BT 3040: BIOINFORMATICS

Assignment 1

Atharva Mandar Phatak | BE21B009
Department of Biotechnology

Indian Institute of Technology
Madras

Q1) Download the EMBOSS package (http://emboss.sourceforge.net/download/) and copy to your Windows system. In case of Linux use the command: sudo apt-get install jemboss. For Mac users, use the online tool links given in the instructions doc.

Installed Emboss following the instructions, on Windows 11 PC. Created Shortcuts in Desktop

Q2) Using EMBOSS, find the complementary strand for the sequence: CTCGGATTTGTAAAGATCATGATCTCATACATAGTACCTAGCCATTG

Used Reverse SQ to find the complementary strand. We can get the non-reversed output by deselecting the respective box under 'Advance Options'

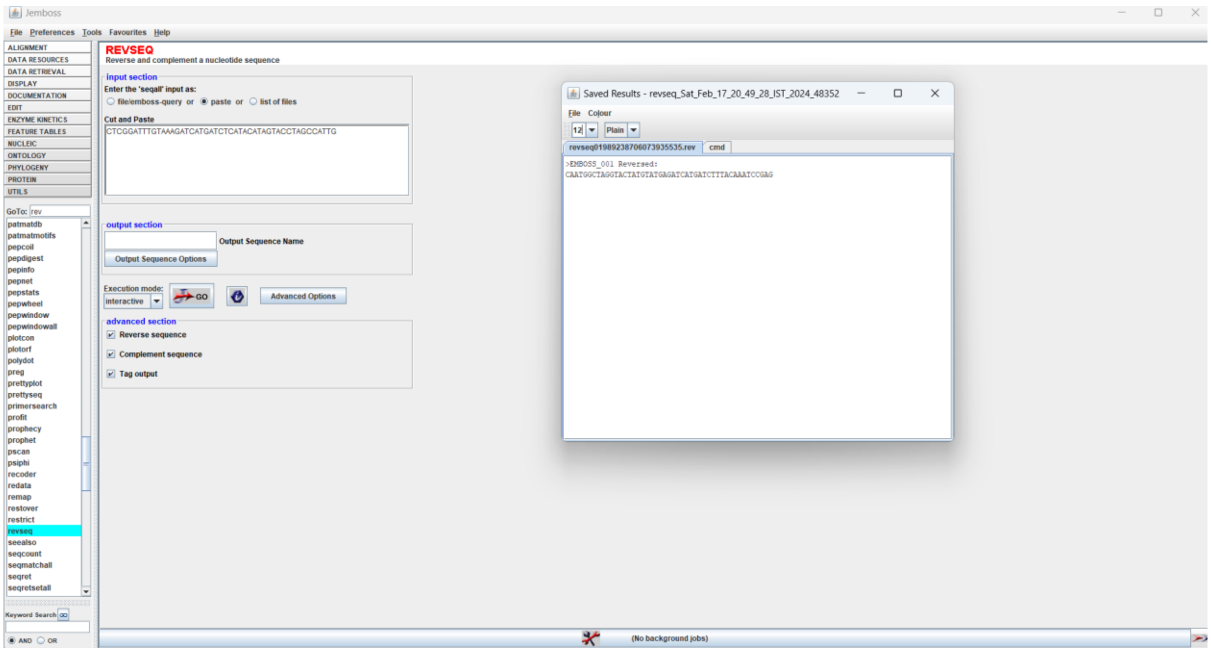
Outputs received:

>EMBOSS_001

GAGCCTAAACATTTCTAGTACTAGAGTATGTATCATGGATCGGTAAC

>EMBOSS_001 Reversed:

CAATGGCTAGGTACTATGTATGAGATCATGATCTTTACAAATCCGAG



REVSEQ
Reverse and complement a nucleotide sequence

input section
Enter the 'seqall' input as:
file/emboss-query or [✓ paste] or list of files
Cut and Paste
CTCGGATTTGTAAAGATCATGATCTCATACATAGTACCTAGCCATTG

output section
Output Sequence Name

Execution mode: interactive

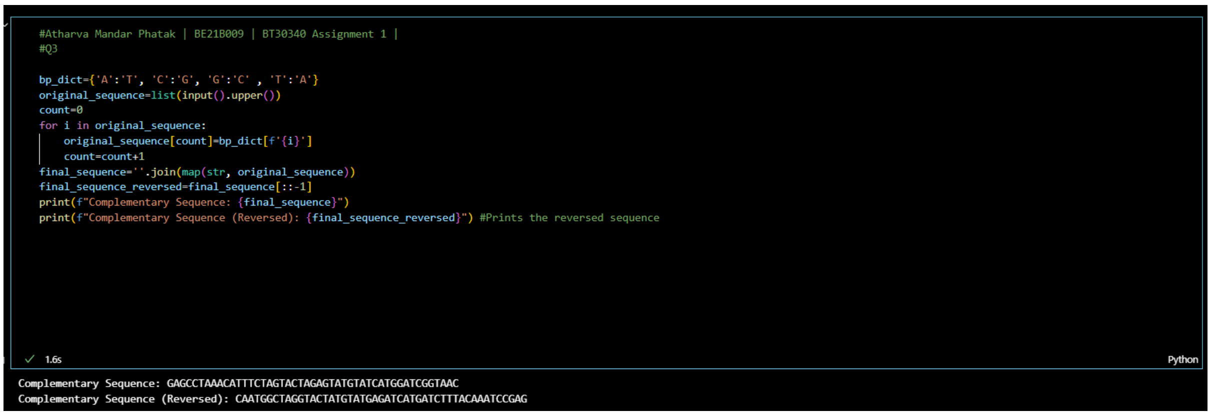
advanced section
[✓] Reverse sequence
[✓] Complement sequence
[✓] Tag output

Saved Results - revseq_Sat_Feb_17_20_49_28_IST_2024_48352

revseq01989238706073935535.rev

```
>EMBOSS_001 Reversed:
CAATGGCTAGGTACTATGTATGAGATCATGATCTTTACAAATCCGAG
```

