }
9667 , {{ { 1670,  1650,  1670,  1650,   20}
9668 , { 180,   -80,    180,   -80,  -870}
9669 , { 830,   810,   830,   810,   20}
9670 , { 1670,  1650,  1670,  1650,  -380}
9671 , { 830,   810,   830,   810,   20}
9672 }
9673 , {{ { 850,   830,   850,   830,   40}
9674 , { 850,   830,   850,   830,   40}
9675 , { 840,   820,   840,   820,   30}
9676 , { 850,   830,   850,   830,   40}
9677 , { -130,  -150,  -130,  -150,  -940}
9678 }
9679 }
9680 }
9681 , {{{ { 2120,  2120,  1990,  2120,  1990}
9682 , { 2120,  1470,  1340,  2120,  1340}
9683 , { 1990,  1340,  1210,  1990,  1210}

```

```
9684 , { 2120, 2120, 1990, 1990, 1990}
9685 , { 1860, 1210, 1080, 1860, 1080}
9686 }
9687 , {{ 2120, 1470, 1340, 2120, 1340}
9688 , { 2120, 1470, 1340, 2120, 1340}
9689 , { 1840, 1190, 1060, 1840, 1060}
9690 , { 180, 60, 180, -210, -60}
9691 , { 1840, 1190, 1060, 1840, 1060}
9692 }
9693 , {{ 1990, 1340, 1210, 1990, 1210}
9694 , { 1990, 1340, 1210, 1990, 1210}
9695 , { 1990, 1340, 1210, 1990, 1210}
9696 , { 1990, 1340, 1210, 1990, 1210}
9697 , { 1860, 1210, 1080, 1860, 1080}
9698 }
9699 , {{ 2120, 2120, 1990, 1840, 1990}
9700 , { -120, -230, -120, -510, -360}
9701 , { 1840, 1190, 1060, 1840, 1060}
9702 , { 2120, 2120, 1990, 1840, 1990}
9703 , { 1840, 1190, 1060, 1840, 1060}
9704 }
9705 , {{ 1990, 1340, 1210, 1990, 1210}
9706 , { 1990, 1340, 1210, 1990, 1210}
9707 , { 1550, 900, 770, 1550, 770}
9708 , { 1990, 1340, 1210, 1990, 1210}
9709 , { 640, 640, 270, 120, 270}
9710 }
9711 }
9712 , {{ { 2120, 2120, 1990, 300, 1990}
9713 , { 1470, 1470, 1340, 190, 1340}
9714 , { 1340, 1340, 1210, 300, 1210}
9715 , { 2120, 2120, 1990, 60, 1990}
9716 , { 1210, 1210, 1080, 180, 1080}
9717 }
9718 , {{ { 1470, 1470, 1340, 190, 1340}
9719 , { 1470, 1470, 1340, 190, 1340}
9720 , { 1190, 1190, 1060, -80, 1060}
9721 , { 60, 60, -60, -1210, -60}
9722 , { 1190, 1190, 1060, -80, 1060}
9723 }
9724 , {{ { 1340, 1340, 1210, 300, 1210}
9725 , { 1340, 1340, 1210, 60, 1210}
9726 , { 1340, 1340, 1210, 300, 1210}
9727 , { 1340, 1340, 1210, 60, 1210}
9728 , { 1210, 1210, 1080, 180, 1080}
9729 }
9730 , {{ { 2120, 2120, 1990, -80, 1990}
9731 , { -230, -230, -360, -1510, -360}
9732 , { 1190, 1190, 1060, -80, 1060}
9733 , { 2120, 2120, 1990, -400, 1990}
9734 , { 1190, 1190, 1060, -80, 1060}
9735 }
9736 , {{ { 1340, 1340, 1210, 60, 1210}
9737 , { 1340, 1340, 1210, 60, 1210}
9738 , { 900, 900, 770, -130, 770}
9739 , { 1340, 1340, 1210, 60, 1210}
9740 , { 640, 640, 270, -870, 270}
9741 }
9742 }
9743 , {{ { 1990, 1840, 1990, 1840, 1750}
9744 , { 1340, 1190, 1340, 1190, 1100}
9745 , { 1210, 1060, 1210, 1060, 970}
9746 , { 1990, 1840, 1990, 1840, 1750}
9747 , { 1080, 930, 1080, 930, 840}
9748 }
9749 , {{ { 1340, 1190, 1340, 1190, 1100}
9750 , { 1340, 1190, 1340, 1190, 1100}
9751 , { 1060, 910, 1060, 910, 820}
9752 , { 180, -210, 180, -210, -60}
9753 , { 1060, 910, 1060, 910, 820}
9754 }
9755 , {{ { 1210, 1060, 1210, 1060, 970}
9756 , { 1210, 1060, 1210, 1060, 970}
9757 , { 1210, 1060, 1210, 1060, 970}
9758 , { 1210, 1060, 1210, 1060, 970}
9759 , { 1080, 930, 1080, 930, 840}
9760 }
9761 , {{ { 1990, 1840, 1990, 1840, 1750}
9762 , { -120, -510, -120, -510, -360}
9763 , { 1060, 910, 1060, 910, 820}
9764 , { 1990, 1840, 1990, 1840, 1750}
9765 , { 1060, 910, 1060, 910, 820}
9766 }
9767 , {{ { 1210, 1060, 1210, 1060, 970}
9768 , { 1210, 1060, 1210, 1060, 970}
9769 , { 770, 620, 770, 620, 530}
9770 , { 1210, 1060, 1210, 1060, 970}
```

```
9771 , { 270, 120, 270, 120, 30}
9772 }
9773 }
9774 ,{{ { 2120, 400, 1990, 2120, 1990}
9775 , { 2120, 290, 1340, 2120, 1340}
9776 , { 1990, 400, 1210, 1990, 1210}
9777 , { 1990, 160, 1990, 1990, 1990}
9778 , { 1860, 270, 1080, 1860, 1080}
9779 }
9780 ,{{ { 2120, 290, 1340, 2120, 1340}
9781 , { 2120, 290, 1340, 2120, 1340}
9782 , { 1840, 10, 1060, 1840, 1060}
9783 , { -60, -1110, -60, -530, -60}
9784 , { 1840, 10, 1060, 1840, 1060}
9785 }
9786 ,{{ { 1990, 400, 1210, 1990, 1210}
9787 , { 1990, 160, 1210, 1990, 1210}
9788 , { 1990, 400, 1210, 1990, 1210}
9789 , { 1990, 160, 1210, 1990, 1210}
9790 , { 1860, 270, 1080, 1860, 1080}
9791 }
9792 ,{{ { 1990, 10, 1990, 1840, 1990}
9793 , { -360, -1410, -360, -830, -360}
9794 , { 1840, 10, 1060, 1840, 1060}
9795 , { 1990, -310, 1990, 270, 1990}
9796 , { 1840, 10, 1060, 1840, 1060}
9797 }
9798 ,{{ { 1990, 160, 1210, 1990, 1210}
9799 , { 1990, 160, 1210, 1990, 1210}
9800 , { 1550, -40, 770, 1550, 770}
9801 , { 1990, 160, 1210, 1990, 1210}
9802 , { 270, -780, 270, -200, 270}
9803 }
9804 }
9805 ,{{ { 1860, 1840, 1860, 1840, 640}
9806 , { 1210, 1190, 1210, 1190, 640}
9807 , { 1080, 1060, 1080, 1060, 270}
9808 , { 1860, 1840, 1860, 1840, 270}
9809 , { 950, 930, 950, 930, 140}
9810 }
9811 ,{{ { 1210, 1190, 1210, 1190, 640}
9812 , { 1210, 1190, 1210, 1190, 640}
9813 , { 930, 910, 930, 910, 120}
9814 , { 50, -210, 50, -210, -1000}
9815 , { 930, 910, 930, 910, 120}
9816 }
9817 ,{{ { 1080, 1060, 1080, 1060, 270}
9818 , { 1080, 1060, 1080, 1060, 270}
9819 , { 1080, 1060, 1080, 1060, 270}
9820 , { 1080, 1060, 1080, 1060, 270}
9821 , { 950, 930, 950, 930, 140}
9822 }
9823 ,{{ { 1860, 1840, 1860, 1840, 120}
9824 , { -250, -510, -250, -510, -1300}
9825 , { 930, 910, 930, 910, 120}
9826 , { 1860, 1840, 1860, 1840, -200}
9827 , { 930, 910, 930, 910, 120}
9828 }
9829 ,{{ { 1080, 1060, 1080, 1060, 270}
9830 , { 1080, 1060, 1080, 1060, 270}
9831 , { 640, 620, 640, 620, -170}
9832 , { 1080, 1060, 1080, 1060, 270}
9833 , { 140, 120, 140, 120, -670}
9834 }
9835 }
9836 }
9837 ,{{ {{ { 2120, 2120, 1990, 2120, 1990}
9838 , { 2120, 1470, 1340, 2120, 1340}
9839 , { 1990, 1340, 1210, 1990, 1210}
9840 , { 2120, 2120, 1990, 1990, 1990}
9841 , { 1860, 1210, 1080, 1860, 1080}
9842 }
9843 ,{{ { 2120, 1470, 1340, 2120, 1340}
9844 , { 2120, 1470, 1340, 2120, 1340}
9845 , { 1840, 1190, 1060, 1840, 1060}
9846 , { 400, 290, 400, 10, 170}
9847 , { 1840, 1190, 1060, 1840, 1060}
9848 }
9849 ,{{ { 1990, 1340, 1210, 1990, 1210}
9850 , { 1990, 1340, 1210, 1990, 1210}
9851 , { 1990, 1340, 1210, 1990, 1210}
9852 , { 1990, 1340, 1210, 1990, 1210}
9853 , { 1860, 1210, 1080, 1860, 1080}
9854 }
9855 ,{{ { 2120, 2120, 1990, 1840, 1990}
9856 , { 540, 190, 540, -80, 70}
9857 , { 1840, 1190, 1060, 1840, 1060}
```

```
9858 , { 2120, 2120, 1990, 1840, 1990}
9859 , { 1840, 1190, 1060, 1840, 1060}
9860 }
9861 , {{ 1990, 1340, 1210, 1990, 1210}
9862 , { 1990, 1340, 1210, 1990, 1210}
9863 , { 1750, 1100, 970, 1750, 970}
9864 , { 1990, 1340, 1210, 1990, 1210}
9865 , { 640, 640, 270, 120, 270}
9866 }
9867 }
9868 , {{{ 2120, 2120, 1990, 540, 1990}
9869 , { 1470, 1470, 1340, 190, 1340}
9870 , { 1340, 1340, 1210, 540, 1210}
9871 , { 2120, 2120, 1990, 60, 1990}
9872 , { 1210, 1210, 1080, 180, 1080}
9873 }
9874 , {{ 1470, 1470, 1340, 190, 1340}
9875 , { 1470, 1470, 1340, 190, 1340}
9876 , { 1190, 1190, 1060, -80, 1060}
9877 , { 290, 290, 160, -980, 160}
9878 , { 1190, 1190, 1060, -80, 1060}
9879 }
9880 , {{ 1340, 1340, 1210, 540, 1210}
9881 , { 1340, 1340, 1210, 60, 1210}
9882 , { 1340, 1340, 1210, 540, 1210}
9883 , { 1340, 1340, 1210, 60, 1210}
9884 , { 1210, 1210, 1080, 180, 1080}
9885 }
9886 , {{ 2120, 2120, 1990, -80, 1990}
9887 , { 190, 190, 60, -1080, 60}
9888 , { 1190, 1190, 1060, -80, 1060}
9889 , { 2120, 2120, 1990, -400, 1990}
9890 , { 1190, 1190, 1060, -80, 1060}
9891 }
9892 , {{ 1340, 1340, 1210, 70, 1210}
9893 , { 1340, 1340, 1210, 60, 1210}
9894 , { 1100, 1100, 970, 70, 970}
9895 , { 1340, 1340, 1210, 60, 1210}
9896 , { 640, 640, 270, -810, 270}
9897 }
9898 }
9899 , {{{ 1990, 1840, 1990, 1840, 1750}
9900 , { 1340, 1190, 1340, 1190, 1100}
9901 , { 1210, 1060, 1210, 1060, 970}
9902 , { 1990, 1840, 1990, 1840, 1750}
9903 , { 1080, 930, 1080, 930, 840}
9904 }
9905 , {{ 1340, 1190, 1340, 1190, 1100}
9906 , { 1340, 1190, 1340, 1190, 1100}
9907 , { 1060, 910, 1060, 910, 820}
9908 , { 400, 10, 400, 10, 170}
9909 , { 1060, 910, 1060, 910, 820}
9910 }
9911 , {{ 1210, 1060, 1210, 1060, 970}
9912 , { 1210, 1060, 1210, 1060, 970}
9913 , { 1210, 1060, 1210, 1060, 970}
9914 , { 1210, 1060, 1210, 1060, 970}
9915 , { 1080, 930, 1080, 930, 840}
9916 }
9917 , {{ 1990, 1840, 1990, 1840, 1750}
9918 , { 540, -80, 540, -80, 70}
9919 , { 1060, 910, 1060, 910, 820}
9920 , { 1990, 1840, 1990, 1840, 1750}
9921 , { 1060, 910, 1060, 910, 820}
9922 }
9923 , {{ 1210, 1060, 1210, 1060, 970}
9924 , { 1210, 1060, 1210, 1060, 970}
9925 , { 970, 820, 970, 820, 740}
9926 , { 1210, 1060, 1210, 1060, 970}
9927 , { 270, 120, 270, 120, 30}
9928 }
9929 }
9930 , {{{ 2120, 400, 1990, 2120, 1990}
9931 , { 2120, 290, 1340, 2120, 1340}
9932 , { 1990, 400, 1210, 1990, 1210}
9933 , { 1990, 160, 1990, 1990, 1990}
9934 , { 1860, 270, 1080, 1860, 1080}
9935 }
9936 , {{ 2120, 290, 1340, 2120, 1340}
9937 , { 2120, 290, 1340, 2120, 1340}
9938 , { 1840, 10, 1060, 1840, 1060}
9939 , { 160, -890, 160, -310, 160}
9940 , { 1840, 10, 1060, 1840, 1060}
9941 }
9942 , {{ 1990, 400, 1210, 1990, 1210}
9943 , { 1990, 160, 1210, 1990, 1210}
9944 , { 1990, 400, 1210, 1990, 1210}
```

```

9945 , { 1990,   160,  1210,  1990,  1210}
9946 , { 1860,   270,  1080,  1860,  1080}
9947 }
9948 , {{ 1990,    10,  1990,  1840,  1990}
9949 , {   60, -980,    60, -400,    60}
9950 , { 1840,    10,  1060,  1840,  1060}
9951 , { 1990, -310,  1990,   270,  1990}
9952 , { 1840,    10,  1060,  1840,  1060}
9953 }
9954 , {{ 1990,   170,  1210,  1990,  1210}
9955 , { 1990,   160,  1210,  1990,  1210}
9956 , { 1750,   170,   970,  1750,   970}
9957 , { 1990,   160,  1210,  1990,  1210}
9958 , { 270, -780,   270, -200,   270}
9959 }
9960 }
9961 , {{{ 1860,  1840,  1860,  1840,   640}
9962 , { 1210,  1190,  1210,  1190,   640}
9963 , { 1080,  1060,  1080,  1060,   270}
9964 , { 1860,  1840,  1860,  1840,   270}
9965 , { 950,   930,   950,   930,   140}
9966 }
9967 , {{ 1210,  1190,  1210,  1190,   640}
9968 , { 1210,  1190,  1210,  1190,   640}
9969 , { 930,   910,   930,   910,   120}
9970 , { 270,    10,   270,    10, -780}
9971 , { 930,   910,   930,   910,   120}
9972 }
9973 , {{ 1080,  1060,  1080,  1060,   270}
9974 , { 1080,  1060,  1080,  1060,   270}
9975 , { 1080,  1060,  1080,  1060,   270}
9976 , { 1080,  1060,  1080,  1060,   270}
9977 , { 950,   930,   950,   930,   140}
9978 }
9979 , {{ 1860,  1840,  1860,  1840,   120}
9980 , { 180,   -80,   180,   -80, -810}
9981 , { 930,   910,   930,   910,   120}
9982 , { 1860,  1840,  1860,  1840, -200}
9983 , { 930,   910,   930,   910,   120}
9984 }
9985 , {{ 1080,  1060,  1080,  1060,   270}
9986 , { 1080,  1060,  1080,  1060,   270}
9987 , { 840,   820,   840,   820,    30}
9988 , { 1080,  1060,  1080,  1060,   270}
9989 , { 140,   120,   140,   120, -670}
9990 }
9991 }
9992 }
9993 }};


```

## 18.170 ViennaRNA/params/io.h File Reference

Read and write energy parameter files.

This graph shows which files directly or indirectly include this file:

### Macros

- `#define VRNA_PARAMETER_FORMAT_DEFAULT 0`  
*Default Energy Parameter File format.*

### Functions

- `int vrna_params_load (const char fname[], unsigned int options)`  
*Load energy parameters from a file.*
- `int vrna_params_save (const char fname[], unsigned int options)`  
*Save energy parameters to a file.*
- `int vrna_params_load_from_string (const char *string, const char *name, unsigned int options)`  
*Load energy paramters from string.*
- `int vrna_params_load_defaults (void)`  
*Load default RNA energy parameter set.*
- `int vrna_params_load_RNA_Turner2004 (void)`  
*Load Turner 2004 RNA energy parameter set.*

- int `vrna_params_load_RNA_Turner1999` (void)  
*Load Turner 1999 RNA energy parameter set.*
- int `vrna_params_load_RNA_Andronescu2007` (void)  
*Load Andronescu 2007 RNA energy parameter set.*
- int `vrna_params_load_RNA_Langdon2018` (void)  
*Load Langdon 2018 RNA energy parameter set.*
- int `vrna_params_load_RNA_misc_special_hairpins` (void)  
*Load Misc Special Hairpin RNA energy parameter set.*
- int `vrna_params_load_DNA_Mathews2004` (void)  
*Load Mathews 2004 DNA energy parameter set.*
- int `vrna_params_load_DNA_Mathews1999` (void)  
*Load Mathews 1999 DNA energy parameter set.*
- const char \* `last_parameter_file` (void)  
*Get the file name of the parameter file that was most recently loaded.*
- void `read_parameter_file` (const char fname[])  
*Read energy parameters from a file.*
- void `write_parameter_file` (const char fname[])  
*Write energy parameters to a file.*

### 18.170.1 Detailed Description

Read and write energy parameter files.

## 18.171 io.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PARAMS_IO_H
2 #define VIENNA_RNA_PACKAGE_PARAMS_IO_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 #  define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
16 #define VRNA_PARAMETER_FORMAT_DEFAULT 0
17
18
19 int
20 vrna_params_load(const char fname[],
21                      unsigned int options);
22
23
24 int
25 vrna_params_save(const char fname[],
26                      unsigned int options);
27
28
29 int
30 vrna_params_load_from_string(const char *string,
31                                 const char *name,
32                                 unsigned int options);
33
34
35 int
36 vrna_params_load_defaults(void);
37
38
39 int
40 vrna_params_load_RNA_Turner2004(void);
41
42
43 int
44 vrna_params_load_RNA_Turner1999(void);
45
46
47 int
48 vrna_params_load_RNA_Andronescu2007(void);
49
50
51 int
52 vrna_params_load_RNA_Langdon2018(void);
53
54
55 int
56 vrna_params_load_RNA_misc_special_hairpins(void);
57
58
59 int
60 vrna_params_load_DNA_Mathews2004(void);
61
62
63 int
64 vrna_params_load_DNA_Mathews1999(void);
65
66
67 int
68 last_parameter_file(void);
69
70
71 int
72 read_parameter_file(const char fname[]);
73
74
75 int
76 write_parameter_file(const char fname[]);
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
307
308
309
309
310
311
312
313
314
315
316
317
317
318
319
319
320
321
322
323
324
325
326
327
327
328
329
329
330
331
332
333
334
335
336
337
337
338
339
339
340
341
342
343
344
345
345
346
347
347
348
349
349
350
351
352
353
354
355
355
356
357
357
358
359
359
360
361
362
363
364
364
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
13
```

```

143 vrna_params_load_RNA_Turner1999(void);
144
145
158 int
159 vrna_params_load_RNA_Andronescu2007(void);
160
161
174 int
175 vrna_params_load_RNA_Langdon2018(void);
176
177
190 int
191 vrna_params_load_RNA_misc_special_hairpins(void);
192
193
206 int
207 vrna_params_load_DNA_Mathews2004(void);
208
209
222 int
223 vrna_params_load_DNA_Mathews1999(void);
224
225
226 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
227
232 enum parset {
233     UNKNOWN= -1, QUIT,
234     S, S_H, HP, HP_H, B, B_H, IL, IL_H, MMH, MMH_H, MMI, MMI_H,
235     MMI1N, MMI1N_H, MMI23, MMI23_H, MMM, MMM_H, MME, MME_H, D5, D5_H, D3, D3_H,
236     INT11, INT11_H, INT21, INT21_H, INT22, INT22_H, ML, TL,
237     TRI, HEX, NIN, MISC
238 };
239
240
246 const char *
247 last_parameter_file(void);
248
249
256 DEPRECATED(void
257             read_parameter_file(const char fname[]),
258             "Use vrna_params_load() instead!");
259
260
267 DEPRECATED(void
268             write_parameter_file(const char fname[]),
269             "Use vrna_params_save() instead!");
270
271
276 enum parset
277 gettype(const char *ident);
278
279
284 char *
285 settype(enum parset s);
286
287
292 #endif
293
294 #endif

```

## 18.172 ViennaRNA/part\_func.h File Reference

Partition function implementations.

Include dependency graph for part\_func.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [vrna\\_dimer\\_pf\\_s](#)  
*Data structure returned by [vrna\\_pf\\_dimer\(\)](#) More...*
- struct [vrna\\_multimer\\_pf\\_s](#)

### Typedefs

- typedef struct [vrna\\_dimer\\_pf\\_s](#) [vrna\\_dimer\\_pf\\_t](#)  
*Typename for the data structure that stores the dimer partition functions, [vrna\\_dimer\\_pf\\_s](#), as returned by [vrna\\_pf\\_dimer\(\)](#)*

- **typedef struct `vrna_dimer_pf_s` `cofoldF`**  
*Backward compatibility typedef for `vrna_dimer_pf_s`.*

## Functions

- **int `vrna_pf_float_precision` (void)**  
*Find out whether partition function computations are using single precision floating points.*
- **float `pf_fold_par` (const char \*sequence, char \*structure, `vrna_exp_param_t` \*parameters, int calculate\_bppm, int is\_constrained, int is\_circular)**  
*Compute the partition function  $Q$  for a given RNA sequence.*
- **float `pf_fold` (const char \*sequence, char \*structure)**  
*Compute the partition function  $Q$  of an RNA sequence.*
- **float `pf_circ_fold` (const char \*sequence, char \*structure)**  
*Compute the partition function of a circular RNA sequence.*
- **char \* `pbacktrack` (char \*sequence)**  
*Sample a secondary structure from the Boltzmann ensemble according its probability.*
- **char \* `pbacktrack5` (char \*sequence, int length)**  
*Sample a sub-structure from the Boltzmann ensemble according its probability.*
- **char \* `pbacktrack_circ` (char \*sequence)**  
*Sample a secondary structure of a circular RNA from the Boltzmann ensemble according its probability.*
- **void `free_pf_arrays` (void)**  
*Free arrays for the partition function recursions.*
- **void `update_pf_params` (int length)**  
*Recalculate energy parameters.*
- **void `update_pf_params_par` (int length, `vrna_exp_param_t` \*parameters)**  
*Recalculate energy parameters.*
- **`FLT_OR_DBL` \* `export_bppm` (void)**  
*Get a pointer to the base pair probability array.*
- **int `get_pf_arrays` (short \*\*S\_p, short \*\*S1\_p, char \*\*ptype\_p, `FLT_OR_DBL` \*\*qb\_p, `FLT_OR_DBL` \*\*qm\_p, `FLT_OR_DBL` \*\*q1k\_p, `FLT_OR_DBL` \*\*qln\_p)**  
*Get the pointers to (almost) all relevant computation arrays used in partition function computation.*
- **double `get_subseq_F` (int i, int j)**  
*Get the free energy of a subsequence from the q[] array.*
- **double `mean_bp_distance` (int length)**  
*Get the mean base pair distance of the last partition function computation.*
- **double `mean_bp_distance_pr` (int length, `FLT_OR_DBL` \*pr)**  
*Get the mean base pair distance in the thermodynamic ensemble.*
- **`vrna_ep_t` \* `stackProb` (double cutoff)**  
*Get the probability of stacks.*
- **void `init_pf_fold` (int length)**  
*Allocate space for `pf_fold()`*
- **char \* `centroid` (int length, double \*dist)**
- **char \* `get_centroid_struct_gquad_pr` (int length, double \*dist)**
- **double `mean_bp_dist` (int length)**
- **double `expLoopEnergy` (int u1, int u2, int type, int type2, short si1, short sj1, short sp1, short sq1)**
- **double `expHairpinEnergy` (int u, int type, short si1, short sj1, const char \*string)**

## Basic global partition function interface

- **`FLT_OR_DBL` `vrna_pf` (`vrna_fold_compound_t` \*vc, char \*structure)**  
*Compute the partition function  $Q$  for a given RNA sequence, or sequence alignment.*
- **`vrna_dimer_pf_t` `vrna_pf_dimer` (`vrna_fold_compound_t` \*vc, char \*structure)**  
*Calculate partition function and base pair probabilities of nucleic acid/nucleic acid dimers.*

- `FLT_OR_DBL * vrna_pf_substrands` (`vrna_fold_compound_t *fc, size_t complex_size`)
- `FLT_OR_DBL vrna_pf_add` (`FLT_OR_DBL dG1, FLT_OR_DBL dG2, double kT`)

### Simplified global partition function computation using sequence(s) or multiple sequence alignment(s)

- `float vrna_pf_fold` (`const char *sequence, char *structure, vrna_ep_t **pl`)  
*Compute Partition function Q (and base pair probabilities) for an RNA sequence using a comparative method.*
- `float vrna_pf_circfold` (`const char *sequence, char *structure, vrna_ep_t **pl`)  
*Compute Partition function Q (and base pair probabilities) for a circular RNA sequences using a comparative method.*
- `float vrna_pf_alifold` (`const char **sequences, char *structure, vrna_ep_t **pl`)  
*Compute Partition function Q (and base pair probabilities) for an RNA sequence alignment using a comparative method.*
- `float vrna_pf_circalifold` (`const char **sequences, char *structure, vrna_ep_t **pl`)  
*Compute Partition function Q (and base pair probabilities) for an alignment of circular RNA sequences using a comparative method.*
- `vrna_dimer_pf_t vrna_pf_co_fold` (`const char *seq, char *structure, vrna_ep_t **pl`)  
*Calculate partition function and base pair probabilities of nucleic acid/nucleic acid dimers.*

## Variables

- `int st_back`  
*Flag indicating that auxiliary arrays are needed throughout the computations. This is essential for stochastic backtracking.*

### 18.172.1 Detailed Description

Partition function implementations.

This file includes (almost) all function declarations within the **RNAlib** that are related to Partition function computations

### 18.172.2 Function Documentation

#### 18.172.2.1 centroid()

```
char * centroid (
    int length,
    double * dist )
```

**Deprecated** This function is deprecated and should not be used anymore as it is not threadsafe!

See also

[get\\_centroid\\_struct\\_pl\(\)](#), [get\\_centroid\\_struct\\_pr\(\)](#)

#### 18.172.2.2 get\_centroid\_struct\_gquad\_pr()

```
char * get_centroid_struct_gquad_pr (
    int length,
    double * dist )
```

**Deprecated** This function is deprecated and should not be used anymore as it is not threadsafe!

See also

[vrna\\_centroid\(\)](#), [vrna\\_centroid\\_from\\_probs\(\)](#), [vrna\\_centroid\\_from plist\(\)](#)

### 18.172.2.3 mean\_bp\_dist()

```
double mean_bp_dist (
    int length )
get the mean pair distance of ensemble
```

**Deprecated** This function is not threadsafe and should not be used anymore. Use [mean\\_bp\\_distance\(\)](#) instead!

### 18.172.2.4 expLoopEnergy()

```
double expLoopEnergy (
    int u1,
    int u2,
    int type,
    int type2,
    short sil,
    short sj1,
    short spl,
    short sq1 )
```

**Deprecated** Use [exp\\_E\\_IntLoop\(\)](#) from [loop\\_energies.h](#) instead

### 18.172.2.5 expHairpinEnergy()

```
double expHairpinEnergy (
    int u,
    int type,
    short sil,
    short sj1,
    const char * string )
```

**Deprecated** Use [exp\\_E\\_Hairpin\(\)](#) from [loop\\_energies.h](#) instead

## 18.173 part\_func.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_PART_FUNC_H
2 #define VIENNA_RNA_PACKAGE_PART_FUNC_H
3
8 typedef struct vrna_dimer_pf_s vrna_dimer_pf_t;
9
10 typedef struct vrna_multimer_pf_s vrna_multimer_pf_t;
11
12 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
13
18 typedef struct vrna_dimer_pf_s cofoldF;
19
20 #endif
21
22
23 #include <ViennaRNA/datastructures/basic.h>
24 #include <ViennaRNA/fold_compound.h>
25 #include <ViennaRNA/utils/structures.h>
26 #include <ViennaRNA/params/basic.h>
27 #include <ViennaRNA/centroid.h>
28 #include <ViennaRNA/equilibrium_probs.h>
29 #include <ViennaRNA/boltzmann_sampling.h>
30
31 #ifdef VRNA_WARN_DEPRECATED
32 # if defined(__clang__)
33 #   define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
34 # elif defined(__GNUC__)
35 #   define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
36 # else
37 #   define DEPRECATED(func, msg) func
```

```

38 # endif
39 #else
40 # define DEPRECATED(func, msg) func
41 #endif
42 /*
53 ##########
54 # PARTITION FUNCTION COMPUTATION #
55 #########
56 */
57
58 struct vrna_dimer_pf_s {
59     /* free energies for: */
60     double FQAB;
61     double FAB;
62     double FcAB;
63     double FA;
64     double FB;
65 };
66
67 struct vrna_multimer_pf_s {
68     /* free energies for: */
69     double F_connected;
70     double *F_monomers;
71     size_t num_monomers;
72 };
73
74 FLT_OR_DBL
75 vrna_pf(vrna_fold_compound_t *vc,
76          char *structure);
77
78
79 vrna_dimer_pf_t
80 vrna_pf_dimer(vrna_fold_compound_t *vc,
81                 char *structure);
82
83
84 FLT_OR_DBL *
85 vrna_pf_substrands(vrna_fold_compound_t *fc,
86                      size_t complex_size);
87
88 FLT_OR_DBL
89 vrna_pf_add(FLT_OR_DBL dG1,
90              FLT_OR_DBL dG2,
91              double kT);
92
93
94 /* End basic global interface */
95
96 float
97 vrna_pf_fold(const char *sequence,
98               char *structure,
99               vrna_ep_t **pl);
100
101
102 float
103 vrna_pf_circfold(const char *sequence,
104                     char *structure,
105                     vrna_ep_t **pl);
106
107
108 float
109 vrna_pf_alifold(const char **sequences,
110                   char *structure,
111                   vrna_ep_t **pl);
112
113
114 float
115 vrna_pf_circalifold(const char **sequences,
116                       char *structure,
117                       vrna_ep_t **pl);
118
119
120 float
121 vrna_dimer_pf_t
122 vrna_pf_co_fold(const char *seq,
123                   char *structure,
124                   vrna_ep_t **pl);
125
126
127 /* End simplified global interface */
128 */
129
130
131 int
132 vrna_pf_float_precision(void);
133
134
135
136
137
138
139
140
141
142
143
144 ##########
145 # OTHER PARTITION FUNCTION RELATED DECLARATIONS #
146 #########
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344 ##########
345 # OTHER PARTITION FUNCTION RELATED DECLARATIONS #
346 #########
347 */
348
349
350
351
352
353
354
355
356
357
358
359
360

```

```

361
362 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
363
364 /*
365 ##### DEPRECATED FUNCTIONS #####
366 # DEPRECATED FUNCTIONS #
367 ##### DEPRECATED FUNCTIONS #####
368 */
369
382 extern int st_back;
383
424 DEPRECATED(float
425             pf_fold_par(const char      *sequence,
426                         char          *structure,
427                         vrna_exp_param_t *parameters,
428                         int           calculate_bppm,
429                         int           is_constrained,
430                         int           is_circular),
431             "Use the new API and vrna_pf() instead");
432
472 DEPRECATED(float
473             pf_fold(const char *sequence,
474                     char      *structure),
475             "Use vrna_pf_fold() or vrna_pf() instead");
476
503 DEPRECATED(float
504             pf_circ_fold(const char  *sequence,
505                           char       *structure),
506             "Use vrna_pf_circfold() or vrna_pf() instead");
507
519 DEPRECATED(char *pbacktrack(char *sequence), "Use vrna_pbacktrack() instead");
520
526 DEPRECATED(char *pbacktrack5(char *sequence,
527                               int length), "Use vrna_pbacktrack5() instead");
528
544 DEPRECATED(char *pbacktrack_circ(char *sequence), "Use vrna_pbacktrack() instead");
545
564 DEPRECATED(void
565             free_pf_arrays(void), "This function is obsolete");
566
578 DEPRECATED(void
579             update_pf_params(int length), "This function is obsolete");
580
589 DEPRECATED(void
590             update_pf_params_par(int length,
591                                   vrna_exp_param_t *parameters),
592             "Use the new API with vrna_fold_compound_t instead");
593
611 DEPRECATED(FLT_OR_DBL * export_bppm(void),
612             "Use the new API with vrna_fold_compound_t instead");
613
614
631 DEPRECATED(int
632             get_pf_arrays(short      **S_p,
633                           short      **S1_p,
634                           char       **ptype_p,
635                           FLT_OR_DBL **qb_p,
636                           FLT_OR_DBL **qm_p,
637                           FLT_OR_DBL **qlk_p,
638                           FLT_OR_DBL **qln_p),
639             "Use the new API with vrna_fold_compound_t instead");
640
646 DEPRECATED(double
647             get_subseq_F(int i,
648                           int j),
649             "Use the new API with vrna_fold_compound_t instead");
650
651
663 DEPRECATED(double
664             mean_bp_distance(int length),
665             "Use vrna_mean_bp_distance() or vrna_mean_bp_distance_pr() instead");
666
684 DEPRECATED(double
685             mean_bp_distance_pr(int      length,
686                                   FLT_OR_DBL *pr),
687             "Use vrna_mean_bp_distance() or vrna_mean_bp_distance_pr() instead");
688
696 DEPRECATED(vrna_ep_t * stackProb(double cutoff), "Use vrna_stack_prob() instead");
697
706 DEPRECATED(void
707             init_pf_fold(int length), "This function is obsolete");
708
713 DEPRECATED(char *centroid(int length,
714                           double *dist),
715             "Use vrna_centroid() instead");
716

```

```

721 DEPRECATED(char *get_centroid_struct_gquad_pr(int length,
722                               double *dist),
723                               "Use vrna_centroid() instead");
724
730 DEPRECATED(double
731               mean_bp_dist(int length),
732               "Use vrna_mean_bp_distance() or vrna_mean_bp_distance_pr() instead");
733
737 DEPRECATED(double
738               expLoopEnergy(int      u1,
739                             int      u2,
740                             int      type,
741                             int      type2,
742                             short    sil,
743                             short    sj1,
744                             short    spl,
745                             short    sq1),
746               "");
747
751 DEPRECATED(double
752               expHairpinEnergy(int      u,
753                             int      type,
754                             short    sil,
755                             short    sj1,
756                             const char *string),
757               "");
758
759 /* this doesn't work if free_pf_arrays() is called before */
760 DEPRECATED(void
761               assign plist_gquad_from_pr(vrna_ep_t **pl,
762                             int      length,
763                             double   cut_off),
764               "Use vrna_plist_from_probs() instead");
765
766 #endif
767
768 #endif

```

## 18.174 ViennaRNA/part\_func\_co.h File Reference

Partition function for two RNA sequences.

Include dependency graph for part\_func\_co.h:

### Functions

- **`vrna_dimer_pf_t co_pf_fold`** (char \*sequence, char \*structure)
 

*Calculate partition function and base pair probabilities.*
- **`vrna_dimer_pf_t co_pf_fold_par`** (char \*sequence, char \*structure, `vrna_exp_param_t` \*parameters, int calculate\_bppm, int is\_constrained)
 

*Calculate partition function and base pair probabilities.*
- **`vrna_ep_t * get plist (vrna_ep_t *pl, int length, double cut_off)`**
- **`void compute_probabilities (double FAB, double FEA, double FEB, vrna_ep_t *prAB, vrna_ep_t *prA, vrna_ep_t *prB, int Alength)`**

*Compute Boltzmann probabilities of dimerization without homodimers.*
- **`void init_co_pf_fold (int length)`**
- **`FLOAT_OR_DOUBLE * export_co_bppm (void)`**

*Get a pointer to the base pair probability array.*
- **`void free_co_pf_arrays (void)`**

*Free the memory occupied by `co_pf_fold()`*
- **`void update_co_pf_params (int length)`**

*Recalculate energy parameters.*
- **`void update_co_pf_params_par (int length, vrna_exp_param_t *parameters)`**

*Recalculate energy parameters.*

## Variables

- int **mirnatog**  
*Toggles no intrabp in 2nd mol.*
- double **F\_monomer [2]**  
*Free energies of the two monomers.*

### 18.174.1 Detailed Description

Partition function for two RNA sequences.

### 18.174.2 Function Documentation

#### 18.174.2.1 get plist()

```
vrna_ep_t * get_plist (
    vrna_ep_t * pl,
    int length,
    double cut_off )
```

DO NOT USE THIS FUNCTION ANYMORE

**Deprecated** { This function is deprecated and will be removed soon!} use [assign plist from pr\(\)](#) instead!

## 18.175 part\_func\_co.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_PART_FUNC_CO_H
2 #define VIENNA_RNA_PACKAGE_PART_FUNC_CO_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 #  define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
52 #include <ViennaRNA/datastructures/basic.h>
53 #include <ViennaRNA/params/basic.h>
54 #include <ViennaRNA/part_func.h>
55 #include <ViennaRNA/equilibrium_probs.h>
56 #include <ViennaRNA/concentrations.h>
57 #include <ViennaRNA/utils/structures.h>
58
62 extern int     mirnatog;
63
67 extern double F_monomer[2];
68
73 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
74 */
75 #####
76 # DEPRECATED FUNCTIONS
77 #####
78 */
80
101 DEPRECATED(vrna_dimer_pf_t co_pf_fold(char *sequence,
102                                char *structure),
103 "Use vrna_pf_co_fold() or vrna_pf_dimer() instead");
104
126 DEPRECATED(vrna_dimer_pf_t co_pf_fold_par(char
127                                *sequence,
128                                *structure,
129                                vrna_exp_param_t
130                                *parameters,
131                                int
132                                calculate_bppm,
133                                int
134                                is_constrained),
135 "Use the new API and vrna_pf_dimer() instead");
136
```

```

138 DEPRECATED(vrna_ep_t *get plist(vrna_ep_t *pl,
139                      int length,
140                      double cut_off),
141 "Use vrna plist() and vrna plist_from_probs() instead");
142
144 DEPRECATED(void compute_probabilities(double FAB,
145                      double FEA,
146                      double FEB,
147                      vrna_ep_t *prAB,
148                      vrna_ep_t *prA,
149                      vrna_ep_t *prB,
150                      int Alength),
151 "Use vrna_pf_dimer_probs() instead");
152
158 DEPRECATED(void init_co_pf_fold(int length),
159 "This function is obsolete");
160
164 DEPRECATED(FLT_OR_DBL *export_co_bppm(void),
165 "Use the new API with vrna_fold_compound_t instead");
166
169 DEPRECATED(void free_co_pf_arrays(void),
170 "This function is obsolete");
171
174 DEPRECATED(void update_co_pf_params(int length),
175 "This function is obsolete");
176
178 DEPRECATED(void update_co_pf_params_par(int length,
179 vrna_exp_param_t *parameters),
180 "Use the new API with vrna_fold_compound_t instead");
181
182 #endif
183
184 #endif

```

## 18.176 ViennaRNA/part\_func\_up.h File Reference

Implementations for accessibility and RNA-RNA interaction as a stepwise process.  
Include dependency graph for part\_func\_up.h:

### Functions

- **pu\_contrib \* pf\_unstru** (char \*sequence, int max\_w)  
*Calculate the partition function over all unpaired regions of a maximal length.*
- **interact \* pf\_interact** (const char \*s1, const char \*s2, pu\_contrib \*p\_c, pu\_contrib \*p\_c2, int max\_w, char \*cstruc, int incr3, int incr5)  
*Calculates the probability of a local interaction between two sequences.*
- **void free\_interact** (interact \*pin)  
*Frees the output of function pf\_interact().*
- **void free\_pu\_contrib\_struct** (pu\_contrib \*pu)  
*Frees the output of function pf\_unstru().*

### 18.176.1 Detailed Description

Implementations for accessibility and RNA-RNA interaction as a stepwise process.

