Practicals 1

-BS19B032

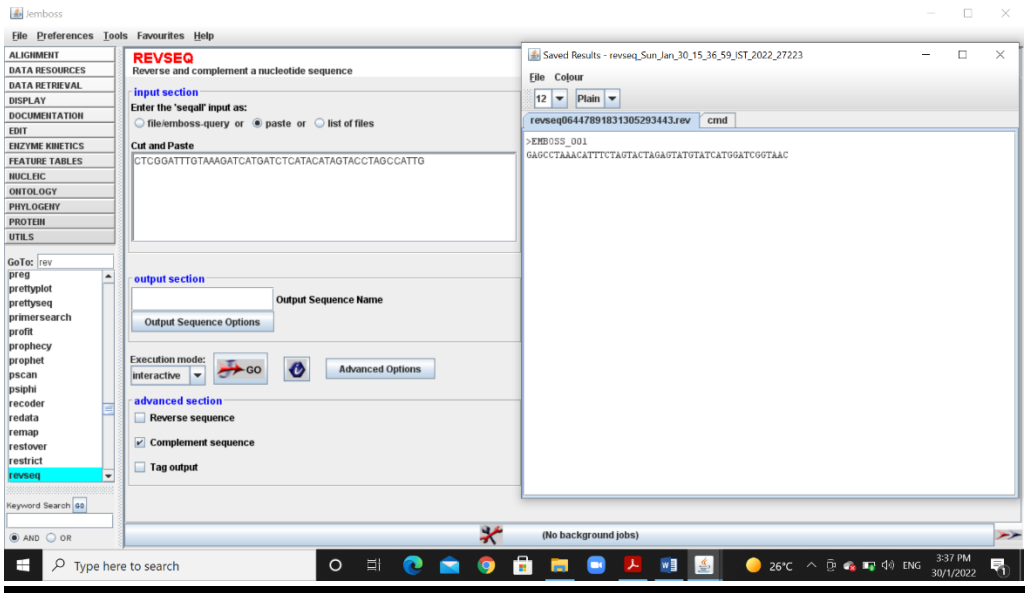
-R. Vasantha Kumar

1) I have downloaded emboss in my computer, as per the demo shown in the class.

2) The complementary strand for the given sequence is:

GAGCCTAAACATTTCTAGTACTAGAGTATGTATCATGGATCGGTAAC

*(*not reversed).*



3) This is a python program that is used to find the complementary strand of given sequence:

```
seq=input("Enter your DNA sequence:")      #taking input sequence
comp_seq=""


for i in range(len(seq)):
    if(seq[i]=='A'):
        comp_seq+='T'          #creating complementary sequence
    if(seq[i]=='T'):
        comp_seq+='A'
    if(seq[i]=='C'):          #A to T; T to A; C to G; G to C
        comp_seq+='G'
    if(seq[i]=='G'):
        comp_seq+='C'


print(comp_seq)
```

The output is:

GAGCCTAAACATTTCTAGTACTAGAGTATGTATCATGGATCGGTAAC

*4**a**) Using EMBOSS the protein sequence for given DNA sequence is:*

Reading Frame 1:

DIVNSKKVHAMRKEQKRKQGKQRSMGSPMDYSPLPIDKHEPEFGPCRRKLDG

Reading Frame 2:

*TL*TVKKSMQCARSRRGSRASSAPWALPWTTLLCPSTSMSLNLVHAEENWMG*

Reading Frame 3:

*HCEQ*KSPCNAQGAEEEAGQAALHGLSHGLLSSAHRQA*A*IWSMQKKTGWX*

Reading Frame -1:

PIQFSSAWTKFRLMLVDGQRRVVHGRAHGALLALLPLLLLAHCMDFFTVHNV

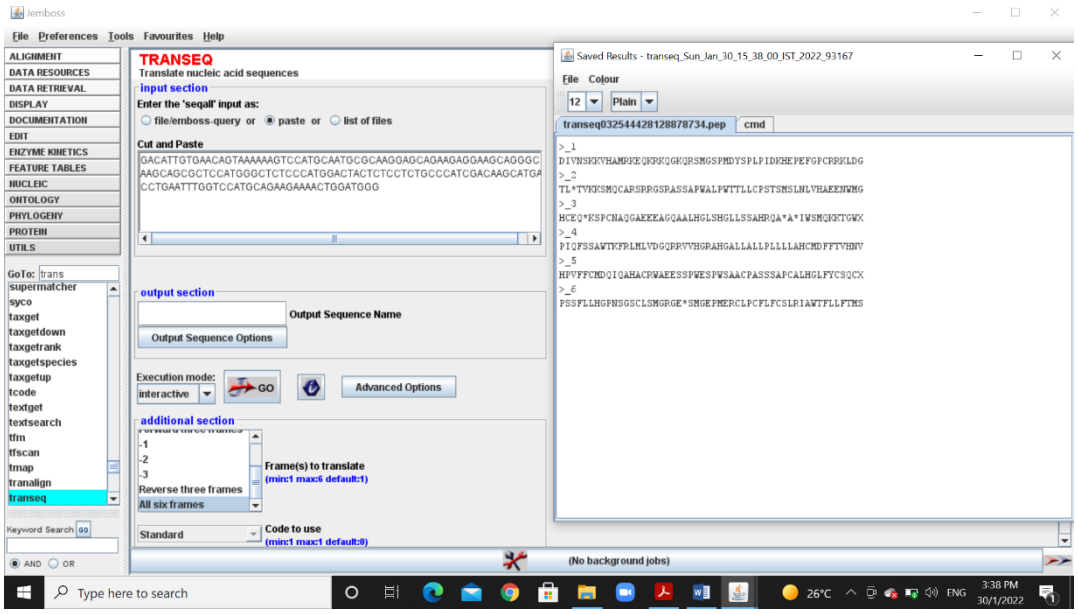
Reading Frame -2:

HPVFFCMDQIQAHACRWAEESSPWESPWSAACPASSSAPCALHGLFYCSQCX

Reading Frame -3:

*PSSFLLHGPNSGSCLSMGRGE*SMGEPMERCLPCFLFCSLRIAWTFLLFTMS*

*4**b**) By checking with the above sequences we got, the reading frame corresponding to given sequence is: **-1.***