Q3) Write a program to find the complementary strand for the sequence given in Q2.



```python
#Atharva Mandar Phatak | BE21B009 | BT30340 Assignment 1 |
#Q3

bp_dict={'A':'T', 'C':'G', 'G':'C' , 'T':'A'}
original_sequence=list(input().upper())
count=0
for i in original_sequence:
    original_sequence[count]=bp_dict[f'{i}']
    count=count+1
final_sequence=''.join(map(str, original_sequence))
final_sequence_reversed=final_sequence[::-1]
print(f"Complementary Sequence: {final_sequence}")
print(f"Complementary Sequence (Reversed): {final_sequence_reversed}") #Prints the reversed sequence
```

```
Complementary Sequence: GAGCCTAAACATTTCTAGTACTAGAGTATGTATCATGGATCGGTAAC
Complementary Sequence (Reversed): CAATGGCTAGGTACTATGTATGAGATCATGATCTTTACAAATCCGAG
```

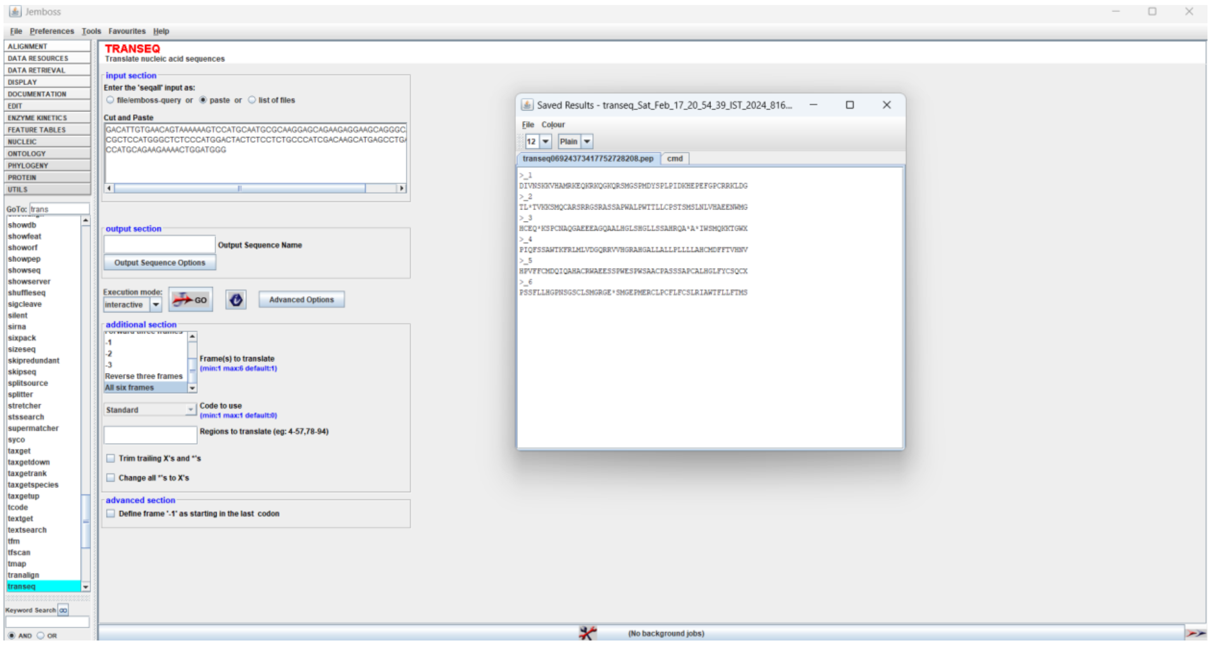
Q4) (i)Using EMBOSS, find the protein sequence for GACATTGTGAACAGTAAAAAAGTCCATGCAATGCGCAAGGAGCAGAAG AGGAAGCAGGGCAAGCAGCGCTCCATGGGCTCTCCCATGGACTACTC TCCTCTGCCCATCGACAAGCATGAGCCTGAATTTGGT CCATGCAGAAGAAAACTGGATGGG

(ii) Identify the reading frame equivalent to the following sequence. PIQFSSAWTKFRLMLVDGQRRVVHGRAHGALLALLPLLLLAHCMDFFTV HNV

(i) Protein sequence Using Emboss. In Advance options select ‘All Six’ to get all the frames

```
>_1

DIVNSKKVHAMRKEQKRKQGKQRSMGSPMDYSPLPIDKHEPEFGPCRRKLDG

>_2

TL*TVKKSMQCARSRRGSRASSAPWALPWTTLLCPSTSMSLNLVHAEENWMG

>_3

HCEQ*KSPCNAQGAEEEAGQAALHGLSHGLLSSAHRQA*A*IWSMQKKTGWX

>_4

PIQFSSAWTKFRLMLVDGQRRVVHGRAHGALLALLPLLLLAHCMDFFTVHNV

>_5

HPVFFCMDQIQAHACRWAEESSPWESPWSAACPASSSAPCALHGLFYCSQCX

>_6

PSSFLLHGPNSGSCLSMGRGE*SMGEPMERCLPCFLFCSLRIAWTFLLFTMS
```

(ii) The reading frame equivalent for the given sequence was found to be the **fourth sequence**. Highlighted in yellow above.



