**Rubric for Assessing Yahtzee Game in Prolog**

Name: Samman Bhetwal

Carefully **highlight** **all** the items that **work correctly**. Incorrect entries may be penalized. Not all the entries may be used for grading.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Setup** | | | | | |
| **Players** | One player is Human | One player is computer |  | Players alternate |  |
| **ScoreCard** | Has 12 categories | Categories are named | Winner of each category is shown | Points earned for the category is shown | Round when the category was filled is shown |
| **First player** | Each player tosses a die | Player with the larger die plays the first turn | The other player plays the second turn |  | If a tie, dice are tossed repeatedly till a first player is determined |
|  | On second and subsequent rounds, player with the lower score plays first | |  | If a tie in the scores, dice are tossed to determine the first player | |
| **Computer player** | | | | | |
| **First roll** | Rolls 5 dice | Announces which dice will be set aside and not re-rolled |  | Lists all the scorecard categories applicable to the dice set aside | Does not list any category already filled |
|  | Announces the category or categories it wants to pursue | Explains why it wants to pursue that category or categories | The explanation is context-specific | Lists the range of points it can earn with the selected category or categories | States whether it wants to stand or roll again |
| **Second roll** | Rolls up to 5 dice | Does not roll dice set aside from the first roll |  | Lists all the scorecard categories applicable to the dice set aside | Does not list any category already filled |
|  | Announces the category or categories it wants to pursue | Explains why it wants to pursue that category or categories | The explanation is context-specific | Lists the range of points it can earn with the selected category or categories | States whether it wants to stand or roll again |
| **Third roll** | Rolls up to 5 dice |  | Does not roll dice set aside in the first roll | Does not roll dice set aside in the second roll |  |
| **End of turn** | Announces the category that will be filled in that turn | Announces the points earned in that category |  | Updates the Scorecard with its name, points and round number |  |
| **Computer’s Strategy** | | | | | |
| **Strategy** | Has strategy for which dice to set aside after each roll | Has strategy for which category to pursue after each roll | Has strategy for whether to roll again or stand |  | Has strategy for which category to fill at the end of the turn |
| **Describe the strategy used to determine the dice to set aside after each roll** | | When the computer receives a dice set, it will search for the available  categories. If the dice set has the potential to fulfill these categories  then the computer decides which dice-values to keep and which to  reroll. | | | |
| **Describe the strategy used to decide which category to prioritize after each roll** | | If sequences categories are available the computer will try to fill  these i.e. prioritise these categories if it recognizes a sequence(3  consecutive numbers) but not to the expense of other categories. For  example if there is yahtzee or four of a kind or full house immediately  then the computer won’t break the dice set in order to get  sequences. if these categories are available it will fill these  categories. | | | |
| **Describe the strategy used to decide whether to roll again or stand** | | if the dice set cannot fill any categories or if there are no patterns  such as no sequences or no duplicates or if there are patterns but  the categories that these patterns can potentially fill are already filled  then it will decide to either roll again or stand. | | | |
| **Describe the strategy used to decide which category to fill at the end of the turn** | | The strategy aims to pursue the best unfilled category with potential  during each roll. If it successfully completes a category at any point,  it selects that category. If unsuccessful after the final roll, it chooses  the category that offers the highest score. | | | |
| **Human player** | | | | | |
| **First roll** | Must roll 5 dice | Will announce which dice will be set aside and not re-rolled | Will list all the scorecard categories applicable to the dice set aside | Cannot list any category already filled | Can list only correct categories |
|  | Will announce the category or categories to pursue | Category to pursue is correct |  |  | Can stand or roll again |
| **Second roll** | Can roll up to 5 dice | Cannot roll dice set aside in the first roll | Will list all the scorecard categories applicable to the dice set aside | Cannot list any category already filled | Can list only correct categories |
|  | Will announce the category or categories to pursue | Category to pursue is correct |  |  | Can stand or roll again |
| **Third roll** | Can roll up to 5 dice |  | Cannot roll dice set aside in the first roll | Cannot roll dice set aside in the second roll |  |
| **End of turn** | Will identify the category that will be filled in that turn | The category to be filled is correct | Will identify the number of points earned for the category | The number of points is correct | Will update the Scorecard with name, points and round number |
| **Playing the Game** | | | | | |
| **End of Turn** | Turn ends after the third roll |  | Turn ends after the player stands |  |  |