5) This is a python program to translate given DNA sequence into protein sequence:

```
seq=input("Enter your DNA sequence:")
mrna_seq=""
protein=""

for i in range(len(seq)):
    if(seq[i]=='T'):
        mrna_seq+='U'
    else:
        mrna_seq+=seq[i]

length=0
if(len(mrna_seq)%3==0):
    length=len(mrna_seq)
elif(len(mrna_seq)%3==1):
```

```
    length=len(mrna_seq)-1
else:
    length=len(mrna_seq)-2

for i in range(0,length,3):
    if(mrna_seq[i]=='A'):
        if(mrna_seq[i+1]=='A'):
            if(mrna_seq[i+2]=='A'):
                protein+='K'
            if(mrna_seq[i+2]=='U'):
                protein+='N'
            if(mrna_seq[i+2]=='C'):
                protein+='N'
            if(mrna_seq[i+2]=='G'):
                protein+='K'
        if(mrna_seq[i+1]=='U'):
            if(mrna_seq[i+2]=='A'):
                protein+='I'
            if(mrna_seq[i+2]=='U'):
                protein+='I'
            if(mrna_seq[i+2]=='C'):
                protein+='I'
            if(mrna_seq[i+2]=='G'):
                protein+='M'
        if(mrna_seq[i+1]=='C'):
            if(mrna_seq[i+2]=='A'):
                protein+='T'
            if(mrna_seq[i+2]=='U'):
                protein+='T'
            if(mrna_seq[i+2]=='C'):
                protein+='T'
            if(mrna_seq[i+2]=='G'):
                protein+='T'
        if(mrna_seq[i+1]=='G'):
```

```
        if(mrna_seq[i+2]=='A'):
            protein+='R'
        if(mrna_seq[i+2]=='U'):
            protein+='S'
        if(mrna_seq[i+2]=='C'):
            protein+='S'
        if(mrna_seq[i+2]=='G'):
            protein+='R'
    if(mrna_seq[i]=='U'):
        if(mrna_seq[i+1]=='A'):
            if(mrna_seq[i+2]=='A'):
                protein+='*'
            if(mrna_seq[i+2]=='U'):
                protein+='Y'
            if(mrna_seq[i+2]=='C'):
                protein+='Y'
            if(mrna_seq[i+2]=='G'):
                protein+='*'
        if(mrna_seq[i+1]=='U'):
            if(mrna_seq[i+2]=='A'):
                protein+='J'
            if(mrna_seq[i+2]=='U'):
                protein+='F'
            if(mrna_seq[i+2]=='C'):
                protein+='F'
            if(mrna_seq[i+2]=='G'):
                protein+='J'
        if(mrna_seq[i+1]=='C'):
            if(mrna_seq[i+2]=='A'):
                protein+='S'
            if(mrna_seq[i+2]=='U'):
                protein+='S'
            if(mrna_seq[i+2]=='C'):
                protein+='S'
```

```
            if(mrna_seq[i+2]=='G'):
                protein+='S'
        if(mrna_seq[i+1]=='G'):
            if(mrna_seq[i+2]=='A'):
                protein+='*'
            if(mrna_seq[i+2]=='U'):
                protein+='C'
            if(mrna_seq[i+2]=='C'):
                protein+='C'
            if(mrna_seq[i+2]=='G'):
                protein+='W'
    if(mrna_seq[i]=='C'):
        if(mrna_seq[i+1]=='A'):
            if(mrna_seq[i+2]=='A'):
                protein+='Q'
            if(mrna_seq[i+2]=='U'):
                protein+='H'
            if(mrna_seq[i+2]=='C'):
                protein+='H'
            if(mrna_seq[i+2]=='G'):
                protein+='Q'
        if(mrna_seq[i+1]=='U'):
            if(mrna_seq[i+2]=='A'):
                protein+='L'
            if(mrna_seq[i+2]=='U'):
                protein+='L'
            if(mrna_seq[i+2]=='C'):
                protein+='L'
            if(mrna_seq[i+2]=='G'):
                protein+='L'
        if(mrna_seq[i+1]=='C'):
            if(mrna_seq[i+2]=='A'):
                protein+='P'
            if(mrna_seq[i+2]=='U'):
```

```
            protein+='P'
        if(mrna_seq[i+2]=='C'):
            protein+='P'
        if(mrna_seq[i+2]=='G'):
            protein+='P'
    if(mrna_seq[i+1]=='G'):
        if(mrna_seq[i+2]=='A'):
            protein+='R'
        if(mrna_seq[i+2]=='U'):
            protein+='R'
        if(mrna_seq[i+2]=='C'):
            protein+='R'
        if(mrna_seq[i+2]=='G'):
            protein+='R'
if(mrna_seq[i]=='G'):
    if(mrna_seq[i+1]=='A'):
        if(mrna_seq[i+2]=='A'):
            protein+='E'
        if(mrna_seq[i+2]=='U'):
            protein+='D'
        if(mrna_seq[i+2]=='C'):
            protein+='D'
        if(mrna_seq[i+2]=='G'):
            protein+='E'
    if(mrna_seq[i+1]=='U'):
        if(mrna_seq[i+2]=='A'):
            protein+='V'
        if(mrna_seq[i+2]=='U'):
            protein+='V'
        if(mrna_seq[i+2]=='C'):
            protein+='V'
        if(mrna_seq[i+2]=='G'):
            protein+='V'
    if(mrna_seq[i+1]=='C'):
```

```
            if(mrna_seq[i+2]=='A'):
                protein+='A'
            if(mrna_seq[i+2]=='U'):
                protein+='A'
            if(mrna_seq[i+2]=='C'):
                protein+='A'
            if(mrna_seq[i+2]=='G'):
                protein+='A'
        if(mrna_seq[i+1]=='G'):
            if(mrna_seq[i+2]=='A'):
                protein+='G'
            if(mrna_seq[i+2]=='U'):
                protein+='G'
            if(mrna_seq[i+2]=='C'):
                protein+='G'
            if(mrna_seq[i+2]=='G'):
                protein+='G'

print(protein)
```

The protein sequence I got is:

DIVNSKKVHAMRKEQKRKQGKQRSMGSPMDYSPLPIDKHEPEFGPCRRKLDG

6) The python code to find matches in the string is:

```
seq=input("Enter your sequence:")
check_str=input("Enter the string to search:")
matches=0

for i in range(len(seq)-len(check_str)+1):
    count=0
    for j in range(len(check_str)):
        if(seq[i+j]==check_str[j]):
```

```
            count=count+1
        if(count==len(check_str)):
            matches=matches+1
            print("Match location:",i)
print("Total matches:",matches)
```