Enter the 'seqall' input as: file/emboss-query or [✓ paste] or list of files

Cut and Paste

```
GACATTGTGAACAGTAAAAAAGTCCATGCAATGCGCAAGGAGCAGAAGAGGAAGCAGGGC
CGCTCCATGGGCTCTCCCATGGACTACTCTCCTCTGCCCATCGACAAGCATGAGCCTG
CCATGCAGAAGAAAACTGGATGGG
```

Execution mode: interactive

Frame(s) to translate: All six frames

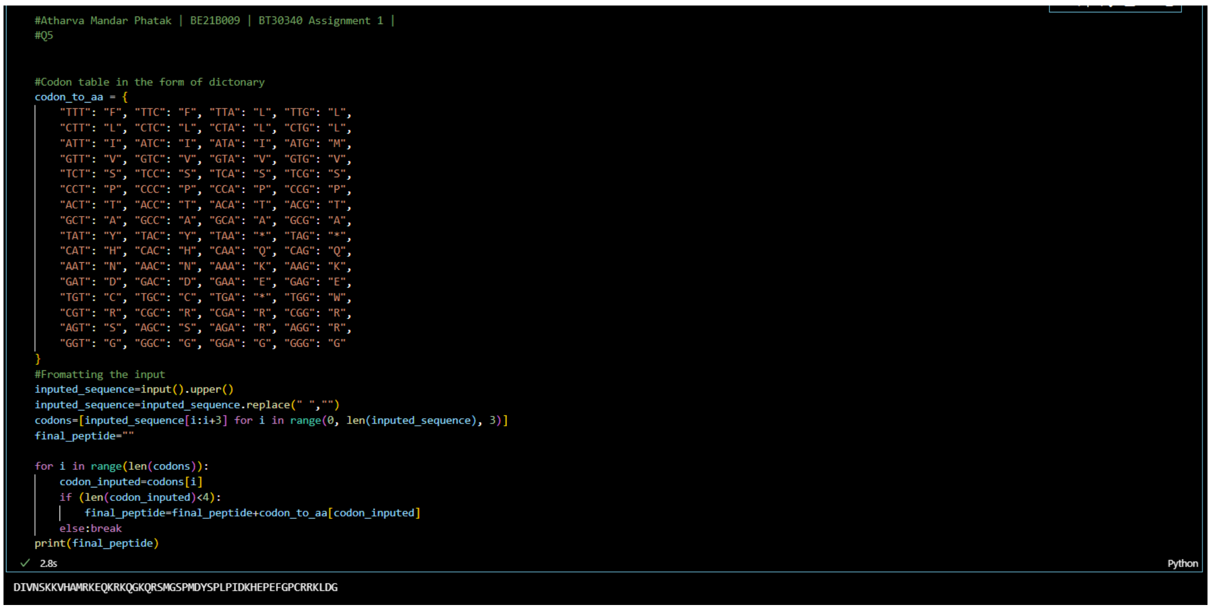
Code to use: Standard

Saved Results - transeq_Sat_Feb_17_20_54_39_IST_2024_816...

transeq06924373417752728208.pep

```
>_1
DIVNSKKVHAMRKEQKRKQGKQRSMGSPMDYSPLPIDKHEPEFGPCRRKLDG
>_2
TL*TVKKSMQCARSRRGSRASSAPWALPWTTLLCPSTSMSLNLVHAEENWMG
>_3
HCEQ*KSPCNAQGAEEEAGQAALHGLSHGLLSSAHRQA*A*IWSMQKKTGWX
>_4
PIQFSSAWTKFRLMLVDGQRRVVHGRAHGALLALLPLLLLAHCMDFFTVHNV
>_5
HPVFFCMDQIQAHACRWAEESSPWESPWSAACPASSSAPCALHGLFYCSQCX
>_6
PSSFLLHGPNSGSCLSMGRGE*SMGEPMERCLPCFLFCSLRIAWTFLLFTMS
```

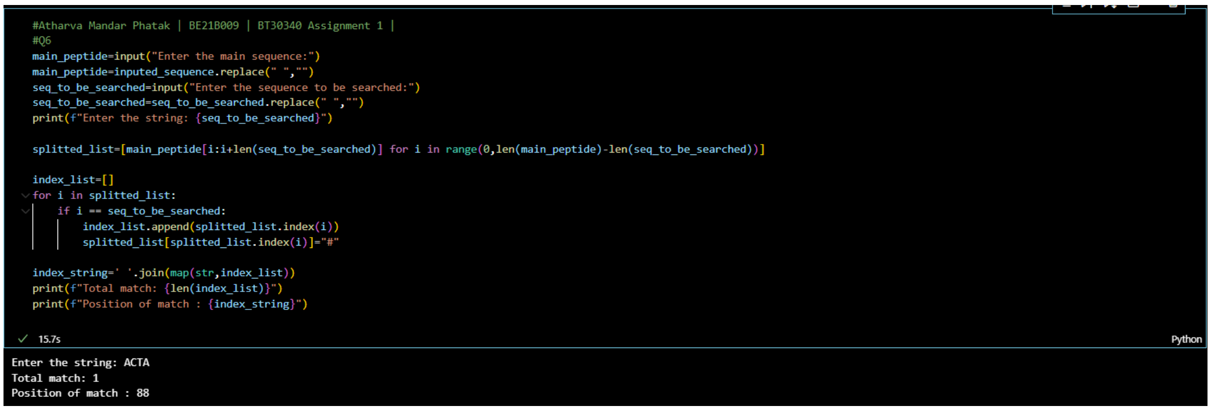
Q5) Write a program to translate the given DNA sequence (refer Q4) to protein sequence.



```python
#Atharva Mandar Phatak | BE21B009 | BT30340 Assignment 1 |
#Q5


#Codon table in the form of dictonary
codon_to_aa = {
    "TTT": "F", "TTC": "F", "TTA": "L", "TTG": "L",
    "CTT": "L", "CTC": "L", "CTA": "L", "CTG": "L",
    "ATT": "I", "ATC": "I", "ATA": "I", "ATG": "M",
    "GTT": "V", "GTC": "V", "GTA": "V", "GTG": "V",
    "TCT": "S", "TCC": "S", "TCA": "S", "TCG": "S",
    "CCT": "P", "CCC": "P", "CCA": "P", "CCG": "P",
    "ACT": "T", "ACC": "T", "ACA": "T", "ACG": "T",
    "GCT": "A", "GCC": "A", "GCA": "A", "GCG": "A",
    "TAT": "Y", "TAC": "Y", "TAA": "*", "TAG": "*",
    "CAT": "H", "CAC": "H", "CAA": "Q", "CAG": "Q",
    "AAT": "N", "AAC": "N", "AAA": "K", "AAG": "K",
    "GAT": "D", "GAC": "D", "GAA": "E", "GAG": "E",
    "TGT": "C", "TGC": "C", "TGA": "*", "TGG": "W",
    "CGT": "R", "CGC": "R", "CGA": "R", "CGG": "R",
    "AGT": "S", "AGC": "S", "AGA": "R", "AGG": "R",
    "GGT": "G", "GGC": "G", "GGA": "G", "GGG": "G"
}
#Fromatting the input
inputed_sequence=input().upper()
inputed_sequence=inputed_sequence.replace(" ","")
codons=[inputed_sequence[i:i+3] for i in range(0, len(inputed_sequence), 3)]
final_peptide=""

for i in range(len(codons)):
    codon_inputed=codons[i]
    if (len(codon_inputed)<4):
        final_peptide=final_peptide+codon_to_aa[codon_inputed]
    else:break
print(final_peptide)
```

```
✓ 2.8s                                                                    Python
DIVNSKKVHAMRKEQKRKQGKQRSMGSPMDYSPLPIDKHEPEFGPCRRKLDG
```