## 18.177 part\_func\_up.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PART_FUNC_UP_H
2 #define VIENNA_RNA_PACKAGE_PART_FUNC_UP_H
3
4 #include <ViennaRNA/datastructures/basic.h>
5
6 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
7
8 #define RNA_UP_MODE_1 1U
9 #define RNA_UP_MODE_2 2U
10 #define RNA_UP_MODE_3 4U
11
12 pu_contrib *pf_unstru(char *sequence,
13                      int max_w);

```

```

63
106 interact *pf_interact(const char *s1,
107             const char *s2,
108             pu_contrib *p_c,
109             pu_contrib *p_c2,
110             int max_w,
111             char *cstruc,
112             int incr3,
113             int incr5);
114
118 void free_interact(interact *pin);
119
123 int Up_plot(pu_contrib *p_c,
124             pu_contrib *p_c_sh,
125             interact *pint,
126             char *ofile,
127             int **unpaired_values,
128             char *select_contrib,
129             char *head,
130             unsigned int mode);
131
135 pu_contrib *get_pu_contrib_struct( unsigned int n,
136                                     unsigned int w);
137
141 void     free_pu_contrib_struct(pu_contrib *pu);
142
143 void
144 free_pu_contrib(pu_contrib *pu);
145
150 #endif
151
152 #endif

```

## 18.178 ViennaRNA/part\_func\_window.h File Reference

Partition function and equilibrium probability implementation for the sliding window algorithm.

Include dependency graph for part\_func\_window.h: This graph shows which files directly or indirectly include this file:

### Macros

- `#define VRNA_EXT_LOOP 1U`  
*Exterior loop.*
- `#define VRNA_HP_LOOP 2U`  
*Hairpin loop.*
- `#define VRNA_INT_LOOP 4U`  
*Internal loop.*
- `#define VRNA_MB_LOOP 8U`  
*Multibranch loop.*
- `#define VRNA_ANY_LOOP (VRNA_EXT_LOOP | VRNA_HP_LOOP | VRNA_INT_LOOP | VRNA_MB_LOOP)`  
*Any loop.*
- `#define VRNA_PROBS_WINDOW_BPP 4096U`  
*Trigger base pairing probabilities.*
- `#define VRNA_PROBS_WINDOW_UP 8192U`  
*Trigger unpaired probabilities.*
- `#define VRNA_PROBS_WINDOW_STACKP 16384U`  
*Trigger base pair stack probabilities.*
- `#define VRNA_PROBS_WINDOW_UP_SPLIT 32768U`  
*Trigger detailed unpaired probabilities split up into different loop type contexts.*
- `#define VRNA_PROBS_WINDOW_PF 65536U`  
*Trigger partition function.*

## Typedefs

- `typedef void() vrna_probs_window_callback(FLT_OR_DBL *pr, int pr_size, int i, int max, unsigned int type, void *data)`  
*Sliding window probability computation callback.*

## Functions

### Basic local partition function interface

- `int vrna_probs_window (vrna_fold_compound_t *fc, int ulength, unsigned int options, vrna_probs_window_callback *cb, void *data)`  
*Compute various equilibrium probabilities under a sliding window approach.*

### Simplified global partition function computation using sequence(s) or multiple sequence alignment(s)

- `vrna_ep_t * vrna_pfl_fold (const char *sequence, int window_size, int max_bp_span, float cutoff)`  
*Compute base pair probabilities using a sliding-window approach.*
- `int vrna_pfl_fold_cb (const char *sequence, int window_size, int max_bp_span, vrna_probs_window_callback *cb, void *data)`  
*Compute base pair probabilities using a sliding-window approach (callback version)*
- `double ** vrna_pfl_fold_up (const char *sequence, int ulength, int window_size, int max_bp_span)`  
*Compute probability of contiguous unpaired segments.*
- `int vrna_pfl_fold_up_cb (const char *sequence, int ulength, int window_size, int max_bp_span, vrna_probs_window_callback *cb, void *data)`  
*Compute probability of contiguous unpaired segments.*

### 18.178.1 Detailed Description

Partition function and equilibrium probability implementation for the sliding window algorithm.

This file contains the implementation for sliding window partition function and equilibrium probabilities. It also provides the unpaired probability implementation from Bernhart et al. 2011 [4]

## 18.179 part\_func\_window.h

### Go to the documentation of this file.

```

1 #ifndef VIENNA_RNA_PACKAGE_PART_FUNC_WINDOW_H
2 #define VIENNA_RNA_PACKAGE_PART_FUNC_WINDOW_H
3
15 #include <ViennaRNA/datastructures/basic.h>
16
63 typedef void (vrna_probs_window_callback) (FLT_OR_DBL *pr,
64                            int pr_size,
65                            int i,
66                            int max,
67                            unsigned int type,
68                            void *data);
69
70 #include <ViennaRNA/fold_compound.h>
71 #include <ViennaRNA/utils/structures.h>
72
76 #define VRNA_EXT_LOOP 1U
77
81 #define VRNA_HP_LOOP 2U
82
86 #define VRNA_INT_LOOP 4U
87
91 #define VRNA_MB_LOOP 8U
92
96 #define VRNA_ANY_LOOP (VRNA_EXT_LOOP | VRNA_HP_LOOP | VRNA_INT_LOOP | VRNA_MB_LOOP)
97
98
111 #define VRNA_PROBS_WINDOW_BPP 4096U
112
125 #define VRNA_PROBS_WINDOW_UP 8192U
126
139 #define VRNA_PROBS_WINDOW_STACKP 16384U
140

```

```

158 #define VRNA_PROBS_WINDOW_UP_SPLIT    32768U
159
160
174 #define VRNA_PROBS_WINDOW_PF        65536U
175
176 int
208 vrna_probs_window(vrna_fold_compound_t      *fc,
209                      int                  ulength,
210                      unsigned int         options,
211                      vrna_probs_window_callback *cb,
212                      void                 *data);
213
214 /* End basic interface */
243 vrna_ep_t *
244 vrna_pfl_fold(const char   *sequence,
245                  int          window_size,
246                  int          max_bp_span,
247                  float        cutoff);
248
249
272 int
273 vrna_pfl_fold_cb(const char           *sequence,
274                      int                  ulength,
275                      int                  max_bp_span,
276                      vrna_probs_window_callback *cb,
277                      void                 *data);
278
279
302 double **
303 vrna_pfl_fold_up(const char *sequence,
304                     int          ulength,
305                     int          window_size,
306                     int          max_bp_span);
307
308
332 int
333 vrna_pfl_fold_up_cb(const char           *sequence,
334                      int                  ulength,
335                      int                  window_size,
336                      int                  max_bp_span,
337                      vrna_probs_window_callback *cb,
338                      void                 *data);
339
340
341 /* End simplified interface */
342 #endif

```

## 18.180 ViennaRNA/perturbation\_fold.h File Reference

Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments.

Include dependency graph for perturbation\_fold.h:

### Macros

- `#define VRNA_OBJECTIVE_FUNCTION_QUADRATIC 0`  
*Use the sum of squared aberrations as objective function.*
- `#define VRNA_OBJECTIVE_FUNCTION_ABSOLUTE 1`  
*Use the sum of absolute aberrations as objective function.*
- `#define VRNA_MINIMIZER_DEFAULT 0`  
*Use a custom implementation of the gradient descent algorithm to minimize the objective function.*
- `#define VRNA_MINIMIZER_CONJUGATE_FR 1`  
*Use the GNU Scientific Library implementation of the Fletcher-Reeves conjugate gradient algorithm to minimize the objective function.*
- `#define VRNA_MINIMIZER_CONJUGATE_PR 2`  
*Use the GNU Scientific Library implementation of the Polak-Ribiere conjugate gradient algorithm to minimize the objective function.*
- `#define VRNA_MINIMIZER_VECTOR_BFGS 3`  
*Use the GNU Scientific Library implementation of the vector Broyden-Fletcher-Goldfarb-Shanno algorithm to minimize the objective function.*
- `#define VRNA_MINIMIZER_VECTOR_BFGS2 4`

*Use the GNU Scientific Library implementation of the vector Broyden-Fletcher-Goldfarb-Shanno algorithm to minimize the objective function.*

- `#define VRNA_MINIMIZER_STEEPEST_DESCENT 5`

*Use the GNU Scientific Library implementation of the steepest descent algorithm to minimize the objective function.*

## Typedefs

- `typedef void(* progress_callback) (int iteration, double score, double *epsilon)`

*Callback for following the progress of the minimization process.*

## Functions

- `void vrna_sc_minimize_perturbation (vrna_fold_compound_t *vc, const double *q_prob_unpaired, int objective_function, double sigma_squared, double tau_squared, int algorithm, int sample_size, double *epsilon, double initialStepSize, double minStepSize, double minImprovement, double minimizerTolerance, progress_callback callback)`

*Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments.*

### 18.180.1 Detailed Description

Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments.

## 18.181 perturbation\_fold.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PERTURBATION_FOLD_H
2 #define VIENNA_RNA_PACKAGE_PERTURBATION_FOLD_H
3
4 #include <ViennaRNA/fold_compound.h>
5
24 #define VRNA_OBJECTIVE_FUNCTION_QUADRATIC 0
25
33 #define VRNA_OBJECTIVE_FUNCTION_ABSOLUTE 1
34
40 #define VRNA_MINIMIZER_DEFAULT 0
41
49 #define VRNA_MINIMIZER_CONJUGATE_FR 1
50
58 #define VRNA_MINIMIZER_CONJUGATE_PR 2
59
67 #define VRNA_MINIMIZER_VECTOR_BFGS 3
68
76 #define VRNA_MINIMIZER_VECTOR_BFGS2 4
77
85 #define VRNA_MINIMIZER_STEEPEST_DESCENT 5
86
96 typedef void (*progress_callback)(int iteration,
97                                     double score,
98                                     double *epsilon);
99
139 void vrna_sc_minimize_perturbation(vrna_fold_compound_t *vc,
140                                       const double *q_prob_unpaired,
141                                       int objective_function,
142                                       double sigma_squared,
143                                       double tau_squared,
144                                       int algorithm,
145                                       int sample_size,
146                                       double *epsilon,
147                                       double initialStepSize,
148                                       double minStepSize,
149                                       double minImprovement,
150                                       double minimizerTolerance,
151                                       progress_callback callback);
152
153
154 #endif

```

## 18.182 pf\_multifold.h

```

1 #ifndef VIENNA_RNA_PACKAGE_PART_FUNC_MULTIFOLD_H
2 #define VIENNA_RNA_PACKAGE_PART_FUNC_MULTIFOLD_H
3
4 #include "ViennaRNA/fold_compound.h"
5
6 int
7 vrna_pf_multifold_prepare(vrna_fold_compound_t *fc);
8
9
10#endif

```

## 18.183 ViennaRNA/pk\_plex.h File Reference

Heuristics for two-step pseudoknot forming interaction predictions.

Include dependency graph for pk\_plex.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [vrna\\_pk\\_plex\\_result\\_s](#)  
*A result of the RNA PKplex interaction prediction. [More...](#)*

### Typedefs

- [typedef int\(\) vrna\\_callback\\_pk\\_plex\\_score](#)(const short \*pt, int start\_5, int end\_5, int start\_3, int end\_3, void \*data)  
*Pseudoknot loop scoring function prototype.*
- [typedef struct vrna\\_pk\\_plex\\_option\\_s \\* vrna\\_pk\\_plex\\_opt\\_t](#)  
*RNA PKplex options object.*
- [typedef struct vrna\\_pk\\_plex\\_result\\_s vrna\\_pk\\_plex\\_t](#)  
*Convenience typedef for results of the RNA PKplex prediction.*

### Functions

- [vrna\\_pk\\_plex\\_t \\* vrna\\_pk\\_plex](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, const int \*\*accessibility, [vrna\\_pk\\_plex\\_opt\\_t](#) options)  
*Predict Pseudoknot interactions in terms of a two-step folding process.*
- [int \\*\\* vrna\\_pk\\_plex\\_accessibility](#) (const char \*sequence, unsigned int [unpaired](#), double cutoff)  
*Obtain a list of opening energies suitable for PKplex computations.*
- [vrna\\_pk\\_plex\\_opt\\_t vrna\\_pk\\_plex\\_opt\\_defaults](#) (void)  
*Default options for PKplex algorithm.*
- [vrna\\_pk\\_plex\\_opt\\_t vrna\\_pk\\_plex\\_opt](#) (unsigned int delta, unsigned int max\_interaction\_length, int pk\_↔ penalty)  
*Simple options for PKplex algorithm.*
- [vrna\\_pk\\_plex\\_opt\\_t vrna\\_pk\\_plex\\_opt\\_fun](#) (unsigned int delta, unsigned int max\_interaction\_length, [vrna\\_callback\\_pk\\_plex\\_score](#) \*scoring\_function, void \*scoring\_data)  
*Simple options for PKplex algorithm.*

### 18.183.1 Detailed Description

Heuristics for two-step pseudoknot forming interaction predictions.

## 18.184 pk\_plex.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PK_PLEX_H
2 #define VIENNA_RNA_PACKAGE_PK_PLEX_H
3
45 typedef int (vrna_callback_pk_plex_score)(const short *pt,
46                                         int start_5,
47                                         int end_5,
48                                         int start_3,
49                                         int end_3,
50                                         void *data);
51
58 typedef struct vrna_pk_plex_option_s *vrna_pk_plex_opt_t;
59
65 typedef struct vrna_pk_plex_result_s vrna_pk_plex_t;
66
67 #include <ViennaRNA/fold_compound.h>
68
74 struct vrna_pk_plex_result_s {
75     char *structure;
76     double energy;
77     double dGpk;
78     double dGint;
79     double dG1;
80     double dG2;
81     unsigned int start_5;
82     unsigned int end_5;
83     unsigned int start_3;
84     unsigned int end_3;
85 };
86
116 vrna_pk_plex_t *
117 vrna_pk_plex(vrna_fold_compound_t *fc,
118                const int **accessibility,
119                vrna_pk_plex_opt_t options);
120
121
132 int **
133 vrna_pk_plex_accessibility(const char *sequence,
134                             unsigned int unpaired,
135                             double cutoff);
136
137
145 vrna_pk_plex_opt_t
146 vrna_pk_plex_opt_defaults(void);
147
148
159 vrna_pk_plex_opt_t
160 vrna_pk_plex_opt(unsigned int delta,
161                     unsigned int max_interaction_length,
162                     int pk_penalty);
163
164
176 vrna_pk_plex_opt_t
177 vrna_pk_plex_opt_fun(unsigned int delta,
178                        unsigned int max_interaction_length,
179                        vrna_callback_pk_plex_score *scoring_function,
180                        void *scoring_data);
181
182
187 #endif

```

## 18.185 PKplex.h

```

1 #ifndef VIENNA_RNA_PACKAGE_PKPLEX_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_PKPLEX_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 # warning "Including deprecated header file <ViennaRNA/PKplex.h>! Use <ViennaRNA/pk_plex.h> instead!"
13 # endif
14
15 #ifdef VRNA_WARN_DEPRECATED
16 # if defined(__clang__)
17 #  define DEPRECATED(func, msg) func __attribute__((deprecated("", msg)))
18 # elif defined(__GNUC__)
19 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
20 # else
21 #  define DEPRECATED(func, msg) func
22 # endif
23 #else
24 # define DEPRECATED(func, msg) func
25 #endif

```

```

26
27 #include <ViennaRNA/datastructures/basic.h>
28
29
30 DEPRECATED(dupVar *
31 PKLduplexfold_XS(const char *s1,
32                     const int **access_s1,
33                     int penalty,
34                     int max_interaction_length,
35                     int delta),
36                     "Use vrna_pk_plex() instead!");
37
38 #include <ViennaRNA/pk_plex.h>
39
40 #endif
41
42 #endif

```

## 18.186 plex.h

```

1 #ifndef VIENNA_RNA_PACKAGE_PLEX_H
2 #define VIENNA_RNA_PACKAGE_PLEX_H
3
4 #include <ViennaRNA/datastructures/basic.h>
5
6 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
7
8 extern int subopt_sorted;
9
13 duplexT** Lduplexfold(const char *s1,
14                         const char *s2,
15                         const int threshold,
16                         const int extension_cost,
17                         const int alignment_length,
18                         const int delta,
19                         const int fast,
20                         const int il_a,
21                         const int il_b,
22                         const int b_a,
23                         const int b_b);
24
28 duplexT** Lduplexfold_XS( const char*s1,
29                            const char* s2,
30                            const int **access_s1,
31                            const int **access_s2,
32                            const int threshold,
33                            const int delta,
34                            const int alignment_length,
35                            const int fast,
36                            const int il_a,
37                            const int il_b,
38                            const int b_a,
39                            const int b_b); /* , const int target_dead, const int query_dead); */
40
44 duplexT** Lduplexfold_C(const char *s1,
45                         const char *s2,
46                         const int threshold,
47                         const int extension_cost,
48                         const int alignment_length,
49                         const int delta,
50                         const int fast,
51                         const char* structure,
52                         const int il_a,
53                         const int il_b,
54                         const int b_a,
55                         const int b_b);
56
61 duplexT** Lduplexfold_CXS(const char*s1,
62                            const char* s2,
63                            const int **access_s1,
64                            const int **access_s2,
65                            const int threshold,
66                            const int delta,
67                            const int alignment_length,
68                            const int fast,
69                            const char* structure,
70                            const int il_a,
71                            const int il_b,
72                            const int b_a,
73                            const int b_b); /* , const int target_dead, const int query_dead); */
74
75
76
77
78 int      arraySize(duplexT** array);

```

```

79 void      freeDuplexT(duplexT** array);
80
81 #endif
82
83 #endif

```

## 18.187 ViennaRNA/plot\_aln.h File Reference

Use [ViennaRNA/plotting/alignments.h](#) instead.

Include dependency graph for plot\_aln.h:

### 18.187.1 Detailed Description

Use [ViennaRNA/plotting/alignments.h](#) instead.

**Deprecated** Use [ViennaRNA/plotting/alignments.h](#) instead

## 18.188 plot\_aln.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PLOT_ALN_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_PLOT_ALN_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 # ifdef VRNA_WARN_DEPRECATED
6 # warning "Including deprecated header file <ViennaRNA/plot_aln.h>! Use <ViennaRNA/plotting/alignments.h>
7 instead!"
8 # endif
9 #include <ViennaRNA/plotting/alignments.h>
10 #endif
11
12 #endif
13
14
15
16
17

```

## 18.189 ViennaRNA/plot\_layouts.h File Reference

Use [ViennaRNA/plotting/layouts.h](#) instead.

Include dependency graph for plot\_layouts.h:

### 18.189.1 Detailed Description

Use [ViennaRNA/plotting/layouts.h](#) instead.

**Deprecated** Use [ViennaRNA/plotting/layouts.h](#) instead

## 18.190 plot\_layouts.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PLOT_LAYOUTS_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_PLOT_LAYOUTS_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 # ifdef VRNA_WARN_DEPRECATED
6 # warning "Including deprecated header file <ViennaRNA/plot_layouts.h>! Use <ViennaRNA/plotting/layouts.h>
7 instead!"
8 # endif
9 #include <ViennaRNA/plotting/layouts.h>
10 #endif
11
12 #endif
13
14
15
16
17

```

## 18.191 ViennaRNA/plot\_structure.h File Reference

Use [ViennaRNA/plotting/structures.h](#) instead.

Include dependency graph for plot\_structure.h:

### 18.191.1 Detailed Description

Use [ViennaRNA/plotting/structures.h](#) instead.

**Deprecated** Use [ViennaRNA/plotting/structures.h](#) instead

## 18.192 plot\_structure.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PLOT_STRUCTURE_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_PLOT_STRUCTURE_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 #warning "Including deprecated header file <ViennaRNA/plot_structure.h>! Use
13 # <ViennaRNA/plotting/structures.h> instead!"
14 #endif
15 #endif
16
17 #endif

```

## 18.193 ViennaRNA/plot\_utils.h File Reference

Use [ViennaRNA/plotting/utils.h](#) instead.

Include dependency graph for plot\_utils.h:

### 18.193.1 Detailed Description

Use [ViennaRNA/plotting/utils.h](#) instead.

**Deprecated** Use [ViennaRNA/plotting/utils.h](#) instead

## 18.194 plot\_utils.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PLOT_UTILS_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_PLOT_UTILS_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 #warning "Including deprecated header file <ViennaRNA/plot_utils.h>! Use <ViennaRNA/plotting/utils.h>
13 # instead!"
14 #endif
15 #include <ViennaRNA/plotting/utils.h>
16 #endif
17 #endif

```

## 18.195 ViennaRNA/plotting alignments.h File Reference

Various functions for plotting Sequence / Structure Alignments.

This graph shows which files directly or indirectly include this file:

### Functions

- int [vrna\\_file\\_PS\\_aln](#) (const char \*filename, const char \*\*seqs, const char \*\*names, const char \*structure, unsigned int columns)
 

*Create an annotated PostScript alignment plot.*
- int [vrna\\_file\\_PS\\_aln\\_slice](#) (const char \*filename, const char \*\*seqs, const char \*\*names, const char \*structure, unsigned int start, unsigned int end, int offset, unsigned int columns)
 

*Create an annotated PostScript alignment plot.*
- int [PS\\_color\\_aln](#) (const char \*structure, const char \*filename, const char \*seqs[], const char \*names[])

*Produce PostScript sequence alignment color-annotated by consensus structure.*

- int `aliPS_color_aln` (const char \*structure, const char \*filename, const char \*seqs[], const char \*names[])
   
*PS\_color\_aln for duplexes.*

### 18.195.1 Detailed Description

Various functions for plotting Sequence / Structure Alignments.

## 18.196 alignments.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PLOT_ALN_H
2 #define VIENNA_RNA_PACKAGE_PLOT_ALN_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 # define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 # define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 # define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
16 int
17 vrna_file_PS_aln(const char    *filename,
18                   const char    **seqs,
19                   const char    **names,
20                   const char    *structure,
21                   unsigned int  columns);
22
23 int
24 vrna_file_PS_aln_slice(const char    *filename,
25                        const char    **seqs,
26                        const char    **names,
27                        const char    *structure,
28                        unsigned int  start,
29                        unsigned int  end,
30                        int          offset,
31                        unsigned int  columns);
32
33 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
34
35 DEPRECATED(int PS_color_aln(const char    *structure,
36                           const char    *filename,
37                           const char    *seqs[],
38                           const char    *names[]),
39             "Use vrna_file_PS_aln() instead!");
40
41 DEPRECATED(int aliPS_color_aln(const char    *structure,
42                           const char    *filename,
43                           const char    *seqs[],
44                           const char    *names[]),
45             "Use vrna_file_PS_aln() instead!");
46
47 #endif
48 #endif
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110

```

## 18.197 ViennaRNA/utils alignments.h File Reference

Various utility- and helper-functions for sequence alignments and comparative structure prediction.

Include dependency graph for alignments.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct `vrna_pinfo_s`
  
*A base pair info structure.* [More...](#)

## Macros

- `#define VRNA_ALN_DEFAULT 0U`  
*Use default alignment settings.*
- `#define VRNA_ALN_RNA 1U`  
*Convert to RNA alphabet.*
- `#define VRNA_ALN_DNA 2U`  
*Convert to DNA alphabet.*
- `#define VRNA_ALN_UPPERCASE 4U`  
*Convert to uppercase nucleotide letters.*
- `#define VRNA_ALN_LOWERCASE 8U`  
*Convert to lowercase nucleotide letters.*
- `#define VRNA_MEASURE_SHANNON_ENTROPY 1U`  
*Flag indicating Shannon Entropy measure.*

## Typedefs

- `typedef struct vrna_pinfo_s vrna_pinfo_t`  
*Typename for the base pair info representing data structure `vrna_pinfo_s`.*
- `typedef struct vrna_pinfo_s pair_info`  
*Old typename of `vrna_pinfo_s`.*

## Functions

- `int vrna_aln_mpi (const char **alignment)`  
*Get the mean pairwise identity in steps from ?to?(ident)*
- `vrna_pinfo_t * vrna_aln_pinfo (vrna_fold_compound_t *vc, const char *structure, double threshold)`  
*Retrieve an array of `vrna_pinfo_t` structures from precomputed pair probabilities.*
- `char ** vrna_aln_slice (const char **alignment, unsigned int i, unsigned int j)`  
*Slice out a subalignment from a larger alignment.*
- `void vrna_aln_free (char **alignment)`  
*Free memory occupied by a set of aligned sequences.*
- `char ** vrna_aln_uppercase (const char **alignment)`  
*Create a copy of an alignment with only uppercase letters in the sequences.*
- `char ** vrna_aln_toRNA (const char **alignment)`  
*Create a copy of an alignment where DNA alphabet is replaced by RNA alphabet.*
- `char ** vrna_aln_copy (const char **alignment, unsigned int options)`  
*Make a copy of a multiple sequence alignment.*
- `float * vrna_aln_conservation_struct (const char **alignment, const char *structure, const vrna_md_t *md)`  
*Compute base pair conservation of a consensus structure.*
- `float * vrna_aln_conservation_col (const char **alignment, const vrna_md_t *md_p, unsigned int options)`  
*Compute nucleotide conservation in an alignment.*
- `char * vrna_aln_consensus_sequence (const char **alignment, const vrna_md_t *md_p)`  
*Compute the consensus sequence for a given multiple sequence alignment.*
- `char * vrna_aln_consensus_mis (const char **alignment, const vrna_md_t *md_p)`  
*Compute the Most Informative Sequence (MIS) for a given multiple sequence alignment.*
- `int get_mpi (char *Alseq[], int n_seq, int length, int *mini)`  
*Get the mean pairwise identity in steps from ?to?(ident)*
- `void encode_ali_sequence (const char *sequence, short *S, short *s5, short *s3, char *ss, unsigned short *as, int circ)`  
*Get arrays with encoded sequence of the alignment.*

- void `alloc_sequence_arrays` (const char \*\*sequences, short \*\*\*S, short \*\*\*S5, short \*\*\*S3, unsigned short \*\*\*a2s, char \*\*\*Ss, int circ)
 

*Allocate memory for sequence array used to deal with aligned sequences.*
- void `free_sequence_arrays` (unsigned int n\_seq, short \*\*\*S, short \*\*\*S5, short \*\*\*S3, unsigned short \*\*\*a2s, char \*\*\*Ss)
 

*Free the memory of the sequence arrays used to deal with aligned sequences.*

## 18.197.1 Detailed Description

Various utility- and helper-functions for sequence alignments and comparative structure prediction.

## 18.198 alignments.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_ALN_UTIL_H
2 #define VIENNA_RNA_PACKAGE_ALN_UTIL_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 # define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 # define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 # define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
29 typedef struct vrna_pinfo_s vrna_pinfo_t;
30
31
35 #define VRNA_ALN_DEFAULT      0U
36
37
41 #define VRNA_ALN_RNA         1U
42
43
47 #define VRNA_ALN_DNA         2U
48
49
53 #define VRNA_ALN_UPPERCASE   4U
54
55
59 #define VRNA_ALN_LOWERCASE   8U
60
66 #define VRNA_MEASURE_SHANNON_ENTROPY 1U
67
68 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
69
70 /* the following typedefs are for backward compatibility only */
71
77 typedef struct vrna_pinfo_s pair_info;
78
79 #endif
80
81 #include <ViennaRNA/fold_compound.h>
82 #include <ViennaRNA/model.h>
83
94 struct vrna_pinfo_s {
95     unsigned i;
96     unsigned j;
97     float p;
98     float ent;
99     short bp[8];
100    char comp;
101 };
102
103
110 int
111 vrna_aln_mpi(const char **alignment);
112
113
127 vrna_pinfo_t *
128 vrna_aln_pinfo(vrna_fold_compound_t *vc,
129                 const char           *structure,
130                 double                threshold);

```

```

131
132
133 int *
134 vrna_aln_pscore(const char **alignment,
135                 vrna_md_t *md);
136
137
138 int
139 vrna_pscore(vrna_fold_compound_t *fc,
140             unsigned int i,
141             unsigned int j);
142
143
144 int
145 vrna_pscore_freq(vrna_fold_compound_t *fc,
146                   const unsigned int *frequencies,
147                   unsigned int pairs);
148
149 char **
150 vrna_aln_slice(const char **alignment,
151                 unsigned int i,
152                 unsigned int j);
153
154 void
155 vrna_aln_free(char **alignment);
156
157
158 char **
159 vrna_aln_uppercase(const char **alignment);
160
161
162 char **
163 vrna_aln_toRNA(const char **alignment);
164
165
166
167
168
169 char **
170 vrna_aln_copy(const char **alignment,
171                 unsigned int options);
172
173
174 float *
175 vrna_aln_conservation_struct(const char **alignment,
176                               const char *structure,
177                               const vrna_md_t *md);
178
179
180 float *
181 vrna_aln_conservation_col(const char **alignment,
182                            const vrna_md_t *md_p,
183                            unsigned int options);
184
185
186 char *
187 vrna_aln_consensus_sequence(const char **alignment,
188                             const vrna_md_t *md_p);
189
190
191 char *
192 vrna_aln_consensus_mis(const char **alignment,
193                         const vrna_md_t *md_p);
194
195
196 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
197
198 #include <stdio.h>
199 DEPRECATED(int read_clustal(FILE *clust,
200                           char *AlignedSeqs[],
201                           char *names[]),
202             "Use vrna_file_msa_read() and vrna_file_msa_read_record() instead");
203
204 DEPRECATED(char *consensus(const char *AS[],
205                           "Use vrna_aln_consensus_sequence() instead!");
206
207 DEPRECATED(char *consens_mis(const char *AS[],
208                           "Use vrna_aln_consensus_mis() instead!");
209
210 DEPRECATED(char *get_ungapped_sequence(const char *seq,
211                           "Use vrna_seq_ungapped() instead!");
212
213 DEPRECATED(int get_mpi(char *Alseq[],
214                           int n_seq,
215                           int length,
216                           int *mini),
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336

```

```

337     "Use vrna_aln_mpi() instead");
338
339 /*
340 ##### some helper functions that might be useful in the library #
341 #####
342 */
343
344
360 DEPRECATED(void encode_ali_sequence(const char      *sequence,
361             short          *S,
362             short          *s5,
363             short          *s3,
364             char           *ss,
365             unsigned short *as,
366             int            circ),
367     "This function is obsolete");
368
369
386 DEPRECATED(void alloc_sequence_arrays(const char    **sequences,
387             short          ***S,
388             short          ***S5,
389             short          ***S3,
390             unsigned short ***a2s,
391             char           ***Ss,
392             int            circ),
393     "This function is obsolete");
394
395
411 DEPRECATED(void free_sequence_arrays(unsigned int   n_seq,
412             short          ***S,
413             short          ***S5,
414             short          ***S3,
415             unsigned short ***a2s,
416             char           ***Ss),
417     "This fucntion is obsolete");
418
419 #endif
420
426 #endif

```

## 18.199 ViennaRNA/plotting/layouts.h File Reference

Secondary structure plot layout algorithms.

Include dependency graph for layouts.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [vrna\\_plot\\_layout\\_s](#)
- struct [COORDINATE](#)

*this is a workaround for the SWIG Perl Wrapper RNA plot function that returns an array of type [COORDINATE](#) More...*

### Macros

- #define [VRNA\\_PLOT\\_TYPE\\_SIMPLE](#) 0  
*Definition of Plot type simple*
- #define [VRNA\\_PLOT\\_TYPE\\_NAVIEW](#) 1  
*Definition of Plot type Naview*
- #define [VRNA\\_PLOT\\_TYPE\\_CIRCULAR](#) 2  
*Definition of Plot type Circular*
- #define [VRNA\\_PLOT\\_TYPE\\_TURTLE](#) 3  
*Definition of Plot type Turtle [24].*
- #define [VRNA\\_PLOT\\_TYPE\\_PUZZLER](#) 4  
*Definition of Plot type RNApuzzler [24].*

### Typedefs

- typedef struct [vrna\\_plot\\_layout\\_s](#) [vrna\\_plot\\_layout\\_t](#)  
*RNA secondary structure figure layout.*

## Functions

- `vrna_plot_layout_t * vrna_plot_layout` (const char \*structure, unsigned int plot\_type)  
*Create a layout (coordinates, etc.) for a secondary structure plot.*
- `vrna_plot_layout_t * vrna_plot_layout_simple` (const char \*structure)  
*Create a layout (coordinates, etc.) for a simple secondary structure plot.*
- `vrna_plot_layout_t * vrna_plot_layout_circular` (const char \*structure)  
*Create a layout (coordinates, etc.) for a circular secondary structure plot.*
- `vrna_plot_layout_t * vrna_plot_layout_turtle` (const char \*structure)  
*Create a layout (coordinates, etc.) for a secondary structure plot using the Turtle Algorithm [24].*
- `vrna_plot_layout_t * vrna_plot_layout_puzzler` (const char \*structure, `vrna_plot_options_puzzler_t` \*options)  
*Create a layout (coordinates, etc.) for a secondary structure plot using the RNAPuzzler Algorithm [24].*
- `void vrna_plot_layout_free` (`vrna_plot_layout_t` \*layout)  
*Free memory occupied by a figure layout data structure.*
- `int vrna_plot_coords` (const char \*structure, float \*\*x, float \*\*y, int plot\_type)  
*Compute nucleotide coordinates for secondary structure plot.*
- `int vrna_plot_coords_pt` (const short \*pt, float \*\*x, float \*\*y, int plot\_type)  
*Compute nucleotide coordinates for secondary structure plot.*
- `int vrna_plot_coords_simple` (const char \*structure, float \*\*x, float \*\*y)  
*Compute nucleotide coordinates for secondary structure plot the Simple way*
- `int vrna_plot_coords_simple_pt` (const short \*pt, float \*\*x, float \*\*y)  
*Compute nucleotide coordinates for secondary structure plot the Simple way*
- `int vrna_plot_coords_circular` (const char \*structure, float \*\*\*x, float \*\*y)  
*Compute coordinates of nucleotides mapped in equal distancies onto a unit circle.*
- `int vrna_plot_coords_circular_pt` (const short \*pt, float \*\*x, float \*\*y)  
*Compute nucleotide coordinates for a Circular Plot*
- `int simple_xy_coordinates` (short \*pair\_table, float \*X, float \*Y)  
*Calculate nucleotide coordinates for secondary structure plot the Simple way*
- `int simple_circplot_coordinates` (short \*pair\_table, float \*x, float \*y)  
*Calculate nucleotide coordinates for Circular Plot*

## Variables

- `int rna_plot_type`  
*Switch for changing the secondary structure layout algorithm.*

### 18.199.1 Detailed Description

Secondary structure plot layout algorithms.

## 18.200 layouts.h

Go to the documentation of this file.

```

1 #ifndef VIENNA_RNA_PACKAGE_PLOT_LAYOUTS_H
2 #define VIENNA_RNA_PACKAGE_PLOT_LAYOUTS_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 #  define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
36 typedef struct vrna_plot_layout_s vrna_plot_layout_t;

```

```
37
38
39 #include <ViennaRNA/datastructures/basic.h>
40
41 #ifdef VRNA_WITH_NAVIEW_LAYOUT
42 #include <ViennaRNA/plotting/naview/naview.h>
43 #endif
44
45 #include "ViennaRNA/plotting/RNApuzzler/RNAturtle.h"
46 #include "ViennaRNA/plotting/RNApuzzler/RNApuzzler.h"
47
48
57 #define VRNA_PLOT_TYPE_SIMPLE      0
58
67 #define VRNA_PLOT_TYPE_NAVIEW     1
68
77 #define VRNA_PLOT_TYPE_CIRCULAR   2
78
83 #define VRNA_PLOT_TYPE_TURTLE    3
84
89 #define VRNA_PLOT_TYPE_PUZZLER   4
90
91 #ifdef VRNA_WITH_NAVIEW_LAYOUT
92 # define VRNA_PLOT_TYPE_DEFAULT  VRNA_PLOT_TYPE_NAVIEW
93 #else
94 # define VRNA_PLOT_TYPE_DEFAULT  VRNA_PLOT_TYPE_PUZZLER
95 #endif
96
97
98 struct vrna_plot_layout_s {
99     unsigned int length;
100    float      *x;
101    float      *y;
102    double     *arcs;
103    int        bbox[4];
104 };
105
106
133 vrna_plot_layout_t *
134 vrna_plot_layout(const char  *structure,
135                    unsigned int plot_type);
136
137
153 vrna_plot_layout_t *
154 vrna_plot_layout_simple(const char *structure);
155
156
157 #ifdef VRNA_WITH_NAVIEW_LAYOUT
173 vrna_plot_layout_t *
174 vrna_plot_layout_naview(const char *structure);
175 #endif
176
192 vrna_plot_layout_t *
193 vrna_plot_layout_circular(const char *structure);
194
195
211 vrna_plot_layout_t *
212 vrna_plot_layout_turtle(const char *structure);
213
214
230 vrna_plot_layout_t *
231 vrna_plot_layout_puzzler(const char           *structure,
232                           vrna_plot_options_puzzler_t *options);
233
234
244 void
245 vrna_plot_layout_free(vrna_plot_layout_t *layout);
246
247
291 int
292 vrna_plot_coords(const char  *structure,
293                     float      **x,
294                     float      **y,
295                     int        plot_type);
296
297
317 int
318 vrna_plot_coords_pt(const short  *pt,
319                      float      **x,
320                      float      **y,
321                      int        plot_type);
322
323
356 int
357 vrna_plot_coords_simple(const char  *structure,
358                           float      **x,
359                           float      **y);
```

```

360
361
380 int
381 vrna_plot_coords_simple_pt(const short *pt,
382                     float ***x,
383                     float ***y);
384
385
416 int
417 vrna_plot_coords_circular(const char *structure,
418                     float ***x,
419                     float ***y);
420
421
440 int
441 vrna_plot_coords_circular_pt(const short *pt,
442                     float ***x,
443                     float ***y);
444
445
451 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
452
463 typedef struct {
464     float X; /* X coords */
465     float Y; /* Y coords */
466 } COORDINATE;
467
468
481 extern int rna_plot_type;
482
483
497 DEPRECATED(int
498             simple_xy_coordinates(short *pair_table,
499                         float *X,
500                         float *Y),
501             "Use vrna_plot_coords_simple_pt() instead!");
502
503
526 DEPRECATED(int
527             simple_circplot_coordinates(short *pair_table,
528                         float *x,
529                         float *y),
530             "Use vrna_plot_coords_circular_pt() instead!");
531
532
537 #endif
538
539
540 #endif

```

## 18.201 ViennaRNA/plotting/probabilities.h File Reference

Various functions for plotting RNA secondary structures, dot-plots and other visualizations.

Include dependency graph for probabilities.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct `vrna_dotplot_auxdata_t`

### Functions

- int `PS_dot_plot_list` (char \*seq, char \*filename, `vrna_ep_t` \*pl, `vrna_ep_t` \*mf, char \*comment)  
*Produce a postscript dot-plot from two pair lists.*
- int `PS_dot_plot` (char \*string, char \*file)  
*Produce postscript dot-plot.*

#### 18.201.1 Detailed Description

Various functions for plotting RNA secondary structures, dot-plots and other visualizations.

## 18.202 probabilities.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_PLOT_PROBABILITIES_H
2 #define VIENNA_RNA_PACKAGE_PLOT_PROBABILITIES_H
3
4
5 #include <ViennaRNA/datastructures/basic.h>
6 #include <ViennaRNA/utils/structures.h>
7
8 #ifdef VRNA_WARN_DEPRECATED
9 # if defined(__clang__)
10 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
11 # elif defined(__GNUC__)
12 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
13 # else
14 #  define DEPRECATED(func, msg) func
15 # endif
16 #else
17 # define DEPRECATED(func, msg) func
18 #endif
19
32 #define VRNA_PLOT_PROBABILITIES_BP      1U
33 #define VRNA_PLOT_PROBABILITIES_ACC     2U
34
35 #define VRNA_PLOT_PROBABILITIES_UD      4U
36 #define VRNA_PLOT_PROBABILITIES_UD_LIN  8U
37
38 #define VRNA_PLOT_PROBABILITIES_SD      16U
39
40 #define VRNA_PLOT_PROBABILITIES_SC_MOTIF 32U
41 #define VRNA_PLOT_PROBABILITIES_SC_UP   64U
42 #define VRNA_PLOT_PROBABILITIES_SC_BP  128U
43
44 #define VRNA_PLOT_PROBABILITIES_DEFAULT  (VRNA_PLOT_PROBABILITIES_BP \
45                                         | VRNA_PLOT_PROBABILITIES_SD \
46                                         | VRNA_PLOT_PROBABILITIES_SC_MOTIF \
47                                         | VRNA_PLOT_PROBABILITIES_UD_LIN)
48
49 typedef struct {
50     char           *comment;
51     char           *title;
52     vrna_data_lin_t **top;
53     char           **top_title;
54
55     vrna_data_lin_t **bottom;
56     char           **bottom_title;
57
58     vrna_data_lin_t **left;
59     char           **left_title;
60
61     vrna_data_lin_t **right;
62     char           **right_title;
63 } vrna_dotplot_auxdata_t;
64
65
66 int
67 vrna_plot_dp_EPS(const char          *filename,
68                   const char          *sequence,
69                   vrna_ep_t            *upper,
70                   vrna_ep_t            *lower,
71                   vrna_dotplot_auxdata_t *auxdata,
72                   unsigned int         options);
73
74
75 int
76 vrna_plot_dp_PS_list(char        *seq,
77                       int          cp,
78                       char        *wastlfile,
79                       vrna_ep_t    *pl,
80                       vrna_ep_t    *mf,
81                       char        *comment);
82
83
84 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
85
86 int
87 PS_color_dot_plot(char        *string,
88                    vrna_cpair_t *pi,
89                    char        *filename);
90
91
92 int
93 PS_color_dot_plot_turn(char       *seq,
94                        vrna_cpair_t *pi,
95                        char        *filename,
96                        int          winSize);
97
98
99 int
```

```

100 PS_dot_plot_turn(char      *seq,
101           vrna_ep_t   *pl,
102           char        *filename,
103           int         winSize);
104
105
125 int PS_dot_plot_list(char      *seq,
126           char        *filename,
127           vrna_ep_t   *pl,
128           vrna_ep_t   *mf,
129           char        *comment);
130
131
147 DEPRECATED(int PS_dot_plot(char *string,
148           char *file),
149           "Use vRNA_plot_dp_EPS() instead");
150
151 #endif
152
157 #endif

```

## 18.203 ViennaRNA/plotting/RNApuzzler/RNApuzzler.h File Reference

Implementation of the RNApuzzler RNA secondary structure layout algorithm [24].

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [vrna\\_plot\\_options\\_puzzler\\_t](#)  
*Options data structure for RNApuzzler algorithm implementation.* [More...](#)

### Functions

- int [vrna\\_plot\\_coords\\_puzzler](#) (const char \*structure, float \*\*x, float \*\*y, double \*\*arc\_coords, [vrna\\_plot\\_options\\_puzzler\\_t](#) \*options)  
*Compute nucleotide coordinates for secondary structure plot using the RNApuzzler algorithm [24].*
- int [vrna\\_plot\\_coords\\_puzzler\\_pt](#) (short const \*const pair\_table, float \*\*x, float \*\*y, double \*\*arc\_coords, [vrna\\_plot\\_options\\_puzzler\\_t](#) \*puzzler)  
*Compute nucleotide coordinates for secondary structure plot using the RNApuzzler algorithm [24].*
- [vrna\\_plot\\_options\\_puzzler\\_t](#) \* [vrna\\_plot\\_options\\_puzzler](#) (void)  
*Create an RNApuzzler options data structure.*
- void [vrna\\_plot\\_options\\_puzzler\\_free](#) ([vrna\\_plot\\_options\\_puzzler\\_t](#) \*options)  
*Free memory occupied by an RNApuzzler options data structure.*

### 18.203.1 Detailed Description

Implementation of the RNApuzzler RNA secondary structure layout algorithm [24].

## 18.204 RNApuzzler.h

[Go to the documentation of this file.](#)

```

1 #ifndef RNAPUZZLER_H
2 #define RNAPUZZLER_H
3
20 typedef struct {
21     /*
22     * variables fixed during operation
23     * drawing behavior
24     */
25     short      drawArcs;
26     double     paired;
27     double     unpaired;
28
29     /* intersection resolution behavior */
30     short      checkAncestorIntersections;
31     short      checkSiblingIntersections;
32     short      checkExteriorIntersections;

```

```

33     short      allowFlipping;
34     short      optimize;
35     int        maximumNumberOfConfigChangesAllowed;
36
37
38     /* import behavior - unused for now */
39     char       *config; /* file path */
40
41     /* other stuff */
42     const char *filename;
43
44     /* variables changed during operation */
45     int         numberOfChangesAppliedToConfig;
46     int         psNumber;
47 } vrna_plot_options_puzzler_t;
48
49
50     int
51     vrna_plot_coords_puzzler(const char           *structure,
52                               float            **x,
53                               float            **y,
54                               double           **arc_coords,
55                               vrna_plot_options_puzzler_t *options);
56
57
58     int
59     vrna_plot_coords_puzzler_pt(short const *const   pair_table,
60                                  float            **x,
61                                  float            **y,
62                                  double           **arc_coords,
63                                  vrna_plot_options_puzzler_t *puzzler);
64
65
66     vrna_plot_options_puzzler_t *
67     vrna_plot_options_puzzler(void);
68
69
70     void
71     vrna_plot_options_puzzler_free(vrna_plot_options_puzzler_t *options);
72
73
74 #endif

```

## 18.205 ViennaRNA/plotting/RNApuzzler/RNAturtle.h File Reference

Implementation of the RNAturtle RNA secondary structure layout algorithm [24].