*The output of the code when we input string in question 4 and the match string as **AAG:***

Match location: 19

Match location: 36

Match location: 45

Match location: 51

Match location: 60

Match location: 111

Match location: 140

Total matches: 7

*The output of the code when we input string in question 4 and the match string as **ACTA:***

Match location: 88

Total matches: 1

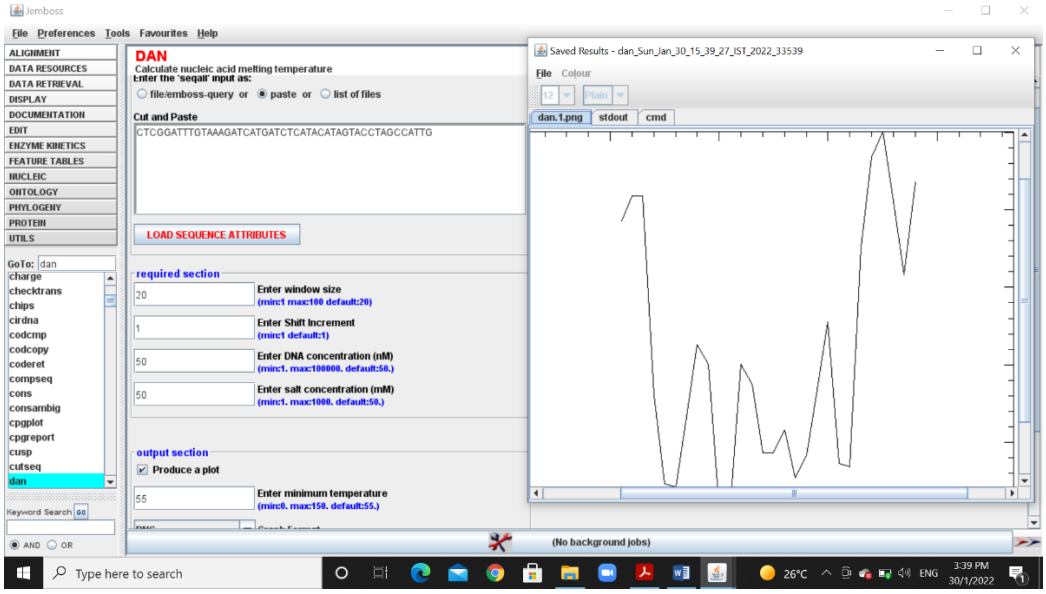
By running the code with strings of different length, I observed that as string length increases, matches decreases.

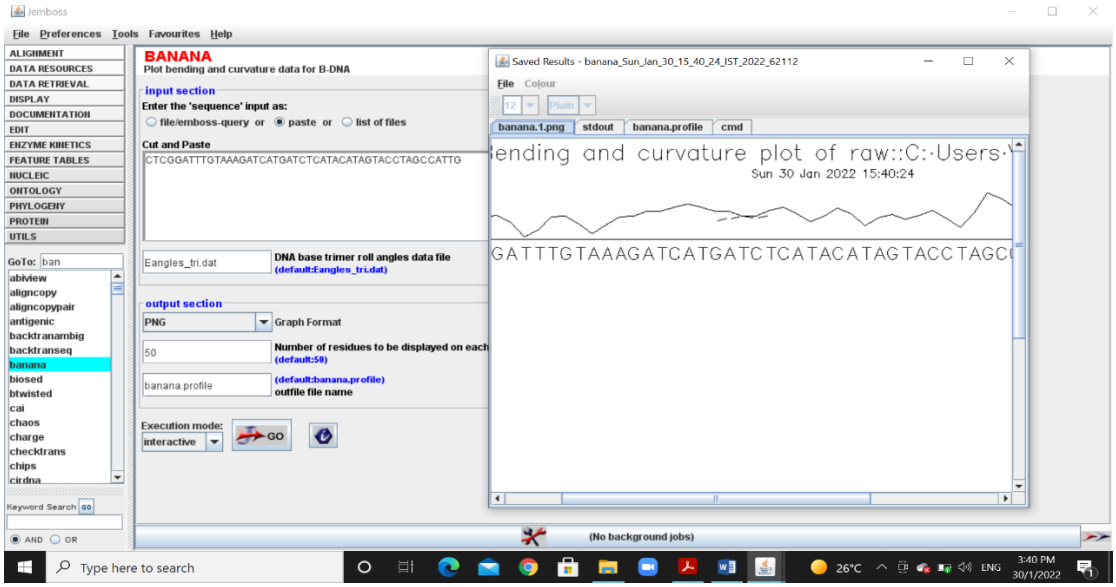
7) I learnt some of these applications of emboss.