Q6) Write a code to search for the following strings 'AAG', 'GTC', 'GAG, 'ACTA', and 'ATAT' in the DNA sequence provided in Q4. The program should print the total number of matches for the each of the given strings and the start positions of the matches.



```python
#Atharva Mandar Phatak | BE21B009 | BT30340 Assignment 1 |
#Q6
main_peptide=input("Enter the main sequence:")
main_peptide=inputed_sequence.replace(" ","")
seq_to_be_searched=input("Enter the sequence to be searched:")
seq_to_be_searched=seq_to_be_searched.replace(" ","")
print(f"Enter the string: {seq_to_be_searched}")

splitted_list=[main_peptide[i:i+len(seq_to_be_searched)] for i in range(0,len(main_peptide)-len(seq_to_be_searched))]

index_list=[]
for i in splitted_list:
    if i == seq_to_be_searched:
        index_list.append(splitted_list.index(i))
        splitted_list[splitted_list.index(i)]="#"

index_string=' '.join(map(str,index_list))
print(f"Total match: {len(index_list)}")
print(f"Position of match : {index_string}")
```

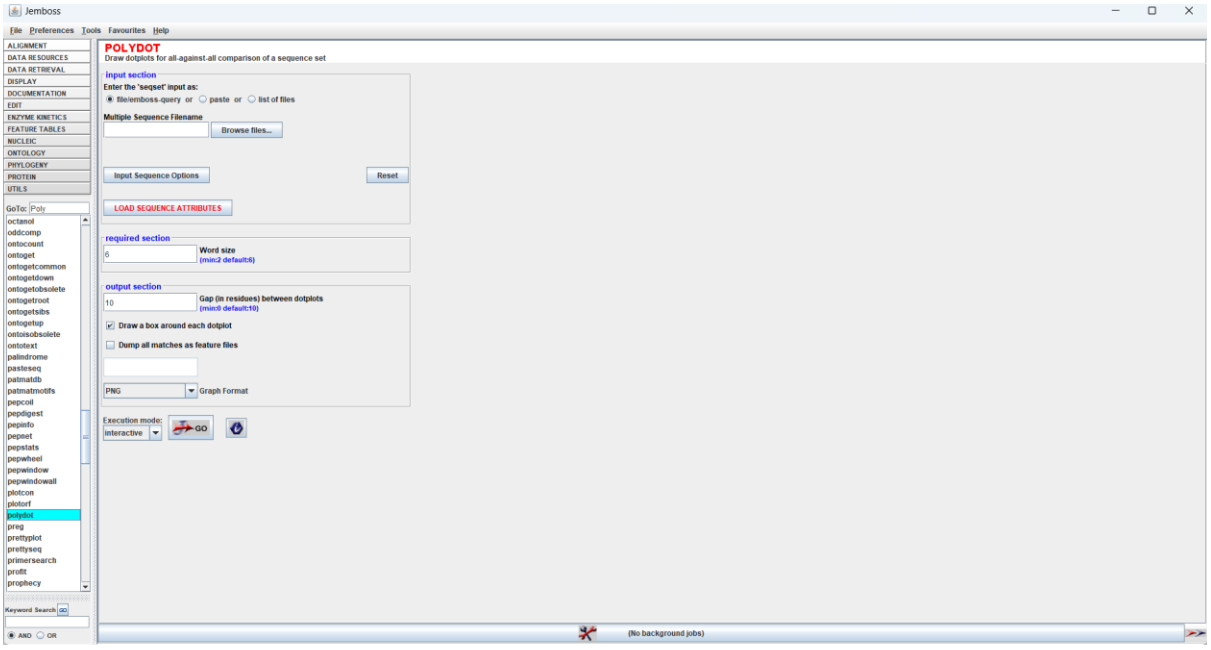
```
✓ 15.7s                                                     Python
```

```
Enter the string: ACTA
Total match: 1
Position of match : 88
```

Q7) Familiarize with other applications in EMBOSS. For example, melting temperature, bending, curvature etc.

a) POLYDOT

Draws dot plot for all-against-all comparison of sequence set



POLYDOT
Draw dotplots for all-against-all comparison of a sequence set

input section
Enter the 'seqset' input as: [✓ file/emboss-query] or paste or list of files
Multiple Sequence Filename:

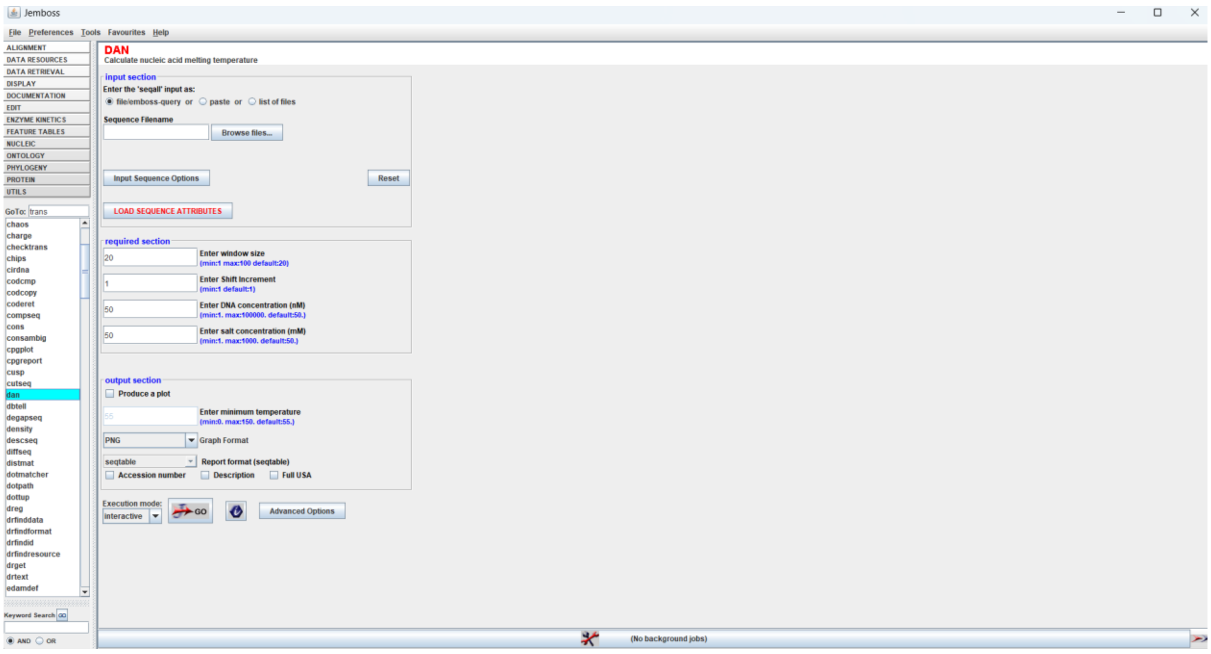
required section
Word size: 6 (min:2 default:6)

output section
Gap (in residues) between dotplots: 10 (min:0 default:10)
[✓] Draw a box around each dotplot
[ ] Dump all matches as feature files
Graph Format: PNG

Execution mode: interactive

b) DAN

Calculates nucleic acid melting temperature



DAN
Calculate nucleic acid melting temperature

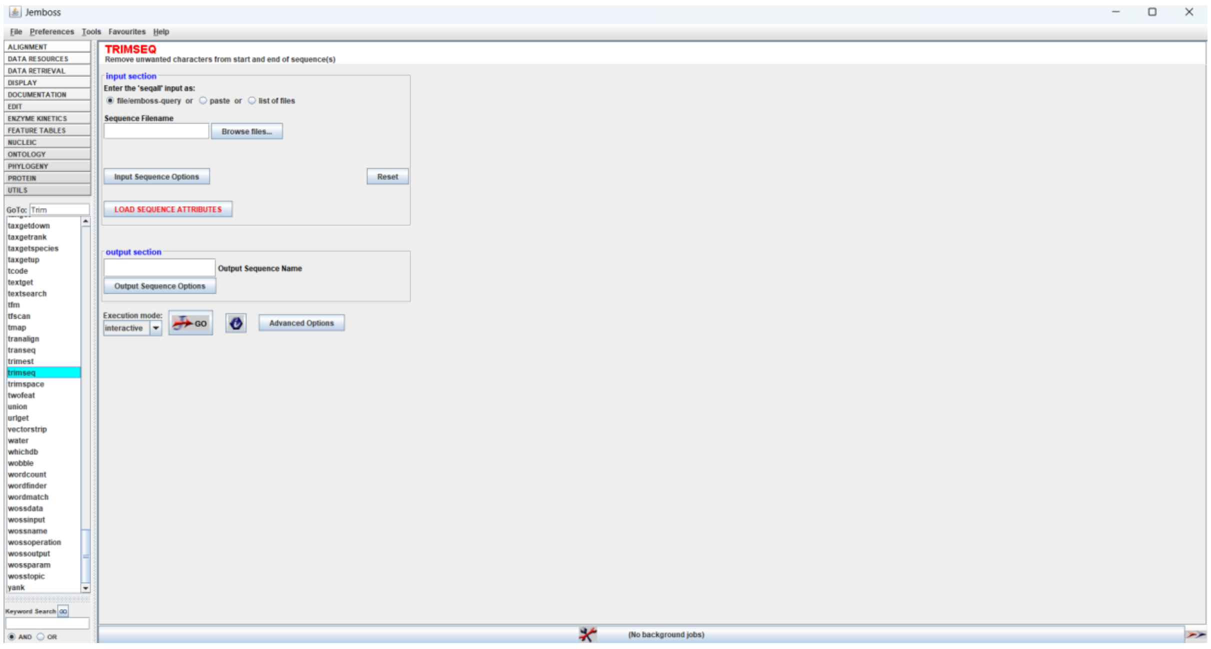
input section
Enter the 'seqall' input as: [✓ file/emboss-query] or paste or list of files
Sequence Filename

required section
20 Enter window size (min:1 max:100 default:20)
1 Enter Shift Increment (min:1 default:1)
50 Enter DNA concentration (nM) (min:1. max:100000. default:50.)
50 Enter salt concentration (mM) (min:1. max:1000. default:50.)

output section
Produce a plot
55 Enter minimum temperature (min:0. max:150. default:55.)
PNG Graph Format
seqtable Report format (seqtable)
Accession number Description Full USA
Execution mode: interactive

c) TRIMSEQ

Removes unwanted characters from start to end.