This graph shows which files directly or indirectly include this file:

### Functions

- int [vrna\\_plot\\_coords\\_turtle](#) (const char \*structure, float \*\*x, float \*\*y, double \*\*arc\_coords)  
*Compute nucleotide coordinates for secondary structure plot using the RNAturtle algorithm [24].*
- int [vrna\\_plot\\_coords\\_turtle\\_pt](#) (short const \*const pair\_table, float \*\*x, float \*\*y, double \*\*arc\_coords)  
*Compute nucleotide coordinates for secondary structure plot using the RNAturtle algorithm [24].*

### 18.205.1 Detailed Description

Implementation of the RNAturtle RNA secondary structure layout algorithm [24].

## 18.206 RNAturtle.h

[Go to the documentation of this file.](#)

```

1 #ifndef RNATURTLE_H
2 #define RNATURTLE_H
3
4     int
5     vrna_plot_coords_turtle(const char  *structure,
6                           float      **x,
7                           float      **y,
8                           double     **arc_coords);
9
10
11     int
12     vrna_plot_coords_turtle_pt(short const *const pair_table,
13                                float      **x,
14                                float      **y,
15                                double     **arc_coords),
16
17
18     vrna_plot_coords_turtle_pt(short const *const pair_table,
19                                float      **x,
20                                float      **y,
21                                double     **arc_coords);
22
23
24     vrna_plot_coords_turtle_pt(short const *const pair_table,
25                                float      **x,
26                                float      **y,
27                                double     **arc_coords);
28
29
30     vrna_plot_coords_turtle_pt(short const *const pair_table,
31                                float      **x,
32                                float      **y,
33                                double     **arc_coords);
34
35
36     vrna_plot_coords_turtle_pt(short const *const pair_table,
37                                float      **x,
38                                float      **y,
39                                double     **arc_coords);
40
41
42     vrna_plot_coords_turtle_pt(short const *const pair_table,
43                                float      **x,
44                                float      **y,
45                                double     **arc_coords);
46
47
48     vrna_plot_coords_turtle_pt(short const *const pair_table,
49                                float      **x,
50                                float      **y,
51                                double     **arc_coords);
52
53
54     vrna_plot_coords_turtle_pt(short const *const pair_table,
55                                float      **x,
56                                float      **y,
57                                double     **arc_coords);
58
59
60     vrna_plot_coords_turtle_pt(short const *const pair_table,
61                                float      **x,
62                                float      **y,
63                                double     **arc_coords);
64
65
66     vrna_plot_coords_turtle_pt(short const *const pair_table,
67                                float      **x,
68                                float      **y,
69                                double     **arc_coords);
70
71
72     vrna_plot_coords_turtle_pt(short const *const pair_table,
73                                float      **x,
74                                float      **y,
75                                double     **arc_coords);
76
77
78     vrna_plot_coords_turtle_pt(short const *const pair_table,
79                                float      **x,
80                                float      **y,
81                                double     **arc_coords);
82
83
84     vrna_plot_coords_turtle_pt(short const *const pair_table,
85                                float      **x,
86                                float      **y,
87                                double     **arc_coords);
88
89
90     vrna_plot_coords_turtle_pt(short const *const pair_table,
91                                float      **x,
92                                float      **y,
93                                double     **arc_coords);
94
95
96     vrna_plot_coords_turtle_pt(short const *const pair_table,
97                                float      **x,
98                                float      **y,
99                                double     **arc_coords);
100
101
102     vrna_plot_coords_turtle_pt(short const *const pair_table,
103                                float      **x,
104                                float      **y,
105                                double     **arc_coords);
106
107
108     vrna_plot_coords_turtle_pt(short const *const pair_table,
109                                float      **x,
110                                float      **y,
111                                double     **arc_coords);
112
113
114     vrna_plot_coords_turtle_pt(short const *const pair_table,
115                                float      **x,
116                                float      **y,
117                                double     **arc_coords);
118
119
120     vrna_plot_coords_turtle_pt(short const *const pair_table,
121                                float      **x,
122                                float      **y,
123                                double     **arc_coords);
124
125
126     vrna_plot_coords_turtle_pt(short const *const pair_table,
127                                float      **x,
128                                float      **y,
129                                double     **arc_coords);
130
131
132     vrna_plot_coords_turtle_pt(short const *const pair_table,
133                                float      **x,
134                                float      **y,
135                                double     **arc_coords);
136
137
138     vrna_plot_coords_turtle_pt(short const *const pair_table,
139                                float      **x,
140                                float      **y,
141                                double     **arc_coords);
142
143
144     vrna_plot_coords_turtle_pt(short const *const pair_table,
145                                float      **x,
146                                float      **y,
147                                double     **arc_coords);
148
149
150     vrna_plot_coords_turtle_pt(short const *const pair_table,
151                                float      **x,
152                                float      **y,
153                                double     **arc_coords);
154
155
156     vrna_plot_coords_turtle_pt(short const *const pair_table,
157                                float      **x,
158                                float      **y,
159                                double     **arc_coords);
160
161
162     vrna_plot_coords_turtle_pt(short const *const pair_table,
163                                float      **x,
164                                float      **y,
165                                double     **arc_coords);
166
167
168     vrna_plot_coords_turtle_pt(short const *const pair_table,
169                                float      **x,
170                                float      **y,
171                                double     **arc_coords);
172
173
174     vrna_plot_coords_turtle_pt(short const *const pair_table,
175                                float      **x,
176                                float      **y,
177                                double     **arc_coords);
178
179
180     vrna_plot_coords_turtle_pt(short const *const pair_table,
181                                float      **x,
182                                float      **y,
183                                double     **arc_coords);
184
185
186     vrna_plot_coords_turtle_pt(short const *const pair_table,
187                                float      **x,
188                                float      **y,
189                                double     **arc_coords);
190
191
192     vrna_plot_coords_turtle_pt(short const *const pair_table,
193                                float      **x,
194                                float      **y,
195                                double     **arc_coords);
196
197
198     vrna_plot_coords_turtle_pt(short const *const pair_table,
199                                float      **x,
200                                float      **y,
201                                double     **arc_coords);
202
203
204     vrna_plot_coords_turtle_pt(short const *const pair_table,
205                                float      **x,
206                                float      **y,
207                                double     **arc_coords);
208
209
210     vrna_plot_coords_turtle_pt(short const *const pair_table,
211                                float      **x,
212                                float      **y,
213                                double     **arc_coords);
214
215
216     vrna_plot_coords_turtle_pt(short const *const pair_table,
217                                float      **x,
218                                float      **y,
219                                double     **arc_coords);
220
221
222     vrna_plot_coords_turtle_pt(short const *const pair_table,
223                                float      **x,
224                                float      **y,
225                                double     **arc_coords);
226
227
228     vrna_plot_coords_turtle_pt(short const *const pair_table,
229                                float      **x,
230                                float      **y,
231                                double     **arc_coords);
232
233
234     vrna_plot_coords_turtle_pt(short const *const pair_table,
235                                float      **x,
236                                float      **y,
237                                double     **arc_coords);
238
239
240     vrna_plot_coords_turtle_pt(short const *const pair_table,
241                                float      **x,
242                                float      **y,
243                                double     **arc_coords);
244
245
246     vrna_plot_coords_turtle_pt(short const *const pair_table,
247                                float      **x,
248                                float      **y,
249                                double     **arc_coords);
250
251
252     vrna_plot_coords_turtle_pt(short const *const pair_table,
253                                float      **x,
254                                float      **y,
255                                double     **arc_coords);
256
257
258     vrna_plot_coords_turtle_pt(short const *const pair_table,
259                                float      **x,
260                                float      **y,
261                                double     **arc_coords);
262
263
264     vrna_plot_coords_turtle_pt(short const *const pair_table,
265                                float      **x,
266                                float      **y,
267                                double     **arc_coords);
268
269
270     vrna_plot_coords_turtle_pt(short const *const pair_table,
271                                float      **x,
272                                float      **y,
273                                double     **arc_coords);
274
275
276     vrna_plot_coords_turtle_pt(short const *const pair_table,
277                                float      **x,
278                                float      **y,
279                                double     **arc_coords);
280
281
282     vrna_plot_coords_turtle_pt(short const *const pair_table,
283                                float      **x,
284                                float      **y,
285                                double     **arc_coords);
286
287
288     vrna_plot_coords_turtle_pt(short const *const pair_table,
289                                float      **x,
290                                float      **y,
291                                double     **arc_coords);
292
293
294     vrna_plot_coords_turtle_pt(short const *const pair_table,
295                                float      **x,
296                                float      **y,
297                                double     **arc_coords);
298
299
299 #endif

```

```

80           float          ***x,
81           float          ***y,
82           double         **arc_coords);
83
84
89 #endif

```

## 18.207 ViennaRNA/plotting/structures.h File Reference

Various functions for plotting RNA secondary structures.

Include dependency graph for structures.h: This graph shows which files directly or indirectly include this file:

### Functions

- int [vrna\\_file\\_PS\\_rnplot](#) (const char \*seq, const char \*structure, const char \*file, [vrna\\_md\\_t](#) \*md\_p)  
*Produce a secondary structure graph in PostScript and write it to 'filename'.*
- int [vrna\\_file\\_PS\\_rnplot\\_a](#) (const char \*seq, const char \*structure, const char \*file, const char \*pre, const char \*post, [vrna\\_md\\_t](#) \*md\_p)  
*Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename'.*
- int [gmlRNA](#) (char \*string, char \*structure, char \*ssfile, char option)  
*Produce a secondary structure graph in Graph Meta Language (gml) and write it to a file.*
- int [ssv\\_rna\\_plot](#) (char \*string, char \*structure, char \*ssfile)  
*Produce a secondary structure graph in SStructView format.*
- int [svg\\_rna\\_plot](#) (char \*string, char \*structure, char \*ssfile)  
*Produce a secondary structure plot in SVG format and write it to a file.*
- int [xrna\\_plot](#) (char \*string, char \*structure, char \*ssfile)  
*Produce a secondary structure plot for further editing in XRNA.*
- int [PS\\_rna\\_plot](#) (char \*string, char \*structure, char \*file)  
*Produce a secondary structure graph in PostScript and write it to 'filename'.*
- int [PS\\_rna\\_plot\\_a](#) (char \*string, char \*structure, char \*file, char \*pre, char \*post)  
*Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename'.*
- int [PS\\_rna\\_plot\\_a\\_gquad](#) (char \*string, char \*structure, char \*ssfile, char \*pre, char \*post)  
*Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename' (detect and draw g-quadruplexes)*

### 18.207.1 Detailed Description

Various functions for plotting RNA secondary structures.

## 18.208 structures.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PLOT_STRUCTURE_H
2 #define VIENNA_RNA_PACKAGE_PLOT_STRUCTURE_H
3
4 #include <ViennaRNA/model.h>
5 #include <ViennaRNA/plotting/layouts.h>
6 #include "ViennaRNA/plotting/RNApuzzler/RNApuzzler.h"
7
8 #ifdef VRNA_WARN_DEPRECATED
9 # if defined(__clang__)
10 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
11 # elif defined(__GNUC__)
12 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
13 # else
14 #  define DEPRECATED(func, msg) func
15 # endif
16 #else
17 # define DEPRECATED(func, msg) func
18 #endif
19
44 int
45 vrna\_file\_PS\_rnplot(const char *seq,

```

```

46         const char *structure,
47         const char *file,
48         vrna_md_t *md_p);
49
50
70 int vrna_file_PS_rnplot_a( const char      *seq,
71                           const char      *structure,
72                           const char      *file,
73                           const char      *pre,
74                           const char      *post,
75                           vrna_md_t       *md_p);
76
77
78 int
79 vrna_file_PS_rnplot_layout(const char      *seq,
80                           const char      *structure,
81                           const char      *ssfile,
82                           const char      *pre,
83                           const char      *post,
84                           vrna_md_t       *md_p,
85                           vrna_plot_layout_t *layout);
86
87 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
88
89 /* write PostScript drawing of structure to file with annotation */
90 int
91 PS_rna_plot_snoop_a(const char  *string,
92                      const char  *structure,
93                      const char  *ssfile,
94                      int          *relative_access,
95                      const char  *seqs[]);
96
97
110 int
111 gmlRNA(char *string,
112         char *structure,
113         char *ssfile,
114         char option);
115
116
127 int
128 ssv_rna_plot(char *string,
129                char *structure,
130                char *ssfile);
131
132
141 int
142 svg_rna_plot(char *string,
143                char *structure,
144                char *ssfile);
145
146
155 int
156 xrna_plot(char *string,
157             char *structure,
158             char *ssfile);
159
160
166 DEPRECATED(int PS_rna_plot(char *string,
167                  char *structure,
168                  char *file),
169                  "Use vrna_file_PS_rnplot() instead");
170
177 DEPRECATED(int PS_rna_plot_a(char *string,
178                  char *structure,
179                  char *file,
180                  char *pre,
181                  char *post),
182                  "Use vrna_file_PS_rnplot_a() instead");
183
190 DEPRECATED(int PS_rna_plot_a_gquad(char *string,
191                  char *structure,
192                  char *ssfile,
193                  char *pre,
194                  char *post),
195                  "Use vrna_file_PS_rnplot_a() instead");
196
197 #endif
198
203 #endif

```

## 18.209 ViennaRNA/utils/structures.h File Reference

Various utility- and helper-functions for secondary structure parsing, converting, etc.

Include dependency graph for structures.h: This graph shows which files directly or indirectly include this file:

## Data Structures

- struct [vrna\\_elem\\_prob\\_s](#)  
*Data structure representing a single entry of an element probability list (e.g. list of pair probabilities)* [More...](#)
- struct [vrna\\_hx\\_s](#)  
*Data structure representing an entry of a helix list.* [More...](#)

## Macros

- [#define VRNA\\_BRACKETS\\_ALPHA 4U](#)  
*Bitflag to indicate secondary structure notations using uppercase/lowercase letters from the latin alphabet.*
- [#define VRNA\\_BRACKETS\\_RND 8U](#)  
*Bitflag to indicate secondary structure notations using round brackets (parenthesis), ()*
- [#define VRNA\\_BRACKETS\\_CLY 16U](#)  
*Bitflag to indicate secondary structure notations using curly brackets, {}*
- [#define VRNA\\_BRACKETS\\_ANG 32U](#)  
*Bitflag to indicate secondary structure notations using angular brackets, <>*
- [#define VRNA\\_BRACKETS\\_SQR 64U](#)  
*Bitflag to indicate secondary structure notations using square brackets, []*
- [#define VRNA\\_BRACKETS\\_DEFAULT](#)  
*Default bitmask to indicate secondary structure notation using any pair of brackets.*
- [#define VRNA\\_BRACKETS\\_ANY](#)  
*Bitmask to indicate secondary structure notation using any pair of brackets or uppercase/lowercase alphabet letters.*
- [#define VRNA\\_PLIST\\_TYPE\\_BASEPAIR 0](#)  
*A Base Pair element.*
- [#define VRNA\\_PLIST\\_TYPE\\_GQUAD 1](#)  
*A G-Quadruplex element.*
- [#define VRNA\\_PLIST\\_TYPE\\_H\\_MOTIF 2](#)  
*A Hairpin loop motif element.*
- [#define VRNA\\_PLIST\\_TYPE\\_I\\_MOTIF 3](#)  
*An Internal loop motif element.*
- [#define VRNA\\_PLIST\\_TYPE\\_UD\\_MOTIF 4](#)  
*An Unstructured Domain motif element.*
- [#define VRNA\\_PLIST\\_TYPE\\_STACK 5](#)  
*A Base Pair stack element.*
- [#define VRNA\\_PLIST\\_TYPE\\_UNPAIRED 6](#)  
*An unpaired base.*
- [#define VRNA\\_PLIST\\_TYPE\\_TRIPLE 7](#)  
*One pair of a base triplet.*
- [#define VRNA\\_STRUCTURE\\_TREE\\_HIT 1U](#)  
*Homeomorphically Irreducible Tree (HIT) representation of a secondary structure.*
- [#define VRNA\\_STRUCTURE\\_TREE\\_SHAPIRO\\_SHORT 2U](#)  
*(short) Coarse Grained representation of a secondary structure*
- [#define VRNA\\_STRUCTURE\\_TREE\\_SHAPIRO 3U](#)  
*(full) Coarse Grained representation of a secondary structure*
- [#define VRNA\\_STRUCTURE\\_TREE\\_SHAPIRO\\_EXT 4U](#)  
*(extended) Coarse Grained representation of a secondary structure*
- [#define VRNA\\_STRUCTURE\\_TREE\\_SHAPIRO\\_WEIGHT 5U](#)  
*(weighted) Coarse Grained representation of a secondary structure*
- [#define VRNA\\_STRUCTURE\\_TREE\\_EXPANDED 6U](#)  
*Expanded Tree representation of a secondary structure.*

## Typedefs

- **typedef struct vrna\_hx\_s vrna\_hx\_t**  
*Convenience typedef for data structure vrna\_hx\_s.*
- **typedef struct vrna\_elem\_prob\_s vrna\_ep\_t**  
*Convenience typedef for data structure vrna\_elem\_prob\_s.*

## Functions

- **char \* vrna\_db\_pack (const char \*struc)**  
*Pack secondary structure, 5:1 compression using base 3 encoding.*
- **char \* vrna\_db\_unpack (const char \*packed)**  
*Unpack secondary structure previously packed with vrna\_db\_pack()*
- **void vrna\_db\_flatten (char \*structure, unsigned int options)**  
*Substitute pairs of brackets in a string with parenthesis.*
- **void vrna\_db\_flatten\_to (char \*string, const char target[3], unsigned int options)**  
*Substitute pairs of brackets in a string with another type of pair characters.*
- **char \* vrna\_db\_from\_ptable (const short \*pt)**  
*Convert a pair table into dot-parenthesis notation.*
- **char \* vrna\_db\_from plist (vrna\_ep\_t \*pairs, unsigned int n)**  
*Convert a list of base pairs into dot-bracket notation.*
- **char \* vrna\_db\_to\_element\_string (const char \*structure)**  
*Convert a secondary structure in dot-bracket notation to a nucleotide annotation of loop contexts.*
- **char \* vrna\_db\_pk\_remove (const char \*structure, unsigned int options)**  
*Remove pseudo-knots from an input structure.*
- **short \* vrna\_ptable (const char \*structure)**  
*Create a pair table from a dot-bracket notation of a secondary structure.*
- **short \* vrna\_ptable\_from\_string (const char \*structure, unsigned int options)**  
*Create a pair table for a secondary structure string.*
- **short \* vrna\_pt\_pk\_get (const char \*structure)**  
*Create a pair table of a secondary structure (pseudo-knot version)*
- **short \* vrna\_pt\_copy (const short \*pt)**  
*Get an exact copy of a pair table.*
- **short \* vrna\_pt\_ali\_get (const char \*structure)**  
*Create a pair table of a secondary structure (snoop align version)*
- **short \* vrna\_pt\_snoop\_get (const char \*structure)**  
*Create a pair table of a secondary structure (snoop version)*
- **short \* vrna\_pt\_pk\_remove (const short \*ptable, unsigned int options)**  
*Remove pseudo-knots from a pair table.*
- **vrna\_ep\_t \* vrna\_plist (const char \*struc, float pr)**  
*Create a vrna\_ep\_t from a dot-bracket string.*
- **vrna\_ep\_t \* vrna plist\_from\_probs (vrna\_fold\_compound\_t \*vc, double cut\_off)**  
*Create a vrna\_ep\_t from base pair probability matrix.*
- **char \* vrna\_db\_from\_WUSS (const char \*wuss)**  
*Convert a WUSS annotation string to dot-bracket format.*
- **char \* vrna\_abstract\_shapes (const char \*structure, unsigned int level)**  
*Convert a secondary structure in dot-bracket notation to its abstract shapes representation.*
- **char \* vrna\_abstract\_shapes\_pt (const short \*pt, unsigned int level)**  
*Convert a secondary structure to its abstract shapes representation.*
- **vrna\_hx\_t \* vrna\_hx\_from\_ptable (short \*pt)**  
*Convert a pair table representation of a secondary structure into a helix list.*
- **vrna\_hx\_t \* vrna\_hx\_merge (const vrna\_hx\_t \*list, int maxdist)**

- `int *vrna_loopidx_from_ptable (const short *pt)`

*Create a merged helix list from another helix list.*
- `int vrna_bp_distance_pt (const short *pt1, const short *pt2)`

*Get a loop index representation of a structure.*
- `int vrna_bp_distance (const char *str1, const char *str2)`

*Compute the "base pair" distance between two pair tables pt1 and pt2 of secondary structures.*
- `unsigned int *vrna_refBPCnt_matrix (const short *reference_pt, unsigned int turn)`

*Make a reference base pair count matrix.*
- `unsigned int *vrna_refBPdist_matrix (const short *pt1, const short *pt2, unsigned int turn)`

*Make a reference base pair distance matrix.*
- `char *vrna_db_from_probs (const FLT_OR_DBL *pr, unsigned int length)`

*Create a dot-bracket like structure string from base pair probability matrix.*
- `char vrna_bpp_symbol (const float *x)`

*Get a pseudo dot bracket notation for a given probability information.*
- `char *vrna_db_from_bp_stack (vrna_bp_stack_t *bp, unsigned int length)`

*Create a dot-backet/parenthesis structure from backtracking stack.*
- `char *vrna_db_to_tree_string (const char *structure, unsigned int type)`

*Convert a Dot-Bracket structure string into tree string representation.*
- `char *vrna_tree_string_unweight (const char *structure)`

*Remove weights from a linear string tree representation of a secondary structure.*
- `char *vrna_tree_string_to_db (const char *tree)`

*Convert a linear tree string representation of a secondary structure back to Dot-Bracket notation.*
- `void assign plist_from_db (vrna_ep_t **pl, const char *struc, float pr)`

*Create a `vrna_ep_t` from a dot-bracket string.*
- `char *pack_structure (const char *struc)`

*Pack secondary secondary structure, 5:1 compression using base 3 encoding.*
- `char *unpack_structure (const char *packed)`

*Unpack secondary structure previously packed with `pack_structure()`*
- `short *make_pair_table (const char *structure)`

*Create a pair table of a secondary structure.*
- `short *copy_pair_table (const short *pt)`

*Get an exact copy of a pair table.*
- `short *alimake_pair_table (const char *structure)`
- `short *make_pair_table_snoop (const char *structure)`
- `int bp_distance (const char *str1, const char *str2)`

*Compute the "base pair" distance between two secondary structures s1 and s2.*
- `unsigned int *make_referenceBP_array (short *reference_pt, unsigned int turn)`

*Make a reference base pair count matrix.*
- `unsigned int *compute_BPdifferences (short *pt1, short *pt2, unsigned int turn)`

*Make a reference base pair distance matrix.*
- `void assign plist_from_pr (vrna_ep_t **pl, FLT_OR_DBL *probs, int length, double cutoff)`

*Create a `vrna_ep_t` from a probability matrix.*
- `void parenthesis_structure (char *structure, vrna_bp_stack_t *bp, int length)`

*Create a dot-backet/parenthesis structure from backtracking stack.*
- `void parenthesis_zuker (char *structure, vrna_bp_stack_t *bp, int length)`

*Create a dot-backet/parenthesis structure from backtracking stack obtained by zuker suboptimal calculation in cofold.c.*
- `void bppm_to_structure (char *structure, FLT_OR_DBL *pr, unsigned int length)`

*Create a dot-bracket like structure string from base pair probability matrix.*
- `char bppm_symbol (const float *x)`

*Get a pseudo dot bracket notation for a given probability information.*

### 18.209.1 Detailed Description

Various utility- and helper-functions for secondary structure parsing, converting, etc.

## 18.210 structures.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_STRUCT_UTILS_H
2 #define VIENNA_RNA_PACKAGE_STRUCT_UTILS_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 #  define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
16
17 #define vrna_hx_s vrna_hx_t;
18
19
20 #define vrna_elem_prob_s vrna_ep_t;
21
22
23 #define VRNA_BRACKETS_ALPHA      4U
24
25
26 #define VRNA_BRACKETS_RND       8U
27
28
29 #define VRNA_BRACKETS_CLY      16U
30
31
32 #define VRNA_BRACKETS_ANG      32U
33
34
35 #define VRNA_BRACKETS_SQR      64U
36
37
38 #define VRNA_BRACKETS_DEFAULT \
39   (VRNA_BRACKETS_RND | \
40    VRNA_BRACKETS_CLY | \
41    VRNA_BRACKETS_ANG | \
42    VRNA_BRACKETS_SQR)
43
44
45 #define VRNA_BRACKETS_ANY \
46   (VRNA_BRACKETS_RND | \
47    VRNA_BRACKETS_CLY | \
48    VRNA_BRACKETS_ANG | \
49    VRNA_BRACKETS_SQR | \
50    VRNA_BRACKETS_ALPHA)
51
52
53 #include <stdio.h>
54
55 #include <ViennaRNA/datastructures/basic.h>
56
57 char *
58 vrna_db_pack(const char *struc);
59
60
61 char *
62 vrna_db_unpack(const char *packed);
63
64
65 void
66 vrna_db_flatten(char          *structure,
67                  unsigned int options);
68
69
70 void
71 vrna_db_flatten_to(char          *string,
72                      const char  target[3],
73                      unsigned int options);
74
75
76 char *
77 vrna_db_from_ptable(const short *pt);
```

```
239 char *
240 vrna_db_from_plist(vrna_ep_t      *pairs,
241                      unsigned int n);
252
253
260 char *
261 vrna_db_to_element_string(const char *structure);
262
263
295 char *
296 vrna_db_pk_remove(const char *structure,
297                      unsigned int options);
298
299 /* End dot-bracket interface */
318 short *
319 vrna_ptable(const char *structure);
320
321
341 short *
342 vrna_ptable_from_string(const char    *structure,
343                           unsigned int  options);
344
345
363 short *
364 vrna_pt_pk_get(const char *structure);
365
366
373 short *
374 vrna_ptable_copy(const short *pt);
375
376
381 short *
382 vrna_pt_ali_get(const char *structure);
383
384
392 short *
393 vrna_pt_snoop_get(const char *structure);
394
395
412 short *
413 vrna_pt_pk_remove(const short   *ptable,
414                      unsigned int options);
415
416
417 /* End pair table interface */
429 #define VRNA_PLIST_TYPE_BASEPAIR      0
430
431
435 #define VRNA_PLIST_TYPE_GQUAD        1
436
437
441 #define VRNA_PLIST_TYPE_H_MOTIF      2
442
443
447 #define VRNA_PLIST_TYPE_I_MOTIF      3
448
449
453 #define VRNA_PLIST_TYPE_UD_MOTIF     4
454
455
459 #define VRNA_PLIST_TYPE_STACK        5
460
461
465 #define VRNA_PLIST_TYPE_UNPAIRED     6
466
467
471 #define VRNA_PLIST_TYPE_TRIPLE       7
472
473
482 struct vrna_elem_prob_s {
483     int    i;
484     int    j;
485     float p;
486     int    type;
487 };
488
504 vrna_ep_t *vrna plist(const char  *struc,
505                         float      pr);
506
507
524 vrna_ep_t *vrna plist_from_probs(vrna_fold_compound_t *vc,
525                                     double      cut_off);
526
527
528 /* End pair list interface */
596 char *
```

```
597 vrna_db_from_WUSS(const char *wuss);
598
599
600 /* End WUSS notation interface */
648 char *
649 vrna_abstract_shapes(const char      *structure,
650                      unsigned int   level);
651
652
668 char *
669 vrna_abstract_shapes_pt(const short   *pt,
670                          unsigned int   level);
671
672
673 /* End abstract shapes interface */
685 struct vrna_hx_s {
686     unsigned int start;
687     unsigned int end;
688     unsigned int length;
689     unsigned int up5;
690     unsigned int up3;
691 };
692
693
700 vrna_hx_t *
701 vrna_hx_from_ptable(short *pt);
702
703
707 vrna_hx_t *
708 vrna_hx_merge(const vrna_hx_t *list,
709                  int           maxdist);
710
711
712 /* End helix list interface */
719 int *
720 vrna_loopidx_from_ptable(const short *pt);
721
722
741 int
742 vrna_bp_distance_pt(const short *pt1,
743                      const short *pt2);
744
759 int
760 vrna_bp_distance(const char *str1,
761                     const char *str2);
762
763
764 double
765 vrna_dist_mountain(const char  *str1,
766                      const char  *str2,
767                      unsigned int p);
768
769
770 /* End metrics interface */
779 unsigned int *
780 vrna_refBPcnt_matrix(const short  *reference_pt,
781                      unsigned int turn);
782
783
791 unsigned int *
792 vrna_refBPDist_matrix(const short   *pt1,
793                      const short   *pt2,
794                      unsigned int  turn);
795
796
800 char *
801 vrna_db_from_probs(const FLT_OR_DBL *pr,
802                      unsigned int   length);
803
804
808 char
809 vrna_bpp_symbol(const float **x);
810
811
823 char *
824 vrna_db_from_bp_stack(vrna_bp_stack_t *bp,
825                         unsigned int   length);
826
827
828 void
829 vrna_letter_structure(char      *structure,
830                        vrna_bp_stack_t *bp,
831                        unsigned int   length);
832
833
919 #define VRNA_STRUCTURE_TREE_HIT 1U
920
```

```

921
926 #define VRNA_STRUCTURE_TREE_SHAPIRO_SHORT 2U
927
928
933 #define VRNA_STRUCTURE_TREE_SHAPIRO 3U
934
935
940 #define VRNA_STRUCTURE_TREE_SHAPIRO_EXT 4U
941
942
947 #define VRNA_STRUCTURE_TREE_SHAPIRO_WEIGHT 5U
948
953 #define VRNA_STRUCTURE_TREE_EXPANDED 6U
954
955
985 char *
986 vrna_db_to_tree_string(const char *structure,
987                         unsigned int type);
988
989
1001 char *
1002 vrna_tree_string_unweight(const char *structure);
1003
1004
1016 char *
1017 vrna_tree_string_to_db(const char *tree);
1018
1019
1020 /* End tree representations */
1023 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
1024
1025 /*#####
1026 /*# deprecated functions below
1027 #####*/
1028
1048 DEPRECATED(void assign plist_from_db(vrna_ep_t **pl,
1049                           const char *struc,
1050                           float pr),
1051               "Use vrna plist() instead");
1052
1066 DEPRECATED(char *pack_structure(const char *struc),
1067               "Use vrna_db_pack() instead");
1068
1080 DEPRECATED(char *unpack_structure(const char *packed),
1081               "Use vrna_db_unpack() instead");
1082
1095 DEPRECATED(short *make_pair_table(const char *structure),
1096               "Use vrna_ptable() instead");
1097
1098 DEPRECATED(short *make_pair_table_pk(const char *structure),
1099               "Use vrna_ptable_from_string() instead");
1100
1110 DEPRECATED(short *copy_pair_table(const short *pt),
1111               "Use vrna_ptable_copy() instead");
1112
1119 DEPRECATED(short *alimake_pair_table(const char *structure),
1120               "Use vrna_pt_ali_get() instead");
1121
1129 DEPRECATED(short *make_pair_table_snoop(const char *structure),
1130               "Use vrna_pt_snoop_get() instead");
1131
1132 DEPRECATED(int *make_loop_index_pt(short *pt),
1133               "Use vrna_loopidx_from_ptable() instead");
1134
1148 DEPRECATED(int bp_distance(const char *str1,
1149                           const char *str2),
1150               "Use vrna_bp_distance() instead");
1151
1161 DEPRECATED(unsigned int *make_referenceBP_array(short *reference_pt,
1162                           unsigned int turn),
1163               "Use vrna_refBPCnt_matrix() instead");
1164
1174 DEPRECATED(unsigned int *compute_BPdifferences(short *pt1,
1175                           short *pt2,
1176                           unsigned int turn),
1177               "Use vrna_refBPdist_matrix() instead");
1178
1199 DEPRECATED(void assign plist_from_pr(vrna_ep_t **pl,
1200                           FLT_OR_DBL *probs,
1201                           int length,
1202                           double cutoff),
1203               "Use vrna plist_from_probs() instead");
1204
1213 DEPRECATED(void parenthesis_structure(char *structure,
1214                           vrna_bp_stack_t *bp,
1215                           int length),
1216               "Use vrna_parenthesis_structure() instead");

```

```

1217 DEPRECATED(void parenthesis_zuker(char           *structure,
1218                           vrna_bp_stack_t *bp,
1219                           int              length),
1220                           "Use vrna_parenthesis_zuker() instead");
1221
1222 DEPRECATED(void letter_structure(char           *structure,
1223                           vrna_bp_stack_t *bp,
1224                           int              length),
1225                           "Use vrna_letter_structure() instead");
1226
1227 DEPRECATED(void bppm_to_structure(char           *structure,
1228                           FLT_OR_DBL   *pr,
1229                           unsigned int  length),
1230                           "Use vrna_db_from_probs() instead");
1231
1232 DEPRECATED(char    bppm_symbol(const float *x),
1233                           "Use vrna_bpp_symbol() instead");
1234
1235 #endif
1236
1237 #endif

```

## 18.211 ProfileAln.h

```

1 #ifndef VIENNA_RNA_PACKAGE_PROFILEALN_H
2 #define VIENNA_RNA_PACKAGE_PROFILEALN_H
3
4 float profile_aln(const float *T1,
5                   const char  *seq1,
6                   const float *T2,
7                   const char  *seq2);
8
9
10 int set_palm_params(double  gap_open,
11                      double  gap_ext,
12                      double  seqweight,
13                      int     free_ends);
14
15
16 #endif

```

## 18.212 ViennaRNA/profiledist.h File Reference

Include dependency graph for profiledist.h:

### Functions

- float **profile\_edit\_distance** (const float \*T1, const float \*T2)  
*Align the 2 probability profiles T1, T2*
- float \* **Make\_bp\_profile\_bppm** (FLT\_OR\_DBL \*bppm, int length)  
*condense pair probability matrix into a vector containing probabilities for unpaired, upstream paired and downstream paired.*
- void **print\_bppm** (const float \*T)  
*print string representation of probability profile*
- void **free\_profile** (float \*T)  
*free space allocated in Make\_bp\_profile*
- float \* **Make\_bp\_profile** (int length)

### 18.212.1 Detailed Description

### 18.212.2 Function Documentation

### 18.212.2.1 profile\_edit\_distance()

```
float profile_edit_distance (
    const float * T1,
    const float * T2 )
```

Align the 2 probability profiles T1, T2

This is like a Needleman-Wunsch alignment, we should really use affine gap-costs ala Gotoh

### 18.212.2.2 Make\_bp\_profile\_bppm()

```
float * Make_bp_profile_bppm (
    FLT_OR_DBL * bppm,
    int length )
```

condense pair probability matrix into a vector containing probabilities for unpaired, upstream paired and downstream paired.

This resulting probability profile is used as input for profile\_edit\_distance

#### Parameters

|               |   |
|---------------|---|
| <i>bppm</i>   | A pointer to the base pair probability matrix |
| <i>length</i> | The length of the sequence                    |

#### Returns

The bp profile

### 18.212.2.3 free\_profile()

```
void free_profile (
    float * T )
```

free space allocated in Make\_bp\_profile

Backward compatibility only. You can just use plain free()

### 18.212.2.4 Make\_bp\_profile()

```
float * Make_bp_profile (
    int length )
```

#### Note

This function is NOT threadsafe

#### See also

[Make\\_bp\\_profile\\_bppm\(\)](#)

**Deprecated** This function is deprecated and will be removed soon! See [Make\\_bp\\_profile\\_bppm\(\)](#) for a replacement

## 18.213 profiledist.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_PROFILEDIST_H
2 #define VIENNA_RNA_PACKAGE_PROFILEDIST_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 #   define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 #   define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
```

```

10 # define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
16 #include <ViennaRNA/datastructures/basic.h>
17
20 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
21
28 float profile_edit_distance(const float *T1,
29                             const float *T2);
30
31
42 float *Make_bp_profile_bppm(FLT_OR_DBL *bppm,
43                             int length);
44
45
49 void print_bppm(const float *T);
50
51
57 void free_profile(float *T);
58
59
68 DEPRECATED(float *Make_bp_profile(int length),
69 "Use Make_bp_profile_bppm() instead");
70
71 #endif
72
73 #endif

```

## 18.214 ViennaRNA/PS\_dot.h File Reference

Use [ViennaRNA/plotting/probabilities.h](#) instead.

Include dependency graph for PS\_dot.h:

### 18.214.1 Detailed Description

Use [ViennaRNA/plotting/probabilities.h](#) instead.

**Deprecated** Use [ViennaRNA/plotting/probabilities.h](#) instead

## 18.215 PS\_dot.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PLOT_PROBABILITIES_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_PLOT_PROBABILITIES_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 # warning "Including deprecated header file <ViennaRNA/PS_dot.h>! Use <ViennaRNA/plotting/probabilities.h>
           instead!"
13 # endif
14 #include <ViennaRNA/plotting/probabilities.h>
15 #include <ViennaRNA/plotting/layouts.h>
16 #include <ViennaRNA/plotting/structures.h>
17 #endif
18
19 #endif

```

## 18.216 ViennaRNA/read\_epars.h File Reference

Use [ViennaRNA/params/io.h](#) instead.

Include dependency graph for read\_epars.h:

### 18.216.1 Detailed Description

Use [ViennaRNA/params/io.h](#) instead.

**Deprecated** Use [ViennaRNA/params/io.h](#) instead

## 18.217 read\_epars.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PARAMS_IO_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_PARAMS_IO_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 #warning "Including deprecated header file <ViennaRNA/read_epars.h>! Use <ViennaRNA/params/io.h>
instead!"
13 # endif
14 #include <ViennaRNA/params/io.h>
15 #endif
16
17 #endif

```

## 18.218 ViennaRNA/ribo.h File Reference

Parse RiboSum Scoring Matrices for Covariance Scoring of Alignments.

This graph shows which files directly or indirectly include this file:

### Functions

- float \*\* **get\_ribosum** (const char \*\*Alseq, int n\_seq, int length)  
*Retrieve a RiboSum Scoring Matrix for a given Alignment.*
- float \*\* **readribosum** (char \*name)  
*Read a RiboSum or other user-defined Scoring Matrix and Store into global Memory.*

### 18.218.1 Detailed Description

Parse RiboSum Scoring Matrices for Covariance Scoring of Alignments.

## 18.219 ribo.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_RIBOSUM_H
2 #define VIENNA_RNA_PACKAGE_RIBOSUM_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11
22 float **get_ribosum(const char **Alseq,
23                      int      n_seq,
24                      int      length);
25
26
31 float **readribosum(char *name);
32
33
37 #endif
38
39 #endif

```

## 18.220 ViennaRNA/RNAstruct.h File Reference

Parsing and Coarse Graining of Structures.

### Functions

- char \* **b2HIT** (const char \*structure)  
*Converts the full structure from bracket notation to the HIT notation including root.*
- char \* **b2C** (const char \*structure)  
*Converts the full structure from bracket notation to the a coarse grained notation using the 'H' 'B' 'I' 'M' and 'R' identifiers.*
- char \* **b2Shapiro** (const char \*structure)

- Converts the full structure from bracket notation to the weighted coarse grained notation using the 'H' 'B' 'I' 'M' 'S' 'E' and 'R' identifiers.*
- `char * add_root (const char *structure)`

*Adds a root to an un-rooted tree in any except bracket notation.*
  - `char * expand_Shapiro (const char *coarse)`

*Inserts missing 'S' identifiers in unweighted coarse grained structures as obtained from `b2C()`.*
  - `char * expand_Full (const char *structure)`

*Convert the full structure from bracket notation to the expanded notation including root.*
  - `char * unexpand_Full (const char *ffull)`

*Restores the bracket notation from an expanded full or HIT tree, that is any tree using only identifiers 'U' 'P' and 'R'.*
  - `char * unweight (const char *wcoarse)`

*Strip weights from any weighted tree.*
  - `void unexpand_aligned_F (char *align[2])`

*Converts two aligned structures in expanded notation.*
  - `void parse_structure (const char *structure)`

*Collects a statistic of structure elements of the full structure in bracket notation.*

## Variables

- `int loop_size [2000]`

*contains a list of all loop sizes. `loop_size[0]` contains the number of external bases.*
- `int helix_size [2000]`

*contains a list of all stack sizes.*
- `int loop_degree [2000]`

*contains the corresponding list of loop degrees.*
- `int loops`

*contains the number of loops ( and therefore of stacks ).*
- `int unpaired`

*contains the number of unpaired bases.*
- `int pairs`

*contains the number of base pairs in the last parsed structure.*

### 18.220.1 Detailed Description

Parsing and Coarse Graining of Structures.  
Example:

```
* .(((((....))...)). is the bracket or full tree
* becomes expanded: - expand_Full() -
* ((U) (((U) (U) (((U) (U) (U) P) P) P) (U) ((U) ((U) (U) P) P) P) (U) R)
* HIT: - b2HIT() -
* ((U1) ((U2) ((U3) P3) (U2) ((U2) P2) P2) (U1) R)
* Coarse: - b2C() -
* ((H) ((H) M) R)
* becomes expanded: - expand_Shapiro() -
* (((H) S) ((H) S) M) S) R)
* weighted Shapiro: - b2Shapiro() -
* (((((H3) S3) ((H2) S2) M4) S2) E2) R)
*
```

## 18.221 RNAstruct.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_RNASTRUCT_H
2 #define VIENNA_RNA_PACKAGE_RNASTRUCT_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 # define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))

```

```

7 # elif defined(__GNUC__)
8 # define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 # define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
40 #define STRUC      2000
41
52 DEPRECATED(char *b2HIT(const char *structure),
53             "Use vrna_db_to_tree_string() instead!");           /* Full    -> HIT    [incl. root] */
54
55
66 DEPRECATED(char *b2C(const char *structure),
67             "Use vrna_db_to_tree_string() instead!");           /* Full    -> Coarse [incl. root] */
68
69
81 DEPRECATED(char *b2Shapiro(const char *structure),
82             "Use vrna_db_to_tree_string() instead!");           /* Full -> weighted Shapiro [i.r.] */
83
84
91 DEPRECATED(char *add_root(const char *structure),
92             "");                                /* {Tree} -> ({Tree}R)          */
93
94
102 DEPRECATED(char *expand_Shapiro(const char *coarse),
103            "Use vrna_db_to_tree_string() instead!");
104
105
106 /* add S for stacks to coarse struct */
114 DEPRECATED(char *expand_Full(const char *structure),
115             "Use vrna_db_to_tree_string() instead!");           /* Full    -> FFull        */
116
117
125 DEPRECATED(char *unexpand_Full(const char *ffull),
126             "Use vrna_tree_string_to_db() instead!");           /* FFull   -> Full        */
127
128
135 DEPRECATED(char *unweight(const char *wcoarse),
136             "Use vrna_tree_string_unweight() instead!");         /* remove weights from coarse struct
137             */
138
148 DEPRECATED(void  unexpand_aligned_F(char *align[2]),
149             "");
150
151
161 DEPRECATED(void  parse_structure(const char *structure),
162             ""); /* make structure statistics */
163
164
169 DEPRECATED(extern int loop_size[STRUC],
170             ""); /* loop sizes of a structure */
171
175 DEPRECATED(extern int helix_size[STRUC],
176             ""); /* helix sizes of a structure */
177
181 DEPRECATED(extern int loop_degree[STRUC],
182             ""); /* loop degrees of a structure */
183
187 DEPRECATED(extern int loops,
188             ""); /* n of loops and stacks */
189
193 DEPRECATED(extern int unpaired,
194             "");
195
199 DEPRECATED(extern int pairs,
200             ""); /* n of unpaired digits and pairs */
201
206 #endif

```

## 18.222 ViennaRNA/search/BoyerMoore.h File Reference

Variants of the Boyer-Moore string search algorithm.

### Functions

- const unsigned int \*[vrna\\_search\\_BMH\\_num](#) (const unsigned int \*needle, size\_t needle\_size, const unsigned int \*haystack, size\_t haystack\_size, size\_t start, size\_t \*badchars, unsigned char cyclic)

*Search for a string of elements in a larger string of elements using the Boyer-Moore-Horspool algorithm.*

- `const char * vrna_search_BMH (const char *needle, size_t needle_size, const char *haystack, size_t haystack_size, size_t start, size_t *badchars, unsigned char cyclic)`

*Search for an ASCII pattern within a larger ASCII string using the Boyer-Moore-Horspool algorithm.*

- `size_t * vrna_search_BM_BCT_num (const unsigned int *pattern, size_t pattern_size, unsigned int num_max)`

*Retrieve a Boyer-Moore Bad Character Table for a pattern of elements represented by natural numbers.*

- `size_t * vrna_search_BM_BCT (const char *pattern)`

*Retrieve a Boyer-Moore Bad Character Table for a NULL-terminated pattern of ASCII characters.*

## 18.222.1 Detailed Description

Variants of the Boyer-Moore string search algorithm.

,

## 18.223 BoyerMoore.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_SEARCH_BOYER_MOORE_H
2 #define VIENNA_RNA_PACKAGE_SEARCH_BOYER_MOORE_H
3
46 const unsigned int *
47 vrna_search_BMH_num(const unsigned int *needle,
48                      size_t needle_size,
49                      const unsigned int *haystack,
50                      size_t haystack_size,
51                      size_t start,
52                      size_t *badchars,
53                      unsigned char cyclic);
54
55
56 const char *
57 vrna_search_BMH(const char *needle,
58                  size_t needle_size,
59                  const char *haystack,
60                  size_t haystack_size,
61                  size_t start,
62                  size_t *badchars,
63                  unsigned char cyclic);
64
65
66 size_t *
67 vrna_search_BM_BCT_num(const unsigned int *pattern,
68                        size_t pattern_size,
69                        unsigned int num_max);
70
71
72 size_t *
73 vrna_search_BM_BCT(const char *pattern);
74
75
76
77
78
79
79 #endif

```

## 18.224 ViennaRNA/sequence.h File Reference

Functions and data structures related to sequence representations ,.