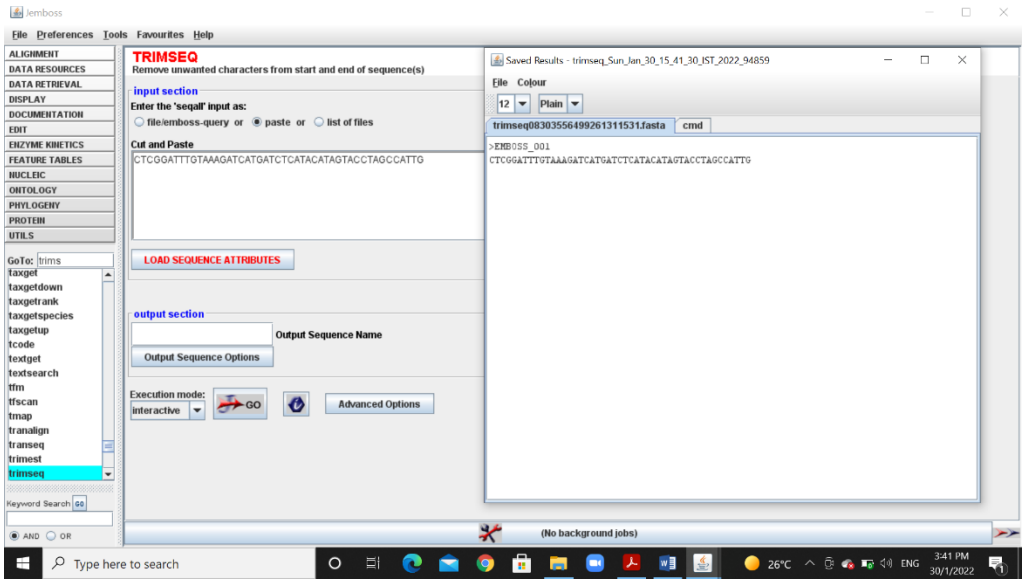
***dan**: This tool in emboss is used to find the melting temperature of the given nucleotide sequence. This tool also provides us with plot on melting temperature.*

***Banana**: This tool in emboss is used to plot bending and curvature of given DNA sequence.*

***trimseq**: This tool helps us to remove unwanted characters like "*" in start and end of the sequence.*


Jemboss
File Preferences Tools Favourites Help
ALIGNMENT
DATA RESOURCES
DATA RETRIEVAL
DISPLAY
DOCUMENTATION
EDIT
ENZYME KINETICS
FEATURE TABLES
NUCLEIC
ONTOLOGY
PHYLOGENY
PROTEIN
UTILS
GoTo: dan
charge
checktrans
chips
cirdna
codcmp
codcopy
coderet
compseq
cons
consambig
cpgplot
cpgreport
cusp
cutseq
dan
Keyword Search GO
AND OR
DAN
Calculate nucleic acid melting temperature
Enter the 'seqall' input as:
file/emboss-query or paste or list of files
Cut and Paste
CTCGGATTTGTAAAGATCATGATCTCATACATAGTACCTAGCCATTG
LOAD SEQUENCE ATTRIBUTES
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.)
Saved Results - dan_Sun_Jan_30_15_39_27_IST_2022_33539
File Colour
dan.1.png
stdout
cmd
(No background jobs)
Type here to search
26°C
ENG
3:39 PM
30/1/2022