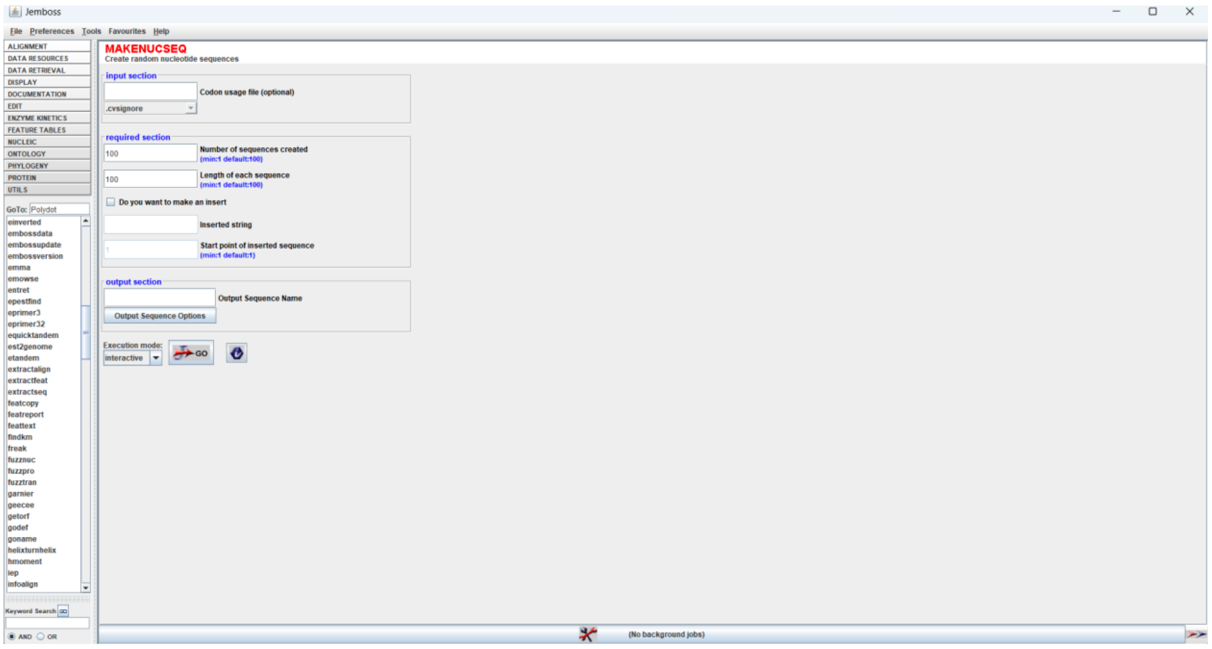
TRIMSEQ
Remove unwanted characters from start and end of sequence(s)

input section
Enter the 'seqall' input as: [✓ file/emboss-query] or paste or list of files
Sequence Filename

output section
Output Sequence Name
Execution mode: interactive

d) MAKEUCESQ

Creates random nucleotide sequences.



MAKENUCSEQ
Create random nucleotide sequences

input section
Codon usage file (optional):
.cvsignore

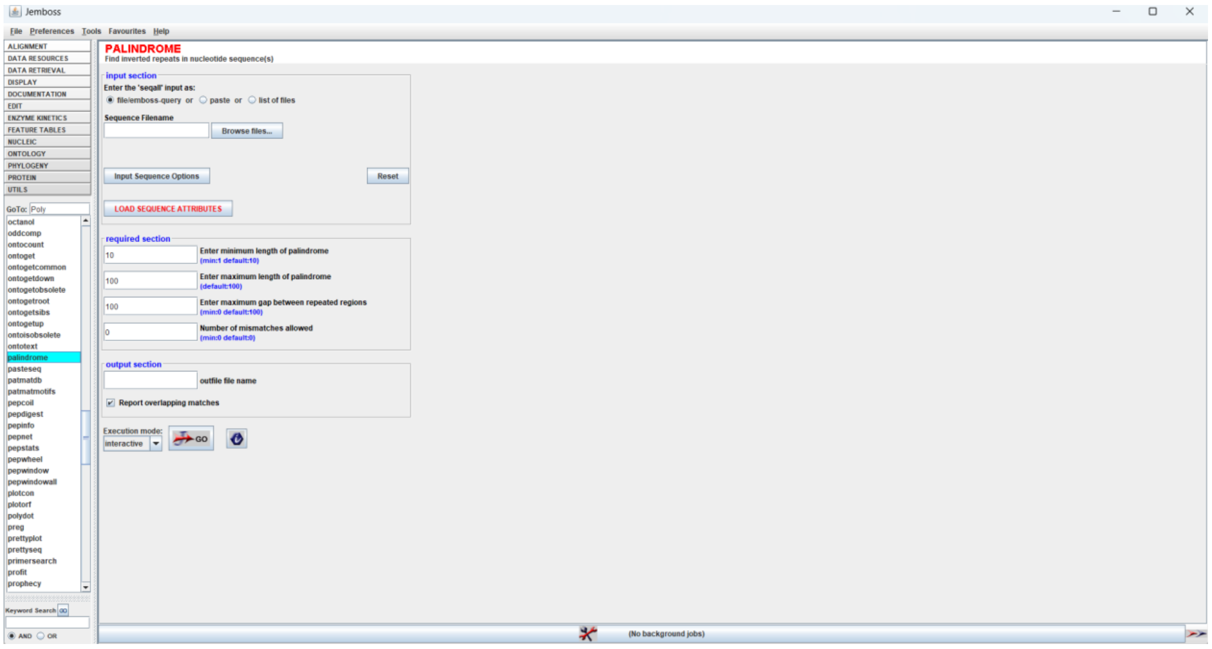
required section
100 Number of sequences created (min:1 default:100)
100 Length of each sequence (min:1 default:100)
☐ Do you want to make an insert
Inserted string:
1 Start point of inserted sequence (min:1 default:1)

output section
Output Sequence Name:

Execution mode: interactive

e) PALINDROME

Finds Palindromes, inverted repeats in the nucleotide sequence(s)



PALINDROME
Find inverted repeats in nucleotide sequence(s)

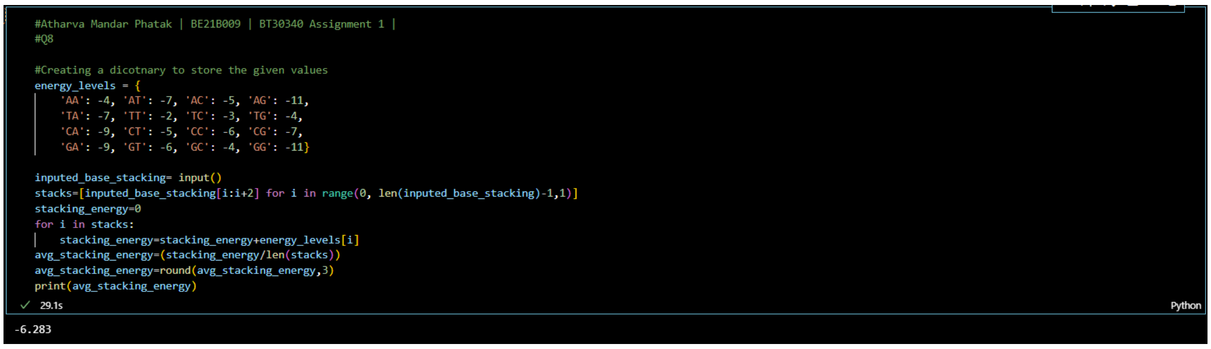
input section
Enter the 'seqall' input as:
[✓ file/emboss-query] or paste or list of files
Sequence Filename:

required section
10 Enter minimum length of palindrome (min:1 default:10)
100 Enter maximum length of palindrome (default:100)
100 Enter maximum gap between repeated regions (min:0 default:100)
0 Number of mismatches allowed (min:0 default:0)

output section
outfile file name:
☑ Report overlapping matches

Execution mode: interactive

Q8) Write a program to compute the average base stacking energy for the sequence given in Q2 (AA: - 4; AT: -7; AC: -5; AG: -11; TA: -7; TT: -2; TC: -3; TG: -4; CA: -9; CT: -5; CC: -6; CG: -7; GA: - 9; GT: -6; GC: -4; GG: -11).



```python
#Atharva Mandar Phatak | BE21B009 | BT30340 Assignment 1 |
#Q8

#Creating a dicotnary to store the given values
energy_levels = {
    'AA': -4, 'AT': -7, 'AC': -5, 'AG': -11,
    'TA': -7, 'TT': -2, 'TC': -3, 'TG': -4,
    'CA': -9, 'CT': -5, 'CC': -6, 'CG': -7,
    'GA': -9, 'GT': -6, 'GC': -4, 'GG': -11}

inputed_base_stacking= input()
stacks=[inputed_base_stacking[i:i+2] for i in range(0, len(inputed_base_stacking)-1,1)]
stacking_energy=0
for i in stacks:
    stacking_energy=stacking_energy+energy_levels[i]
avg_stacking_energy=(stacking_energy/len(stacks))
avg_stacking_energy=round(avg_stacking_energy,3)
print(avg_stacking_energy)
```

✓ 29.1s

Python

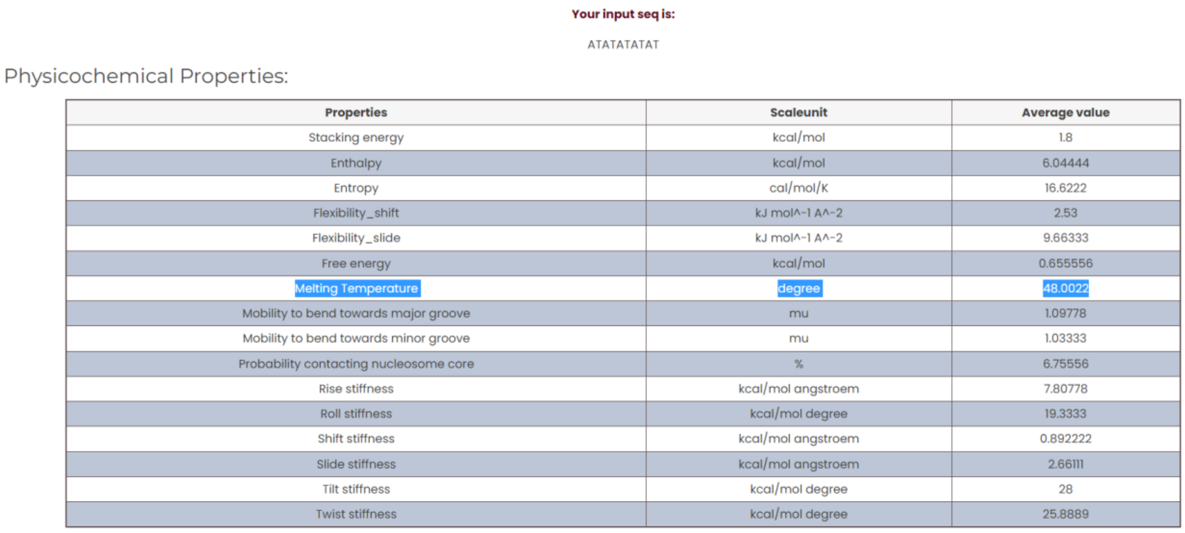
```
-6.283
```

Q9) Compute the average melting temperature of the following sequences using Seq2Feature tool (https://www.iitm.ac.in/bioinfo/SBFE/index.html) and comment on the results (Enter one sequence at a time in fasta format)
(i) ATATATATAT ii) GCGCGCGCGC

i) Average Melting Temperature: 48.0022° C

ii) Average Melting Temperature: 107.867° C

(i)