Include dependency graph for sequence.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct `vrna_sequence_s`

*Data structure representing a nucleotide sequence. [More...](#)*
- struct `vrna_alignment_s`

## TypeDefs

- `typedef struct vrna_sequence_s vrna_seq_t`

Type name for nucleotide sequence representation data structure `vrna_sequence_s`.

## Enumerations

- enum vrna\_seq\_type\_e { VRNA\_SEQ\_UNKNOWN , VRNA\_SEQ\_RNA , VRNA\_SEQ\_DNA }

A enumerator used in `vrna` sequence `s` to distinguish different nucleotide sequences.

## **18.224.1 Detailed Description**

Functions and data structures related to sequence representations ..

18.225 sequence.h

[Go to the documentation of this file.](#)

```

81 vrna_sequence_remove_all(vrna_fold_compound_t *fc);
82
83
84 void
85 vrna_sequence_prepare(vrna_fold_compound_t *fc);
86
87
88 int
89 vrna_sequence_order_update(vrna_fold_compound_t *fc,
90                             const unsigned int *order);
91
92
93 int
94 vrna_msa_add( vrna_fold_compound_t      *fc,
95                 const char           **alignment,
96                 const char           **names,
97                 const unsigned char   *orientation,
98                 const unsigned long long *start,
99                 const unsigned long long *genome_size,
100                unsigned int          options);
101
102
107 #endif

```

## 18.226 snofold.h

```

1 /* function from fold.c */
2 #ifndef VIENNA_RNA_PACKAGE_SNOFOLD_H
3 #define VIENNA_RNA_PACKAGE_SNOFOLD_H
4
5 #include <ViennaRNA/datastructures/basic.h>
6
7 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
8
9 /* Normal fold */
10
14 int snofold(const char *sequence,
15             char *structure,
16             const int max_assym,
17             const int threshold,
18             const int min_s2,
19             const int max_s2,
20             const int half_stem,
21             const int max_half_stem);
22
23
28 void snofree_arrays(const int length); /* free arrays for mfe folding */
29
30
31 void snoinitialize_fold(int length); /* allocate arrays for folding */
32
33
34 void snoupdate_fold_params(void); /* recalculate parameters */
35
36
37 int snoloop_energy(short *ptable,
38                     short *s,
39                     short *s1,
40                     int i);
41
42
43 void snoexport_fold_arrays(int    **indx_p,
44                            int    **mLoop_p,
45                            int    **cLoop,
46                            folden ***fold_p,
47                            folden ***fold_p_xs);
48
49
50 char *snobacktrack_fold_from_pair(const char *sequence,
51                                     int i,
52                                     int j);
53
54
55 /* alifold */
56 float alisnofold(const char **strings,
57                   const int max_assym,
58                   const int threshloop,
59                   const int min_s2,
60                   const int max_s2,
61                   const int half_stem,
62                   const int max_half_stem);
63
64
65 void alisnofree_arrays(const int length);
66

```

```

67
68 char *alisnobacktrack_fold_from_pair(const char **sequence,
69             int      i,
70             int      j,
71             int      *cov);
72
73
74 extern double cv_fact /* =1 */;
75 extern double nc_fact /* =1 */;
76
77 /* max number of mismatch >>>..(( ))..>> */
78 #define MISMATCH 3
79
80 #endif
81
82 #endif

```

## 18.227 snoop.h

```

1 #ifndef VIENNA_RNA_PACKAGE_SNOOP_H
2 #define VIENNA_RNA_PACKAGE_SNOOP_H
3
4 #include <ViennaRNA/datastructures/basic.h>
5
6 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
7
12 snoopT snoopfold(const char *s1,
13             const char *s2,
14             const int  penalty,
15             const int  threshloop,
16             const int  threshLE,
17             const int  threshRE,
18             const int  threshDE,
19             const int  threshD,
20             const int  half_stem,
21             const int  max_half_stem,
22             const int  min_s2,
23             const int  max_s2,
24             const int  min_s1,
25             const int  max_s1,
26             const int  min_d1,
27             const int  min_d2,
28             const int  fullStemEnergy);
29
30
36 snoopT *snoop_subopt(const char *s1,
37             const char *s2,
38             int      delta,
39             int      w,
40             const int  penalty,
41             const int  threshloop,
42             const int  threshLE,
43             const int  threshRE,
44             const int  threshDE,
45             const int  threshTE,
46             const int  threshSE,
47             const int  threshD,
48             const int  distance,
49             const int  half_stem,
50             const int  max_half_stem,
51             const int  min_s2,
52             const int  max_s2,
53             const int  min_s1,
54             const int  max_s1,
55             const int  min_d1,
56             const int  min_d2,
57             const int  fullStemEnergy);
58
59
65 void Lsnoop_subopt(const char *s1,
66             const char *s2,
67             int      delta,
68             int      w,
69             const int  penalty,
70             const int  threshloop,
71             const int  threshLE,
72             const int  threshRE,
73             const int  threshDE,
74             const int  threshTE,
75             const int  threshSE,
76             const int  threshD,
77             const int  distance,
78             const int  half_stem,
79             const int  max_half_stem,
80             const int  min_s2,

```

```
81         const int max_s2,
82         const int min_s1,
83         const int max_s1,
84         const int min_d1,
85         const int min_d2,
86         const int alignment_length,
87         const char *name,
88         const int fullStemEnergy);
89
90
96 void Lsnoop_subopt_list(const char *s1,
97                         const char *s2,
98                         int delta,
99                         int w,
100                        const int penalty,
101                        const int threshloop,
102                        const int threshLE,
103                        const int threshRE,
104                        const int threshDE,
105                        const int threshTE,
106                        const int threshSE,
107                        const int threshD,
108                        const int distance,
109                        const int half_stem,
110                        const int max_half_stem,
111                        const int min_s2,
112                        const int max_s2,
113                        const int min_s1,
114                        const int max_s1,
115                        const int min_d1,
116                        const int min_d2,
117                        const int alignment_length,
118                        const char *name,
119                        const int fullStemEnergy);
120
121
127 void Lsnoop_subopt_list_XS(const char *s1,
128                           const char *s2,
129                           const int **access_s1,
130                           int delta,
131                           int w,
132                           const int penalty,
133                           const int threshloop,
134                           const int threshLE,
135                           const int threshRE,
136                           const int threshDE,
137                           const int threshTE,
138                           const int threshSE,
139                           const int threshD,
140                           const int distance,
141                           const int half_stem,
142                           const int max_half_stem,
143                           const int min_s2,
144                           const int max_s2,
145                           const int min_s1,
146                           const int max_s1,
147                           const int min_d1,
148                           const int min_d2,
149                           const int alignment_length,
150                           const char *name,
151                           const int fullStemEnergy);
152
153
159 void snoop_subopt_XS(const char *s1,
160                       const char *s2,
161                       const int **access_s1,
162                       int delta,
163                       int w,
164                       const int penalty,
165                       const int threshloop,
166                       const int threshLE,
167                       const int threshRE,
168                       const int threshDE,
169                       const int threshTE,
170                       const int threshSE,
171                       const int threshD,
172                       const int distance,
173                       const int half_stem,
174                       const int max_half_stem,
175                       const int min_s2,
176                       const int max_s2,
177                       const int min_s1,
178                       const int max_s1,
179                       const int min_d1,
180                       const int min_d2,
181                       const int alignment_length,
182                       const char *name,
```

```
183         const int fullStemEnergy);
184
185
190 snoopT *alisnoop_subopt(const char **s1,
191                         const char **s2,
192                         int delta,
193                         int w,
194                         const int penalty,
195                         const int threshloop,
196                         const int threshLE,
197                         const int threshRE,
198                         const int threshDE,
199                         const int threshTE,
200                         const int threshSE,
201                         const int threshD,
202                         const int distance,
203                         const int half_stem,
204                         const int max_half_stem,
205                         const int min_s2,
206                         const int max_s2,
207                         const int min_s1,
208                         const int max_s1,
209                         const int min_d1,
210                         const int min_d2);
211
212
218 snoopT *aliLsnoop_subopt_list(const char **s1,
219                                 const char **s2,
220                                 int delta,
221                                 int w,
222                                 const int penalty,
223                                 const int threshloop,
224                                 const int threshLE,
225                                 const int threshRE,
226                                 const int threshDE,
227                                 const int threshTE,
228                                 const int threshSE,
229                                 const int threshD,
230                                 const int distance,
231                                 const int half_stem,
232                                 const int max_half_stem,
233                                 const int min_s2,
234                                 const int max_s2,
235                                 const int min_s1,
236                                 const int max_s1,
237                                 const int min_d1,
238                                 const int min_d2,
239                                 const int alignment_length);
240
241
247 snoopT alisnooopfold(const char **s1,
248                         const char **s2,
249                         const int penalty,
250                         const int threshloop,
251                         const int threshLE,
252                         const int threshRE,
253                         const int threshDE,
254                         const int threshD,
255                         const int half_stem,
256                         const int max_half_stem,
257                         const int min_s2,
258                         const int max_s2,
259                         const int min_s1,
260                         const int max_s1,
261                         const int min_d1,
262                         const int min_d2);
263
264
269 snoopT snoopfold_XS(const char *s1,
270                       const char *s2,
271                       const int **access_s1,
272                       const int pos,
273                       const int max_pos_j,
274                       const int penalty,
275                       const int threshloop,
276                       const int threshLE,
277                       const int threshRE,
278                       const int threshDE,
279                       const int threshD,
280                       const int half_stem,
281                       const int max_half_stem,
282                       const int min_s2,
283                       const int max_s2,
284                       const int min_s1,
285                       const int max_s1,
286                       const int min_d1,
287                       const int min_d2,
```

```

288             const int    fullStemEnergy);
289
290
291 extern int snoop_subopt_sorted;
292 #endif
293
294 #endif

```

## 18.228 special\_const.h

```

1 extern const char  wall;
2 extern const char  bg;
3 extern const char  star;
4 extern const char  probe;
5 extern const char  start[];
6 extern const char  end[];
7 extern const char  success[];
8 extern const char  injector[];
9 extern unsigned int injector_len;
10 extern const char  flash[];
11 extern const char  head1[];
12 extern const char  head2[];
13 extern const char  l1str[];
14 extern const char  inv[];

```

## 18.229 ViennaRNA/datastructures/stream\_output.h File Reference

An implementation of a buffered, ordered stream output data structure.  
 This graph shows which files directly or indirectly include this file:

### Typedefs

- `typedef struct vrna_ordered_stream_s *vrna_ostream_t`  
*An ordered output stream structure with unordered insert capabilities.*
- `typedef void() vrna_callback_stream_output(void *auxdata, unsigned int i, void *data)`  
*Ordered stream processing callback.*

### Functions

- `vrna_ostream_t vrna_ostream_init (vrna_callback_stream_output *output, void *auxdata)`  
*Get an initialized ordered output stream.*
- `void vrna_ostream_free (vrna_ostream_t dat)`  
*Free an initialized ordered output stream.*
- `void vrna_ostream_request (vrna_ostream_t dat, unsigned int num)`  
*Request index in ordered output stream.*
- `void vrna_ostream_provide (vrna_ostream_t dat, unsigned int i, void *data)`  
*Provide output stream data for a particular index.*

### 18.229.1 Detailed Description

An implementation of a buffered, ordered stream output data structure.

,

## 18.230 stream\_output.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_STREAM_OUTPUT_H
2 #define VIENNA_RNA_PACKAGE_STREAM_OUTPUT_H
3
18 typedef struct vrna_ordered_stream_s *vrna_ostream_t;
19
34 typedef void (vrna_callback_stream_output)(void           *auxdata,
35                                         unsigned int i,
36                                         void         *data);

```

```

37
47 vrna_ostream_t
48 vrna_ostream_init(vrna_callback_stream_output *output,
49             void                         *auxdata);
50
51
59 void
60 vrna_ostream_free(vrna_ostream_t dat);
61
62
63 int
64 vrna_ostream_threadsafe(void);
65
66
79 void
80 vrna_ostream_request(vrna_ostream_t dat,
81             unsigned int num);
82
83
96 void
97 vrna_ostream_provide(vrna_ostream_t dat,
98             unsigned int i,
99             void         *data);
100
101
107 #endif

```

## 18.231 ViennaRNA/stream\_output.h File Reference

Use [ViennaRNA/datastructures/stream\\_output.h](#) instead.

Include dependency graph for stream\_output.h:

### 18.231.1 Detailed Description

Use [ViennaRNA/datastructures/stream\\_output.h](#) instead.

**Deprecated** Use [ViennaRNA/datastructures/stream\\_output.h](#) instead

## 18.232 stream\_output.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_DATA_STRUCTURES_STREAM_OUTPUT_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_DATA_STRUCTURES_STREAM_OUTPUT_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 # warning "Including deprecated header file <ViennaRNA/stream_output.h>! Use
13 #           <ViennaRNA/datastructures/stream_output.h> instead!"
14 # endif
15 #include <ViennaRNA/datastructures/stream_output.h>
16 #endif
17 #endif

```

## 18.233 ViennaRNA/string\_utils.h File Reference

Use [ViennaRNA/utils/strings.h](#) instead.

Include dependency graph for string\_utils.h:

### 18.233.1 Detailed Description

Use [ViennaRNA/utils/strings.h](#) instead.

**Deprecated** Use [ViennaRNA/utils/strings.h](#) instead

## 18.234 string\_utils.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_STRING_UTILS_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_STRING_UTILS_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 # ifdef VRNA_WARN_DEPRECATED
6 # warning "Including deprecated header file <ViennaRNA/string_utils.h>! Use <ViennaRNA/utils/strings.h>
7 instead!"
8 # endif
9 #include <ViennaRNA/utils/strings.h>
10#endif
11
12#endif

```

## 18.235 ViennaRNA/stringdist.h File Reference

Functions for String Alignment.

Include dependency graph for stringdist.h:

### Functions

- `swString * Make_swString (char *string)`  
*Convert a structure into a format suitable for `string_edit_distance()`.*
- `float string_edit_distance (swString *T1, swString *T2)`  
*Calculate the string edit distance of T1 and T2.*

### 18.235.1 Detailed Description

Functions for String Alignment.

### 18.235.2 Function Documentation

#### 18.235.2.1 Make\_swString()

```
swString * Make_swString (
    char * string )
```

Convert a structure into a format suitable for `string_edit_distance()`.

Parameters

|                     |  |
|---------------------|--|
| <code>string</code> |  |
|---------------------|--|

Returns

#### 18.235.2.2 string\_edit\_distance()

```
float string_edit_distance (
    swString * T1,
    swString * T2 )
```

Calculate the string edit distance of T1 and T2.

Parameters

|                 |  |
|-----------------|--|
| <code>T1</code> |  |
| <code>T2</code> |  |

Returns

## 18.236 stringdist.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_STRING_DIST_H
2 #define VIENNA_RNA_PACKAGE_STRING_DIST_H
3
4 #include <ViennaRNA/dist_vars.h>
5
6
7 swString *Make_swString(char *string);
8
9 float     string_edit_distance( swString *T1,
10                                swString *T2);
11
12
13#endif
```

## 18.237 ViennaRNA/structure\_utils.h File Reference

Use [ViennaRNA/utils/structures.h](#) instead.

Include dependency graph for structure\_utils.h:

### 18.237.1 Detailed Description

Use [ViennaRNA/utils/structures.h](#) instead.

**Deprecated** Use [ViennaRNA/utils/structures.h](#) instead

## 18.238 structure\_utils.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_STRUCTURE_UTILS_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_STRUCTURE_UTILS_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 # ifdef VRNA_WARN_DEPRECATED
6 # warning "Including deprecated header file <ViennaRNA/structure_utils.h>! Use
7      <ViennaRNA/utils/structures.h> instead!"
8 # endif
9 #include <ViennaRNA/utils/structures.h>
10
11#endif
```

## 18.239 ViennaRNA/structured\_domains.h File Reference

This module provides interfaces that deal with additional structured domains in the folding grammar.  
This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [vrna\\_structured\\_domains\\_s](#)

### 18.239.1 Detailed Description

This module provides interfaces that deal with additional structured domains in the folding grammar.

## 18.240 structured\_domains.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_STRUCTURAL_DOMAINS_H
```

```

2 #define VIENNA_RNA_PACKAGE_STRUCTURAL_DOMAINS_H
3
23 typedef struct vrna_structured_domains_s vrna_sd_t;
24
25
26 struct vrna_structured_domains_s {
27     char __placeholder; /* dummy placeholder to not leave this struct empty */
28 };
29
30 #endif

```

## 18.241 ViennaRNA/subopt.h File Reference

RNAsubopt and density of states declarations.

Include dependency graph for subopt.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [vrna\\_subopt\\_sol\\_s](#)  
*Solution element from subopt.c.*

### Macros

- #define **MAXDOS** 1000  
*Maximum density of states discretization for subopt.*

### Typedefs

- typedef struct [vrna\\_subopt\\_sol\\_s](#) **vrna\_subopt\_solution\_t**  
*Typename for the subopt solution list repesenting data structure [vrna\\_subopt\\_sol\\_s](#).*
- typedef void() [vrna\\_subopt\\_callback](#)(const char \*structure, float energy, void \*data)  
*Callback for [vrna\\_subopt\\_cb\(\)](#)*
- typedef struct [vrna\\_subopt\\_sol\\_s](#) **SOLUTION**  
*Backward compatibility typedef for [vrna\\_subopt\\_sol\\_s](#).*

### Functions

- [vrna\\_subopt\\_solution\\_t](#) \* [vrna\\_subopt](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int delta, int sorted, FILE \*fp)  
*Returns list of subopt structures or writes to fp.*
- void [vrna\\_subopt\\_cb](#) ([vrna\\_fold\\_compound\\_t](#) \*fc, int delta, [vrna\\_subopt\\_callback](#) \*cb, void \*data)  
*Generate suboptimal structures within an energy band arround the MFE.*
- **SOLUTION** \* [subopt](#) (char \*seq, char \*structure, int delta, FILE \*fp)  
*Returns list of subopt structures or writes to fp.*
- **SOLUTION** \* [subopt\\_par](#) (char \*seq, char \*structure, [vrna\\_param\\_t](#) \*parameters, int delta, int is\_left\_constrained, int is\_circular, FILE \*fp)  
*Returns list of subopt structures or writes to fp.*
- **SOLUTION** \* [subopt\\_circ](#) (char \*seq, char \*sequence, int delta, FILE \*fp)  
*Returns list of circular subopt structures or writes to fp.*
- **SOLUTION** \* [zukersubopt](#) (const char \*string)  
*Compute Zuker type suboptimal structures.*
- **SOLUTION** \* [zukersubopt\\_par](#) (const char \*string, [vrna\\_param\\_t](#) \*parameters)  
*Compute Zuker type suboptimal structures.*

## Variables

- double **print\_energy**  
*printing threshold for use with logML*
- int **subopt\_sorted**  
*Sort output by energy.*
- int **density\_of\_states** [MAXDOS+1]  
*The Density of States.*

### 18.241.1 Detailed Description

RNAsubopt and density of states declarations.

### 18.241.2 Typedef Documentation

#### 18.241.2.1 SOLUTION

```
typedef struct vrna_subopt_sol_s SOLUTION
Backward compatibility typedef for vrna_subopt_sol_s.
```

**Deprecated** Use [vrna\\_subopt\\_solution\\_t](#) instead!

## 18.242 subopt.h

[Go to the documentation of this file.](#)

```
1 /* subopt.h */
2 #ifndef VIENNA_RNA_PACKAGE_SUBOPT_H
3 #define VIENNA_RNA_PACKAGE_SUBOPT_H
4
5 #ifdef VRNA_WARN_DEPRECATED
6 # if defined(__clang__)
7 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
8 # elif defined(__GNUC__)
9 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
10 # else
11 #  define DEPRECATED(func, msg) func
12 # endif
13 #else
14 # define DEPRECATED(func, msg) func
15 #endif
16
31 typedef struct vrna_subopt_sol_s vrna_subopt_solution_t;
32
48 typedef void (vrna_subopt_callback)(const char *structure,
49                           float      energy,
50                           void       *data);
51
52 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
53
58 typedef struct vrna_subopt_sol_s SOLUTION;
59
60 #endif
61
62 #include <stdio.h>
63
64 #include <ViennaRNA/datastructures/basic.h>
65 #include <ViennaRNA/fold_compound.h>
66 #include <ViennaRNA/params/basic.h>
67
68
72 struct vrna_subopt_sol_s {
73     float energy;
74     char *structure;
75 };
76
80 #define MAXDOS          1000
81
89 #define VRNA_UNSORTED    0
90 #define VRNA_SORT_BY_ENERGY_LEXICOGRAPHIC_ASC 1
91 #define VRNA_SORT_BY_ENERGY_ASC        2
92
```

```

123 vrna_subopt_solution_t *
124 vrna_subopt(vrna_fold_compound_t *fc,
125     int delta,
126     int sorted,
127     FILE *fp);
128
129
164 void
165 vrna_subopt_cb(vrna_fold_compound_t *fc,
166     int delta,
167     vrna_subopt_callback *cb,
168     void *data);
169
170
177 extern double print_energy;
178
185 extern int subopt_sorted;
186
203 extern int density_of_states[MAXDOS + 1];
204 /* End of group dos */
206
207 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
208
226 DEPRECATED(SOLUTION * subopt(char *seq, char *structure, int delta, FILE * fp),
227     "Use vrna_subopt() or vrna_subopt_cb() instead");
228
234 DEPRECATED(SOLUTION *
235     subopt_par(char *seq, char *structure, vrna_param_t * parameters, int delta,
236     int is_constrained,
237     int is_circular, FILE * fp),
238     "Use vrna_subopt() or vrna_subopt_cb() instead");
239
254 DEPRECATED(SOLUTION * subopt_circ(char *seq, char *sequence, int delta, FILE * fp),
255     "Use vrna_subopt() or vrna_subopt_cb() instead");
256
271 DEPRECATED(SOLUTION * zukersubopt(const char *string),
272     "Use vrna_subopt_zuker() instead");
273
282 DEPRECATED(SOLUTION * zukersubopt_par(const char *string, vrna_param_t * parameters),
283     "Use vrna_subopt_zuker() instead");
284
285
286 #endif
287
288 #endif

```

## 18.243 subopt\_zuker.h

```

1 /* subopt_zuker.h */
2 #ifndef VIENNA_RNA_PACKAGE_SUBOPT_ZUKER_H
3 #define VIENNA_RNA_PACKAGE_SUBOPT_ZUKER_H
4
5 #include <ViennaRNA/fold_compound.h>
6 #include <ViennaRNA/subopt.h>
7
32 vrna_subopt_solution_t *
33 vrna_subopt_zuker(vrna_fold_compound_t *fc);
34
35
36 #endif

```

## 18.244 ViennaRNA/svm\_utils.h File Reference

Use [ViennaRNA/utils/svm.h](#) instead.

Include dependency graph for svm\_utils.h:

### 18.244.1 Detailed Description

Use [ViennaRNA/utils/svm.h](#) instead.

**Deprecated** Use [ViennaRNA/utils/svm.h](#) instead

## 18.245 svm\_utils.h

Go to the documentation of this file.

```

1 #ifndef VIENNA_RNA_PACKAGE_UTILS_SVM_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_UTILS_SVM_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 # warning "Including deprecated header file <ViennaRNA/svm_utils.h>! Use <ViennaRNA/utils/svm.h> instead!"
13 # endif
14 #include <ViennaRNA/utils/svm.h>
15 #endif
16
17 #endif

```

## 18.246 ViennaRNA/treedist.h File Reference

Functions for [Tree](#) Edit Distances.

Include dependency graph for treedist.h:

### Functions

- [Tree \\* make\\_tree \(char \\*struc\)](#)  
*Constructs a [Tree](#) ( essentially the postorder list ) of the structure 'struc', for use in [tree\\_edit\\_distance\(\)](#).*
- [float tree\\_edit\\_distance \(Tree \\*T1, Tree \\*T2\)](#)  
*Calculates the edit distance of the two trees.*
- [void print\\_tree \(Tree \\*t\)](#)  
*Print a tree (mainly for debugging)*
- [void free\\_tree \(Tree \\*t\)](#)  
*Free the memory allocated for [Tree](#) t.*

### 18.246.1 Detailed Description

Functions for [Tree](#) Edit Distances.

### 18.246.2 Function Documentation

#### 18.246.2.1 make\_tree()

```
Tree * make_tree (
    char * struc )
```

Constructs a [Tree](#) ( essentially the postorder list ) of the structure 'struc', for use in [tree\\_edit\\_distance\(\)](#).

##### Parameters

|              |   |
|--------------|---|
| <i>struc</i> | may be any rooted structure representation. |
|--------------|---|

##### Returns

#### 18.246.2.2 tree\_edit\_distance()

```
float tree_edit_distance (
    Tree * T1,
    Tree * T2 )
```

Calculates the edit distance of the two trees.

**Parameters**

|    |  |
|----|--|
| T1 |  |
| T2 |  |

**Returns****18.246.2.3 free\_tree()**

```
void free_tree (
    Tree * t )
Free the memory allocated for Tree t.
```

**Parameters**

|   |  |
|---|--|
| t |  |
|---|--|

**18.247 treedist.h**[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_TREE_DIST_H
2 #define VIENNA_RNA_PACKAGE_TREE_DIST_H
3
9 #include <ViennaRNA/dist_vars.h>
10
18 Tree *make_tree(char *struc);
19
20
28 float tree_edit_distance(Tree *T1,
29                           Tree *T2);
30
31
35 void print_tree(Tree *t);
36
37
43 void free_tree(Tree *t);
44
45
46 #endif
```

**18.248 ViennaRNA/units.h File Reference**Use [ViennaRNA/utils/units.h](#) instead.

Include dependency graph for units.h:

**18.248.1 Detailed Description**Use [ViennaRNA/utils/units.h](#) instead.**Deprecated** Use [ViennaRNA/utils/units.h](#) instead**18.249 units.h**[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_UNITS_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_UNITS_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
```

```

12 #warning "Including deprecated header file <ViennaRNA/units.h>! Use <ViennaRNA/utils/units.h> instead!"
13 # endif
14 #include <ViennaRNA/utils/units.h>
15 #endif
16
17 #endif

```

## 18.250 ViennaRNA/utils/units.h File Reference

Physical Units and Functions to convert them into each other.

This graph shows which files directly or indirectly include this file:

### Enumerations

- enum `vrna_unit_energy_e` {
 `VRNA_UNIT_J`, `VRNA_UNIT_KJ`, `VRNA_UNIT_CAL_IT`, `VRNA_UNIT_DACAL_IT`,
 `VRNA_UNIT_KCAL_IT`, `VRNA_UNIT_CAL`, `VRNA_UNIT_DACAL`, `VRNA_UNIT_KCAL`,
 `VRNA_UNIT_G_TNT`, `VRNA_UNIT_KG_TNT`, `VRNA_UNIT_T_TNT`, `VRNA_UNIT_EV`,
 `VRNA_UNIT_WH`, `VRNA_UNIT_KWH` }
- Energy / Work Units.*
- enum `vrna_unit_temperature_e` {
 `VRNA_UNIT_K`, `VRNA_UNIT_DEG_C`, `VRNA_UNIT_DEG_F`, `VRNA_UNIT_DEG_R`,
 `VRNA_UNIT_DEG_N`, `VRNA_UNIT_DEG_DE`, `VRNA_UNIT_DEG_RE`, `VRNA_UNIT_DEG_RO` }
- Temperature Units.*

### Functions

- double `vrna_convert_energy` (double energy, `vrna_unit_energy_e` from, `vrna_unit_energy_e` to)
 

*Convert between energy / work units.*
- double `vrna_convert_temperature` (double temp, `vrna_unit_temperature_e` from, `vrna_unit_temperature_e` to)
 

*Convert between temperature units.*
- int `vrna_convert_kcal_to_dcal` (double energy)
 

*Convert floating point energy value into integer representation.*
- double `vrna_convert_dcal_to_kcal` (int energy)
 

*Convert an integer representation of free energy in deka-cal/mol to kcal/mol.*

### 18.250.1 Detailed Description

Physical Units and Functions to convert them into each other.

,

## 18.251 units.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_UNITS_H
2 #define VIENNA_RNA_PACKAGE_UNITS_H
3
21 typedef enum {
22     VRNA_UNIT_J,
23     VRNA_UNIT_KJ,
24     VRNA_UNIT_CAL_IT,
25     VRNA_UNIT_DACAL_IT,
26     VRNA_UNIT_KCAL_IT,
27     VRNA_UNIT_CAL,
28     VRNA_UNIT_DACAL,
29     VRNA_UNIT_KCAL,
30     VRNA_UNIT_G_TNT,
31     VRNA_UNIT_KG_TNT,
32     VRNA_UNIT_T_TNT,
33     VRNA_UNIT_EV,
34     VRNA_UNIT_WH,
35     VRNA_UNIT_KWH,

```

```

36 } vrna_unit_energy_e;
37
38
44 typedef enum {
45     VRNA_UNIT_K,
46     VRNA_UNIT_DEG_C,
47     VRNA_UNIT_DEG_F,
48     VRNA_UNIT_DEG_R,
49     VRNA_UNIT_DEG_N,
50     VRNA_UNIT_DEG_DB,
51     VRNA_UNIT_DEG_RB,
52     VRNA_UNIT_DEG_RO,
53 } vrna_unit_temperature_e;
54
55
65 double
66 vrna_convert_energy(double energy,
67                      vrna_unit_energy_e from,
68                      vrna_unit_energy_e to);
69
70
80 double
81 vrna_convert_temperature(double temp,
82                           vrna_unit_temperature_e from,
83                           vrna_unit_temperature_e to);
84
85
97 int
98 vrna_convert_kcal_to_dcal(double energy);
99
100
111 double
112 vrna_convert_dcal_to_kcal(int energy);
113
114
119 #endif

```

## 18.252 ViennaRNA/unstructured\_domains.h File Reference

Functions to modify unstructured domains, e.g. to incorporate ligands binding to unpaired stretches.

Include dependency graph for unstructured\_domains.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [vrna\\_unstructured\\_domain\\_s](#)  
*Data structure to store all functionality for ligand binding. [More...](#)*
- struct [vrna\\_unstructured\\_domain\\_motif\\_s](#)

### Macros

- #define **VRNA\_UNSTRUCTURED\_DOMAIN\_EXT\_LOOP** 1U  
*Flag to indicate ligand bound to unpaired stretch in the exterior loop.*
- #define **VRNA\_UNSTRUCTURED\_DOMAIN\_HP\_LOOP** 2U  
*Flag to indicate ligand bound to unpaired stretch in a hairpin loop.*
- #define **VRNA\_UNSTRUCTURED\_DOMAIN\_INT\_LOOP** 4U  
*Flag to indicate ligand bound to unpaired stretch in an interior loop.*
- #define **VRNA\_UNSTRUCTURED\_DOMAIN\_MB\_LOOP** 8U  
*Flag to indicate ligand bound to unpaired stretch in a multibranch loop.*
- #define **VRNA\_UNSTRUCTURED\_DOMAIN\_MOTIF** 16U  
*Flag to indicate ligand binding without additional unbound nucleotides (motif-only)*
- #define **VRNA\_UNSTRUCTURED\_DOMAIN\_ALL\_LOOPS**  
*Flag to indicate ligand bound to unpaired stretch in any loop (convenience macro)*

## Typedefs

- `typedef struct vrna_unstructured_domain_s vrna_ud_t`  
*Typename for the ligand binding extension data structure `vrna_unstructured_domain_s`.*
- `typedef int() vrna_callback_ud_energy(vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, void *data)`  
*Callback to retrieve binding free energy of a ligand bound to an unpaired sequence segment.*
- `typedef FLT_OR_DBL() vrna_callback_ud_exp_energy(vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, void *data)`  
*Callback to retrieve Boltzmann factor of the binding free energy of a ligand bound to an unpaired sequence segment.*
- `typedef void() vrna_callback_ud_production(vrna_fold_compound_t *vc, void *data)`  
*Callback for pre-processing the production rule of the ligand binding to unpaired stretches feature.*
- `typedef void() vrna_callback_ud_exp_production(vrna_fold_compound_t *vc, void *data)`  
*Callback for pre-processing the production rule of the ligand binding to unpaired stretches feature (partition function variant)*
- `typedef void() vrna_callback_ud_probs_add(vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, FLT_OR_DBL exp_energy, void *data)`  
*Callback to store/add equilibrium probability for a ligand bound to an unpaired sequence segment.*
- `typedef FLT_OR_DBL() vrna_callback_ud_probs_get(vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, int motif, void *data)`  
*Callback to retrieve equilibrium probability for a ligand bound to an unpaired sequence segment.*

## Functions

- `vrna_ud_motif_t * vrna_ud_motifs_centroid (vrna_fold_compound_t *fc, const char *structure)`  
*Detect unstructured domains in centroid structure.*
- `vrna_ud_motif_t * vrna_ud_motifs_MEA (vrna_fold_compound_t *fc, const char *structure, vrna_ep_t *probability_list)`  
*Detect unstructured domains in MEA structure.*
- `vrna_ud_motif_t * vrna_ud_motifs_MFE (vrna_fold_compound_t *fc, const char *structure)`  
*Detect unstructured domains in MFE structure.*
- `void vrna_ud_add_motif (vrna_fold_compound_t *vc, const char *motif, double motif_en, const char *motif_name, unsigned int loop_type)`  
*Add an unstructured domain motif, e.g. for ligand binding.*
- `int * vrna_ud_get_motif_size_at (vrna_fold_compound_t *vc, int i, unsigned int loop_type)`  
*Get a list of unique motif sizes that start at a certain position within the sequence.*
- `void vrna_ud_remove (vrna_fold_compound_t *vc)`  
*Remove ligand binding to unpaired stretches.*
- `void vrna_ud_set_data (vrna_fold_compound_t *vc, void *data, vrna_callback_free_auxdata *free_cb)`  
*Attach an auxiliary data structure.*
- `void vrna_ud_set_prod_rule_cb (vrna_fold_compound_t *vc, vrna_callback_ud_production *pre_cb, vrna_callback_ud_energy *e_cb)`  
*Attach production rule callbacks for free energies computations.*
- `void vrna_ud_set_exp_prod_rule_cb (vrna_fold_compound_t *vc, vrna_callback_ud_exp_production *pre_cb, vrna_callback_ud_exp_energy *exp_e_cb)`  
*Attach production rule for partition function.*
- `void vrna_ud_set_prob_cb (vrna_fold_compound_t *vc, vrna_callback_ud_probs_add *setter, vrna_callback_ud_probs_get *getter)`

### 18.252.1 Detailed Description

Functions to modify unstructured domains, e.g. to incorporate ligands binding to unpaired stretches.

## 18.252.2 Function Documentation

### 18.252.2.1 vrna\_ud\_set\_prob\_cb()

```
void vrna_ud_set_prob_cb (
    vrna_fold_compound_t * vc,
    vrna_callback_ud_probs_add * setter,
    vrna_callback_ud_probs_get * getter )
```

**SWIG Wrapper Notes** This function is attached as method **ud\_set\_prob\_cb()** to objects of type *fold\_compound*

## 18.253 unstructured\_domains.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_UNSTRUCTURED_DOMAIN_H
2 #define VIENNA_RNA_PACKAGE_UNSTRUCTURED_DOMAIN_H
3
69 typedef struct vrna_unstructured_domain_s vrna_ud_t;
70
71 typedef struct vrna_unstructured_domain_motif_s vrna_ud_motif_t;
72
73 #include <ViennaRNA/datastructures/basic.h>
74 #include <ViennaRNA/fold_compound.h>
75 #include <ViennaRNA/utils/structures.h>
76
95 typedef int (vrna_callback_ud_energy)(vrna_fold_compound_t *vc,
96                                         int i,
97                                         int j,
98                                         unsigned int loop_type,
99                                         void *data);
100
119 typedef FLT_OR_DBL (vrna_callback_ud_exp_energy)(vrna_fold_compound_t *vc,
120                                                 int i,
121                                                 int j,
122                                                 unsigned int loop_type,
123                                                 void *data);
124
135 typedef void (vrna_callback_ud_production)(vrna_fold_compound_t *vc,
136                                              void *data);
137
148 typedef void (vrna_callback_ud_exp_production)(vrna_fold_compound_t *vc,
149                                                 void *data);
150
151
161 typedef void (vrna_callback_ud_probs_add)(vrna_fold_compound_t *vc,
162                                         int i,
163                                         int j,
164                                         unsigned int loop_type,
165                                         FLT_OR_DBL exp_energy,
166                                         void *data);
167
177 typedef FLT_OR_DBL (vrna_callback_ud_probs_get)(vrna_fold_compound_t *vc,
178                                                 int i,
179                                                 int j,
180                                                 unsigned int loop_type,
181                                                 int motif,
182                                                 void *data);
183
184
189 #define VRNA_UNSTRUCTURED_DOMAIN_EXT_LOOP 1U
190
195 #define VRNA_UNSTRUCTURED_DOMAIN_HP_LOOP 2U
196
201 #define VRNA_UNSTRUCTURED_DOMAIN_INT_LOOP 4U
202
207 #define VRNA_UNSTRUCTURED_DOMAIN_MB_LOOP 8U
208
213 #define VRNA_UNSTRUCTURED_DOMAIN_MOTIF 16U
214
219 #define VRNA_UNSTRUCTURED_DOMAIN_ALL_LOOPS (VRNA_UNSTRUCTURED_DOMAIN_EXT_LOOP | \
220                                             VRNA_UNSTRUCTURED_DOMAIN_HP_LOOP | \
221                                             VRNA_UNSTRUCTURED_DOMAIN_INT_LOOP | \
222                                             VRNA_UNSTRUCTURED_DOMAIN_MB_LOOP)
223
228 struct vrna_unstructured_domain_s {
229     /*
230     ****
231     * Keep track of all motifs added
```

```

232  ****
233  */
234  int      uniq_motif_count;
235  unsigned int *uniq_motif_size;
237  int      motif_count;
238  char     **motif;
239  char     **motif_name;
240  unsigned int *motif_size;
241  double   *motif_en;
242  unsigned int *motif_type;
244  /*
245  ****
246  * Grammar extension for ligand
247  * binding
248  ****
249  */
250  vrna_callback_ud_production    *prod_cb;
254  vrna_callback_ud_exp_production *exp_prod_cb;
255  vrna_callback_ud_energy       *energy_cb;
256  vrna_callback_ud_exp_energy   *exp_energy_cb;
257  void    *data;
258  vrna_callback_free_auxdata   *free_data;
259  vrna_callback_ud_probs_add    *probs_add;
260  vrna_callback_ud_probs_get    *probs_get;
261 };
262
263
264 struct vrna_unstructured_domain_motif_s {
265  int start;
266  int number;
267 };
268
269
289 vrna_ud_motif_t *
290 vrna_ud_motifs_centroid(vrna_fold_compound_t *fc,
291                           const char           *structure);
292
293
314 vrna_ud_motif_t *
315 vrna_ud_motifs_MEA(vrna_fold_compound_t *fc,
316                      const char           *structure,
317                      vrna_ep_t            *probability_list);
318
319
338 vrna_ud_motif_t *
339 vrna_ud_motifs_MFE(vrna_fold_compound_t *fc,
340                      const char           *structure);
341
342
368 void  vrna_ud_add_motif(vrna_fold_compound_t *vc,
369                           const char           *motif,
370                           double              motif_en,
371                           const char           *motif_name,
372                           unsigned int         loop_type);
373
374
379 int *vrna_ud_get_motif_size_at(vrna_fold_compound_t *vc,
380                                   int                 i,
381                                   unsigned int        loop_type);
382
383
384 int *
385 vrna_ud_get_motifs_at(vrna_fold_compound_t *vc,
386                        int                 i,
387                        unsigned int        loop_type);
388
389
390 vrna_ud_motif_t *
391 vrna_ud_detect_motifs(vrna_fold_compound_t *vc,
392                        const char           *structure);
393
394
405 void  vrna_ud_remove(vrna_fold_compound_t *vc);
406
407
424 void  vrna_ud_set_data(vrna_fold_compound_t      *vc,
425                           void                *data,
426                           vrna_callback_free_auxdata *free_cb);
427
428
463 void  vrna_ud_set_prod_rule_cb(vrna_fold_compound_t      *vc,
464                                  vrna_callback_ud_production *pre_cb,
465                                  vrna_callback_ud_energy     *e_cb);
466
467
492 void  vrna_ud_set_exp_prod_rule_cb(vrna_fold_compound_t      *vc,
493                                       vrna_callback_ud_exp_production *pre_cb,

```

```

494                               vrna_callback_ud_exp_energy      *exp_e_cb);
495
496
497 void  vrna_ud_set_prob_cb(vrna_fold_compound_t          *vc,
498                           vrna_callback_ud_probs_add   *setter,
499                           vrna_callback_ud_probs_get   *getter);
500
501
502 #endif

```

## 18.254 ViennaRNA/io/utils.h File Reference

Several utilities for file handling.