Jemboss
File Preferences Tools Favourites Help
ALIGNMENT
DATA RESOURCES
DATA RETRIEVAL
DISPLAY
DOCUMENTATION
EDIT
ENZYME KINETICS
FEATURE TABLES
NUCLEIC
ONTOLOGY
PHYLOGENY
PROTEIN
UTILS
GoTo: ban
abiview
aligncopy
aligncopypair
antigenic
backtranambig
backtranseq
banana
biosed
btwisted
cai
chaos
charge
checktrans
chips
cirdna
Keyword Search GO
AND OR
BANANA
Plot bending and curvature data for B-DNA
input section
Enter the 'sequence' input as:
file/emboss-query or paste or list of files
Cut and Paste
CTCGGATTTGTAAAGATCATGATCTCATACATAGTACCTAGCCATTG
Eangles_tri.dat
DNA base trimer roll angles data file
(default:Eangles_tri.dat)
output section
PNG
Graph Format
50
Number of residues to be displayed on each
(default:50)
banana.profile
(default:banana.profile)
outfile file name
Execution mode:
interactive
GO
Saved Results - banana_Sun_Jan_30_15_40_24_IST_2022_62112
File Colour
banana.1.png
stdout
banana.profile
cmd
ending and curvature plot of raw::C:·Users·
Sun 30 Jan 2022 15:40:24
GATTTGTAAAGATCATGATCTCATACATAGTACCTAGC
(No background jobs)
Type here to search
26°C
ENG
3:40 PM
30/1/2022



8) The python program to find average stacking energy of given sequence is:

```
seq=input("Enter your sequence:")
stack_energy=0


for i in range(len(seq)-1):
    if(seq[i]=='A'):
        if(seq[i+1]=='A'):
            stack_energy=stack_energy + (-4)
        if(seq[i+1]=='T'):
            stack_energy=stack_energy + (-7)
        if(seq[i+1]=='C'):
            stack_energy=stack_energy + (-5)
        if(seq[i+1]=='G'):
            stack_energy=stack_energy + (-11)
    if(seq[i]=='T'):
        if(seq[i+1]=='A'):
            stack_energy=stack_energy + (-7)
        if(seq[i+1]=='T'):
            stack_energy=stack_energy + (-2)
        if(seq[i+1]=='C'):
```

```
            stack_energy=stack_energy + (-3)
        if(seq[i+1]=='G'):
            stack_energy=stack_energy + (-4)
    if(seq[i]=='C'):
        if(seq[i+1]=='A'):
            stack_energy=stack_energy + (-9)
        if(seq[i+1]=='T'):
            stack_energy=stack_energy + (-5)
        if(seq[i+1]=='C'):
            stack_energy=stack_energy + (-6)
        if(seq[i+1]=='G'):
            stack_energy=stack_energy + (-7)
    if(seq[i]=='G'):
        if(seq[i+1]=='A'):
            stack_energy=stack_energy + (-9)
        if(seq[i+1]=='T'):
            stack_energy=stack_energy + (-6)
        if(seq[i+1]=='C'):
            stack_energy=stack_energy + (-4)
        if(seq[i+1]=='G'):
            stack_energy=stack_energy + (11)

avg_stack_energy=stack_energy/(len(seq)-1)

print("The average stacking energy of given sequence is:",avg_stack_energy)
```

When we provide the input sequence given in the question 2, the output average stacking energy we get is:

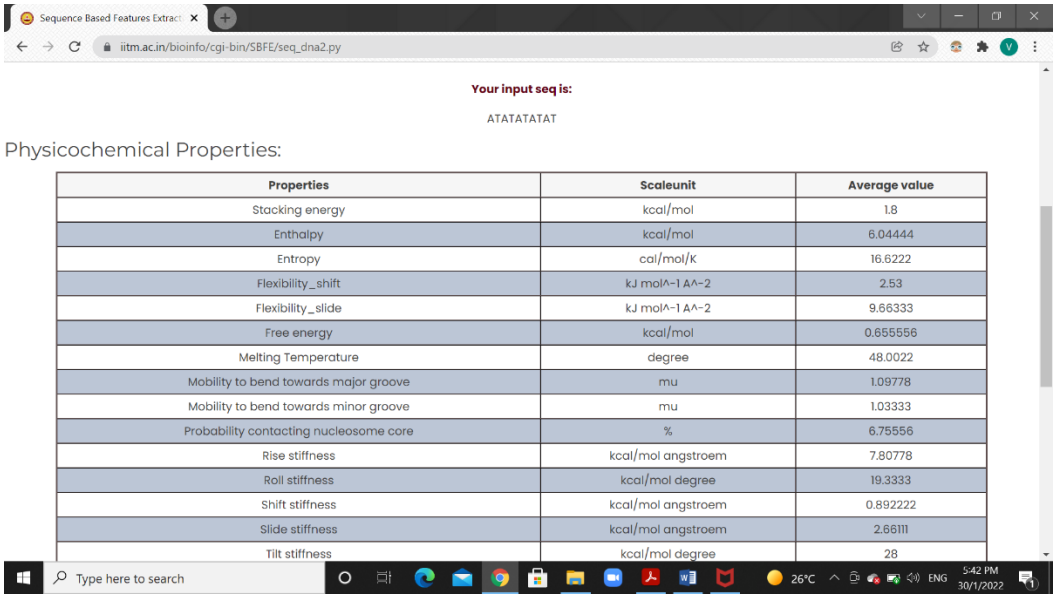
-5.804347826086956

9) Using the seq2feature app, I calculated the melting point for given sequences:

(i) ATATATATAT: 48.0022 degrees

(ii) GCGCGCGC: 107.867 degrees

As we know that, in a DNA double strand G and C forms three hydrogen bonds between them, whereas, A and T forms only two hydrogen bonds. So, since G and C sequence has many bonds, their melting temperature is higher than A and T sequence.



Sequence Based Features Extract

iitm.ac.in/bioinfo/cgi-bin/SBFE/seq_dna2.py

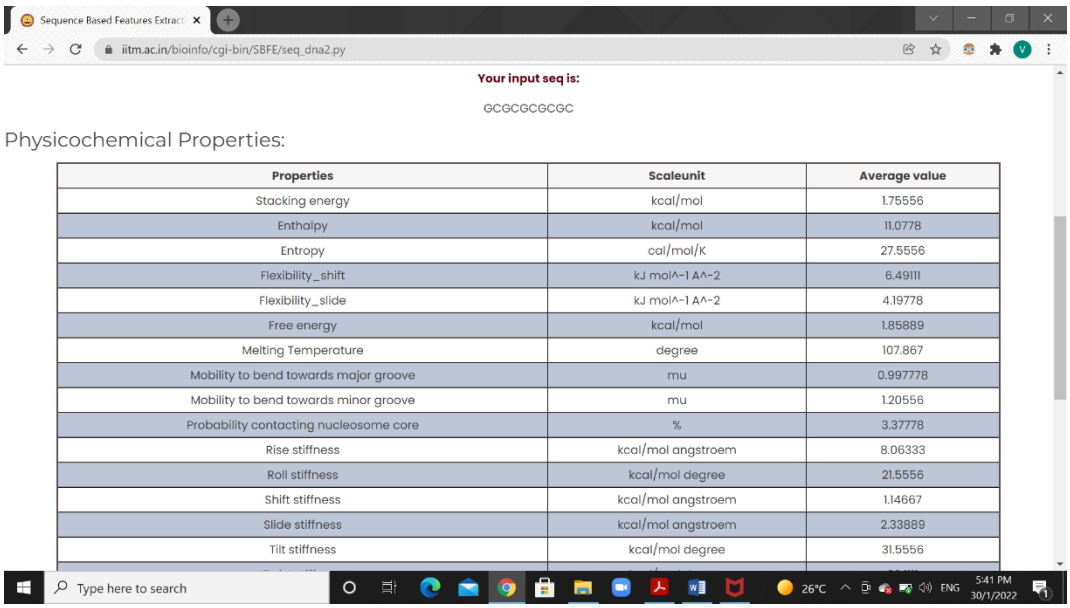
Your input seq is:

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

Type here to search 26°C ENG 5:42 PM 30/1/2022



Sequence Based Features Extract

iitm.ac.in/bioinfo/cgi-bin/SBFE/seq_dna2.py

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

Type here to search 26°C ENG 5:41 PM 30/1/2022

10) Using seq2feature tool I found the AT and GC content in the given sequence:

AAATGGCCCTAA:

- *AT content – 58.333333*
- *GC content – 41.666667*