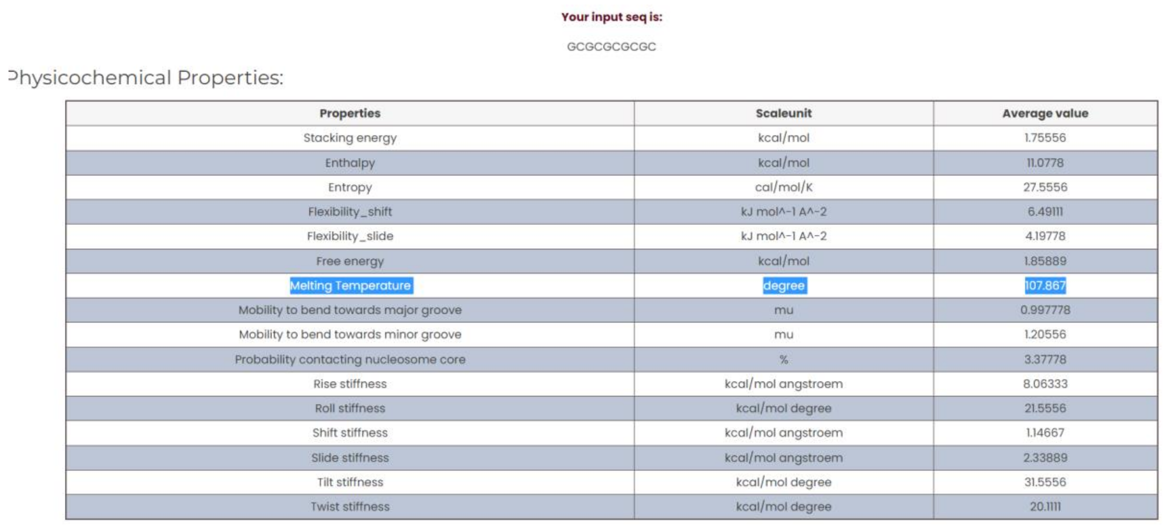
Your input seq is:

ATATATATAT

Physicochemical Properties:

Properties	Scaleunit	Average value
Stacking energy	kcal/mol	1.8
Enthalpy	kcal/mol	6.04444
Entropy	cal/mol/K	16.6222
Flexibility_shift	kJ mol^-1 A^-2	2.53
Flexibility_slide	kJ mol^-1 A^-2	9.66333
Free energy	kcal/mol	0.655556
Melting Temperature	degree	48.0022
Mobility to bend towards major groove	mu	1.09778
Mobility to bend towards minor groove	mu	1.03333
Probability contacting nucleosome core	%	6.75556
Rise stiffness	kcal/mol angstroem	7.80778
Roll stiffness	kcal/mol degree	19.3333
Shift stiffness	kcal/mol angstroem	0.892222
Slide stiffness	kcal/mol angstroem	2.66111
Tilt stiffness	kcal/mol degree	28
Twist stiffness	kcal/mol degree	25.8889

ii)



Your input seq is:

GCGCGCGCGC

Physicochemical Properties:

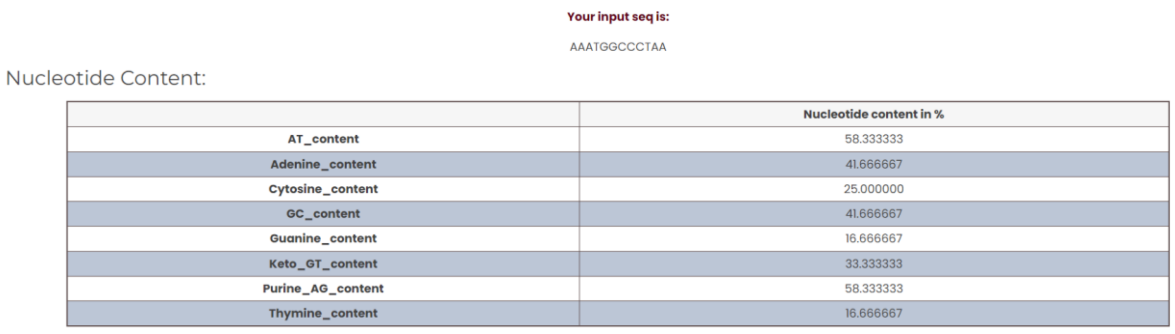
Properties	Scaleunit	Average value
Stacking energy	kcal/mol	1.75556
Enthalpy	kcal/mol	11.0778
Entropy	cal/mol/K	27.5556
Flexibility_shift	kJ mol^-1 A^-2	6.49111
Flexibility_slide	kJ mol^-1 A^-2	4.19778
Free energy	kcal/mol	1.85889
Melting Temperature	degree	107.867
Mobility to bend towards major groove	mu	0.997778
Mobility to bend towards minor groove	mu	1.20556
Probability contacting nucleosome core	%	3.37778
Rise stiffness	kcal/mol angstroem	8.06333
Roll stiffness	kcal/mol degree	21.5556
Shift stiffness	kcal/mol angstroem	1.14667
Slide stiffness	kcal/mol angstroem	2.33889
Tilt stiffness	kcal/mol degree	31.5556
Twist stiffness	kcal/mol degree	20.1111

Thus i) sequence shows lower melting point as compared to the second sequence.
Reason : G-C pairs have a higher stacking energy than A-T pairs because they make three hydrogen bonds in water whereas AT only forms two hydrogen bonds resulting into GC pair have high melting point. Thus, the sequence with more GC content will have a higher melting point.

Q10) Calculate the AT and GC content of the sequence AAATGGCCCTAA using Seq2Feature too

AT Content = 5.3334 %

GC Content = 41.6667 %



Your input seq is:

AAATGGCCCTAA

Nucleotide Content:

	Nucleotide content in %
AT_content	58.333333
Adenine_content	41.666667
Cytosine_content	25.000000
GC_content	41.666667
Guanine_content	16.666667
Keto_GT_content	33.333333
Purine_AG_content	58.333333
Thymine_content	16.666667