Include dependency graph for utils.h: This graph shows which files directly or indirectly include this file:

### Functions

- void **vrna\_file\_copy** (FILE \*from, FILE \*to)  
*Inefficient 'cp'.*
- char \* **vrna\_read\_line** (FILE \*fp)  
*Read a line of arbitrary length from a stream.*
- int **vrna\_mkdir\_p** (const char \*path)  
*Recursively create a directory tree.*
- char \* **vrna\_basename** (const char \*path)  
*Extract the filename from a file path.*
- char \* **vrna dirname** (const char \*path)  
*Extract the directory part of a file path.*
- char \* **vrna\_filename\_sanitize** (const char \*name, const char \*replacement)  
*Sanitize a file name.*
- int **vrna\_file\_exists** (const char \*filename)  
*Check if a file already exists in the file system.*

### 18.254.1 Detailed Description

Several utilities for file handling.

## 18.255 utils.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_FILE_UTILS_H
2 #define VIENNA_RNA_PACKAGE_FILE_UTILS_H
3
10 #include <stdio.h>
11
21 void vrna_file_copy(FILE *from,
22                      FILE *to);
23
24
35 char *vrna_read_line(FILE *fp);
36
37
41 int vrna_mkdir_p(const char *path);
42
43
47 char *vrna_basename(const char *path);
48
49
53 char *vrna dirname(const char *path);
54
55
97 char *vrna_filename_sanitize(const char *name,
98                               const char *replacement);
99
100
107 int

```

```

108 vrna_file_exists(const char *filename);
109
110
115 #endif

```

## 18.256 ViennaRNA/plotting/utils.h File Reference

Various utilities to assist in plotting secondary structures and consensus structures.

Include dependency graph for utils.h: This graph shows which files directly or indirectly include this file:

### Functions

- `char ** vrna_annotation_covar_db (const char **alignment, const char *structure, vrna_md_t *md_p)`  
*Produce covariance annotation for an alignment given a secondary structure.*
- `vrna_cpair_t * vrna_annotation_covar_pairs (const char **alignment, vrna_ep_t *pl, vrna_ep_t *mfel, double threshold, vrna_md_t *md)`  
*Produce covariance annotation for an alignment given a set of base pairs.*

### 18.256.1 Detailed Description

Various utilities to assist in plotting secondary structures and consensus structures.

,

## 18.257 utils.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_PLOT_UTILS_H
2 #define VIENNA_RNA_PACKAGE_PLOT_UTILS_H
3
10 #include <ViennaRNA/datastructures/basic.h>
11 #include <ViennaRNA/model.h>
12 #include <ViennaRNA/utils/structures.h>
13
32 char **
33 vrna_annotation_covar_db(const char    **alignment,
34                           const char    *structure,
35                           vrna_md_t    *md_p);
36
37
38 char **
39 vrna_annotation_covar_db_extended(const char    **alignment,
40                                   const char    *structure,
41                                   vrna_md_t    *md_p,
42                                   unsigned int options);
43
44
49 vrna_cpair_t *
50 vrna_annotation_covar_pairs(const char    **alignment,
51                            vrna_ep_t    *pl,
52                            vrna_ep_t    *mfel,
53                            double      threshold,
54                            vrna_md_t    *md);
55
56
61 #endif

```

## 18.258 ViennaRNA/utils.h File Reference

Use [ViennaRNA/utils/basic.h](#) instead.

Include dependency graph for utils.h:

### 18.258.1 Detailed Description

Use [ViennaRNA/utils/basic.h](#) instead.

**Deprecated** Use [ViennaRNA/utils/basic.h](#) instead

**Deprecated** Use [ViennaRNA/utils/basic.h](#) instead

## 18.259 utils.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_UTILS_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_UTILS_DEPRECATED_H
3
4 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
5 # ifdef VRNA_WARN_DEPRECATED
6 # warning "Including deprecated header file <ViennaRNA/utils.h>! Use <ViennaRNA/utils/basic.h> instead!"
7 # endif
8 #include <ViennaRNA/utils/basic.h>
9 #include <ViennaRNA/utils/strings.h>
10 #include <ViennaRNA/utils/structures.h>
11 #include <ViennaRNA/io/utils.h>
12 #include <ViennaRNA/alphabet.h>
13 #endif
14
15 #endif
16
17 #endif
18
19 #endif
20
21 #endif

```

## 18.260 cpu.h

```

1 #ifndef VIENNA_RNA_PACKAGE_UTILS_CPU_H
2 #define VIENNA_RNA_PACKAGE_UTILS_CPU_H
3
4 #define VRNA_CPU SIMD_NONE      0U
5 #define VRNA_CPU SIMD_SSE2     1U
6 #define VRNA_CPU SIMD_SSE3     2U
7 #define VRNA_CPU SIMD_SSE41    4U
8 #define VRNA_CPU SIMD_SSE42    8U
9 #define VRNA_CPU SIMD_AVX      16U
10 #define VRNA_CPU SIMD_AVX2     32U
11 #define VRNA_CPU SIMD_AVX512F  64U
12
13
14 char *
15 vrna_cpu_vendor_string(void);
16
17
18 unsigned int
19 vrna_cpu_simd_capabilities(void);
20
21
22 #endif

```

## 18.261 higher\_order\_functions.h

```

1 #ifndef VIENNA_RNA_PACKAGE_UTILS_FUN_H
2 #define VIENNA_RNA_PACKAGE_UTILS_FUN_H
3
4 void
5 vrna_fun_dispatch_disable(void);
6
7
8 void
9 vrna_fun_dispatch_enable(void);
10
11
12 int
13 vrna_fun_zip_add_min(const int *e1,
14                      const int *e2,
15                      int       count);
16
17
18 #endif

```

## 18.262 ViennaRNA/utils/strings.h File Reference

General utility- and helper-functions for RNA sequence and structure strings used throughout the ViennaRNA Package.

Include dependency graph for strings.h: This graph shows which files directly or indirectly include this file:

### Macros

- `#define XSTR(s) STR(s)`

- `#define STR(s) #s`

*Stringify a macro after expansion.*
- `#define FILENAME_MAX_LENGTH 80`

*Maximum length of filenames that are generated by our programs.*
- `#define FILENAME_ID_LENGTH 42`

*Maximum length of id taken from fasta header for filename generation.*
- `#define VRNA_TRIM.LEADING 1U`

*Trim only characters leading the string.*
- `#define VRNA_TRIM.TRAILING 2U`

*Trim only characters trailing the string.*
- `#define VRNA_TRIM.IN.BETWEEN 4U`

*Trim only characters within the string.*
- `#define VRNA_TRIM.SUBST_BY_FIRST 8U`

*Replace remaining characters after trimming with the first delimiter in list.*
- `#define VRNA_TRIM.DEFAULT ( VRNA_TRIM.LEADING | VRNA_TRIM.TRAILING )`

*Default settings for trimming, i.e. trim leading and trailing.*
- `#define VRNA_TRIM.ALL ( VRNA_TRIM.DEFAULT | VRNA_TRIM.IN.BETWEEN )`

*Trim characters anywhere in the string.*

## Functions

- `char * vrna_strdup_printf (const char *format,...)`

*Safely create a formatted string.*
- `char * vrna_strdup_vprintf (const char *format, va_list argp)`

*Safely create a formatted string.*
- `int vrna_strcat_printf (char **dest, const char *format,...)`

*Safely append a formatted string to another string.*
- `int vrna_strcat_vprintf (char **dest, const char *format, va_list args)`

*Safely append a formatted string to another string.*
- `unsigned int vrna_strtrim (char *string, const char *delimiters, unsigned int keep, unsigned int options)`

*Trim a string by removing (multiple) occurrences of a particular character.*
- `char ** vrna_strsplit (const char *string, const char *delimiter)`

*Split a string into tokens using a delimiting character.*
- `char * vrna_random_string (int l, const char symbols[ ])`

*Create a random string using characters from a specified symbol set.*
- `int vrna_hamming_distance (const char *s1, const char *s2)`

*Calculate hamming distance between two sequences.*
- `int vrna_hamming_distance_bound (const char *s1, const char *s2, int n)`

*Calculate hamming distance between two sequences up to a specified length.*
- `void vrna_seq_toRNA (char *sequence)`

*Convert an input sequence (possibly containing DNA alphabet characters) to RNA alphabet.*
- `void vrna_seq_toupper (char *sequence)`

*Convert an input sequence to uppercase.*
- `void vrna_seq_reverse (char *sequence)`

*Reverse a string in-place.*
- `char * vrna_DNA_complement (const char *sequence)`

*Retrieve a DNA sequence which resembles the complement of the input sequence.*
- `char * vrna_seq_ungapped (const char *sequence)`

*Remove gap characters from a nucleotide sequence.*
- `char * vrna_cut_point_insert (const char *string, int cp)`

- Add a separating '&' character into a string according to cut-point position.
- `char * vrna_cut_point_remove (const char *string, int *cp)`  
Remove a separating '&' character from a string.
  - `void str_uppercase (char *sequence)`  
Convert an input sequence to uppercase.
  - `void str_DNA2RNA (char *sequence)`  
Convert a DNA input sequence to RNA alphabet.
  - `char * random_string (int l, const char symbols[])`  
Create a random string using characters from a specified symbol set.
  - `int hamming (const char *s1, const char *s2)`  
Calculate hamming distance between two sequences.
  - `int hamming_bound (const char *s1, const char *s2, int n)`  
Calculate hamming distance between two sequences up to a specified length.

## 18.262.1 Detailed Description

General utility- and helper-functions for RNA sequence and structure strings used throughout the ViennaRNA Package.

,

## 18.262.2 Function Documentation

### 18.262.2.1 str\_uppercase()

```
void str_uppercase (
    char * sequence )
```

Convert an input sequence to uppercase.

**Deprecated** Use `vrna_seq_toupper()` instead!

### 18.262.2.2 str\_DNA2RNA()

```
void str_DNA2RNA (
    char * sequence )
```

Convert a DNA input sequence to RNA alphabet.

**Deprecated** Use `vrna_seq_toRNA()` instead!

### 18.262.2.3 random\_string()

```
char * random_string (
    int l,
    const char symbols[] )
```

Create a random string using characters from a specified symbol set.

**Deprecated** Use `vrna_random_string()` instead!

#### 18.262.2.4 hamming()

```
int hamming (
    const char * s1,
    const char * s2 )
```

Calculate hamming distance between two sequences.

**Deprecated** Use [vrna\\_hamming\\_distance\(\)](#) instead!

#### 18.262.2.5 hamming\_bound()

```
int hamming_bound (
    const char * s1,
    const char * s2,
    int n )
```

Calculate hamming distance between two sequences up to a specified length.

**Deprecated** Use [vrna\\_hamming\\_distance\\_bound\(\)](#) instead!

### 18.263 strings.h

[Go to the documentation of this file.](#)

```
1 #ifndef VIENNA_RNA_PACKAGE_STRING_UTILS_H
2 #define VIENNA_RNA_PACKAGE_STRING_UTILS_H
3
4 #ifdef VRNA_WARN_DEPRECATED
5 # if defined(__clang__)
6 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
7 # elif defined(__GNUC__)
8 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
9 # else
10 #  define DEPRECATED(func, msg) func
11 # endif
12 #else
13 # define DEPRECATED(func, msg) func
14 #endif
15
28 #include <stdarg.h>
29 #include <ViennaRNA/datastructures/basic.h>
30
34 #define XSTR(s) STR(s)
35
39 #define STR(s) #s
40
41 #ifndef FILENAME_MAX_LENGTH
42
49 #define FILENAME_MAX_LENGTH 80
50
57 #define FILENAME_ID_LENGTH 42
58
59 #endif
60
61 #ifdef HAVE_CONFIG_H
62 #include <config.h>
63 #ifndef HAVE_STRDUP
64 char *
65 strdup(const char *s);
66
67
68 #endif
69 #endif
70
89 char *
90 vrna_strdup_printf(const char *format,
91                     ...);
92
93
108 char *
109 vrna_strdup_vprintf(const char *format,
110                      va_list argp);
111
112
131 int
132 vrna_strcat_printf(char **dest,
133                      const char *format,
```

```
134         ...);
135
136
137 int
138 vrna_strcat_vprintf(char      **dest,
139                      const char *format,
140                      va_list    args);
141
142
143 #define VRNA_TRIM_LEADING     1U
144
145 #define VRNA_TRIM_TRAILING    2U
146
147 #define VRNA_TRIM_IN_BETWEEN   4U
148
149 #define VRNA_TRIM_SUBST_BY_FIRST 8U
150
151 #define VRNA_TRIM_DEFAULT      ( VRNA_TRIM_LEADING | VRNA_TRIM_TRAILING )
152
153 #define VRNA_TRIM_ALL          ( VRNA_TRIM_DEFAULT | VRNA_TRIM_IN_BETWEEN )
154
155
156 unsigned int
157 vrna_trim(char      *string,
158             const char *delimiters,
159             unsigned int keep,
160             unsigned int options);
161
162
163
164 char **
165 vrna_strsplit(const char *string,
166                 const char *delimiter);
167
168
169
170 char *
171 vrna_strjoin(const char **strings,
172                const char *delimiter);
173
174
175
176 char *
177 vrna_random_string(int      l,
178                      const char symbols[]);
179
180
181
182 int
183 vrna_hamming_distance(const char *s1,
184                         const char *s2);
185
186
187
188 int
189 vrna_hamming_distance_bound(const char *s1,
190                               const char *s2,
191                               int        n);
192
193
194
195 void
196 vrna_seq_toRNA(char *sequence);
197
198
199
200 void
201 vrna_seq_toupper(char *sequence);
202
203
204
205 void
206 vrna_seq_reverse(char *sequence);
207
208
209
210 char *
211 vrna_DNA_complement(const char *sequence);
212
213
214
215 char *
216 vrna_seq_ungapped(const char *sequence);
217
218
219
220 char *
221 vrna_cut_point_insert(const char *string,
222                         int       cp);
223
224
225
226 char *
227 vrna_cut_point_remove(const char *string,
228                         int       *cp);
229
230
231
232 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
233
234
235 DEPRECATED(void
```

```

454         str_uppercase(char *sequence),
455         "Use vrna_seq_toupper() instead");
456
462 DEPRECATED(void
463         str_DNA2RNA(char *sequence),
464         "Use vrna_seq_toRNA() instead");
465
471 DEPRECATED(char *random_string(int l,
472         const char symbols[]),
473         "Use vrna_random_string() instead");
474
480 DEPRECATED(int
481         hamming(const char *s1,
482                 const char *s2),
483         "Use vrna_hamming_distance() instead");
484
490 DEPRECATED(int
491         hamming_bound(const char *s1,
492                     const char *s2,
493                     int n),
494         "Use vrna_hamming_distance_bound() instead");
495
496 #endif
497
498 #endif

```

## 18.264 svm.h

```

1 #ifndef VIENNA_RNA_PACKAGE_UTILS_SVM_H
2 #define VIENNA_RNA_PACKAGE_UTILS_SVM_H
3
4 extern char *avg_model_string;
5 extern char *sd_model_string;
6
7 float     get_z(char *sequence,
8                  double energy);
9 double    avg_regression (int N,
10                      int A,
11                      int C,
12                      int G,
13                      int T,
14                      struct svm_model *avg_model,
15                      int *info );
16 double    sd_regression (int N,
17                      int A,
18                      int C,
19                      int G,
20                      int T,
21                      struct svm_model *sd_model);
22 double    minimal_sd   (int N,
23                      int A,
24                      int C,
25                      int G,
26                      int T);
27 struct svm_model *svm_load_model_string(char *modelString);
28 int       *get_seq_composition( short *S,
29                               unsigned int start,
30                               unsigned int stop,
31                               unsigned int length);
32
33 #endif

```

## 18.265 vrna\_config.h

```

1 #ifndef VIENNA_RNA_PACKAGE_CONFIG_H
2 #define VIENNA_RNA_PACKAGE_CONFIG_H
3
4 /* version number */
5 #define VRNA_VERSION "2.5.1"
6
7 #define VRNA_VERSION_MAJOR 2
8 #define VRNA_VERSION_MINOR 5
9 #define VRNA_VERSION_PATCH 1
10
11 /*
12  * The following pre-processor definitions specify whether
13  * or not certain features were activated upon build-time
14  */
15
16 /*
17  * Build with deactivated C11 Features
18  *

```

```

19 * It this feature is missing, the next line defines
20 * 'VRNA_DISABLE_C11_FEATURES'
21 */
22
23 /*
24 * Build with OpenMP support
25 *
26 * If this feature is present, the next line defines
27 * 'VRNA_WITH_OPENMP'
28 */
29
30 #define VRNA_WITH_OPENMP
31
32 /*
33 * Build with single precision partition function
34 *
35 * If this feature is present, the next line defines
36 * 'USE_FLOAT_PF'
37 */
38
39 /*
40 * Build with JSON input/output support
41 *
42 * If this feature is present, the next line defines
43 * 'VRNA_WITH_JSON_SUPPORT'
44 */
45
46 #define VRNA_WITH_JSON_SUPPORT
47
48 /*
49 * Build with Support Vector Machine (SVM) Z-score feature in RNALfold
50 *
51 * If this feature is present, the next line defines
52 * 'VRNA_WITH_SVM'
53 */
54 #define VRNA_WITH_SVM
55
56 /*
57 * Build with GSL minimizers
58 *
59 * If this feature is present, the next line defines
60 * 'VRNA_WITH_GSL'
61 */
62 #define VRNA_WITH_GSL
63
64 /*
65 * Build with colored TTY output
66 *
67 * If this feature is missing, the next line defines
68 * 'VRNA_WITHOUT_TTY_COLORS'
69 */
70
71 /*
72 * Build with Link Time Optimization support
73 *
74 * If this feature is enabled, the next line defines
75 * 'VRNA_WITH_LTO'
76 */
77
78
79 /*
80 * Build with Naview Layout algorithm of Bruccoleri 1988
81 *
82 * If this feature is enabled, the next line defines
83 * 'VRNA_WITH_NAVIEW_LAYOUT'
84 */
85
86 #define VRNA_WITH_NAVIEW_LAYOUT
87
88
89
90 #endif

```

## 18.266 ViennaRNA/landscape/walk.h File Reference

Methods to generate particular paths such as gradient or random walks through the energy landscape of an RNA sequence.

Include dependency graph for walk.h: This graph shows which files directly or indirectly include this file:

## Macros

- `#define VRNA_PATH_STEEPEST_DESCENT 128`  
*Option flag to request a steepest descent / gradient path.*
- `#define VRNA_PATH_RANDOM 256`  
*Option flag to request a random walk path.*
- `#define VRNA_PATH_NO_TRANSITION_OUTPUT 512`  
*Option flag to omit returning the transition path.*
- `#define VRNA_PATH_DEFAULT (VRNA_PATH_STEEPEST_DESCENT | VRNA_MOVESET_DEFAULT)`  
*Option flag to request defaults (steepest descent / default move set)*

## Functions

- `vrna_move_t * vrna_path (vrna_fold_compound_t *vc, short *pt, unsigned int steps, unsigned int options)`  
*Compute a path, store the final structure, and return a list of transition moves from the start to the final structure.*
- `vrna_move_t * vrna_path_gradient (vrna_fold_compound_t *vc, short *pt, unsigned int options)`  
*Compute a steepest descent / gradient path, store the final structure, and return a list of transition moves from the start to the final structure.*
- `vrna_move_t * vrna_path_random (vrna_fold_compound_t *vc, short *pt, unsigned int steps, unsigned int options)`  
*Generate a random walk / path of a given length, store the final structure, and return a list of transition moves from the start to the final structure.*

### 18.266.1 Detailed Description

Methods to generate particular paths such as gradient or random walks through the energy landscape of an RNA sequence.

## 18.267 walk.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_WALK_H
2 #define VIENNA_RNA_PACKAGE_WALK_H
3
11 #include <ViennaRNA/fold_compound.h>
12 #include <ViennaRNA/landscape/move.h>
13
24 #define VRNA_PATH_STEEPEST_DESCENT 128
25
30 #define VRNA_PATH_RANDOM 256
31
36 #define VRNA_PATH_NO_TRANSITION_OUTPUT 512
37
43 #define VRNA_PATH_DEFAULT (VRNA_PATH_STEEPEST_DESCENT | VRNA_MOVESET_DEFAULT)
44
73 vrna_move_t *
74 vrna_path(vrna_fold_compound_t *vc,
75             short *pt,
76             unsigned int steps,
77             unsigned int options);
78
79
101 vrna_move_t *
102 vrna_path_gradient(vrna_fold_compound_t *vc,
103                      short *pt,
104                      unsigned int options);
105
106
129 vrna_move_t *
130 vrna_path_random(vrna_fold_compound_t *vc,
131                   short *pt,
132                   unsigned int steps,
133                   unsigned int options);
134
135
140#endif /* VIENNA_RNA_PACKAGE_WALK_H */

```

## 18.268 ViennaRNA/walk.h File Reference

Use [ViennaRNA/landscape/walk.h](#) instead.

Include dependency graph for walk.h:

### 18.268.1 Detailed Description

Use [ViennaRNA/landscape/walk.h](#) instead.

**Deprecated** Use [ViennaRNA/landscape/walk.h](#) instead

## 18.269 walk.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIENNA_RNA_PACKAGE_WALK_DEPRECATED_H
2 #define VIENNA_RNA_PACKAGE_WALK_DEPRECATED_H
3
10 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
11 # ifdef VRNA_WARN_DEPRECATED
12 # warning "Including deprecated header file <ViennaRNA/walk.h>! Use <ViennaRNA/landscape/walk.h> instead!"
13 # endif
14 #include <ViennaRNA/landscape/walk.h>
15 #include <ViennaRNA/landscape/neighbor.h>
16 #endif
17
18#endif

```

## 18.270 wrap\_dlib.h

```

1 #ifndef VIENNARNA_DLIP_WRAPPER_H
2 #define VIENNARNA_DLIP_WRAPPER_H
3
4 #ifdef __cplusplus
5 extern "C" {
6 #endif
7
8 double *
9 vrna_equilibrium_conc(const double           *eq_constants,
10                      double                 *concentration_strands,
11                      const unsigned int    **A,
12                      size_t                num_strands,
13                      size_t                num_complexes);
14
15
16 #ifdef __cplusplus
17 }
18 #endif
19
20#endif

```

## 18.271 zscore.h

```

1 #ifndef VIENNA_RNA_PACKAGE_ZSCORE_H
2 #define VIENNA_RNA_PACKAGE_ZSCORE_H
3
4 typedef struct vrna_zsc_dat_s *vrna_zsc_dat_t;
5
6 #define VRNA_ZSCORE_OPTIONS_NONE      1U
7 #define VRNA_ZSCORE_FILTER_ON        2U
8 #define VRNA_ZSCORE_PRE_FILTER      4U
9 #define VRNA_ZSCORE_REPORT_SUBSUMED  8U
10 #define VRNA_ZSCORE_MODEL_DEFAULT   16U
11 #define VRNA_ZSCORE_SETTINGS_DEFAULT (VRNA_ZSCORE_FILTER_ON | VRNA_ZSCORE_MODEL_DEFAULT)
12
13 int
14 vrna_zsc_filter_init(vrna_fold_compound_t *fc,
15                      double               min_z,
16                      unsigned int         options);
17
18
19 int
20 vrna_zsc_filter_update(vrna_fold_compound_t *fc,
21                      double               min_z,
22                      unsigned int         options);
23

```