| **End of Round** | Round ends when each player has taken one turn | Scorecard correctly updated for both players | The total score of each player is announced | The total score announced is correct | Round can end after only one player has taken turn |
| **End of Tournament** | Tournament ends when all the Scorecard categories have been filled |  | The completed Scorecard is displayed | The total score of each player is announced | The total score announced is correct |
|  | Player with the greatest score is announced as the winner of the tournament | Tournament is a draw if both players have the same number of wins |  |  | Program exits after announcing winner of the tournament |
| **Implementation Features** | | | | | |
| **Serialization** | Provides option to save the game after each round | Game is saved into text file | Correct format used for text file | Game state correctly saved | Game quits upon serialization |
|  | Provides option to resume the game from the text file | Prompts for the name of the text file |  | Resumes loaded game |  |
| Correctly restores round number | Correctly restores the Scorecard |  |  |  |
| **Dice** | For each roll, asks whether to generate or manually input dice | If generate, generates dice randomly |  | If manual input, reads the required number of dice from the user | Validates dice input |
| **Help Mode** | User has the option to ask for help after each roll |  | Help advises on the dice to set aside from that roll | Help lists all the scorecard categories applicable to the dice set aside so far |  |
|  | Help recommends the category or categories that should be pursued | Help explains why it recommends the category or categories | The explanation is context-specific | Help lists the range of points that can be earned for each category being pursued |  |
|  | Help recommends whether to roll again or stand | At the end of the turn, help recommends the category that should be filled |  |  | Computer uses its own strategies to provide help |
| **Game Features** | | | | | |
| **Validates input from human player** | Whether to start a game using a text file | The name of the text file from which to resume a game |  | Whether to suspend a game after a turn | The name of the file in which to save the game |
|  | Manual input of dice |  | Asking for help from the computer |  |  |
|  | Input of dice to set aside/re-roll | Input of categories applicable to the dice set aside so far | Input of category or categories to be pursued |  | Input of whether to roll or stand |
|  | Input of the Scorecard category to be filled at the end of the turn | Input of points earned in the turn to be entered into the Scorecard | Input of round number to be entered into the Scorecard |  |  |
| **Output** | Scorecard displayed in tabular format | Scorecard correctly updated after each turn |  | Dice specified in terms of their face value | Next player clearly identified |
|  | Total score of both players listed at the end of each round | Total score of both players listed at the end of the tournament |  | Computer’s move is described in user-friendly format | Computer’s help displayed in user-friendly format |
| **Design** | | | | | |
| **Logical design** | Each clause in charge of only one logical task |  | Code in clauses properly factored out | Code for repeated execution separated from  code for single execution | Display issues separated from problem logic (Model Vs View) |
| **Code Design – Data flow** | Data flow is through parameters only | Appropriate list structures used to hold data | Appropriate accessors and manipulators provided for each data structure |  | Changes to data always validated |
| **Code Design – Control flow** | Overall design is hierarchical, top-down decomposed | Clause call chart is a balanced tree, not a tail-recursive chain | Overall structure evident in each top-level clause |  |  |
| **Implementation** | | | | | |
| **Identifiers** | All variables have names corresponding to nouns in problem description | All clauses have names corresponding to verbs in problem description | Any abbreviations in the names are readable |  |  |
| **Clauses** | Parameters of clauses are logical | Order of parameters in clauses logical, consistent | Names clarify whether parameters are in or out parameters | List used to group related parameters |  |
| **Coding style** | No imperative constructs used (repeat, if, etc.) | Cut (!), abort not used | Assert, retract not used | All literal constants are explained at *each* occurrence | No singleton variables |
| **Courtesy Programming** | | | | | |
| **Listing** | Code is indented properly | Code does not wrap around to the next line | All the clauses with the same signature are listed together | Clauses are grouped according to their functionality | Clauses are listed in the order in which they are called |
| **Documentation** | Every clause has a complete header | Within each clause, code is properly commented – steps in the algorithm are listed | Comments in the code describe semantics, not syntax | Comments in the code do not have spelling/ grammatical errors. |  |
| **Submission - Manual** | | | | | |
| **Screen shots of:** | First player of the round being determined | Computer’s move being explained | Computer providing help |  | Winner of the tournament being announced |
| **Includes:** | Bug report | Missing features report |  | Project log |  |
|  | Description of data structures |  |  | Source and documentation are placed in a directory and the directory is zipped | |
| **Milestones uploaded?** | First: | | Second: | | |

***Do not delete the blank pages below***