```
24
25 void
26 vrna_zsc_filter_free(vrna_fold_compound_t *fc);
27
28
29 int
30 vrna_zsc_filter_on(vrna_fold_compound_t *fc);
31
32
33 double
34 vrna_zsc_filter_threshold(vrna_fold_compound_t *fc);
35
36
37 double
38 vrna_zsc_compute(vrna_fold_compound_t *fc,
39                 unsigned int      i,
40                 unsigned int      j,
41                 int              e);
42
43
44 double
45 vrna_zsc_compute_raw(vrna_fold_compound_t *fc,
46                      unsigned int      i,
47                      unsigned int      j,
48                      int              e,
49                      double           *avg,
50                      double           *sd);
51
52
53 #endif
```

# Bibliography

- [1] S.H. Bernhart, I.L. Hofacker, S. Will, A.R. Gruber, and P.F. Stadler. RNAalifold: Improved consensus structure prediction for RNA alignments. *BMC bioinformatics*, 9(1):474, 2008. [27](#)
- [2] S.H. Bernhart, H. Tafer, U. Mückstein, C. Flamm, P.F. Stadler, and I.L. Hofacker. Partition function and base pairing probabilities of RNA heterodimers. *Algorithms for Molecular Biology*, 1(1):3, 2006. [378](#)
- [3] Stephan H Bernhart, Ivo L Hofacker, and Peter F Stadler. Local RNA base pairing probabilities in large sequences. *Bioinformatics*, 22(5):614–615, 2005. [268](#)
- [4] Stephan H Bernhart, Ullrike Mückstein, and Ivo L Hofacker. RNA accessibility in cubic time. *Algorithms for Molecular Biology*, 6(1):3, 2011. [268, 708, 1199](#)
- [5] R.E. Bruccoleri and G. Heinrich. An improved algorithm for nucleic acid secondary structure display. *Computer applications in the biosciences: CABIOS*, 4(1):167–173, 1988. [463, 574](#)
- [6] Katherine E. Deigan, Tian W. Li, David H. Mathews, and Kevin M. Weeks. Accurate SHAPE-directed RNA structure determination. *PNAS*, 106:97–102, 2009. [348](#)
- [7] Christoph Flamm, Ivo L Hofacker, Sebastian Maurer-Stroh, Peter F Stadler, and Martin Zehl. Design of multi-stable RNA molecules. *RNA*, 7(02):254–265, 2001. [339, 340, 341, 342](#)
- [8] W. Fontana, P.F. Stadler, E.G. Bornberg-Bauer, T. Griesmacher, I.L. Hofacker, M. Tacker, P. Tarazona, E.D. Weinberger, and P. Schuster. RNA folding and combinatorial landscapes. *Physical review E*, 47(3):2083, 1993. [33, 34, 424, 427](#)
- [9] Eva Freyhult, Vincent Moulton, and Paul Gardner. Predicting RNA structure using mutual information. *Applied bioinformatics*, 4(1):53–59, 2005. [435](#)
- [10] Robert Giegerich, Björn Voß, and Marc Rehmsmeier. Abstract shapes of RNA. *Nucleic Acids Research*, 32(16):4843–4851, 2004. [33, 406, 421, 422](#)
- [11] I.L. Hofacker, M. Fekete, and P.F. Stadler. Secondary structure prediction for aligned RNA sequences. *Journal of molecular biology*, 319(5):1059–1066, 2002. [27](#)
- [12] I.L. Hofacker, W. Fontana, P.F. Stadler, L.S. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie/Chemical Monthly*, 125(2):167–188, 1994. [1](#)
- [13] I.L. Hofacker and P.F. Stadler. Memory efficient folding algorithms for circular RNA secondary structures. *Bioinformatics*, 22(10):1172–1176, 2006. [24, 252, 253, 264, 265](#)
- [14] Ronny Lorenz, Stephan H. Bernhart, Christian Höner zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F. Stadler, and Ivo L. Hofacker. ViennaRNA package 2.0. *Algorithms for Molecular Biology*, 6(1):26, 2011. [1](#)
- [15] Ronny Lorenz, Christoph Flamm, and Ivo L. Hofacker. 2d projections of RNA folding landscapes. In Ivo Grosse, Steffen Neumann, Stefan Posch, Falk Schreiber, and Peter F. Stadler, editors, *German Conference on Bioinformatics 2009*, volume 157 of *Lecture Notes in Informatics*, pages 11–20, Bonn, September 2009. Gesellschaft f. Informatik. [27, 304](#)
- [16] Ronny Lorenz, Ivo L. Hofacker, and Peter F. Stadler. RNA folding with hard and soft constraints. *Algorithms for Molecular Biology*, 11(1):1–13, 2016. [22](#)
- [17] Ronny Lorenz, Dominik Luntzer, Ivo L. Hofacker, Peter F. Stadler, and Michael T. Wolfinger. Shape directed rna folding. *Bioinformatics*, 32(1):145–147, 2016. [347](#)

- [18] J.S. McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29(6-7):1105–1119, 1990. [25](#)
- [19] Juraj Michálik, Hélène Touzet, and Yann Ponty. Efficient approximations of RNA kinetics landscape using non-redundant sampling. *Bioinformatics*, 33(14):i283–i292, 2017. [281](#), [283](#), [284](#), [285](#), [287](#), [289](#), [290](#), [291](#), [292](#), [294](#), [295](#), [297](#), [298](#)
- [20] Joe Sawada. A fast algorithm to generate necklaces with fixed content. *Theoretical Computer Science*, 301(1):477–489, 2003. [480](#)
- [21] B.A. Shapiro. An algorithm for comparing multiple RNA secondary structures. *Computer applications in the biosciences: CABIOS*, 4(3):387–393, 1988. [33](#), [34](#), [424](#), [427](#)
- [22] B.A. Shapiro and K. Zhang. Comparing multiple RNA secondary structures using tree comparisons. *Computer applications in the biosciences: CABIOS*, 6(4):309–318, 1990. [17](#)
- [23] Stefan Washietl, Ivo L. Hofacker, Peter F. Stadler, and Manolis Kellis. RNA folding with soft constraints: reconciliation of probing data and thermodynamics secondary structure prediction. *Nucleic Acids Research*, 40(10):4261–4272, 2012. [353](#)
- [24] Daniel Wiegreffe, Daniel Alexander, Peter F Stadler, and Dirk Zeckzer. RNAPuzzler: efficient outerplanar drawing of RNA-secondary structures. *Bioinformatics*, 2018. [131](#), [455](#), [461](#), [462](#), [465](#), [466](#), [471](#), [473](#), [1211](#), [1212](#), [1216](#), [1217](#)
- [25] S. Wuchty, W. Fontana, I. L. Hofacker, and P. Schuster. Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers*, 49(2):145–165, February 1999. [277](#), [278](#)
- [26] Kourosh Zarringhalam, Michelle M. Meyer, Ivan Dotu, Jeffrey H. Chuang, and Peter Clote. Integrating chemical footprinting data into RNA secondary structure prediction. *PLOS ONE*, 7(10), 2012. [349](#)
- [27] M. Zuker. On finding all suboptimal foldings of an RNA molecule. *Science*, 244(4900):48–52, April 1989. [275](#)
- [28] M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic acids research*, 9(1):133–148, 1981. [24](#)

# Index

(Abstract) Data Structures, [485](#)  
bondT, [490](#)  
cpair, [489](#)  
PAIR, [489](#)  
plist, [489](#)  
sect, [489](#)  
vrna\_C11\_features, [490](#)

(Nucleic Acid Sequence) String Utilitites, [396](#)  
FILENAME\_ID\_LENGTH, [397](#)  
FILENAME\_MAX\_LENGTH, [397](#)  
vrna\_cut\_point\_insert, [405](#)  
vrna\_cut\_point\_remove, [405](#)  
vrna\_DNA\_complement, [404](#)  
vrna\_hamming\_distance, [403](#)  
vrna\_hamming\_distance\_bound, [403](#)  
vrna\_random\_string, [402](#)  
vrna\_seq\_reverse, [404](#)  
vrna\_seq\_toRNA, [403](#)  
vrna\_seq\_toupper, [404](#)  
vrna\_seq\_ungapped, [405](#)  
vrna\_strcat\_printf, [400](#)  
vrna\_strcat\_vprintf, [400](#)  
vrna\_strdup\_printf, [398](#)  
vrna\_strdup\_vprintf, [399](#)  
vrna\_strsplit, [402](#)  
vrna\_trim, [401](#)  
VRNA\_TRIM\_ALL, [398](#)  
VRNA\_TRIM\_DEFAULT, [398](#)  
VRNA\_TRIM\_IN\_BETWEEN, [398](#)  
VRNA\_TRIM.LEADING, [397](#)  
VRNA\_TRIM\_SUBST\_BY\_FIRST, [398](#)  
VRNA\_TRIM.TRAILING, [397](#)

(Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima, [335](#)  
vrna\_path\_free, [337](#)  
vrna\_path\_options\_free, [337](#)  
VRNA\_PATH\_TYPE\_DOT\_BRACKET, [337](#)  
VRNA\_PATH\_TYPE\_MOVES, [337](#)  
\_struct\_en, [577](#)

2Dpfold.h  
destroy\_TwoDpfold\_variables, [584](#)  
get\_TwoDpfold\_variables, [584](#)  
TwoDpfold\_pbacktrack, [585](#)  
TwoDpfold\_pbacktrack5, [586](#)  
TwoDpfoldList, [585](#)

Abstract Shapes Representation of Secondary Structures, [421](#)  
vrna\_abstract\_shapes, [422](#)  
vrna\_abstract\_shapes\_pt, [422](#)

add\_root  
Deprecated Interface for Secondary Structure Utilities, [566](#)

alifold  
Deprecated Interface for Global MFE Prediction, [534](#)

alifold.h  
cv\_fact, [591](#)  
energy\_of\_alistruct, [590](#)  
nc\_fact, [591](#)  
update\_alifold\_params, [591](#)

Alignment Plots, [474](#)  
vrna\_file\_PS\_aln, [475](#)  
vrna\_file\_PS\_aln\_slice, [475](#)

alimake\_pair\_table  
Deprecated Interface for Secondary Structure Utilities, [570](#)

alipbacktrack  
Deprecated Interface for Global Partition Function Computation, [556](#)

alipf\_circ\_fold  
Deprecated Interface for Global Partition Function Computation, [555](#)

alipf\_fold  
Deprecated Interface for Global Partition Function Computation, [554](#)

alipf\_fold\_par  
Deprecated Interface for Global Partition Function Computation, [544](#)

aliPS\_color\_aln  
Deprecated Interface for Plotting Utilities, [573](#)

alloc\_sequence\_arrays  
Deprecated Interface for Multiple Sequence Alignment Utilities, [563](#)

alpha  
vrna\_exp\_param\_s, [199](#)

Annotation, [474](#)

assign\_plist\_from\_db  
Deprecated Interface for Global Partition Function Computation, [554](#)

assign\_plist\_from\_pr  
Deprecated Interface for Global Partition Function Computation, [554](#)

auxdata  
vrna\_fc\_s, [501](#)

b2C  
Deprecated Interface for Secondary Structure Utilities, [566](#)

b2HIT

Deprecated Interface for Secondary Structure Utilities, 565

b2Shapiro  
Deprecated Interface for Secondary Structure Utilities, 566

backtrack\_GQuad\_IntLoop  
G-Quadruplexes, 362

backtrack\_GQuad\_IntLoop\_L  
G-Quadruplexes, 362

backtrack\_type  
Fine-tuning of the Implemented Models, 196

Backtracking MFE structures, 258

- vrna\_backtrack5, 259
- vrna\_BT\_hp\_loop, 259
- vrna\_BT\_mb\_loop, 259

base\_pair  
fold\_vars.h, 661

basic.h

- filecopy, 896
- get\_line, 894
- init\_rand, 895
- int\_urn, 895
- nerror, 895
- print\_tty\_input\_seq, 894
- print\_tty\_input\_seq\_str, 894
- space, 895
- time\_stamp, 896
- urn, 895
- warn\_user, 894
- xrealloc, 895

bondT  
(Abstract) Data Structures, 490

BONUS

- constants.h, 899

bp\_distance  
Deprecated Interface for Secondary Structure Utilities, 570

bppm\_symbol  
Deprecated Interface for Secondary Structure Utilities, 572

bppm\_to\_structure  
Deprecated Interface for Secondary Structure Utilities, 571

bt

- vrna\_sc\_s, 239

Buffers, 528

- vrna\_callback\_stream\_output, 529
- vrna\_cstr, 529
- vrna\_cstr\_close, 530
- vrna\_cstr\_discard, 529
- vrna\_cstr\_fflush, 530
- vrna\_cstr\_free, 530
- vrna\_ostream\_free, 531
- vrna\_ostream\_init, 531
- vrna\_ostream\_provide, 532
- vrna\_ostream\_request, 531

centroid  
part\_func.h, 1191

centroid.h

- get\_centroid\_struct\_pl, 598
- get\_centroid\_struct\_pr, 599

circalifold  
Deprecated Interface for Global MFE Prediction, 541

circfold  
Deprecated Interface for Global MFE Prediction, 538

Classified Dynamic Programming Variants, 303

co\_pf\_fold  
Deprecated Interface for Global Partition Function Computation, 551

co\_pf\_fold\_par  
Deprecated Interface for Global Partition Function Computation, 551

cofold  
Deprecated Interface for Global MFE Prediction, 534

cofold\_par  
Deprecated Interface for Global MFE Prediction, 534

Combinatorics Algorithms, 479

- vrna\_boustrophedon, 484
- vrna\_boustrophedon\_pos, 485
- vrna\_enumerate\_necklaces, 479
- vrna\_n\_multichoose\_k, 484
- vrna\_rotational\_symmetry, 481
- vrna\_rotational\_symmetry\_db, 482
- vrna\_rotational\_symmetry\_db\_pos, 483
- vrna\_rotational\_symmetry\_num, 480
- vrna\_rotational\_symmetry\_pos, 482
- vrna\_rotational\_symmetry\_pos\_num, 480

Command Files, 451

- VRNA\_CMD\_PARSE\_DEFAULTS, 453
- VRNA\_CMD\_PARSE\_HC, 452
- VRNA\_CMD\_PARSE\_SC, 452
- VRNA\_CMD\_PARSE\_SD, 452
- VRNA\_CMD\_PARSE\_UD, 452
- vrna\_commands\_apply, 454
- vrna\_commands\_free, 454
- vrna\_file\_commands\_apply, 453
- vrna\_file\_commands\_read, 453

Compute the Centroid Structure, 301

- vrna\_centroid, 302
- vrna\_centroid\_from plist, 302
- vrna\_centroid\_from\_probs, 303

Compute the Density of States, 322

- density\_of\_states, 322

Compute the Structure with Maximum Expected Accuracy (MEA), 299

- MEA, 301
- vrna\_MEA, 300
- vrna\_MEA\_from plist, 300

compute\_BPdifferences  
Deprecated Interface for Secondary Structure Utilities, 571

compute\_probabilities

Deprecated Interface for Global Partition Function Computation, 552

Computing MFE representatives of a Distance Based Partitioning, 304

destroy\_TwoDfold\_variables, 308

get\_TwoDfold\_variables, 308

TwoDfold\_backtrack\_f5, 309

TwoDfold\_vars, 306

TwoDfoldList, 309

vrna\_backtrack5\_TwoD, 307

vrna\_mfe\_TwoD, 307

vrna\_sol\_TwoD\_t, 306

Computing Partition Functions of a Distance Based Partitioning, 310

vrna\_pf\_TwoD, 311

vrna\_sol\_TwoD\_pf\_t, 310

concentrations.h

    get\_concentrations, 608

cons\_seq

    vrna\_fc\_s, 503

constants.h

    BONUS, 899

    FORBIDDEN, 899

    GASCONST, 899

    INF, 899

    K0, 899

    MAXLOOP, 899

    NBPAIRS, 899

    TURN, 899

constrain, 489

constrain\_ptypes

    hard.h, 613

Constraining the RNA Folding Grammar, 216

    VRNA\_CONSTRAINT\_FILE, 219

    VRNA\_CONSTRAINT\_SOFT\_MFE, 220

    VRNA\_CONSTRAINT\_SOFT\_PF, 220

    vrna\_constraints\_add, 227

    VRNA\_DECOMP\_EXT\_EXT, 225

    VRNA\_DECOMP\_EXT\_EXT\_EXT, 226

    VRNA\_DECOMP\_EXT\_EXT\_STEM, 226

    VRNA\_DECOMP\_EXT\_EXT\_STEM1, 227

    VRNA\_DECOMP\_EXT\_STEM, 225

    VRNA\_DECOMP\_EXT\_STEM\_EXT, 226

    VRNA\_DECOMP\_EXT\_UP, 225

    VRNA\_DECOMP\_ML\_COAXIAL, 224

    VRNA\_DECOMP\_ML\_COAXIAL\_ENC, 224

    VRNA\_DECOMP\_ML\_ML, 223

    VRNA\_DECOMP\_ML\_ML\_ML, 222

    VRNA\_DECOMP\_ML\_ML\_STEM, 223

    VRNA\_DECOMP\_ML\_STEM, 222

    VRNA\_DECOMP\_ML\_UP, 223

    VRNA\_DECOMP\_PAIR\_HP, 220

    VRNA\_DECOMP\_PAIR\_IL, 221

    VRNA\_DECOMP\_PAIR\_ML, 221

    vrna\_message\_constraint\_options, 228

    vrna\_message\_constraint\_options\_all, 228

convert\_parameter\_file

    Converting Energy Parameter Files, 392

    Converting Energy Parameter Files, 388

    convert\_parameter\_file, 392

    VRNA\_CONVERT\_OUTPUT\_ALL, 389

    VRNA\_CONVERT\_OUTPUT\_BULGE, 391

    VRNA\_CONVERT\_OUTPUT\_DANGLE3, 390

    VRNA\_CONVERT\_OUTPUT\_DANGLE5, 390

    VRNA\_CONVERT\_OUTPUT\_DUMP, 392

    VRNA\_CONVERT\_OUTPUT\_HP, 389

    VRNA\_CONVERT\_OUTPUT\_INT, 391

    VRNA\_CONVERT\_OUTPUT\_INT\_11, 390

    VRNA\_CONVERT\_OUTPUT\_INT\_21, 391

    VRNA\_CONVERT\_OUTPUT\_INT\_22, 391

    VRNA\_CONVERT\_OUTPUT\_MISC, 391

    VRNA\_CONVERT\_OUTPUT\_ML, 391

    VRNA\_CONVERT\_OUTPUT\_MM\_EXT, 390

    VRNA\_CONVERT\_OUTPUT\_MM\_HP, 389

    VRNA\_CONVERT\_OUTPUT\_MM\_INT, 390

    VRNA\_CONVERT\_OUTPUT\_MM\_INT\_1N, 390

    VRNA\_CONVERT\_OUTPUT\_MM\_INT\_23, 390

    VRNA\_CONVERT\_OUTPUT\_MM\_MULTI, 390

    VRNA\_CONVERT\_OUTPUT\_NINIO, 392

    VRNA\_CONVERT\_OUTPUT\_SPECIAL\_HP, 391

    VRNA\_CONVERT\_OUTPUT\_STACK, 389

    VRNA\_CONVERT\_OUTPUT\_VANILLA, 391

COORDINATE, 572

copy\_pair\_table

    Deprecated Interface for Secondary Structure Utilities, 569

cost\_matrix

    dist\_vars.h, 630

cpair

    (Abstract) Data Structures, 489

cut\_point

    fold\_vars.h, 661

cv\_fact

    alifold.h, 591

dangles

    Fine-tuning of the Implemented Models, 195

    vrna\_md\_s, 175

density\_of\_states

    Compute the Density of States, 322

Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers, 574

    find\_saddle, 575

    free\_path, 575

    get\_path, 576

    path\_t, 575

Deprecated Interface for Free Energy Evaluation, 156

    E\_IntLoop, 166

    E\_Stem, 164

    energy\_of\_circ\_struct, 164

    energy\_of\_circ\_struct\_par, 159

    energy\_of\_circ\_structure, 158

    energy\_of\_move, 161

    energy\_of\_move\_pt, 161

    energy\_of\_struct, 162

    energy\_of\_struct\_par, 158

    energy\_of\_struct\_pt, 163

energy\_of\_struct\_pt\_par, 160  
 energy\_of\_structure, 157  
 energy\_of\_structure\_pt, 160  
 exp\_E\_ExtLoop, 165  
 exp\_E\_IntLoop, 167  
 exp\_E\_Stem, 165  
 loop\_energy, 162

Deprecated Interface for Global MFE Prediction, 532  
 alifold, 534  
 circalifold, 541  
 circfold, 538  
 cofold, 534  
 cofold\_par, 534  
 export\_circfold\_arrays, 540  
 export\_circfold\_arrays\_par, 540  
 export\_cofold\_arrays, 536  
 export\_cofold\_arrays\_gq, 535  
 export\_fold\_arrays, 539  
 export\_fold\_arrays\_par, 540  
 fold, 538  
 fold\_par, 537  
 free\_alifold\_arrays, 542  
 free\_arrays, 539  
 free\_co\_arrays, 535  
 HairpinE, 541  
 initialize\_cofold, 537  
 initialize\_fold, 541  
 LoopEnergy, 540  
 update\_cofold\_params, 535  
 update\_cofold\_params\_par, 535  
 update\_fold\_params, 539  
 update\_fold\_params\_par, 539

Deprecated Interface for Global Partition Function Computation, 543  
 alipbacktrack, 556  
 alipf\_circ\_fold, 555  
 alipf\_fold, 554  
 alipf\_fold\_par, 544  
 assign plist\_from\_db, 554  
 assign plist\_from\_pr, 554  
 co\_pf\_fold, 551  
 co\_pf\_fold\_par, 551  
 compute\_probabilities, 552  
 export\_ali\_bppm, 555  
 export\_bppm, 548  
 export\_co\_bppm, 552  
 free\_alipf\_arrays, 556  
 free\_co\_pf\_arrays, 553  
 free\_pf\_arrays, 547  
 get\_alipf\_arrays, 556  
 get\_pf\_arrays, 549  
 init\_co\_pf\_fold, 552  
 init\_pf\_fold, 550  
 mean\_bp\_distance, 549  
 mean\_bp\_distance\_pr, 550  
 pf\_circ\_fold, 547  
 pf\_fold, 546  
 pf\_fold\_par, 545

stackProb, 550  
 update\_co\_pf\_params, 553  
 update\_co\_pf\_params\_par, 553  
 update\_pf\_params, 548  
 update\_pf\_params\_par, 548

Deprecated Interface for Local (Sliding Window) MFE Prediction, 542  
 Lfold, 542  
 Lfoldz, 542

Deprecated Interface for Local (Sliding Window) Partition Function Computation, 557  
 pfl\_fold, 558  
 putoutpU\_prob, 559  
 putoutpU\_prob\_bin, 559  
 update\_pf\_paramsLP, 558

Deprecated Interface for Multiple Sequence Alignment Utilities, 561  
 alloc\_sequence\_arrays, 563  
 encode\_ali\_sequence, 562  
 free\_sequence\_arrays, 563  
 get\_mpi, 562  
 pair\_info, 561

Deprecated Interface for Plotting Utilities, 572  
 aliPS\_color\_aln, 573  
 PS\_color\_aln, 572  
 rna\_plot\_type, 574  
 simple\_circplot\_coordinates, 573  
 simple\_xy\_coordinates, 573

Deprecated Interface for Secondary Structure Utilities, 564  
 add\_root, 566  
 alimake\_pair\_table, 570  
 b2C, 566  
 b2HIT, 565  
 b2Shapiro, 566  
 bp\_distance, 570  
 bppm\_symbol, 572  
 bppm\_to\_structure, 571  
 compute\_BPdifferences, 571  
 copy\_pair\_table, 569  
 expand\_Full, 567  
 expand\_Shapiro, 567  
 make\_pair\_table, 569  
 make\_pair\_table\_snoop, 570  
 make\_referenceBP\_array, 570  
 pack\_structure, 568  
 parenthesis\_structure, 571  
 parenthesis\_zuker, 571  
 parse\_structure, 568  
 unexpand\_aligned\_F, 568  
 unexpand\_Full, 567  
 unpack\_structure, 569  
 unweight, 567

Deprecated Interface for Stochastic Backtracking, 559  
 pbbacktrack, 560  
 pbbacktrack\_circ, 560  
 st\_back, 561  
 destroy\_TwoDfold\_variables

Computing MFE representatives of a Distance Based Partitioning, 308  
destroy\_TwoDpfold\_variables  
    2Dpfold.h, 584  
Direct Refolding Paths between two Secondary Structures, 338  
    vrna\_path\_direct, 342  
    vrna\_path\_direct\_ub, 342  
    vrna\_path\_findpath, 340  
    vrna\_path\_findpath\_saddle, 338  
    vrna\_path\_findpath\_saddle\_ub, 339  
    vrna\_path\_findpath\_ub, 340  
    vrna\_path\_options\_findpath, 341  
dist\_vars.h  
    cost\_matrix, 630  
    edit\_backtrack, 630  
Distance Based Partitioning of the Secondary Structure Space, 304  
Distance measures between Secondary Structures, 428  
    vrna\_bp\_distance, 429  
    vrna\_bp\_distance\_pt, 428  
do\_backtrack  
    Fine-tuning of the Implemented Models, 196  
Dot-Bracket Notation of Secondary Structures, 408  
    VRNA\_BRACKETS\_ALPHA, 409  
    VRNA\_BRACKETS\_ANG, 409  
    VRNA\_BRACKETS\_ANY, 410  
    VRNA\_BRACKETS\_CLY, 409  
    VRNA\_BRACKETS\_DEFAULT, 410  
    VRNA\_BRACKETS\_RND, 409  
    VRNA\_BRACKETS\_SQR, 409  
    vrna\_db\_flatten, 411  
    vrna\_db\_flatten\_to, 412  
    vrna\_db\_from plist, 412  
    vrna\_db\_from\_ptable, 412  
    vrna\_db\_pack, 410  
    vrna\_db\_pk\_remove, 413  
    vrna\_db\_to\_element\_string, 413  
    vrna\_db\_unpack, 411  
duplexT, 489  
dupVar, 489  
  
E\_Hairpin  
    Hairpin Loops, 374  
E\_IntLoop  
    Deprecated Interface for Free Energy Evaluation, 166  
E\_Stem  
    Deprecated Interface for Free Energy Evaluation, 164  
edit\_backtrack  
    dist\_vars.h, 630  
encode\_ali\_sequence  
    Deprecated Interface for Multiple Sequence Alignment Utilities, 562  
energy  
    vrna\_ht\_entry\_db\_t, 516  
Energy Evaluation for Atomic Moves, 155  
    vrna\_eval\_move, 155  
    vrna\_eval\_move\_pt, 156  
Energy Evaluation for Individual Loops, 153  
    vrna\_eval\_loop\_pt, 154  
    vrna\_eval\_loop\_pt\_v, 154  
Energy Parameters, 196  
    get\_boltzmann\_factor\_copy, 205  
    get\_boltzmann\_factors, 205  
    get\_boltzmann\_factors\_ali, 206  
    get\_scaled\_alipf\_parameters, 206  
    get\_scaled\_parameters, 206  
    get\_scaled\_pf\_parameters, 204  
    paramT, 199  
    pf\_paramT, 199  
    scale\_parameters, 206  
    vrna\_exp\_params, 200  
    vrna\_exp\_params\_comparative, 201  
    vrna\_exp\_params\_copy, 201  
    vrna\_exp\_params\_rescale, 202  
    vrna\_exp\_params\_reset, 204  
    vrna\_exp\_params\_subst, 202  
    vrna\_params, 199  
    vrna\_params\_copy, 200  
    vrna\_params\_reset, 203  
    vrna\_params\_subst, 202  
energy\_of\_alistruct  
    alifold.h, 590  
energy\_of\_circ\_struct  
    Deprecated Interface for Free Energy Evaluation, 164  
energy\_of\_circ\_struct\_par  
    Deprecated Interface for Free Energy Evaluation, 159  
energy\_of\_circ\_structure  
    Deprecated Interface for Free Energy Evaluation, 158  
energy\_of\_move  
    Deprecated Interface for Free Energy Evaluation, 161  
energy\_of\_move\_pt  
    Deprecated Interface for Free Energy Evaluation, 161  
energy\_of\_struct  
    Deprecated Interface for Free Energy Evaluation, 162  
energy\_of\_struct\_par  
    Deprecated Interface for Free Energy Evaluation, 158  
energy\_of\_struct\_pt  
    Deprecated Interface for Free Energy Evaluation, 163  
energy\_of\_struct\_pt\_par  
    Deprecated Interface for Free Energy Evaluation, 160  
energy\_of\_structure  
    Deprecated Interface for Free Energy Evaluation, 157  
energy\_of\_structure\_pt  
    Deprecated Interface for Free Energy Evaluation,

160

**energy\_set**  
    Fine-tuning of the Implemented Models, 196

**exp\_E\_ExtLoop**  
    Deprecated Interface for Free Energy Evaluation, 165

**exp\_E\_Hairpin**  
    Hairpin Loops, 375

**exp\_E\_IntLoop**  
    Deprecated Interface for Free Energy Evaluation, 167

**exp\_E\_Stem**  
    Deprecated Interface for Free Energy Evaluation, 165

**exp\_f**  
    vrna\_sc\_s, 239

**expand\_Full**  
    Deprecated Interface for Secondary Structure Utilities, 567

**expand\_Shapiro**  
    Deprecated Interface for Secondary Structure Utilities, 567

**Experimental Structure Probing Data**, 347

**expHairpinEnergy**  
    part\_func.h, 1192

**expLoopEnergy**  
    part\_func.h, 1192

**expMLbase**  
    vrna\_mx\_pf\_s, 512

**export\_ali\_bppm**  
    Deprecated Interface for Global Partition Function Computation, 555

**export\_bppm**  
    Deprecated Interface for Global Partition Function Computation, 548

**export\_circfold\_arrays**  
    Deprecated Interface for Global MFE Prediction, 540

**export\_circfold\_arrays\_par**  
    Deprecated Interface for Global MFE Prediction, 540

**export\_co\_bppm**  
    Deprecated Interface for Global Partition Function Computation, 552

**export\_cofold\_arrays**  
    Deprecated Interface for Global MFE Prediction, 536

**export\_cofold\_arrays\_gq**  
    Deprecated Interface for Global MFE Prediction, 535

**export\_fold\_arrays**  
    Deprecated Interface for Global MFE Prediction, 539

**export\_fold\_arrays\_par**  
    Deprecated Interface for Global MFE Prediction, 540

**Extending the Folding Grammar with Additional Domains**, 207

**Exterior Loops**, 369

    vrna\_E\_ext\_stem, 370

    vrna\_eval\_ext\_stem, 371

    vrna\_exp\_E\_ext\_stem, 371

    vrna\_mx\_pf\_aux\_el\_t, 370

**f**

    vrna\_sc\_s, 239

**filecopy**  
    basic.h, 896

**FILENAME\_ID\_LENGTH**  
    (Nucleic Acid Sequence) String Utilities, 397

**FILENAME\_MAX\_LENGTH**  
    (Nucleic Acid Sequence) String Utilities, 397

**Files and I/O**, 436

    vrna\_file\_exists, 438

    vrna\_filename\_sanitize, 437

    vrna\_read\_line, 437

**final\_cost**  
    Inverse Folding (Design), 324

**find\_saddle**  
    Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers, 575

**Fine-tuning of the Implemented Models**, 169

    backtrack\_type, 196

    dangles, 195

    do\_backtrack, 196

    energy\_set, 196

    max\_bp\_span, 196

    noLonelyPairs, 196

    nonstandards, 196

    pf\_scale, 195

    set\_model\_details, 194

    temperature, 195

    tetra\_loop, 195

    vrna\_md\_copy, 180

    vrna\_md\_defaults\_backtrack, 188

    vrna\_md\_defaults\_backtrack\_get, 188

    vrna\_md\_defaults\_backtrack\_type, 189

    vrna\_md\_defaults\_backtrack\_type\_get, 189

    vrna\_md\_defaults\_betaScale, 182

    vrna\_md\_defaults\_betaScale\_get, 182

    vrna\_md\_defaults\_circ, 186

    vrna\_md\_defaults\_circ\_get, 186

    vrna\_md\_defaults\_compute\_bpp, 189

    vrna\_md\_defaults\_compute\_bpp\_get, 190

    vrna\_md\_defaults\_cv\_fact, 193

    vrna\_md\_defaults\_cv\_fact\_get, 193

    vrna\_md\_defaults\_dangles, 182

    vrna\_md\_defaults\_dangles\_get, 183

    vrna\_md\_defaults\_energy\_set, 188

    vrna\_md\_defaults\_energy\_set\_get, 188

    vrna\_md\_defaults\_gquad, 186

    vrna\_md\_defaults\_gquad\_get, 187

    vrna\_md\_defaults\_logML, 185

    vrna\_md\_defaults\_logML\_get, 186

    vrna\_md\_defaults\_max\_bp\_span, 190

    vrna\_md\_defaults\_max\_bp\_span\_get, 190

    vrna\_md\_defaults\_min\_loop\_size, 190

vrna\_md\_defaults\_min\_loop\_size\_get, 191  
vrna\_md\_defaults\_nc\_fact, 193  
vrna\_md\_defaults\_nc\_fact\_get, 194  
vrna\_md\_defaults\_noGU, 184  
vrna\_md\_defaults\_noGU\_get, 184  
vrna\_md\_defaults\_noGUclosure, 185  
vrna\_md\_defaults\_noGUclosure\_get, 185  
vrna\_md\_defaults\_noLP, 184  
vrna\_md\_defaults\_noLP\_get, 184  
vrna\_md\_defaults\_oldAliEn, 192  
vrna\_md\_defaults\_oldAliEn\_get, 192  
vrna\_md\_defaults\_reset, 181  
vrna\_md\_defaults\_ribo, 192  
vrna\_md\_defaults\_ribo\_get, 192  
vrna\_md\_defaults\_sfact, 194  
vrna\_md\_defaults\_sfact\_get, 194  
vrna\_md\_defaults\_special\_hp, 183  
vrna\_md\_defaults\_special\_hp\_get, 183  
vrna\_md\_defaults\_temperature, 181  
vrna\_md\_defaults\_temperature\_get, 182  
vrna\_md\_defaults\_uniq\_ML, 187  
vrna\_md\_defaults\_uniq\_ML\_get, 187  
vrna\_md\_defaults\_window\_size, 191  
vrna\_md\_defaults\_window\_size\_get, 191  
vrna\_md\_option\_string, 181  
vrna\_md\_set\_default, 180  
vrna\_md\_update, 180  
VRNA\_MODEL\_DEFAULT\_ALI\_CV\_FACT, 179  
VRNA\_MODEL\_DEFAULT\_ALI\_NC\_FACT, 179  
VRNA\_MODEL\_DEFAULT\_ALI\_OLD\_EN, 179  
VRNA\_MODEL\_DEFAULT\_ALI\_RIBO, 179  
VRNA\_MODEL\_DEFAULT\_BACKTRACK, 178  
VRNA\_MODEL\_DEFAULT\_BACKTRACK\_TYPE, 178  
VRNA\_MODEL\_DEFAULT\_BETA\_SCALE, 176  
VRNA\_MODEL\_DEFAULT\_CIRC, 177  
VRNA\_MODEL\_DEFAULT\_COMPUTE\_BPP, 178  
VRNA\_MODEL\_DEFAULT\_DANGLES, 176  
VRNA\_MODEL\_DEFAULT\_ENERGY\_SET, 178  
VRNA\_MODEL\_DEFAULT\_GQUAD, 177  
VRNA\_MODEL\_DEFAULT\_LOG\_ML, 179  
VRNA\_MODEL\_DEFAULT\_MAX\_BP\_SPAN, 178  
VRNA\_MODEL\_DEFAULT\_NO\_GU, 177  
VRNA\_MODEL\_DEFAULT\_NO\_GU\_CLOSURE, 177  
VRNA\_MODEL\_DEFAULT\_NO\_LP, 177  
VRNA\_MODEL\_DEFAULT\_PF\_SCALE, 176  
VRNA\_MODEL\_DEFAULT\_SPECIAL\_HP, 177  
VRNA\_MODEL\_DEFAULT\_TEMPERATURE, 176  
VRNA\_MODEL\_DEFAULT\_UNIQ\_ML, 178  
VRNA\_MODEL\_DEFAULT\_WINDOW\_SIZE, 179

fold  
Deprecated Interface for Global MFE Prediction, 538

fold\_par  
Deprecated Interface for Global MFE Prediction, 537

fold\_vars.h

base\_pair, 661  
cut\_point, 661  
iindx, 661  
james\_rule, 661  
logML, 661  
pr, 661  
RibosumFile, 661

Folding Paths that start at a single Secondary Structure, 343

vrna\_path, 344  
VRNA\_PATH\_DEFAULT, 344  
vrna\_path\_gradient, 345  
VRNA\_PATH\_NO\_TRANSITION\_OUTPUT, 344  
VRNA\_PATH\_RANDOM, 344  
vrna\_path\_random, 346  
VRNA\_PATH\_STEEPEST\_DESCENT, 344

FORBIDDEN  
constants.h, 899

Free Energy Evaluation, 133

vrna\_eval\_circ\_consensus\_structure, 145  
vrna\_eval\_circ\_consensus\_structure\_v, 148  
vrna\_eval\_circ\_gquad\_consensus\_structure, 146  
vrna\_eval\_circ\_gquad\_consensus\_structure\_v, 150  
vrna\_eval\_circ\_gquad\_structure, 141  
vrna\_eval\_circ\_gquad\_structure\_v, 144  
vrna\_eval\_circ\_structure, 140  
vrna\_eval\_circ\_structure\_v, 143  
vrna\_eval\_consensus\_structure\_pt\_simple, 152  
vrna\_eval\_consensus\_structure\_pt\_simple\_v, 153  
vrna\_eval\_consensus\_structure\_pt\_simple\_verbose, 153  
vrna\_eval\_consensus\_structure\_simple, 145  
vrna\_eval\_consensus\_structure\_simple\_v, 148  
vrna\_eval\_consensus\_structure\_simple\_verbose, 147  
vrna\_eval\_covar\_structure, 136  
vrna\_eval\_gquad\_consensus\_structure, 146  
vrna\_eval\_gquad\_consensus\_structure\_v, 149  
vrna\_eval\_gquad\_structure, 140  
vrna\_eval\_gquad\_structure\_v, 143  
vrna\_eval\_structure, 136  
vrna\_eval\_structure\_pt, 138  
vrna\_eval\_structure\_pt\_simple, 151  
vrna\_eval\_structure\_pt\_simple\_v, 151  
vrna\_eval\_structure\_pt\_simple\_verbose, 151  
vrna\_eval\_structure\_pt\_v, 139  
vrna\_eval\_structure\_pt\_verbose, 138  
vrna\_eval\_structure\_simple, 139  
vrna\_eval\_structure\_simple\_v, 142  
vrna\_eval\_structure\_simple\_verbose, 142  
vrna\_eval\_structure\_v, 137  
vrna\_eval\_structure\_verbose, 137

free\_alifold\_arrays  
Deprecated Interface for Global MFE Prediction, 542

free\_alipf\_arrays

Deprecated Interface for Global Partition Function Computation, 556  
**free\_arrays**  
     Deprecated Interface for Global MFE Prediction, 539  
**free\_auxdata**  
     vrna\_fc\_s, 501  
**free\_co\_arrays**  
     Deprecated Interface for Global MFE Prediction, 535  
**free\_co\_pf\_arrays**  
     Deprecated Interface for Global Partition Function Computation, 553  
**free\_data**  
     vrna\_hc\_s, 231  
**free\_path**  
     Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers, 575  
**free\_pf\_arrays**  
     Deprecated Interface for Global Partition Function Computation, 547  
**free\_profile**  
     profiledist.h, 1228  
**free\_sequence\_arrays**  
     Deprecated Interface for Multiple Sequence Alignment Utilities, 563  
**free\_tree**  
     treedist.h, 1247  
**G-Quadruplexes**, 361  
     backtrack\_GQuad\_IntLoop, 362  
     backtrack\_GQuad\_IntLoop\_L, 362  
     get\_gquad\_matrix, 361  
     parse\_gquad, 362  
**GASCONST**  
     constants.h, 899  
**Generate Soft Constraints from Data**, 350  
     progress\_callback, 352  
     VRNA\_MINIMIZER\_CONJUGATE\_FR, 351  
     VRNA\_MINIMIZER\_CONJUGATE\_PR, 351  
     VRNA\_MINIMIZER\_STEEPEST\_DESCENT, 352  
     VRNA\_MINIMIZER\_VECTOR\_BFGS, 352  
     VRNA\_MINIMIZER\_VECTOR\_BFGS2, 352  
     VRNA\_OBJECTIVE\_FUNCTION\_ABSOLUTE, 351  
     VRNA\_OBJECTIVE\_FUNCTION\_QUADRATIC, 351  
     vrna\_sc\_minimize\_perturbation, 352  
**get\_alipf\_arrays**  
     Deprecated Interface for Global Partition Function Computation, 556  
**get\_boltzmann\_factor\_copy**  
     Energy Parameters, 205  
**get\_boltzmann\_factors**  
     Energy Parameters, 205  
**get\_boltzmann\_factors\_ali**  
     Energy Parameters, 206  
**get\_centroid\_struct\_gquad\_pr**  
     part\_func.h, 1191  
     get\_centroid\_struct\_pl  
         centroid.h, 598  
**get\_centroid\_struct\_pr**  
         centroid.h, 599  
**get\_concentrations**  
         concentrations.h, 608  
**get\_gquad\_matrix**  
         G-Quadruplexes, 361  
**get\_input\_line**  
         Utilities, 368  
**get\_line**  
         basic.h, 894  
**get\_mpi**  
         Deprecated Interface for Multiple Sequence Alignment Utilities, 562  
**get\_path**  
         Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers, 576  
**get\_pf\_arrays**  
         Deprecated Interface for Global Partition Function Computation, 549  
**get\_plist**  
         part\_func\_co.h, 1196  
**get\_scaled\_alipf\_parameters**  
         Energy Parameters, 206  
**get\_scaled\_parameters**  
         Energy Parameters, 206  
**get\_scaled\_pf\_parameters**  
         Energy Parameters, 204  
**get\_TwoDfold\_variables**  
         Computing MFE representatives of a Distance Based Partitioning, 308  
**get\_TwoDpfold\_variables**  
         2Dpfold.h, 584  
**give\_up**  
         Inverse Folding (Design), 324  
**Global MFE Prediction**, 249  
     vrna\_alifold, 252  
     vrna\_circalifold, 253  
     vrna\_circfold, 252  
     vrna\_cofold, 253  
     vrna\_fold, 251  
     vrna\_mfe, 250  
     vrna\_mfe\_dimer, 250  
**Global Partition Function and Equilibrium Probabilities**, 260  
     vrna\_pf, 262  
     vrna\_pf\_alifold, 264  
     vrna\_pf\_circalifold, 265  
     vrna\_pf\_circfold, 264  
     vrna\_pf\_co\_fold, 266  
     vrna\_pf\_dimer, 262  
     vrna\_pf\_fold, 263  
     vrna\_plist\_from\_probs, 266  
**gmlRNA**  
     Plotting, 459  
**Hairpin Loops**, 372  
     E\_Hairpin, 374

exp\_E\_Hairpin, 375  
vrna\_E\_ext\_hp\_loop, 373  
vrna\_E\_hp\_loop, 373  
vrna\_eval\_hp\_loop, 373  
vrna\_exp\_E\_hp\_loop, 376

HairpinE  
    Deprecated Interface for Global MFE Prediction, 541

hamming  
    strings.h, 1257

hamming\_bound  
    strings.h, 1258

Hard Constraints, 229  
    vrna\_callback\_hc\_evaluate, 234  
    VRNA\_CONSTRAINT\_DB, 231  
    VRNA\_CONSTRAINT\_DB\_DEFAULT, 233  
    VRNA\_CONSTRAINT\_DB\_DOT, 232  
    VRNA\_CONSTRAINT\_DB\_ENFORCE\_BP, 232  
    VRNA\_CONSTRAINT\_DB\_GQUAD, 233  
    VRNA\_CONSTRAINT\_DB\_INTERMOL, 233  
    VRNA\_CONSTRAINT\_DB\_INTRAMOL, 233  
    VRNA\_CONSTRAINT\_DB\_PIPE, 232  
    VRNA\_CONSTRAINT\_DB\_RND\_BRACK, 232  
    VRNA\_CONSTRAINT\_DB\_WUSS, 233  
    VRNA\_CONSTRAINT\_DB\_X, 232  
    vrna\_hc\_add\_bp, 236  
    vrna\_hc\_add\_bp\_nonspecific, 236  
    vrna\_hc\_add\_from\_db, 237  
    vrna\_hc\_add\_up, 235  
    vrna\_hc\_add\_up\_batch, 235  
    vrna\_hc\_free, 237  
    vrna\_hc\_init, 235

hard.h  
    constrain\_ptypes, 613  
    print\_tty\_constraint, 613  
    print\_tty\_constraint\_full, 613  
    VRNA\_CONSTRAINT\_DB\_ANG\_BRACK, 612  
    VRNA\_CONSTRAINT\_NO\_HEADER, 612  
    vrna\_hc\_add\_data, 612  
    VRNA\_HC\_DEFAULT, 612  
    vrna\_hc\_type\_e, 612  
    VRNA\_HC\_WINDOW, 612

Hash Tables, 514  
    vrna\_callback\_ht\_compare\_entries, 516  
    vrna\_callback\_ht\_free\_entry, 517  
    vrna\_callback\_ht\_hash\_function, 516  
    vrna\_hash\_table\_t, 516  
    vrna\_ht\_clear, 519  
    vrna\_ht\_collisions, 518  
    vrna\_ht\_db\_comp, 520  
    vrna\_ht\_db\_free\_entry, 521  
    vrna\_ht\_db\_hash\_func, 520  
    vrna\_ht\_free, 520  
    vrna\_ht\_get, 518  
    vrna\_ht\_init, 517  
    vrna\_ht\_insert, 519  
    vrna\_ht\_remove, 519  
    vrna\_ht\_size, 518

Heaps, 521  
    vrna\_callback\_heap\_cmp, 522  
    vrna\_callback\_heap\_get\_pos, 524  
    vrna\_callback\_heap\_set\_pos, 524  
    vrna\_heap\_free, 525  
    vrna\_heap\_init, 524  
    vrna\_heap\_insert, 526  
    vrna\_heap\_pop, 526  
    vrna\_heap\_remove, 527  
    vrna\_heap\_size, 525  
    vrna\_heap\_t, 522  
    vrna\_heap\_top, 526  
    vrna\_heap\_update, 527

Helix List Representation of Secondary Structures, 423  
    vrna\_hx\_from\_ptable, 423

id  
    vrna\_exp\_param\_s, 199

iindx  
    fold\_vars.h, 661

Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints, 354  
    vrna\_sc\_add\_hi\_motif, 355

INF  
    constants.h, 899

init\_co\_pf\_fold  
    Deprecated Interface for Global Partition Function Computation, 552

init\_pf\_fold  
    Deprecated Interface for Global Partition Function Computation, 550

init\_pf\_foldLP  
    LPfold.h, 708

init\_rand  
    basic.h, 895

initialize\_cofold  
    Deprecated Interface for Global MFE Prediction, 537

initialize\_fold  
    Deprecated Interface for Global MFE Prediction, 541

int\_urn  
    basic.h, 895

interact, 488

Internal Loops, 376  
    vrna\_eval\_int\_loop, 376

inv\_verbose  
    Inverse Folding (Design), 324

Inverse Folding (Design), 323  
    final\_cost, 324  
    give\_up, 324  
    inv\_verbose, 324  
    inverse\_fold, 323  
    inverse\_pf\_fold, 324

inverse\_fold  
    Inverse Folding (Design), 323

inverse\_pf\_fold  
    Inverse Folding (Design), 324

james\_rule  
 fold\_vars.h, 661

K0  
 constants.h, 899

last\_parameter\_file  
 Reading/Writing Energy Parameter Sets from/to File, 387

Layouts and Coordinates, 461  
 vrna\_plot\_coords, 466  
 vrna\_plot\_coords\_circular, 469  
 vrna\_plot\_coords\_circular\_pt, 470  
 vrna\_plot\_coords\_pt, 467  
 vrna\_plot\_coords\_puzzler, 470  
 vrna\_plot\_coords\_puzzler\_pt, 471  
 vrna\_plot\_coords\_simple, 468  
 vrna\_plot\_coords\_simple\_pt, 469  
 vrna\_plot\_coords\_turtle, 473  
 vrna\_plot\_coords\_turtle\_pt, 473  
 vrna\_plot\_layout, 463  
 vrna\_plot\_layout\_circular, 465  
 vrna\_plot\_layout\_free, 466  
 vrna\_plot\_layout\_puzzler, 466  
 vrna\_plot\_layout\_simple, 464  
 vrna\_plot\_layout\_t, 463  
 vrna\_plot\_layout\_turtle, 465  
 vrna\_plot\_options\_puzzler, 472  
 vrna\_plot\_options\_puzzler\_free, 472  
 VRNA\_PLOT\_TYPE\_CIRCULAR, 463  
 VRNA\_PLOT\_TYPE\_NAVIEW, 463  
 VRNA\_PLOT\_TYPE\_SIMPLE, 463

length  
 vrna\_mx\_pf\_s, 512

Lfold  
 Deprecated Interface for Local (Sliding Window) MFE Prediction, 542

Lfoldz  
 Deprecated Interface for Local (Sliding Window) MFE Prediction, 542

Ligands Binding to RNA Structures, 354

Ligands Binding to Unstructured Domains, 354

LIST, 577

Local (sliding window) MFE Prediction, 254  
 vrna\_Lfold, 257  
 vrna\_Lfoldz, 257  
 vrna\_mfe\_window, 256  
 vrna\_mfe\_window\_callback, 255  
 vrna\_mfe\_window\_zscore, 256

Local (sliding window) Partition Function and Equilibrium Probabilities, 267  
 vrna\_pfl\_fold, 271  
 vrna\_pfl\_fold\_cb, 272  
 vrna\_pfl\_fold\_up, 272  
 vrna\_pfl\_fold\_up\_cb, 273  
 vrna\_probs\_window, 270  
 VRNA\_PROBS\_WINDOW\_BPP, 268  
 vrna\_probs\_window\_callback, 269  
 VRNA\_PROBS\_WINDOW\_PF, 269

VRNA\_PROBS\_WINDOW\_STACKP, 268  
 VRNA\_PROBS\_WINDOW\_UP, 268  
 VRNA\_PROBS\_WINDOW\_UP\_SPLIT, 269

logML  
 fold\_vars.h, 661

loop\_energy  
 Deprecated Interface for Free Energy Evaluation, 162

LoopEnergy  
 Deprecated Interface for Global MFE Prediction, 540

LPfold.h  
 init\_pf\_foldLP, 708

LST\_BUCKET, 577

Make\_bp\_profile  
 profiledist.h, 1228

Make\_bp\_profile\_bppm  
 profiledist.h, 1228

make\_pair\_table  
 Deprecated Interface for Secondary Structure Utilities, 569

make\_pair\_table\_snoop  
 Deprecated Interface for Secondary Structure Utilities, 570

make\_referenceBP\_array  
 Deprecated Interface for Secondary Structure Utilities, 570

Make\_swString  
 stringdist.h, 1241

make\_tree  
 treedist.h, 1246

max\_bp\_span  
 Fine-tuning of the Implemented Models, 196

MAXLOOP  
 constants.h, 899

MEA  
 Compute the Structure with Maximum Expected Accuracy (MEA), 301

mean\_bp\_dist  
 part\_func.h, 1191

mean\_bp\_distance  
 Deprecated Interface for Global Partition Function Computation, 549

mean\_bp\_distance\_pr  
 Deprecated Interface for Global Partition Function Computation, 550

Messages, 490  
 vrna\_message\_error, 491  
 vrna\_message\_info, 492  
 vrna\_message\_input\_seq, 493  
 vrna\_message\_input\_seq\_simple, 493  
 vrna\_message\_verror, 491  
 vrna\_message\_vinfo, 493  
 vrna\_message\_vwarning, 492  
 vrna\_message\_warning, 492

min\_loop\_size  
 vrna\_md\_s, 176

Minimum Free Energy (MFE) Algorithms, 247

mm.h  
    vrna\_maximum\_matching, 714  
    vrna\_maximum\_matching\_simple, 714

Multibranch Loops, 377  
    vrna\_E\_mb\_loop\_stack, 378  
    vrna\_mx\_pf\_aux\_ml\_t, 377

Multiple Sequence Alignment Utilities, 429  
    vrna\_aln\_consensus\_mis, 435  
    vrna\_aln\_consensus\_sequence, 435  
    vrna\_aln\_conservation\_col, 434  
    vrna\_aln\_conservation\_struct, 434  
    vrna\_aln\_copy, 433  
    vrna\_aln\_free, 433  
    vrna\_aln\_mpi, 431  
    vrna\_aln\_pinfo, 432  
    vrna\_aln\_slice, 432  
    vrna\_aln\_toRNA, 433  
    vrna\_aln\_uppercase, 433  
    VRNA\_MEASURE\_SHANNON\_ENTROPY, 431

Multiple Sequence Alignments, 444  
    VRNA\_FILE\_FORMAT\_MSA\_APPEND, 447  
    VRNA\_FILE\_FORMAT\_MSA\_CLUSTAL, 445  
    VRNA\_FILE\_FORMAT\_MSA\_DEFAULT, 446  
    VRNA\_FILE\_FORMAT\_MSA\_FASTA, 445  
    VRNA\_FILE\_FORMAT\_MSA\_MAF, 446  
    VRNA\_FILE\_FORMAT\_MSA\_MIS, 446  
    VRNA\_FILE\_FORMAT\_MSA\_NOCHECK, 446  
    VRNA\_FILE\_FORMAT\_MSA QUIET, 447  
    VRNA\_FILE\_FORMAT\_MSA\_SILENT, 447  
    VRNA\_FILE\_FORMAT\_MSA STOCKHOLM, 445  
    VRNA\_FILE\_FORMAT\_MSA UNKNOWN, 446  
    vrna\_file\_msa\_detect\_format, 450  
    vrna\_file\_msa\_read, 447  
    vrna\_file\_msa\_read\_record, 448  
    vrna\_file\_msa\_write, 450

n\_seq  
    vrna\_fc\_s, 503

NBPAIRS  
    constants.h, 899

nc\_fact  
    alifold.h, 591

Neighborhood Relation and Move Sets for Secondary Structures, 324  
    vrna\_callback\_move\_update, 329  
    vrna\_loopidx\_update, 332  
    vrna\_move\_apply, 330  
    vrna\_move\_compare, 331  
    vrna\_move\_init, 330  
    vrna\_move\_is\_insertion, 331  
    vrna\_move\_is\_removal, 330  
    vrna\_move\_is\_shift, 331  
    vrna\_move\_list\_free, 330  
    vrna\_move\_neighor\_diff, 334  
    vrna\_move\_neighor\_diff\_cb, 333  
    VRNA\_MOVESET\_DEFAULT, 328  
    VRNA\_MOVESET\_DELETION, 328  
    VRNA\_MOVESET\_INSERTION, 328  
    VRNA\_MOVESET\_NO\_LP, 328

VRNA\_MOVESET\_SHIFT, 328  
VRNA\_NEIGHBOR\_CHANGE, 329  
VRNA\_NEIGHBOR\_INVALID, 329  
VRNA\_NEIGHBOR\_NEW, 329  
vrna\_neighbors, 332  
vrna\_neighbors\_successive, 333

node, 489

noLonelyPairs  
    Fine-tuning of the Implemented Models, 196

nonstandards  
    Fine-tuning of the Implemented Models, 196

nrerror  
    basic.h, 895

Nucleic Acid Sequences and Structures, 438  
    read\_record, 444  
    VRNA\_CONSTRAINT\_MULTILINE, 439  
    vrna\_extract\_record\_rest\_constraint, 443  
    vrna\_extract\_record\_rest\_structure, 442  
    vrna\_file\_bpseq, 440  
    vrna\_file\_connect, 440  
    vrna\_file\_fasta\_read\_record, 441  
    vrna\_file\_helixlist, 439  
    vrna\_file\_json, 441  
    vrna\_file\_SHAPE\_read, 443  
    VRNA\_OPTION\_MULTILINE, 439

pack\_structure  
    Deprecated Interface for Secondary Structure Utilities, 568

PAIR  
    (Abstract) Data Structures, 489

Pair List Representation of Secondary Structures, 419  
    vrna\_plist, 420

Pair Table Representation of Secondary Structures, 415  
    vrna\_pt\_pk\_get, 418  
    vrna\_pt\_pk\_remove, 419  
    vrna\_pt\_snoop\_get, 419  
    vrna\_ptable, 416  
    vrna\_ptable\_copy, 418  
    vrna\_ptable\_from\_string, 416

pair\_info  
    Deprecated Interface for Multiple Sequence Alignment Utilities, 561

paramT  
    Energy Parameters, 199

parenthesis\_structure  
    Deprecated Interface for Secondary Structure Utilities, 571

parenthesis\_zuker  
    Deprecated Interface for Secondary Structure Utilities, 571

parse\_gquad  
    G-Quadruplexes, 362

parse\_structure  
    Deprecated Interface for Secondary Structure Utilities, 568

part\_func.h  
    centroid, 1191  
    expHairpinEnergy, 1192

expLoopEnergy, 1192  
 get\_centroid\_struct\_gquad\_pr, 1191  
 mean\_bp\_dist, 1191  
**part\_func\_co.h**  
 get\_plist, 1196  
 Partition Function and Equilibrium Properties, 248  
 vrna\_pf\_float\_precision, 248  
 Partition Function for Two Hybridized Sequences, 378  
 vrna\_pf\_co\_fold, 379  
 vrna\_pf\_dimer\_concentrations, 380  
 Partition Function for two Hybridized Sequences as a Stepwise Process, 380  
 pf\_interact, 381  
 pf\_unstru, 381  
**path\_t**  
 Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers, 575  
**pbacktrack**  
 Deprecated Interface for Stochastic Backtracking, 560  
**pbacktrack\_circ**  
 Deprecated Interface for Stochastic Backtracking, 560  
**pf\_circ\_fold**  
 Deprecated Interface for Global Partition Function Computation, 547  
**pf\_fold**  
 Deprecated Interface for Global Partition Function Computation, 546  
**pf\_fold\_par**  
 Deprecated Interface for Global Partition Function Computation, 545  
**pf\_interact**  
 Partition Function for two Hybridized Sequences as a Stepwise Process, 381  
**pf\_paramT**  
 Energy Parameters, 199  
**pf\_scale**  
 Fine-tuning of the Implemented Models, 195  
**pf\_unstru**  
 Partition Function for two Hybridized Sequences as a Stepwise Process, 381  
**pfl\_fold**  
 Deprecated Interface for Local (Sliding Window) Partition Function Computation, 558  
**plist**  
 (Abstract) Data Structures, 489  
**Plotting, 455**  
 gmlRNA, 459  
 PS\_dot\_plot, 456  
 PS\_dot\_plot\_list, 456  
 PS\_rna\_plot, 460  
 PS\_rna\_plot\_a, 460  
 PS\_rna\_plot\_a\_gquad, 461  
 ssv\_rna\_plot, 459  
 svg\_rna\_plot, 460  
 vrna\_file\_PS\_rnplot, 457  
 vrna\_file\_PS\_rnplot\_a, 457  
 xrna\_plot, 460  
**Postorder\_list, 577**  
**pr**  
 fold\_vars.h, 661  
 Predicting various thermodynamic properties, 313  
 vrna\_ensemble\_defect, 317  
 vrna\_ensemble\_defect\_pt, 316  
 vrna\_heat\_capacity, 320  
 vrna\_heat\_capacity\_callback, 315  
 vrna\_heat\_capacity\_cb, 321  
 vrna\_heat\_capacity\_simple, 321  
 vrna\_heat\_capacity\_t, 315  
 vrna\_mean\_bp\_distance, 316  
 vrna\_mean\_bp\_distance\_pr, 315  
 vrna\_pf\_dimer\_probs, 319  
 vrna\_positional\_entropy, 318  
 vrna\_pr\_energy, 320  
 vrna\_pr\_structure, 319  
 vrna\_stack\_prob, 318  
**print\_tty\_constraint**  
 hard.h, 613  
**print\_tty\_constraint\_full**  
 hard.h, 613  
**print\_tty\_input\_seq**  
 basic.h, 894  
**print\_tty\_input\_seq\_str**  
 basic.h, 894  
**prod\_cb**  
 vrna\_unstructured\_domain\_s, 210  
**profile\_edit\_distance**  
 profiledist.h, 1227  
**profiledist.h**  
 free\_profile, 1228  
 Make\_bp\_profile, 1228  
 Make\_bp\_profile\_bppm, 1228  
 profile\_edit\_distance, 1227  
**progress\_callback**  
 Generate Soft Constraints from Data, 352  
**PS\_color\_aln**  
 Deprecated Interface for Plotting Utilities, 572  
**PS\_dot\_plot**  
 Plotting, 456  
**PS\_dot\_plot\_list**  
 Plotting, 456  
**PS\_rna\_plot**  
 Plotting, 460  
**PS\_rna\_plot\_a**  
 Plotting, 460  
**PS\_rna\_plot\_a\_gquad**  
 Plotting, 461  
**pscore**  
 vrna\_fc\_s, 504  
**pscore\_local**  
 vrna\_fc\_s, 504  
**pscore\_pf\_compat**  
 vrna\_fc\_s, 504  
**Pseudoknots, 356**  
 vrna\_callback\_pk\_plex\_score, 358

vrna\_pk\_plex, 359  
vrna\_pk\_plex\_accessibility, 359  
vrna\_pk\_plex\_opt, 360  
vrna\_pk\_plex\_opt\_defaults, 360  
vrna\_pk\_plex\_opt\_fun, 360  
vrna\_pk\_plex\_opt\_t, 358  
vrna\_pk\_plex\_t, 358

ptype  
  vrna\_fc\_s, 502

ptype\_pf\_compat  
  vrna\_fc\_s, 502

pu\_contrib, 487

pu\_out, 488

putoutpU\_prob  
  Deprecated Interface for Local (Sliding Window)  
    Partition Function Computation, 559

putoutpU\_prob\_bin  
  Deprecated Interface for Local (Sliding Window)  
    Partition Function Computation, 559

Random Structure Samples from the Ensemble, 279  
  vrna\_boltzmann\_sampling\_callback, 281  
  vrna\_pbacktrack, 288  
  vrna\_pbacktrack5, 282  
  vrna\_pbacktrack5\_cb, 284  
  vrna\_pbacktrack5\_num, 283  
  vrna\_pbacktrack5\_resume, 285  
  vrna\_pbacktrack5\_resume\_cb, 286  
  vrna\_pbacktrack\_cb, 289  
  VRNA\_PBACKTRACK\_DEFAULT, 281  
  vrna\_pbacktrack\_mem\_free, 299  
  vrna\_pbacktrack\_mem\_t, 281  
  VRNA\_PBACKTRACK\_NON\_REDUNDANT, 281  
  vrna\_pbacktrack\_num, 288  
  vrna\_pbacktrack\_resume, 290  
  vrna\_pbacktrack\_resume\_cb, 292  
  vrna\_pbacktrack\_sub, 293  
  vrna\_pbacktrack\_sub\_cb, 295  
  vrna\_pbacktrack\_sub\_num, 294  
  vrna\_pbacktrack\_sub\_resume, 296  
  vrna\_pbacktrack\_sub\_resume\_cb, 298

random\_string  
  strings.h, 1257

read\_parameter\_file  
  Reading/Writing Energy Parameter Sets from/to  
    File, 388

read\_record  
  Nucleic Acid Sequences and Structures, 444

Reading/Writing Energy Parameter Sets from/to File,  
  382

  last\_parameter\_file, 387  
  read\_parameter\_file, 388  
  VRNA\_PARAMETER\_FORMAT\_DEFAULT, 383  
  vrna\_params\_load, 383  
  vrna\_params\_load\_defaults, 385  
  vrna\_params\_load\_DNA\_Mathews1999, 387  
  vrna\_params\_load\_DNA\_Mathews2004, 387  
  vrna\_params\_load\_from\_string, 384  
  vrna\_params\_load\_RNA\_Andronescu2007, 386

  vrna\_params\_load\_RNA\_Langdon2018, 386  
  vrna\_params\_load\_RNA\_misc\_special\_hairpins,  
    386  
  vrna\_params\_load\_RNA\_Turner1999, 385  
  vrna\_params\_load\_RNA\_Turner2004, 385  
  vrna\_params\_save, 384  
  write\_parameter\_file, 388

RibosumFile  
  fold\_vars.h, 661

RNA-RNA Interaction, 303

rna\_plot\_type  
  Deprecated Interface for Plotting Utilities, 574

S  
  vrna\_fc\_s, 503

S3  
  vrna\_fc\_s, 503

S5  
  vrna\_fc\_s, 503

S\_cons  
  vrna\_fc\_s, 503

sc  
  vrna\_fc\_s, 502

scale  
  vrna\_mx\_pf\_s, 512

scale\_parameters  
  Energy Parameters, 206

scs  
  vrna\_fc\_s, 504

Search Algorithms, 476  
  vrna\_search\_BM\_BCT, 478  
  vrna\_search\_BM\_BCT\_num, 478  
  vrna\_search\_BMH, 477  
  vrna\_search\_BMH\_num, 476

Secondary Structure Utilities, 406  
  vrna\_db\_from\_bp\_stack, 407  
  vrna\_db\_from\_probs, 407  
  vrna\_refBPcnt\_matrix, 407  
  vrna\_refBPdist\_matrix, 407

sect  
  (Abstract) Data Structures, 489

sequence  
  vrna\_fc\_s, 501

sequence\_encoding  
  vrna\_fc\_s, 502

sequences  
  vrna\_fc\_s, 502

set\_model\_details  
  Fine-tuning of the Implemented Models, 194

SHAPE Reactivity Data, 347  
  vrna\_sc\_add\_SHAPE\_deigan, 348  
  vrna\_sc\_add\_SHAPE\_deigan\_ali, 348  
  vrna\_sc\_add\_SHAPE\_zarringhalam, 349  
  vrna\_sc\_SHAPE\_to\_pr, 350

SHAPE.h  
  vrna\_sc\_SHAPE\_parse\_method, 618

simple\_circplot\_coordinates  
  Deprecated Interface for Plotting Utilities, 573

simple\_xy\_coordinates

Deprecated Interface for Plotting Utilities, 573  
**snoopT**, 489  
**Soft Constraints**, 237  
 vrna\_callback\_sc\_backtrack, 241  
 vrna\_callback\_sc\_energy, 240  
 vrna\_callback\_sc\_exp\_energy, 240  
 vrna\_sc\_add\_bp, 243  
 vrna\_sc\_add\_bt, 246  
 vrna\_sc\_add\_data, 245  
 vrna\_sc\_add\_exp\_f, 246  
 vrna\_sc\_add\_f, 245  
 vrna\_sc\_add\_up, 244  
 vrna\_sc\_free, 245  
 vrna\_sc\_init, 242  
 vrna\_sc\_remove, 244  
 vrna\_sc\_set\_bp, 242  
 vrna\_sc\_set\_up, 243  
**soft.h**  
 VRNA\_SC\_DEFAULT, 620  
 vrna\_sc\_type\_e, 620  
 VRNA\_SC\_WINDOW, 621  
**SOLUTION**  
 subopt.h, 1244  
**space**  
 basic.h, 895  
**ssv\_rna\_plot**  
 Plotting, 459  
**st\_back**  
 Deprecated Interface for Stochastic Backtracking, 561  
**stackProb**  
 Deprecated Interface for Global Partition Function Computation, 550  
**stat\_cb**  
 vrna\_fc\_s, 501  
**Stochastic Backtracking of Structures from Distance Based Partitioning**, 311  
 vrna\_pbacktrack5\_TwoD, 312  
 vrna\_pbacktrack\_TwoD, 312  
**str\_DNA2RNA**  
 strings.h, 1257  
**str\_uppercase**  
 strings.h, 1257  
**strands**  
 vrna\_mx\_mfe\_s, 511  
**string\_edit\_distance**  
 stringdist.h, 1241  
**stringdist.h**  
 Make\_swString, 1241  
 string\_edit\_distance, 1241  
**strings.h**  
 hamming, 1257  
 hamming\_bound, 1258  
 random\_string, 1257  
 str\_DNA2RNA, 1257  
 str\_uppercase, 1257  
**structure**  
 vrna\_ht\_entry\_db\_t, 515

Structure Modules and Pseudoknots, 356  
**Structured Domains**, 216  
**subopt**  
 Suboptimal Structures within an Energy Band around the MFE, 278  
**subopt.h**  
 SOLUTION, 1244  
**subopt\_circ**  
 Suboptimal Structures within an Energy Band around the MFE, 279  
 Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989, 274  
 vrna\_subopt\_zuker, 275  
 zukersubopt, 275  
 zukersubopt\_par, 275  
 Suboptimal Structures within an Energy Band around the MFE, 276  
 subopt, 278  
 subopt\_circ, 279  
 vrna\_subopt, 277  
 vrna\_subopt\_callback, 276  
 vrna\_subopt\_cb, 277  
 Suboptimals and Representative Structures, 274  
**svg\_rna\_plot**  
 Plotting, 460  
**swString**, 577  
**temperature**  
 Fine-tuning of the Implemented Models, 195  
**tetra\_loop**  
 Fine-tuning of the Implemented Models, 195  
**The Dynamic Programming Matrices**, 510  
 VRNA\_MX\_2DFOLD, 513  
 vrna\_mx\_add, 513  
 VRNA\_MX\_DEFAULT, 513  
 vrna\_mx\_mfe\_free, 514  
 vrna\_mx\_pf\_free, 514  
 vrna\_mx\_type\_e, 512  
 VRNA\_MX\_WINDOW, 513  
**The Fold Compound**, 497  
 vrna\_callback\_free\_auxdata, 506  
 vrna\_callback\_recursion\_status, 506  
 VRNA\_FC\_TYPE\_COMPARATIVE, 507  
 vrna\_fc\_type\_e, 507  
 VRNA\_FC\_TYPE\_SINGLE, 507  
 vrna\_fold\_compound, 507  
 vrna\_fold\_compound\_add\_auxdata, 509  
 vrna\_fold\_compound\_add\_callback, 510  
 vrna\_fold\_compound\_comparative, 508  
 vrna\_fold\_compound\_free, 509  
 VRNA\_OPTION\_EVAL\_ONLY, 505  
 VRNA\_OPTION\_MFE, 505  
 VRNA\_OPTION\_PF, 505  
 VRNA\_STATUS\_MFE\_POST, 504  
 VRNA\_STATUS\_MFE\_PRE, 504  
 VRNA\_STATUS\_PF\_POST, 505  
 VRNA\_STATUS\_PF\_PRE, 505  
 The RNA Folding Grammar, 168  
 The RNA Secondary Structure Landscape, 247

time\_stamp  
  basic.h, 896

Tree, 578

Tree Representation of Secondary Structures, 424  
  vrna\_db\_to\_tree\_string, 426  
  VRNA\_STRUCTURE\_TREE\_EXPANDED, 426  
  VRNA\_STRUCTURE\_TREE\_HIT, 425  
  VRNA\_STRUCTURE\_TREE\_SHAPIRO, 426  
  VRNA\_STRUCTURE\_TREE\_SHAPIRO\_EXT, 426  
  VRNA\_STRUCTURE\_TREE\_SHAPIRO\_SHORT, 425  
  VRNA\_STRUCTURE\_TREE\_SHAPIRO\_WEIGHT, 426  
  vrna\_tree\_string\_to\_db, 428  
  vrna\_tree\_string\_unweight, 427

tree\_edit\_distance  
  treedist.h, 1246

treedist.h  
  free\_tree, 1247  
  make\_tree, 1246  
  tree\_edit\_distance, 1246

TURN  
  constants.h, 899

TwoDfold\_backtrack\_f5  
  Computing MFE representatives of a Distance Based Partitioning, 309

TwoDfold\_vars, 305  
  Computing MFE representatives of a Distance Based Partitioning, 306

TwoDfoldList  
  Computing MFE representatives of a Distance Based Partitioning, 309

TwoDpfold\_pbacktrack  
  2Dpfold.h, 585

TwoDpfold\_pbacktrack5  
  2Dpfold.h, 586

TwoDpfold\_vars, 578

TwoDpfoldList  
  2Dpfold.h, 585

type  
  vrna\_fc\_s, 501  
  vrna\_mx\_mfe\_s, 511  
  vrna\_mx\_pf\_s, 512  
  vrna\_path\_s, 337

unexpand\_aligned\_F  
  Deprecated Interface for Secondary Structure Utilities, 568

unexpand\_Full  
  Deprecated Interface for Secondary Structure Utilities, 567

Unit Conversion, 494  
  vrna\_convert\_dcal\_to\_kcal, 497  
  vrna\_convert\_energy, 495  
  vrna\_convert\_kcal\_to\_dcal, 496  
  vrna\_convert\_temperature, 496  
  VRNA\_UNIT\_CAL, 495  
  VRNA\_UNIT\_CAL\_IT, 495  
  VRNA\_UNIT\_DACAL, 495  
  VRNA\_UNIT\_DACAL\_IT, 495  
  VRNA\_UNIT\_DEG\_C, 495  
  VRNA\_UNIT\_DEG\_DE, 495  
  VRNA\_UNIT\_DEG\_F, 495  
  VRNA\_UNIT\_DEG\_N, 495  
  VRNA\_UNIT\_DEG\_R, 495  
  VRNA\_UNIT\_DEG\_RE, 495  
  VRNA\_UNIT\_DEG\_RO, 495  
  vrna\_unit\_energy\_e, 494  
  VRNA\_UNIT\_EV, 495  
  VRNA\_UNIT\_G\_TNT, 495  
  VRNA\_UNIT\_J, 494  
  VRNA\_UNIT\_K, 495  
  VRNA\_UNIT\_KCAL, 495  
  VRNA\_UNIT\_KCAL\_IT, 495  
  VRNA\_UNIT\_KG\_TNT, 495  
  VRNA\_UNIT\_KJ, 495  
  VRNA\_UNIT\_KWH, 495  
  VRNA\_UNIT\_T\_TNT, 495  
  vrna\_unit\_temperature\_e, 495  
  VRNA\_UNIT\_WH, 495

unpack\_structure  
  Deprecated Interface for Secondary Structure Utilities, 569

Unstructured Domains, 207  
  vrna\_callback\_ud\_energy, 210  
  vrna\_callback\_ud\_exp\_energy, 210  
  vrna\_callback\_ud\_exp\_production, 211  
  vrna\_callback\_ud\_probs\_add, 211  
  vrna\_callback\_ud\_probs\_get, 211  
  vrna\_callback\_ud\_production, 211  
  vrna\_ud\_add\_motif, 213  
  vrna\_ud\_motifs\_centroid, 211  
  vrna\_ud\_motifs\_MEA, 212  
  vrna\_ud\_motifs\_MFE, 212  
  vrna\_ud\_remove, 214  
  vrna\_ud\_set\_data, 214  
  vrna\_ud\_set\_exp\_prod\_rule\_cb, 215  
  vrna\_ud\_set\_prod\_rule\_cb, 214

unstructured\_domains.h  
  vrna\_ud\_set\_prob\_cb, 1251

unweight  
  Deprecated Interface for Secondary Structure Utilities, 567

update\_alifold\_params  
  alifold.h, 591

update\_co\_pf\_params  
  Deprecated Interface for Global Partition Function Computation, 553

update\_co\_pf\_params\_par  
  Deprecated Interface for Global Partition Function Computation, 553

update\_cofold\_params  
  Deprecated Interface for Global MFE Prediction, 535

update\_cofold\_params\_par  
  Deprecated Interface for Global MFE Prediction, 535

update\_fold\_params  
 Deprecated Interface for Global MFE Prediction, 539

update\_fold\_params\_par  
 Deprecated Interface for Global MFE Prediction, 539

update\_pf\_params  
 Deprecated Interface for Global Partition Function Computation, 548

update\_pf\_params\_par  
 Deprecated Interface for Global Partition Function Computation, 548

update\_pf\_paramsLP  
 Deprecated Interface for Local (Sliding Window) Partition Function Computation, 558

urn  
 basic.h, 895

Utilities, 363  
 get\_input\_line, 368  
 vrna\_alloc, 366  
 vrna\_idx\_col\_wise, 369  
 vrna\_idx\_row\_wise, 368  
 vrna\_init\_rand, 366  
 vrna\_init\_rand\_seed, 366  
 VRNA\_INPUT\_FASTA\_HEADER, 365  
 vrna\_int\_urn, 367  
 vrna\_realloc, 366  
 vrna\_time\_stamp, 368  
 vrna\_urn, 367  
 xsubi, 369

Utilities to deal with Nucleotide Alphabets, 392  
 vrna\_nucleotide\_decode, 395  
 vrna\_nucleotide\_encode, 395  
 vrna\_ptypes, 394  
 VRNA\_SEQ\_DNA, 394  
 vrna\_seq\_encode, 394  
 VRNA\_SEQ\_RNA, 394  
 vrna\_seq\_type\_e, 394  
 VRNA\_SEQ\_UNKNOWN, 394

ViennaRNA/2Dfold.h, 581  
 ViennaRNA/2Dpfold.h, 583, 587  
 ViennaRNA/ali\_plex.h, 589  
 ViennaRNA/alifold.h, 589, 592  
 ViennaRNA/aln\_util.h, 593  
 ViennaRNA/alphabet.h, 593, 594  
 ViennaRNA/boltzmann\_sampling.h, 595, 596  
 ViennaRNA/centroid.h, 598, 599  
 ViennaRNA/char\_stream.h, 600  
 ViennaRNA/cofold.h, 603  
 ViennaRNA/combinatorics.h, 604, 605  
 ViennaRNA/commands.h, 606  
 ViennaRNA/concentrations.h, 607, 608  
 ViennaRNA/constraints.h, 609  
 ViennaRNA/constraints/basic.h, 882, 883  
 ViennaRNA/constraints/hard.h, 610, 614  
 ViennaRNA/constraints/ligand.h, 617  
 ViennaRNA/constraints/SHAPE.h, 618, 619  
 ViennaRNA/constraints/soft.h, 619, 621

ViennaRNA/constraints\_hard.h, 623  
 ViennaRNA/constraints\_ligand.h, 623, 624  
 ViennaRNA/constraints\_SHAPE.h, 624  
 ViennaRNA/constraints\_soft.h, 624  
 ViennaRNA/convert\_epars.h, 625  
 ViennaRNA/data\_structures.h, 625  
 ViennaRNA/datastructures/basic.h, 883, 885  
 ViennaRNA/datastructures/char\_stream.h, 600  
 ViennaRNA/datastructures/hash\_tables.h, 626, 627  
 ViennaRNA/datastructures/heap.h, 628  
 ViennaRNA/datastructures/lists.h, 629  
 ViennaRNA/datastructures/stream\_output.h, 1239  
 ViennaRNA/dist\_vars.h, 630, 631  
 ViennaRNA/dp\_matrices.h, 631, 632  
 ViennaRNA/duplex.h, 635  
 ViennaRNA/edit\_cost.h, 636  
 ViennaRNA/energy\_const.h, 637  
 ViennaRNA/energy\_par.h, 637, 638  
 ViennaRNA/equilibrium\_probs.h, 638, 639  
 ViennaRNA/eval.h, 639, 642  
 ViennaRNA/exterior\_loops.h, 646, 647  
 ViennaRNA/file\_formats.h, 647  
 ViennaRNA/file\_formats\_msa.h, 649, 650  
 ViennaRNA/file\_utils.h, 652  
 ViennaRNA/findpath.h, 652  
 ViennaRNA/fold.h, 654, 655  
 ViennaRNA/fold\_compound.h, 656, 658  
 ViennaRNA/fold\_vars.h, 660, 661  
 ViennaRNA/gquad.h, 662  
 ViennaRNA/grammar.h, 682  
 ViennaRNA/hairpin\_loops.h, 683  
 ViennaRNA/heat\_capacity.h, 684  
 ViennaRNA/interior\_loops.h, 685  
 ViennaRNA/inverse.h, 685, 686  
 ViennaRNA/io/file\_formats.h, 647, 648  
 ViennaRNA/io/file\_formats\_msa.h, 650, 651  
 ViennaRNA/io/utils.h, 1253  
 ViennaRNA/landscape/findpath.h, 653  
 ViennaRNA/landscape/move.h, 686, 687  
 ViennaRNA/landscape/neighbor.h, 726  
 ViennaRNA/landscape/paths.h, 688, 689  
 ViennaRNA/landscape/walk.h, 1261, 1262  
 ViennaRNA/Lfold.h, 690  
 ViennaRNA/loop\_energies.h, 690, 691  
 ViennaRNA/loops/all.h, 691  
 ViennaRNA/loops/external.h, 691, 692  
 ViennaRNA/loops/hairpin.h, 694  
 ViennaRNA/loops/internal.h, 697  
 ViennaRNA/loops/multibranch.h, 705  
 ViennaRNA/LPfold.h, 707, 708  
 ViennaRNA/MEA.h, 709, 710  
 ViennaRNA/mfe.h, 710, 711  
 ViennaRNA/mfe\_window.h, 712, 713  
 ViennaRNA/mm.h, 714, 715  
 ViennaRNA/model.h, 715, 719  
 ViennaRNA/move\_set.h, 723  
 ViennaRNA/multibranch\_loops.h, 725  
 ViennaRNA/naview.h, 725

ViennaRNA/neighbor.h, 727  
ViennaRNA/pair\_mat.h, 728  
ViennaRNA/params.h, 730  
ViennaRNA/params/1.8.4\_epars.h, 730  
ViennaRNA/params/1.8.4\_intloops.h, 734, 735  
ViennaRNA/params/basic.h, 888, 889  
ViennaRNA/params/constants.h, 898, 900  
ViennaRNA/params/convert.h, 900, 901  
ViennaRNA/params/default.h, 901  
ViennaRNA/params/intl11.h, 902  
ViennaRNA/params/intl11dH.h, 907  
ViennaRNA/params/intl21.h, 911  
ViennaRNA/params/intl21dH.h, 934  
ViennaRNA/params/intl22.h, 957  
ViennaRNA/params/intl22dH.h, 1072  
ViennaRNA/params/io.h, 1187, 1188  
ViennaRNA/part\_func.h, 1189, 1192  
ViennaRNA/part\_func\_co.h, 1195, 1196  
ViennaRNA/part\_func\_up.h, 1197  
ViennaRNA/part\_func\_window.h, 1198, 1199  
ViennaRNA/perturbation\_fold.h, 1200, 1201  
ViennaRNA/pf\_multifold.h, 1202  
ViennaRNA/pk\_plex.h, 1202, 1203  
ViennaRNA/PKplex.h, 1203  
ViennaRNA/plex.h, 1204  
ViennaRNA/plot\_aln.h, 1205  
ViennaRNA/plot\_layouts.h, 1205  
ViennaRNA/plot\_structure.h, 1205, 1206  
ViennaRNA/plot\_utils.h, 1206  
ViennaRNA/plotting alignments.h, 1206, 1207  
ViennaRNA/plotting/layouts.h, 1211, 1212  
ViennaRNA/plotting/probabilities.h, 1214  
ViennaRNA/plotting/RNApuzzler/RNApuzzler.h, 1216  
ViennaRNA/plotting/RNApuzzler/RNAturtle.h, 1217  
ViennaRNA/plotting/structures.h, 1218  
ViennaRNA/plotting/utils.h, 1254  
ViennaRNA/ProfileAln.h, 1227  
ViennaRNA/profiledist.h, 1227, 1228  
ViennaRNA/PS\_dot.h, 1229  
ViennaRNA/read\_epars.h, 1229, 1230  
ViennaRNA/ribo.h, 1230  
ViennaRNA/RNAstruct.h, 1230, 1231  
ViennaRNA/search/BoyerMoore.h, 1232, 1233  
ViennaRNA/sequence.h, 1233, 1234  
ViennaRNA/snfold.h, 1235  
ViennaRNA/snoop.h, 1236  
ViennaRNA/special\_const.h, 1239  
ViennaRNA/stream\_output.h, 1240  
ViennaRNA/string\_utils.h, 1240  
ViennaRNA/stringdist.h, 1241, 1242  
ViennaRNA/structure\_utils.h, 1242  
ViennaRNA/structured\_domains.h, 1242  
ViennaRNA/subopt.h, 1243, 1244  
ViennaRNA/subopt\_zuker.h, 1245  
ViennaRNA/svm\_utils.h, 1245  
ViennaRNA/treedist.h, 1246, 1247  
ViennaRNA/units.h, 1247  
ViennaRNA/unstructured\_domains.h, 1249, 1251  
ViennaRNA/utils.h, 1254, 1255  
ViennaRNA/utils/alignments.h, 1207, 1209  
ViennaRNA/utils/basic.h, 892, 896  
ViennaRNA/utils/cpu.h, 1255  
ViennaRNA/utils/higher\_order\_functions.h, 1255  
ViennaRNA/utils/strings.h, 1255, 1258  
ViennaRNA/utils/structures.h, 1219, 1223  
ViennaRNA/utils/svm.h, 1260  
ViennaRNA/utils/units.h, 1248  
ViennaRNA/vrna\_config.h, 1260  
ViennaRNA/walk.h, 1263  
ViennaRNA/wrap\_dlib.h, 1263  
ViennaRNA/zscore.h, 1263  
vrna\_abstract\_shapes  
    Abstract Shapes Representation of Secondary Structures, 422  
vrna\_abstract\_shapes\_pt  
    Abstract Shapes Representation of Secondary Structures, 422  
vrna\_alifold  
    Global MFE Prediction, 252  
vrna\_alignment\_s, 393  
vrna\_alloc  
    Utilities, 366  
vrna\_aln\_consensus\_mis  
    Multiple Sequence Alignment Utilities, 435  
vrna\_aln\_consensus\_sequence  
    Multiple Sequence Alignment Utilities, 435  
vrna\_aln\_conservation\_col  
    Multiple Sequence Alignment Utilities, 434  
vrna\_aln\_conservation\_struct  
    Multiple Sequence Alignment Utilities, 434  
vrna\_aln\_copy  
    Multiple Sequence Alignment Utilities, 433  
vrna\_aln\_free  
    Multiple Sequence Alignment Utilities, 433  
vrna\_aln\_mpi  
    Multiple Sequence Alignment Utilities, 431  
vrna\_aln\_pinfo  
    Multiple Sequence Alignment Utilities, 432  
vrna\_aln\_slice  
    Multiple Sequence Alignment Utilities, 432  
vrna\_aln\_toRNA  
    Multiple Sequence Alignment Utilities, 433  
vrna\_aln\_uppercase  
    Multiple Sequence Alignment Utilities, 433  
vrna\_backtrack5  
    Backtracking MFE structures, 259  
vrna\_backtrack5\_TwoD  
    Computing MFE representatives of a Distance Based Partitioning, 307  
vrna\_basepair\_s, 487  
vrna\_boltzmann\_sampling\_callback  
    Random Structure Samples from the Ensemble, 281  
vrna\_boustrophedon  
    Combinatorics Algorithms, 484  
vrna\_boustrophedon\_pos

Combinatorics Algorithms, 485

`vrna_bp_distance`  
Distance measures between Secondary Structures, 429

`vrna_bp_distance_pt`  
Distance measures between Secondary Structures, 428

`vrna_bp_stack_s`, 487

`VRNA_BRACKETS_ALPHA`  
Dot-Bracket Notation of Secondary Structures, 409

`VRNA_BRACKETS_ANG`  
Dot-Bracket Notation of Secondary Structures, 409

`VRNA_BRACKETS_ANY`  
Dot-Bracket Notation of Secondary Structures, 410

`VRNA_BRACKETS_CLY`  
Dot-Bracket Notation of Secondary Structures, 409

`VRNA_BRACKETS_DEFAULT`  
Dot-Bracket Notation of Secondary Structures, 410

`VRNA_BRACKETS_RND`  
Dot-Bracket Notation of Secondary Structures, 409

`VRNA_BRACKETS_SQR`  
Dot-Bracket Notation of Secondary Structures, 409

`vrna_BT_hp_loop`  
Backtracking MFE structures, 259

`vrna_BT_mb_loop`  
Backtracking MFE structures, 259

`vrna_C11_features`  
(Abstract) Data Structures, 490

`vrna_callback_free_auxdata`  
The Fold Compound, 506

`vrna_callback_hc_evaluate`  
Hard Constraints, 234

`vrna_callback_heap_cmp`  
Heaps, 522

`vrna_callback_heap_get_pos`  
Heaps, 524

`vrna_callback_heap_set_pos`  
Heaps, 524

`vrna_callback_ht_compare_entries`  
Hash Tables, 516

`vrna_callback_ht_free_entry`  
Hash Tables, 517

`vrna_callback_ht_hash_function`  
Hash Tables, 516

`vrna_callback_move_update`  
Neighborhood Relation and Move Sets for Secondary Structures, 329

`vrna_callback_pk_plex_score`  
Pseudoknots, 358

`vrna_callback_recursion_status`  
The Fold Compound, 506

`vrna_callback_sc_backtrack`  
Soft Constraints, 241

`vrna_callback_sc_energy`  
Soft Constraints, 240

`vrna_callback_sc_exp_energy`  
Soft Constraints, 240

`vrna_callback_stream_output`

Buffers, 529

`vrna_callback_ud_energy`  
Unstructured Domains, 210

`vrna_callback_ud_exp_energy`  
Unstructured Domains, 210

`vrna_callback_ud_exp_production`  
Unstructured Domains, 211

`vrna_callback_ud_probs_add`  
Unstructured Domains, 211

`vrna_callback_ud_probs_get`  
Unstructured Domains, 211

`vrna_callback_ud_production`  
Unstructured Domains, 211

`vrna_centroid`  
Compute the Centroid Structure, 302

`vrna_centroid_from_plist`  
Compute the Centroid Structure, 302

`vrna_centroid_from_probs`  
Compute the Centroid Structure, 303

`vrna_circalifold`  
Global MFE Prediction, 253

`vrna_circfold`  
Global MFE Prediction, 252

`VRNA_CMD_PARSE_DEFAULTS`  
Command Files, 453

`VRNA_CMD_PARSE_HC`  
Command Files, 452

`VRNA_CMD_PARSE_SC`  
Command Files, 452

`VRNA_CMD_PARSE_SD`  
Command Files, 452

`VRNA_CMD_PARSE_UD`  
Command Files, 452

`vrna_cofold`  
Global MFE Prediction, 253

`vrna_color_s`, 487

`vrna_commands_apply`  
Command Files, 454

`vrna_commands_free`  
Command Files, 454

`VRNA_CONSTRAINT_DB`  
Hard Constraints, 231

`VRNA_CONSTRAINT_DB_ANG_BRACK`  
`hard.h`, 612

`VRNA_CONSTRAINT_DB_DEFAULT`  
Hard Constraints, 233

`VRNA_CONSTRAINT_DB_DOT`  
Hard Constraints, 232

`VRNA_CONSTRAINT_DB_ENFORCE_BP`  
Hard Constraints, 232

`VRNA_CONSTRAINT_DB_GQUAD`  
Hard Constraints, 233

`VRNA_CONSTRAINT_DB_INTERMOL`  
Hard Constraints, 233

`VRNA_CONSTRAINT_DB_INTRAMOL`  
Hard Constraints, 233

`VRNA_CONSTRAINT_DB_PIPE`  
Hard Constraints, 232

VRNA\_CONSTRAINT\_DB\_RND\_BRACK  
    Hard Constraints, 232

VRNA\_CONSTRAINT\_DB\_WUSS  
    Hard Constraints, 233

VRNA\_CONSTRAINT\_DB\_X  
    Hard Constraints, 232

VRNA\_CONSTRAINT\_FILE  
    Constraining the RNA Folding Grammar, 219

VRNA\_CONSTRAINT\_MULTILINE  
    Nucleic Acid Sequences and Structures, 439

VRNA\_CONSTRAINT\_NO\_HEADER  
    hard.h, 612

VRNA\_CONSTRAINT\_SOFT\_MFE  
    Constraining the RNA Folding Grammar, 220

VRNA\_CONSTRAINT\_SOFT\_PF  
    Constraining the RNA Folding Grammar, 220

vrna\_constraints\_add  
    Constraining the RNA Folding Grammar, 227

vrna\_convert\_dcal\_to\_kcal  
    Unit Conversion, 497

vrna\_convert\_energy  
    Unit Conversion, 495

vrna\_convert\_kcal\_to\_dcal  
    Unit Conversion, 496

VRNA\_CONVERT\_OUTPUT\_ALL  
    Converting Energy Parameter Files, 389

VRNA\_CONVERT\_OUTPUT\_BULGE  
    Converting Energy Parameter Files, 391

VRNA\_CONVERT\_OUTPUT\_DANGLE3  
    Converting Energy Parameter Files, 390

VRNA\_CONVERT\_OUTPUT\_DANGLE5  
    Converting Energy Parameter Files, 390

VRNA\_CONVERT\_OUTPUT\_DUMP  
    Converting Energy Parameter Files, 392

VRNA\_CONVERT\_OUTPUT\_HP  
    Converting Energy Parameter Files, 389

VRNA\_CONVERT\_OUTPUT\_INT  
    Converting Energy Parameter Files, 391

VRNA\_CONVERT\_OUTPUT\_INT\_11  
    Converting Energy Parameter Files, 390

VRNA\_CONVERT\_OUTPUT\_INT\_21  
    Converting Energy Parameter Files, 391

VRNA\_CONVERT\_OUTPUT\_INT\_22  
    Converting Energy Parameter Files, 391

VRNA\_CONVERT\_OUTPUT\_MISC  
    Converting Energy Parameter Files, 391

VRNA\_CONVERT\_OUTPUT\_DL  
    Converting Energy Parameter Files, 391

VRNA\_CONVERT\_OUTPUT\_MM\_EXT  
    Converting Energy Parameter Files, 390

VRNA\_CONVERT\_OUTPUT\_MM\_HP  
    Converting Energy Parameter Files, 389

VRNA\_CONVERT\_OUTPUT\_MM\_INT  
    Converting Energy Parameter Files, 390

VRNA\_CONVERT\_OUTPUT\_MM\_INT\_1N  
    Converting Energy Parameter Files, 390

VRNA\_CONVERT\_OUTPUT\_MM\_INT\_23  
    Converting Energy Parameter Files, 390

VRNA\_CONVERT\_OUTPUT\_MM\_MULTI  
    Converting Energy Parameter Files, 390

VRNA\_CONVERT\_OUTPUT\_NINIO  
    Converting Energy Parameter Files, 392

VRNA\_CONVERT\_OUTPUT\_SPECIAL\_HP  
    Converting Energy Parameter Files, 391

VRNA\_CONVERT\_OUTPUT\_STACK  
    Converting Energy Parameter Files, 389

VRNA\_CONVERT\_OUTPUT\_VANILLA  
    Converting Energy Parameter Files, 391

vrna\_convert\_temperature  
    Unit Conversion, 496

vrna\_cpair\_s, 487

vrna\_cstr  
    Buffers, 529

vrna\_cstr\_close  
    Buffers, 530

vrna\_cstr\_discard  
    Buffers, 529

vrna\_cstr\_fflush  
    Buffers, 530

vrna\_cstr\_free  
    Buffers, 530

vrna\_cut\_point\_insert  
    (Nucleic Acid Sequence) String Utilities, 405

vrna\_cut\_point\_remove  
    (Nucleic Acid Sequence) String Utilities, 405

vrna\_data\_linear\_s, 487

vrna\_db\_flatten  
    Dot-Bracket Notation of Secondary Structures, 411

vrna\_db\_flatten\_to  
    Dot-Bracket Notation of Secondary Structures, 412

vrna\_db\_from\_bp\_stack  
    Secondary Structure Utilities, 407

vrna\_db\_from\_plist  
    Dot-Bracket Notation of Secondary Structures, 412

vrna\_db\_from\_probs  
    Secondary Structure Utilities, 407

vrna\_db\_from\_ptable  
    Dot-Bracket Notation of Secondary Structures, 412

vrna\_db\_from\_WUSS  
    Washington University Secondary Structure (WUSS) notation, 415

vrna\_db\_pack  
    Dot-Bracket Notation of Secondary Structures, 410

vrna\_db\_pk\_remove  
    Dot-Bracket Notation of Secondary Structures, 413

vrna\_db\_to\_element\_string  
    Dot-Bracket Notation of Secondary Structures, 413

vrna\_db\_to\_tree\_string  
    Tree Representation of Secondary Structures, 426

vrna\_db\_unpack  
    Dot-Bracket Notation of Secondary Structures, 411

VRNA\_DECOMP\_EXT\_EXT  
    Constraining the RNA Folding Grammar, 225

VRNA\_DECOMP\_EXT\_EXT\_EXT  
    Constraining the RNA Folding Grammar, 226

VRNA\_DECOMP\_EXT\_EXT\_STEM

Constrained the RNA Folding Grammar, 226  
**VRNA\_DECOMP\_EXT\_EXT\_STEM1**  
 Constrained the RNA Folding Grammar, 227  
**VRNA\_DECOMP\_EXT\_STEM**  
 Constrained the RNA Folding Grammar, 225  
**VRNA\_DECOMP\_EXT\_STEM\_EXT**  
 Constrained the RNA Folding Grammar, 226  
**VRNA\_DECOMP\_EXT\_UP**  
 Constrained the RNA Folding Grammar, 225  
**VRNA\_DECOMP\_ML\_COAXIAL**  
 Constrained the RNA Folding Grammar, 224  
**VRNA\_DECOMP\_ML\_COAXIAL\_ENC**  
 Constrained the RNA Folding Grammar, 224  
**VRNA\_DECOMP\_ML\_ML**  
 Constrained the RNA Folding Grammar, 223  
**VRNA\_DECOMP\_ML\_ML\_ML**  
 Constrained the RNA Folding Grammar, 222  
**VRNA\_DECOMP\_ML\_ML\_STEM**  
 Constrained the RNA Folding Grammar, 223  
**VRNA\_DECOMP\_ML\_STEM**  
 Constrained the RNA Folding Grammar, 222  
**VRNA\_DECOMP\_ML\_UP**  
 Constrained the RNA Folding Grammar, 223  
**VRNA\_DECOMP\_PAIR\_HP**  
 Constrained the RNA Folding Grammar, 220  
**VRNA\_DECOMP\_PAIR\_IL**  
 Constrained the RNA Folding Grammar, 221  
**VRNA\_DECOMP\_PAIR\_ML**  
 Constrained the RNA Folding Grammar, 221  
**vrna\_dimer\_conc\_s**, 579  
**vrna\_dimer\_pf\_s**, 261  
**vrna\_DNA\_complement**  
 (Nucleic Acid Sequence) String Utilities, 404  
**vrna\_dotplot\_auxdata\_t**, 456  
**vrna\_E\_ext\_hp\_loop**  
 Hairpin Loops, 373  
**vrna\_E\_ext\_stem**  
 Exterior Loops, 370  
**vrna\_E\_hp\_loop**  
 Hairpin Loops, 373  
**vrna\_E\_mb\_loop\_stack**  
 Multibranch Loops, 378  
**vrna\_elem\_prob\_s**, 420  
**vrna\_ensemble\_defect**  
 Predicting various thermodynamic properties, 317  
**vrna\_ensemble\_defect\_pt**  
 Predicting various thermodynamic properties, 316  
**vrna\_enumerate\_necklaces**  
 Combinatorics Algorithms, 479  
**vrna\_eval\_circ\_consensus\_structure**  
 Free Energy Evaluation, 145  
**vrna\_eval\_circ\_consensus\_structure\_v**  
 Free Energy Evaluation, 148  
**vrna\_eval\_circ\_gquad\_consensus\_structure**  
 Free Energy Evaluation, 146  
**vrna\_eval\_circ\_gquad\_consensus\_structure\_v**  
 Free Energy Evaluation, 150  
**vrna\_eval\_circ\_gquad\_structure**  
 Free Energy Evaluation, 141  
**vrna\_eval\_circ\_gquad\_structure\_v**  
 Free Energy Evaluation, 144  
**vrna\_eval\_circ\_structure**  
 Free Energy Evaluation, 140  
**vrna\_eval\_circ\_structure\_v**  
 Free Energy Evaluation, 143  
**vrna\_eval\_consensus\_structure\_pt\_simple**  
 Free Energy Evaluation, 152  
**vrna\_eval\_consensus\_structure\_pt\_simple\_v**  
 Free Energy Evaluation, 153  
**vrna\_eval\_consensus\_structure\_pt\_simple\_verbose**  
 Free Energy Evaluation, 153  
**vrna\_eval\_consensus\_structure\_simple**  
 Free Energy Evaluation, 145  
**vrna\_eval\_consensus\_structure\_simple\_v**  
 Free Energy Evaluation, 148  
**vrna\_eval\_consensus\_structure\_simple\_verbose**  
 Free Energy Evaluation, 147  
**vrna\_eval\_covar\_structure**  
 Free Energy Evaluation, 136  
**vrna\_eval\_ext\_stem**  
 Exterior Loops, 371  
**vrna\_eval\_gquad\_consensus\_structure**  
 Free Energy Evaluation, 146  
**vrna\_eval\_gquad\_consensus\_structure\_v**  
 Free Energy Evaluation, 149  
**vrna\_eval\_gquad\_structure**  
 Free Energy Evaluation, 140  
**vrna\_eval\_gquad\_structure\_v**  
 Free Energy Evaluation, 143  
**vrna\_eval\_hp\_loop**  
 Hairpin Loops, 373  
**vrna\_eval\_int\_loop**  
 Internal Loops, 376  
**vrna\_eval\_loop\_pt**  
 Energy Evaluation for Individual Loops, 154  
**vrna\_eval\_loop\_pt\_v**  
 Energy Evaluation for Individual Loops, 154  
**vrna\_eval\_move**  
 Energy Evaluation for Atomic Moves, 155  
**vrna\_eval\_move\_pt**  
 Energy Evaluation for Atomic Moves, 156  
**vrna\_eval\_structure**  
 Free Energy Evaluation, 136  
**vrna\_eval\_structure\_pt**  
 Free Energy Evaluation, 138  
**vrna\_eval\_structure\_pt\_simple**  
 Free Energy Evaluation, 151  
**vrna\_eval\_structure\_pt\_simple\_v**  
 Free Energy Evaluation, 151  
**vrna\_eval\_structure\_pt\_simple\_verbose**  
 Free Energy Evaluation, 151  
**vrna\_eval\_structure\_pt\_v**  
 Free Energy Evaluation, 139  
**vrna\_eval\_structure\_pt\_verbose**  
 Free Energy Evaluation, 138  
**vrna\_eval\_structure\_simple**

Free Energy Evaluation, 139  
vrna\_eval\_structure\_simple\_v  
    Free Energy Evaluation, 142  
vrna\_eval\_structure\_simple\_verbose  
    Free Energy Evaluation, 142  
vrna\_eval\_structure\_v  
    Free Energy Evaluation, 137  
vrna\_eval\_structure\_verbose  
    Free Energy Evaluation, 137  
vrna\_exp\_E\_ext\_stem  
    Exterior Loops, 371  
vrna\_exp\_E\_hp\_loop  
    Hairpin Loops, 376  
vrna\_exp\_param\_s, 198  
    alpha, 199  
    id, 199  
vrna\_exp\_params  
    Energy Parameters, 200  
vrna\_exp\_params\_comparative  
    Energy Parameters, 201  
vrna\_exp\_params\_copy  
    Energy Parameters, 201  
vrna\_exp\_params\_rescale  
    Energy Parameters, 202  
vrna\_exp\_params\_reset  
    Energy Parameters, 204  
vrna\_exp\_params\_subst  
    Energy Parameters, 202  
vrna\_extract\_record\_rest\_constraint  
    Nucleic Acid Sequences and Structures, 443  
vrna\_extract\_record\_rest\_structure  
    Nucleic Acid Sequences and Structures, 442  
vrna\_fc\_s, 499  
    auxdata, 501  
    cons\_seq, 503  
    free\_auxdata, 501  
    n\_seq, 503  
    pscore, 504  
    pscore\_local, 504  
    pscore\_pf\_compat, 504  
    ptype, 502  
    ptype\_pf\_compat, 502  
    S, 503  
    S3, 503  
    S5, 503  
    S\_cons, 503  
    sc, 502  
    scs, 504  
    sequence, 501  
    sequence\_encoding, 502  
    sequences, 502  
    stat\_cb, 501  
    type, 501  
VRNA\_FC\_TYPE\_COMPARATIVE  
    The Fold Compound, 507  
vrna\_fc\_type\_e  
    The Fold Compound, 507  
VRNA\_FC\_TYPE\_SINGLE  
    The Fold Compound, 507  
vrna\_file\_bpseq  
    Nucleic Acid Sequences and Structures, 440  
vrna\_file\_commands\_apply  
    Command Files, 453  
vrna\_file\_commands\_read  
    Command Files, 453  
vrna\_file\_connect  
    Nucleic Acid Sequences and Structures, 440  
vrna\_file\_exists  
    Files and I/O, 438  
vrna\_file\_fasta\_read\_record  
    Nucleic Acid Sequences and Structures, 441  
VRNA\_FILE\_FORMAT\_MSA\_APPEND  
    Multiple Sequence Alignments, 447  
VRNA\_FILE\_FORMAT\_MSA\_CLUSTAL  
    Multiple Sequence Alignments, 445  
VRNA\_FILE\_FORMAT\_MSA\_DEFAULT  
    Multiple Sequence Alignments, 446  
VRNA\_FILE\_FORMAT\_MSA\_FASTA  
    Multiple Sequence Alignments, 445  
VRNA\_FILE\_FORMAT\_MSA\_MAF  
    Multiple Sequence Alignments, 446  
VRNA\_FILE\_FORMAT\_MSA\_MIS  
    Multiple Sequence Alignments, 446  
VRNA\_FILE\_FORMAT\_MSA\_NOCHECK  
    Multiple Sequence Alignments, 446  
VRNA\_FILE\_FORMAT\_MSA QUIET  
    Multiple Sequence Alignments, 447  
VRNA\_FILE\_FORMAT\_MSA\_SILENT  
    Multiple Sequence Alignments, 447  
VRNA\_FILE\_FORMAT\_MSA\_STOCKHOLM  
    Multiple Sequence Alignments, 445  
VRNA\_FILE\_FORMAT\_MSA\_UNKNOWN  
    Multiple Sequence Alignments, 446  
vrna\_file\_helixlist  
    Nucleic Acid Sequences and Structures, 439  
vrna\_file\_json  
    Nucleic Acid Sequences and Structures, 441  
vrna\_file\_msa\_detect\_format  
    Multiple Sequence Alignments, 450  
vrna\_file\_msa\_read  
    Multiple Sequence Alignments, 447  
vrna\_file\_msa\_read\_record  
    Multiple Sequence Alignments, 448  
vrna\_file\_msa\_write  
    Multiple Sequence Alignments, 450  
vrna\_file\_PS\_aln  
    Alignment Plots, 475  
vrna\_file\_PS\_aln\_slice  
    Alignment Plots, 475  
vrna\_file\_PS\_rnplot  
    Plotting, 457  
vrna\_file\_PS\_rnplot\_a  
    Plotting, 457  
vrna\_file\_SHAPE\_read  
    Nucleic Acid Sequences and Structures, 443  
vrna\_filename\_sanitize

Files and I/O, 437  
**vrna\_fold**  
 Global MFE Prediction, 251  
**vrna\_fold\_compound**  
 The Fold Compound, 507  
**vrna\_fold\_compound\_add\_auxdata**  
 The Fold Compound, 509  
**vrna\_fold\_compound\_add\_callback**  
 The Fold Compound, 510  
**vrna\_fold\_compound\_comparative**  
 The Fold Compound, 508  
**vrna\_fold\_compound\_free**  
 The Fold Compound, 509  
**vrna\_gr\_aux\_s**, 169  
**vrna\_hamming\_distance**  
 (Nucleic Acid Sequence) String Utilitites, 403  
**vrna\_hamming\_distance\_bound**  
 (Nucleic Acid Sequence) String Utilitites, 403  
**vrna\_hash\_table\_t**  
 Hash Tables, 516  
**vrna\_hc\_add\_bp**  
 Hard Constraints, 236  
**vrna\_hc\_add\_bp\_nonspecific**  
 Hard Constraints, 236  
**vrna\_hc\_add\_data**  
 hard.h, 612  
**vrna\_hc\_add\_from\_db**  
 Hard Constraints, 237  
**vrna\_hc\_add\_up**  
 Hard Constraints, 235  
**vrna\_hc\_add\_up\_batch**  
 Hard Constraints, 235  
**VRNA\_HC\_DEFAULT**  
 hard.h, 612  
**vrna\_hc\_free**  
 Hard Constraints, 237  
**vrna\_hc\_init**  
 Hard Constraints, 235  
**vrna\_hc\_s**, 230  
 free\_data, 231  
**vrna\_hc\_type\_e**  
 hard.h, 612  
**vrna\_hc\_up\_s**, 231  
**VRNA\_HC\_WINDOW**  
 hard.h, 612  
**vrna\_heap\_free**  
 Heaps, 525  
**vrna\_heap\_init**  
 Heaps, 524  
**vrna\_heap\_insert**  
 Heaps, 526  
**vrna\_heap\_pop**  
 Heaps, 526  
**vrna\_heap\_remove**  
 Heaps, 527  
**vrna\_heap\_size**  
 Heaps, 525  
**vrna\_heap\_t**  
 Heaps, 522  
**vrna\_heap\_top**  
 Heaps, 526  
**vrna\_heap\_update**  
 Heaps, 527  
**vrna\_heat\_capacity**  
 Predicting various thermodynamic properties, 320  
**vrna\_heat\_capacity\_callback**  
 Predicting various thermodynamic properties, 315  
**vrna\_heat\_capacity\_cb**  
 Predicting various thermodynamic properties, 321  
**vrna\_heat\_capacity\_s**, 315  
**vrna\_heat\_capacity\_simple**  
 Predicting various thermodynamic properties, 321  
**vrna\_heat\_capacity\_t**  
 Predicting various thermodynamic properties, 315  
**vrna\_ht\_clear**  
 Hash Tables, 519  
**vrna\_ht\_collisions**  
 Hash Tables, 518  
**vrna\_ht\_db\_comp**  
 Hash Tables, 520  
**vrna\_ht\_db\_free\_entry**  
 Hash Tables, 521  
**vrna\_ht\_db\_hash\_func**  
 Hash Tables, 520  
**vrna\_ht\_entry\_db\_t**, 515  
 energy, 516  
 structure, 515  
**vrna\_ht\_free**  
 Hash Tables, 520  
**vrna\_ht\_get**  
 Hash Tables, 518  
**vrna\_ht\_init**  
 Hash Tables, 517  
**vrna\_ht\_insert**  
 Hash Tables, 519  
**vrna\_ht\_remove**  
 Hash Tables, 519  
**vrna\_ht\_size**  
 Hash Tables, 518  
**vrna\_hx\_from\_ptable**  
 Helix List Representation of Secondary Structures, 423  
**vrna\_hx\_s**, 423  
**vrna\_idx\_col\_wise**  
 Utilities, 369  
**vrna\_idx\_row\_wise**  
 Utilities, 368  
**vrna\_init\_rand**  
 Utilities, 366  
**vrna\_init\_rand\_seed**  
 Utilities, 366  
**VRNA\_INPUT\_FASTA\_HEADER**  
 Utilities, 365  
**vrna\_int\_urn**  
 Utilities, 367  
**vrna\_Lfold**

Local (sliding window) MFE Prediction, 257  
vrna\_Lfoldz  
    Local (sliding window) MFE Prediction, 257  
vrna\_loapidx\_update  
    Neighborhood Relation and Move Sets for Secondary Structures, 332  
vrna\_maximum\_matching  
    mm.h, 714  
vrna\_maximum\_matching\_simple  
    mm.h, 714  
vrna\_md\_copy  
    Fine-tuning of the Implemented Models, 180  
vrna\_md\_defaults\_backtrack  
    Fine-tuning of the Implemented Models, 188  
vrna\_md\_defaults\_backtrack\_get  
    Fine-tuning of the Implemented Models, 188  
vrna\_md\_defaults\_backtrack\_type  
    Fine-tuning of the Implemented Models, 189  
vrna\_md\_defaults\_backtrack\_type\_get  
    Fine-tuning of the Implemented Models, 189  
vrna\_md\_defaults\_betaScale  
    Fine-tuning of the Implemented Models, 182  
vrna\_md\_defaults\_betaScale\_get  
    Fine-tuning of the Implemented Models, 182  
vrna\_md\_defaults\_circ  
    Fine-tuning of the Implemented Models, 186  
vrna\_md\_defaults\_circ\_get  
    Fine-tuning of the Implemented Models, 186  
vrna\_md\_defaults\_compute\_bpp  
    Fine-tuning of the Implemented Models, 189  
vrna\_md\_defaults\_compute\_bpp\_get  
    Fine-tuning of the Implemented Models, 190  
vrna\_md\_defaults\_cv\_fact  
    Fine-tuning of the Implemented Models, 193  
vrna\_md\_defaults\_cv\_fact\_get  
    Fine-tuning of the Implemented Models, 193  
vrna\_md\_defaults\_dangles  
    Fine-tuning of the Implemented Models, 182  
vrna\_md\_defaults\_dangles\_get  
    Fine-tuning of the Implemented Models, 183  
vrna\_md\_defaults\_energy\_set  
    Fine-tuning of the Implemented Models, 188  
vrna\_md\_defaults\_energy\_set\_get  
    Fine-tuning of the Implemented Models, 188  
vrna\_md\_defaults\_gquad  
    Fine-tuning of the Implemented Models, 186  
vrna\_md\_defaults\_gquad\_get  
    Fine-tuning of the Implemented Models, 187  
vrna\_md\_defaults\_logML  
    Fine-tuning of the Implemented Models, 185  
vrna\_md\_defaults\_logML\_get  
    Fine-tuning of the Implemented Models, 186  
vrna\_md\_defaults\_max\_bp\_span  
    Fine-tuning of the Implemented Models, 190  
vrna\_md\_defaults\_max\_bp\_span\_get  
    Fine-tuning of the Implemented Models, 190  
vrna\_md\_defaults\_min\_loop\_size  
    Fine-tuning of the Implemented Models, 190  
vrna\_md\_defaults\_min\_loop\_size\_get  
    Fine-tuning of the Implemented Models, 191  
vrna\_md\_defaults\_nc\_fact  
    Fine-tuning of the Implemented Models, 193  
vrna\_md\_defaults\_nc\_fact\_get  
    Fine-tuning of the Implemented Models, 194  
vrna\_md\_defaults\_noGU  
    Fine-tuning of the Implemented Models, 184  
vrna\_md\_defaults\_noGU\_get  
    Fine-tuning of the Implemented Models, 184  
vrna\_md\_defaults\_noGUclosure  
    Fine-tuning of the Implemented Models, 185  
vrna\_md\_defaults\_noGUclosure\_get  
    Fine-tuning of the Implemented Models, 185  
vrna\_md\_defaults\_noLP  
    Fine-tuning of the Implemented Models, 184  
vrna\_md\_defaults\_noLP\_get  
    Fine-tuning of the Implemented Models, 184  
vrna\_md\_defaults\_oldAliEn  
    Fine-tuning of the Implemented Models, 192  
vrna\_md\_defaults\_oldAliEn\_get  
    Fine-tuning of the Implemented Models, 192  
vrna\_md\_defaults\_reset  
    Fine-tuning of the Implemented Models, 181  
vrna\_md\_defaults\_ribo  
    Fine-tuning of the Implemented Models, 192  
vrna\_md\_defaults\_ribo\_get  
    Fine-tuning of the Implemented Models, 192  
vrna\_md\_defaults\_sfact  
    Fine-tuning of the Implemented Models, 194  
vrna\_md\_defaults\_sfact\_get  
    Fine-tuning of the Implemented Models, 194  
vrna\_md\_defaults\_special\_hp  
    Fine-tuning of the Implemented Models, 183  
vrna\_md\_defaults\_special\_hp\_get  
    Fine-tuning of the Implemented Models, 183  
vrna\_md\_defaults\_temperature  
    Fine-tuning of the Implemented Models, 181  
vrna\_md\_defaults\_temperature\_get  
    Fine-tuning of the Implemented Models, 182  
vrna\_md\_defaults\_uniq\_ML  
    Fine-tuning of the Implemented Models, 187  
vrna\_md\_defaults\_uniq\_ML\_get  
    Fine-tuning of the Implemented Models, 187  
vrna\_md\_defaults\_window\_size  
    Fine-tuning of the Implemented Models, 191  
vrna\_md\_defaults\_window\_size\_get  
    Fine-tuning of the Implemented Models, 191  
vrna\_md\_option\_string  
    Fine-tuning of the Implemented Models, 181  
vrna\_md\_s, 173  
    dangles, 175  
    min\_loop\_size, 176  
vrna\_md\_set\_default  
    Fine-tuning of the Implemented Models, 180  
vrna\_md\_update  
    Fine-tuning of the Implemented Models, 180  
vrna\_MEA

Compute the Structure with Maximum Expected Accuracy (MEA), 300

`vrna_MEAl_from_plist`  
Compute the Structure with Maximum Expected Accuracy (MEA), 300

`vrna_mean_bp_distance`  
Predicting various thermodynamic properties, 316

`vrna_mean_bp_distance_pr`  
Predicting various thermodynamic properties, 315

`VRNA_MEASURE_SHANNON_ENTROPY`  
Multiple Sequence Alignment Utilities, 431

`vrna_message_constraint_options`  
Constraining the RNA Folding Grammar, 228

`vrna_message_constraint_options_all`  
Constraining the RNA Folding Grammar, 228

`vrna_message_error`  
Messages, 491

`vrna_message_info`  
Messages, 492

`vrna_message_input_seq`  
Messages, 493

`vrna_message_input_seq_simple`  
Messages, 493

`vrna_message_verror`  
Messages, 491

`vrna_message_vinfo`  
Messages, 493

`vrna_message_vwarning`  
Messages, 492

`vrna_message_warning`  
Messages, 492

`vrna_mfe`  
Global MFE Prediction, 250

`vrna_mfe_dimer`  
Global MFE Prediction, 250

`vrna_mfe_TwoD`  
Computing MFE representatives of a Distance Based Partitioning, 307

`vrna_mfe_window`  
Local (sliding window) MFE Prediction, 256

`vrna_mfe_window_callback`  
Local (sliding window) MFE Prediction, 255

`vrna_mfe_window_zscore`  
Local (sliding window) MFE Prediction, 256

`VRNA_MINIMIZER_CONJUGATE_FR`  
Generate Soft Constraints from Data, 351

`VRNA_MINIMIZER_CONJUGATE_PR`  
Generate Soft Constraints from Data, 351

`VRNA_MINIMIZER_STEEPEST_DESCENT`  
Generate Soft Constraints from Data, 352

`VRNA_MINIMIZER_VECTOR_BFGS`  
Generate Soft Constraints from Data, 352

`VRNA_MINIMIZER_VECTOR_BFGS2`  
Generate Soft Constraints from Data, 352

`VRNA_MODEL_DEFAULT_ALI_CV_FACT`  
Fine-tuning of the Implemented Models, 179

`VRNA_MODEL_DEFAULT_ALI_NC_FACT`  
Fine-tuning of the Implemented Models, 179

`VRNA_MODEL_DEFAULT_ALI_OLD_EN`  
Fine-tuning of the Implemented Models, 179

`VRNA_MODEL_DEFAULT_ALI_RIBO`  
Fine-tuning of the Implemented Models, 179

`VRNA_MODEL_DEFAULT_BACKTRACK`  
Fine-tuning of the Implemented Models, 178

`VRNA_MODEL_DEFAULT_BACKTRACK_TYPE`  
Fine-tuning of the Implemented Models, 178

`VRNA_MODEL_DEFAULT_BETA_SCALE`  
Fine-tuning of the Implemented Models, 176

`VRNA_MODEL_DEFAULT_CIRC`  
Fine-tuning of the Implemented Models, 177

`VRNA_MODEL_DEFAULT_COMPUTE_BPP`  
Fine-tuning of the Implemented Models, 178

`VRNA_MODEL_DEFAULT_DANGLES`  
Fine-tuning of the Implemented Models, 176

`VRNA_MODEL_DEFAULT_ENERGY_SET`  
Fine-tuning of the Implemented Models, 178

`VRNA_MODEL_DEFAULT_GQUAD`  
Fine-tuning of the Implemented Models, 177

`VRNA_MODEL_DEFAULT_LOG_ML`  
Fine-tuning of the Implemented Models, 179

`VRNA_MODEL_DEFAULT_MAX_BP_SPAN`  
Fine-tuning of the Implemented Models, 178

`VRNA_MODEL_DEFAULT_NO_GU`  
Fine-tuning of the Implemented Models, 177

`VRNA_MODEL_DEFAULT_NO_GU_CLOSURE`  
Fine-tuning of the Implemented Models, 177

`VRNA_MODEL_DEFAULT_NO_LP`  
Fine-tuning of the Implemented Models, 177

`VRNA_MODEL_DEFAULT_PF_SCALE`  
Fine-tuning of the Implemented Models, 176

`VRNA_MODEL_DEFAULT_SPECIAL_HP`  
Fine-tuning of the Implemented Models, 177

`VRNA_MODEL_DEFAULT_TEMPERATURE`  
Fine-tuning of the Implemented Models, 176

`VRNA_MODEL_DEFAULT_UNIQ_ML`  
Fine-tuning of the Implemented Models, 178

`VRNA_MODEL_DEFAULT_WINDOW_SIZE`  
Fine-tuning of the Implemented Models, 179

`vrna_move_apply`  
Neighborhood Relation and Move Sets for Secondary Structures, 330

`vrna_move_compare`  
Neighborhood Relation and Move Sets for Secondary Structures, 331

`vrna_move_init`  
Neighborhood Relation and Move Sets for Secondary Structures, 330

`vrna_move_is_insertion`  
Neighborhood Relation and Move Sets for Secondary Structures, 331

`vrna_move_is_removal`  
Neighborhood Relation and Move Sets for Secondary Structures, 330

`vrna_move_is_shift`  
Neighborhood Relation and Move Sets for Secondary Structures, 331

vrna\_move\_list\_free  
    Neighborhood Relation and Move Sets for Secondary Structures, 330

vrna\_move\_neighbor\_diff  
    Neighborhood Relation and Move Sets for Secondary Structures, 334

vrna\_move\_neighbor\_diff\_cb  
    Neighborhood Relation and Move Sets for Secondary Structures, 333

vrna\_move\_s, 327

VRNA\_MOVESET\_DEFAULT  
    Neighborhood Relation and Move Sets for Secondary Structures, 328

VRNA\_MOVESET\_DELETION  
    Neighborhood Relation and Move Sets for Secondary Structures, 328

VRNA\_MOVESET\_INSERTION  
    Neighborhood Relation and Move Sets for Secondary Structures, 328

VRNA\_MOVESET\_NO\_LP  
    Neighborhood Relation and Move Sets for Secondary Structures, 328

VRNA\_MOVESET\_SHIFT  
    Neighborhood Relation and Move Sets for Secondary Structures, 328

vrna\_multimer\_pf\_s, 262

VRNA\_MX\_2DFOLD  
    The Dynamic Programming Matrices, 513

vrna\_mx\_add  
    The Dynamic Programming Matrices, 513

VRNA\_MX\_DEFAULT  
    The Dynamic Programming Matrices, 513

vrna\_mx\_mfe\_free  
    The Dynamic Programming Matrices, 514

vrna\_mx\_mfe\_s, 511  
    strands, 511  
    type, 511

vrna\_mx\_pf\_aux\_el\_t  
    Exterior Loops, 370

vrna\_mx\_pf\_aux\_ml\_t  
    Multibranch Loops, 377

vrna\_mx\_pf\_free  
    The Dynamic Programming Matrices, 514

vrna\_mx\_pf\_s, 512  
    expMLbase, 512  
    length, 512  
    scale, 512  
    type, 512

vrna\_mx\_type\_e  
    The Dynamic Programming Matrices, 512

VRNA\_MX\_WINDOW  
    The Dynamic Programming Matrices, 513

vrna\_n\_multichoose\_k  
    Combinatorics Algorithms, 484

VRNA\_NEIGHBOR\_CHANGE  
    Neighborhood Relation and Move Sets for Secondary Structures, 329

VRNA\_NEIGHBOR\_INVALID  
    Neighborhood Relation and Move Sets for Secondary Structures, 329

VRNA\_NEIGHBOR\_NEW  
    Neighborhood Relation and Move Sets for Secondary Structures, 329

vrna\_neighbors  
    Neighborhood Relation and Move Sets for Secondary Structures, 332

vrna\_neighbors\_successive  
    Neighborhood Relation and Move Sets for Secondary Structures, 333

vrna\_nucleotide\_decode  
    Utilities to deal with Nucleotide Alphabets, 395

vrna\_nucleotide\_encode  
    Utilities to deal with Nucleotide Alphabets, 395

VRNA\_OBJECTIVE\_FUNCTION\_ABSOLUTE  
    Generate Soft Constraints from Data, 351

VRNA\_OBJECTIVE\_FUNCTION\_QUADRATIC  
    Generate Soft Constraints from Data, 351

VRNA\_OPTION\_EVAL\_ONLY  
    The Fold Compound, 505

VRNA\_OPTION\_MFE  
    The Fold Compound, 505

VRNA\_OPTION\_MULTILINE  
    Nucleic Acid Sequences and Structures, 439

VRNA\_OPTION\_PF  
    The Fold Compound, 505

vrna\_oiostream\_free  
    Buffers, 531

vrna\_oiostream\_init  
    Buffers, 531

vrna\_oiostream\_provide  
    Buffers, 532

vrna\_oiostream\_request  
    Buffers, 531

vrna\_param\_s, 198

VRNA\_PARAMETER\_FORMAT\_DEFAULT  
    Reading/Writing Energy Parameter Sets from/to File, 383

vrna\_params  
    Energy Parameters, 199

vrna\_params\_copy  
    Energy Parameters, 200

vrna\_params\_load  
    Reading/Writing Energy Parameter Sets from/to File, 383

vrna\_params\_load\_defaults  
    Reading/Writing Energy Parameter Sets from/to File, 385

vrna\_params\_load\_DNA\_Mathews1999  
    Reading/Writing Energy Parameter Sets from/to File, 387

vrna\_params\_load\_DNA\_Mathews2004  
    Reading/Writing Energy Parameter Sets from/to File, 387

vrna\_params\_load\_from\_string  
    Reading/Writing Energy Parameter Sets from/to File, 384

**vrna\_params\_load\_RNA\_Andronescu2007**  
 Reading/Writing Energy Parameter Sets from/to File, [386](#)
  
**vrna\_params\_load\_RNA\_Langdon2018**  
 Reading/Writing Energy Parameter Sets from/to File, [386](#)
  
**vrna\_params\_load\_RNA\_misc\_special\_hairpins**  
 Reading/Writing Energy Parameter Sets from/to File, [386](#)
  
**vrna\_params\_load\_RNA\_Turner1999**  
 Reading/Writing Energy Parameter Sets from/to File, [385](#)
  
**vrna\_params\_load\_RNA\_Turner2004**  
 Reading/Writing Energy Parameter Sets from/to File, [385](#)
  
**vrna\_params\_reset**  
 Energy Parameters, [203](#)
  
**vrna\_params\_save**  
 Reading/Writing Energy Parameter Sets from/to File, [384](#)
  
**vrna\_params\_subst**  
 Energy Parameters, [202](#)
  
**vrna\_path**  
 Folding Paths that start at a single Secondary Structure, [344](#)
  
**VRNA\_PATH\_DEFAULT**  
 Folding Paths that start at a single Secondary Structure, [344](#)
  
**vrna\_path\_direct**  
 Direct Refolding Paths between two Secondary Structures, [342](#)
  
**vrna\_path\_direct\_ub**  
 Direct Refolding Paths between two Secondary Structures, [342](#)
  
**vrna\_path\_findpath**  
 Direct Refolding Paths between two Secondary Structures, [340](#)
  
**vrna\_path\_findpath\_saddle**  
 Direct Refolding Paths between two Secondary Structures, [338](#)
  
**vrna\_path\_findpath\_saddle\_ub**  
 Direct Refolding Paths between two Secondary Structures, [339](#)
  
**vrna\_path\_findpath\_ub**  
 Direct Refolding Paths between two Secondary Structures, [340](#)
  
**vrna\_path\_free**  
 (Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima, [337](#)
  
**vrna\_path\_gradient**  
 Folding Paths that start at a single Secondary Structure, [345](#)
  
**VRNA\_PATH\_NO\_TRANSITION\_OUTPUT**  
 Folding Paths that start at a single Secondary Structure, [344](#)
  
**vrna\_path\_options\_findpath**  
 Direct Refolding Paths between two Secondary Structures, [341](#)
  
**vrna\_path\_options\_free**  
 (Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima, [337](#)
  
**VRNA\_PATH\_RANDOM**  
 Folding Paths that start at a single Secondary Structure, [344](#)
  
**vrna\_path\_random**  
 Folding Paths that start at a single Secondary Structure, [346](#)
  
**vrna\_path\_s**, [336](#)  
 type, [337](#)
  
**VRNA\_PATH\_STEEPEST\_DESCENT**  
 Folding Paths that start at a single Secondary Structure, [344](#)
  
**VRNA\_PATH\_TYPE\_DOT\_BRACKET**  
 (Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima, [337](#)
  
**VRNA\_PATH\_TYPE\_MOVES**  
 (Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima, [337](#)
  
**vrna\_pbacktrack**  
 Random Structure Samples from the Ensemble, [288](#)
  
**vrna\_pbacktrack5**  
 Random Structure Samples from the Ensemble, [282](#)
  
**vrna\_pbacktrack5\_cb**  
 Random Structure Samples from the Ensemble, [284](#)
  
**vrna\_pbacktrack5\_num**  
 Random Structure Samples from the Ensemble, [283](#)
  
**vrna\_pbacktrack5\_resume**  
 Random Structure Samples from the Ensemble, [285](#)
  
**vrna\_pbacktrack5\_resume\_cb**  
 Random Structure Samples from the Ensemble, [286](#)
  
**vrna\_pbacktrack5\_TwoD**  
 Stochastic Backtracking of Structures from Distance Based Partitioning, [312](#)
  
**vrna\_pbacktrack\_cb**  
 Random Structure Samples from the Ensemble, [289](#)
  
**VRNA\_PBACKTRACK\_DEFAULT**  
 Random Structure Samples from the Ensemble, [281](#)
  
**vrna\_pbacktrack\_mem\_free**  
 Random Structure Samples from the Ensemble, [299](#)
  
**vrna\_pbacktrack\_mem\_t**  
 Random Structure Samples from the Ensemble, [281](#)
  
**VRNA\_PBACKTRACK\_NON\_REDUNDANT**  
 Random Structure Samples from the Ensemble, [281](#)
  
**vrna\_pbacktrack\_num**  
 Random Structure Samples from the Ensemble,

vrna\_pbacktrack\_resume  
Random Structure Samples from the Ensemble,  
**290**

vrna\_pbacktrack\_resume\_cb  
Random Structure Samples from the Ensemble,  
**292**

vrna\_pbacktrack\_sub  
Random Structure Samples from the Ensemble,  
**293**

vrna\_pbacktrack\_sub\_cb  
Random Structure Samples from the Ensemble,  
**295**

vrna\_pbacktrack\_sub\_num  
Random Structure Samples from the Ensemble,  
**294**

vrna\_pbacktrack\_sub\_resume  
Random Structure Samples from the Ensemble,  
**296**

vrna\_pbacktrack\_sub\_resume\_cb  
Random Structure Samples from the Ensemble,  
**298**

vrna\_pbacktrack\_TwoD  
Stochastic Backtracking of Structures from Dis-  
tance Based Partitioning, **312**

vrna\_pf  
Global Partition Function and Equilibrium Probabil-  
ities, **262**

vrna\_pf\_alifold  
Global Partition Function and Equilibrium Probabil-  
ities, **264**

vrna\_pf\_circalifold  
Global Partition Function and Equilibrium Probabil-  
ities, **265**

vrna\_pf\_circfold  
Global Partition Function and Equilibrium Probabil-  
ities, **264**

vrna\_pf\_co\_fold  
Global Partition Function and Equilibrium Probabil-  
ities, **266**

Partition Function for Two Hybridized Sequences,  
**379**

vrna\_pf\_dimer  
Global Partition Function and Equilibrium Probabil-  
ities, **262**

vrna\_pf\_dimer\_concentrations  
Partition Function for Two Hybridized Sequences,  
**380**

vrna\_pf\_dimer\_probs  
Predicting various thermodynamic properties, **319**

vrna\_pf\_float\_precision  
Partition Function and Equilibrium Properties, **248**

vrna\_pf\_fold  
Global Partition Function and Equilibrium Probabil-  
ities, **263**

vrna\_pf\_TwoD  
Computing Partition Functions of a Distance Based  
Partitioning, **311**

vrna\_pfl\_fold  
Local (sliding window) Partition Function and Equi-  
librium Probabilities, **271**

vrna\_pfl\_fold\_cb  
Local (sliding window) Partition Function and Equi-  
librium Probabilities, **272**

vrna\_pfl\_fold\_up  
Local (sliding window) Partition Function and Equi-  
librium Probabilities, **272**

vrna\_pfl\_fold\_up\_cb  
Local (sliding window) Partition Function and Equi-  
librium Probabilities, **273**

vrna\_pinfo\_s, **431**

vrna\_pk\_plex  
Pseudoknots, **359**

vrna\_pk\_plex\_accessibility  
Pseudoknots, **359**

vrna\_pk\_plex\_opt  
Pseudoknots, **360**

vrna\_pk\_plex\_opt\_defaults  
Pseudoknots, **360**

vrna\_pk\_plex\_opt\_fun  
Pseudoknots, **360**

vrna\_pk\_plex\_opt\_t  
Pseudoknots, **358**

vrna\_pk\_plex\_result\_s, **357**

vrna\_pk\_plex\_t  
Pseudoknots, **358**

vrna\_plist  
Pair List Representation of Secondary Structures,  
**420**

vrna\_plist\_from\_probs  
Global Partition Function and Equilibrium Probabil-  
ities, **266**

vrna\_plot\_coords  
Layouts and Coordinates, **466**

vrna\_plot\_coords\_circular  
Layouts and Coordinates, **469**

vrna\_plot\_coords\_circular\_pt  
Layouts and Coordinates, **470**

vrna\_plot\_coords\_pt  
Layouts and Coordinates, **467**

vrna\_plot\_coords\_puzzler  
Layouts and Coordinates, **470**

vrna\_plot\_coords\_puzzler\_pt  
Layouts and Coordinates, **471**

vrna\_plot\_coords\_simple  
Layouts and Coordinates, **468**

vrna\_plot\_coords\_simple\_pt  
Layouts and Coordinates, **469**

vrna\_plot\_coords\_turtle  
Layouts and Coordinates, **473**

vrna\_plot\_coords\_turtle\_pt  
Layouts and Coordinates, **473**

vrna\_plot\_layout  
Layouts and Coordinates, **463**

vrna\_plot\_layout\_circular  
Layouts and Coordinates, **465**

**vrna\_plot\_layout\_free**  
 Layouts and Coordinates, [466](#)  
**vrna\_plot\_layout\_puzzler**  
 Layouts and Coordinates, [466](#)  
**vrna\_plot\_layout\_s**, [462](#)  
**vrna\_plot\_layout\_simple**  
 Layouts and Coordinates, [464](#)  
**vrna\_plot\_layout\_t**  
 Layouts and Coordinates, [463](#)  
**vrna\_plot\_layout\_turtle**  
 Layouts and Coordinates, [465](#)  
**vrna\_plot\_options\_puzzler**  
 Layouts and Coordinates, [472](#)  
**vrna\_plot\_options\_puzzler\_free**  
 Layouts and Coordinates, [472](#)  
**vrna\_plot\_options\_puzzler\_t**, [462](#)  
**VRNA\_PLOT\_TYPE\_CIRCULAR**  
 Layouts and Coordinates, [463](#)  
**VRNA\_PLOT\_TYPE\_NAVIEW**  
 Layouts and Coordinates, [463](#)  
**VRNA\_PLOT\_TYPE\_SIMPLE**  
 Layouts and Coordinates, [463](#)  
**vrna\_positional\_entropy**  
 Predicting various thermodynamic properties, [318](#)  
**vrna\_pr\_energy**  
 Predicting various thermodynamic properties, [320](#)  
**vrna\_pr\_structure**  
 Predicting various thermodynamic properties, [319](#)  
**vrna\_probs\_window**  
 Local (sliding window) Partition Function and Equilibrium Probabilities, [270](#)  
**VRNA\_PROBS\_WINDOW\_BPP**  
 Local (sliding window) Partition Function and Equilibrium Probabilities, [268](#)  
**vrna\_probs\_window\_callback**  
 Local (sliding window) Partition Function and Equilibrium Probabilities, [269](#)  
**VRNA\_PROBS\_WINDOW\_PF**  
 Local (sliding window) Partition Function and Equilibrium Probabilities, [269](#)  
**VRNA\_PROBS\_WINDOW\_STACKP**  
 Local (sliding window) Partition Function and Equilibrium Probabilities, [268](#)  
**VRNA\_PROBS\_WINDOW\_UP**  
 Local (sliding window) Partition Function and Equilibrium Probabilities, [268](#)  
**VRNA\_PROBS\_WINDOW\_UP\_SPLIT**  
 Local (sliding window) Partition Function and Equilibrium Probabilities, [269](#)  
**vrna\_pt\_pk\_get**  
 Pair Table Representation of Secondary Structures, [418](#)  
**vrna\_pt\_pk\_remove**  
 Pair Table Representation of Secondary Structures, [419](#)  
**vrna\_pt\_snoop\_get**  
 Pair Table Representation of Secondary Structures, [419](#)  
**vrna\_ptable**  
 Pair Table Representation of Secondary Structures, [416](#)  
**vrna\_ptable\_copy**  
 Pair Table Representation of Secondary Structures, [418](#)  
**vrna\_ptable\_from\_string**  
 Pair Table Representation of Secondary Structures, [416](#)  
**vrna\_ptypes**  
 Utilities to deal with Nucleotide Alphabets, [394](#)  
**vrna\_random\_string**  
 (Nucleic Acid Sequence) String Utilities, [402](#)  
**vrna\_read\_line**  
 Files and I/O, [437](#)  
**vrna\_realloc**  
 Utilities, [366](#)  
**vrna\_refBPcnt\_matrix**  
 Secondary Structure Utilities, [407](#)  
**vrna\_refBPDist\_matrix**  
 Secondary Structure Utilities, [407](#)  
**vrna\_rotational\_symmetry**  
 Combinatorics Algorithms, [481](#)  
**vrna\_rotational\_symmetry\_db**  
 Combinatorics Algorithms, [482](#)  
**vrna\_rotational\_symmetry\_db\_pos**  
 Combinatorics Algorithms, [483](#)  
**vrna\_rotational\_symmetry\_num**  
 Combinatorics Algorithms, [480](#)  
**vrna\_rotational\_symmetry\_pos**  
 Combinatorics Algorithms, [482](#)  
**vrna\_rotational\_symmetry\_pos\_num**  
 Combinatorics Algorithms, [480](#)  
**vrna\_sc\_add\_bp**  
 Soft Constraints, [243](#)  
**vrna\_sc\_add\_bt**  
 Soft Constraints, [246](#)  
**vrna\_sc\_add\_data**  
 Soft Constraints, [245](#)  
**vrna\_sc\_add\_exp\_f**  
 Soft Constraints, [246](#)  
**vrna\_sc\_add\_f**  
 Soft Constraints, [245](#)  
**vrna\_sc\_add\_hi\_motif**  
 Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints, [355](#)  
**vrna\_sc\_add\_SHAPE\_deigan**  
 SHAPE Reactivity Data, [348](#)  
**vrna\_sc\_add\_SHAPE\_deigan\_ali**  
 SHAPE Reactivity Data, [348](#)  
**vrna\_sc\_add\_SHAPE\_zarringhalam**  
 SHAPE Reactivity Data, [349](#)  
**vrna\_sc\_add\_up**  
 Soft Constraints, [244](#)  
**vrna\_sc\_bp\_storage\_t**, [579](#)  
**VRNA\_SC\_DEFAULT**  
`soft.h`, [620](#)

vrna\_sc\_free  
    Soft Constraints, 245

vrna\_sc\_init  
    Soft Constraints, 242

vrna\_sc\_minimize\_perturbation  
    Generate Soft Constraints from Data, 352

vrna\_sc\_motif\_s, 355

vrna\_sc\_remove  
    Soft Constraints, 244

vrna\_sc\_s, 238  
    bt, 239  
    exp\_f, 239  
    f, 239

vrna\_sc\_set\_bp  
    Soft Constraints, 242

vrna\_sc\_set\_up  
    Soft Constraints, 243

vrna\_sc\_SHAPE\_parse\_method  
    SHAPE.h, 618

vrna\_sc\_SHAPE\_to\_pr  
    SHAPE Reactivity Data, 350

vrna\_sc\_type\_e  
    soft.h, 620

VRNA\_SC\_WINDOW  
    soft.h, 621

vrna\_search\_BM\_BCT  
    Search Algorithms, 478

vrna\_search\_BM\_BCT\_num  
    Search Algorithms, 478

vrna\_search\_BMH  
    Search Algorithms, 477

vrna\_search\_BMH\_num  
    Search Algorithms, 476

vrna\_sect\_s, 487

VRNA\_SEQ\_DNA  
    Utilities to deal with Nucleotide Alphabets, 394

vrna\_seq\_encode  
    Utilities to deal with Nucleotide Alphabets, 394

vrna\_seq\_reverse  
    (Nucleic Acid Sequence) String Utilitites, 404

VRNA\_SEQ\_RNA  
    Utilities to deal with Nucleotide Alphabets, 394

vrna\_seq\_toRNA  
    (Nucleic Acid Sequence) String Utilitites, 403

vrna\_seq\_toupper  
    (Nucleic Acid Sequence) String Utilitites, 404

vrna\_seq\_type\_e  
    Utilities to deal with Nucleotide Alphabets, 394

vrna\_seq\_ungapped  
    (Nucleic Acid Sequence) String Utilitites, 405

VRNA\_SEQ\_UNKNOWN  
    Utilities to deal with Nucleotide Alphabets, 394

vrna\_sequence\_s, 393

vrna\_sol\_TwoD\_pf\_t, 310  
    Computing Partition Functions of a Distance Based Partitioning, 310

vrna\_sol\_TwoD\_t, 305

Computing MFE representatives of a Distance Based Partitioning, 306

vrna\_stack\_prob  
    Predicting various thermodynamic properties, 318

VRNA\_STATUS\_MFE\_POST  
    The Fold Compound, 504

VRNA\_STATUS\_MFE\_PRE  
    The Fold Compound, 504

VRNA\_STATUS\_PF\_POST  
    The Fold Compound, 505

VRNA\_STATUS\_PF\_PRE  
    The Fold Compound, 505

vrna\_strcat\_printf  
    (Nucleic Acid Sequence) String Utilitites, 400

vrna\_strcat\_vprintf  
    (Nucleic Acid Sequence) String Utilitites, 400

vrna\_strdup\_printf  
    (Nucleic Acid Sequence) String Utilitites, 398

vrna\_strdup\_vprintf  
    (Nucleic Acid Sequence) String Utilitites, 399

vrna\_strsplit  
    (Nucleic Acid Sequence) String Utilitites, 402

vrna\_trim  
    (Nucleic Acid Sequence) String Utilitites, 401

VRNA\_STRUCTURE\_TREE\_EXPANDED  
    Tree Representation of Secondary Structures, 426

VRNA\_STRUCTURE\_TREE\_HIT  
    Tree Representation of Secondary Structures, 425

VRNA\_STRUCTURE\_TREE\_SHAPIRO  
    Tree Representation of Secondary Structures, 426

VRNA\_STRUCTURE\_TREE\_SHAPIRO\_EXT  
    Tree Representation of Secondary Structures, 426

VRNA\_STRUCTURE\_TREE\_SHAPIRO\_SHORT  
    Tree Representation of Secondary Structures, 425

VRNA\_STRUCTURE\_TREE\_SHAPIRO\_WEIGHT  
    Tree Representation of Secondary Structures, 426

vrna\_structured\_domains\_s, 579

vrna\_subopt  
    Suboptimal Structures within an Energy Band around the MFE, 277

vrna\_subopt\_callback  
    Suboptimal Structures within an Energy Band around the MFE, 276

vrna\_subopt\_cb  
    Suboptimal Structures within an Energy Band around the MFE, 277

vrna\_subopt\_sol\_s, 579

vrna\_subopt\_zuker  
    Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989, 275

vrna\_time\_stamp  
    Utilities, 368

vrna\_tree\_string\_to\_db  
    Tree Representation of Secondary Structures, 428

vrna\_tree\_string\_unweight  
    Tree Representation of Secondary Structures, 427

VRNA\_TRIM\_ALL  
    (Nucleic Acid Sequence) String Utilitites, 398

VRNA\_TRIM\_DEFAULT  
     (Nucleic Acid Sequence) String Utilitites, 398

VRNA\_TRIM\_IN\_BETWEEN  
     (Nucleic Acid Sequence) String Utilitites, 398

VRNA\_TRIM.LEADING  
     (Nucleic Acid Sequence) String Utilitites, 397

VRNA\_TRIM\_SUBST\_BY\_FIRST  
     (Nucleic Acid Sequence) String Utilitites, 398

VRNA\_TRIM\_TRAILING  
     (Nucleic Acid Sequence) String Utilitites, 397

vrna\_ud\_add\_motif  
     Unstructured Domains, 213

vrna\_ud\_motifs\_centroid  
     Unstructured Domains, 211

vrna\_ud\_motifs\_MEA  
     Unstructured Domains, 212

vrna\_ud\_motifs\_MFE  
     Unstructured Domains, 212

vrna\_ud\_remove  
     Unstructured Domains, 214

vrna\_ud\_set\_data  
     Unstructured Domains, 214

vrna\_ud\_set\_exp\_prod\_rule\_cb  
     Unstructured Domains, 215

vrna\_ud\_set\_prob\_cb  
     unstructured\_domains.h, 1251

vrna\_ud\_set\_prod\_rule\_cb  
     Unstructured Domains, 214

VRNA\_UNIT\_CAL  
     Unit Conversion, 495

VRNA\_UNIT\_CAL\_IT  
     Unit Conversion, 495

VRNA\_UNIT\_DACAL  
     Unit Conversion, 495

VRNA\_UNIT\_DEG\_C  
     Unit Conversion, 495

VRNA\_UNIT\_DEG\_DE  
     Unit Conversion, 495

VRNA\_UNIT\_DEG\_F  
     Unit Conversion, 495

VRNA\_UNIT\_DEG\_N  
     Unit Conversion, 495

VRNA\_UNIT\_DEG\_R  
     Unit Conversion, 495

VRNA\_UNIT\_DEG\_RE  
     Unit Conversion, 495

VRNA\_UNIT\_DEG\_RO  
     Unit Conversion, 495

vrna\_unit\_energy\_e  
     Unit Conversion, 494

VRNA\_UNIT\_EV  
     Unit Conversion, 495

VRNA\_UNIT\_G\_TNT  
     Unit Conversion, 495

VRNA\_UNIT\_J  
     Unit Conversion, 494

VRNA\_UNIT\_K  
     Unit Conversion, 495

VRNA\_UNIT\_KCAL  
     Unit Conversion, 495

VRNA\_UNIT\_KCAL\_IT  
     Unit Conversion, 495

VRNA\_UNIT\_KG\_TNT  
     Unit Conversion, 495

VRNA\_UNIT\_KJ  
     Unit Conversion, 495

VRNA\_UNIT\_KWH  
     Unit Conversion, 495

VRNA\_UNIT\_T\_TNT  
     Unit Conversion, 495

vrna\_unit\_temperature\_e  
     Unit Conversion, 495

VRNA\_UNIT\_WH  
     Unit Conversion, 495

vrna\_unstructured\_domain\_motif\_s, 580

vrna\_unstructured\_domain\_s, 209  
     prod\_cb, 210

vrna\_urn  
     Utilities, 367

warn\_user  
     basic.h, 894

Washington University Secondary Structure (WUSS) notation, 414  
     vrna\_db\_from\_WUSS, 415

write\_parameter\_file  
     Reading/Writing Energy Parameter Sets from/to File, 388

xrealloc  
     basic.h, 895

xrna\_plot  
     Plotting, 460

xsubi  
     Utilities, 369

zukersubopt  
     Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989, 275

zukersubopt\_par  
     Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989, 275