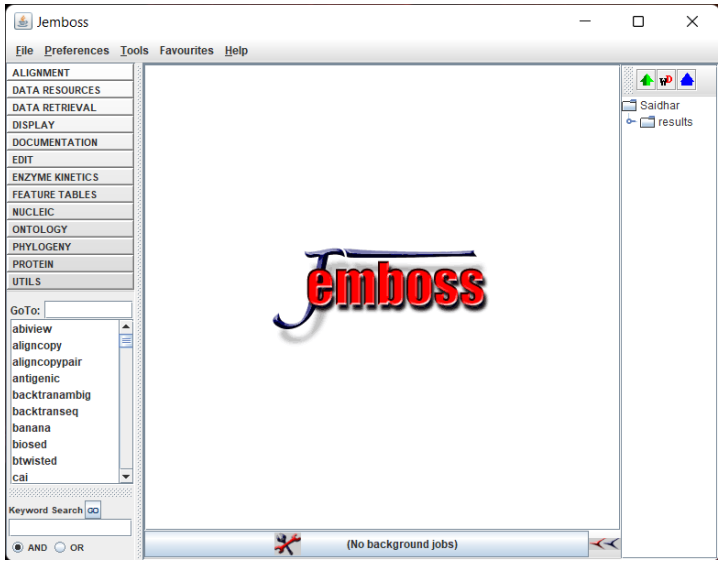
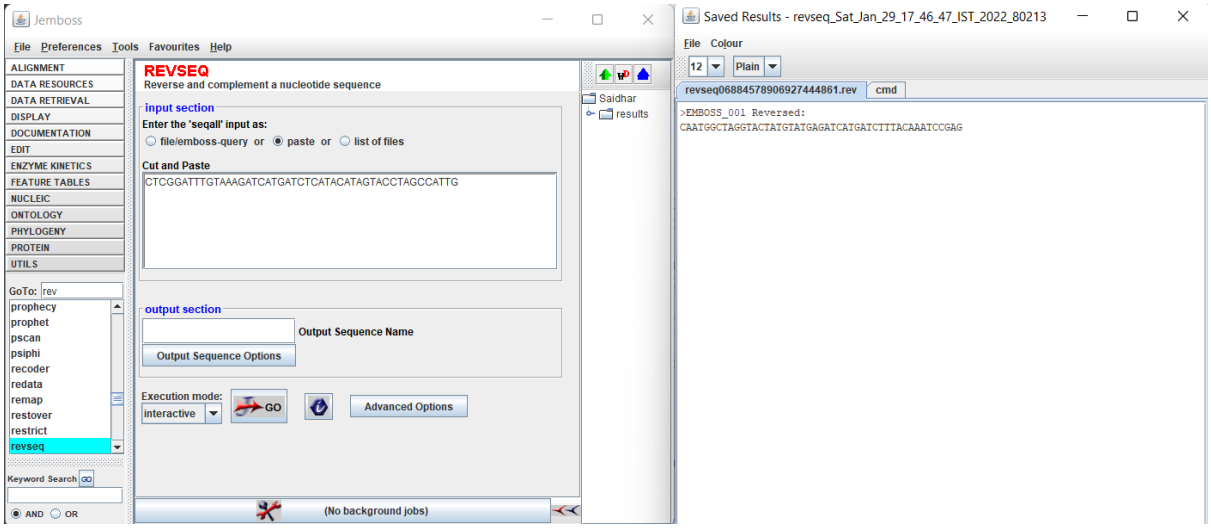
Assignment 1 BT3040

- Note: Though the codes are typed in this document, separate python files of code are submitted in the submission.

1. Steps taken to download the EMBOSS package:
   - Visit the given website and scroll down to “Platform-Dependent Notes” section where we can find the ftp address to the package.
   - Go to “My Computer/My PC” and right click and select “Add a Network Location” to add the FTP address in it.
   - Proceed with the given steps to get the FTP address and download the package after the steps are completed.
   - Install the EMBOSS package along with Java 32-bit version to get it running.



2. To find complementary sequence using EMBOSS:
   - Open EMBOSS and search for the REVSEQ function
   - Check the paste radio button and paste the given sequence.
   - Click Go to get the results.

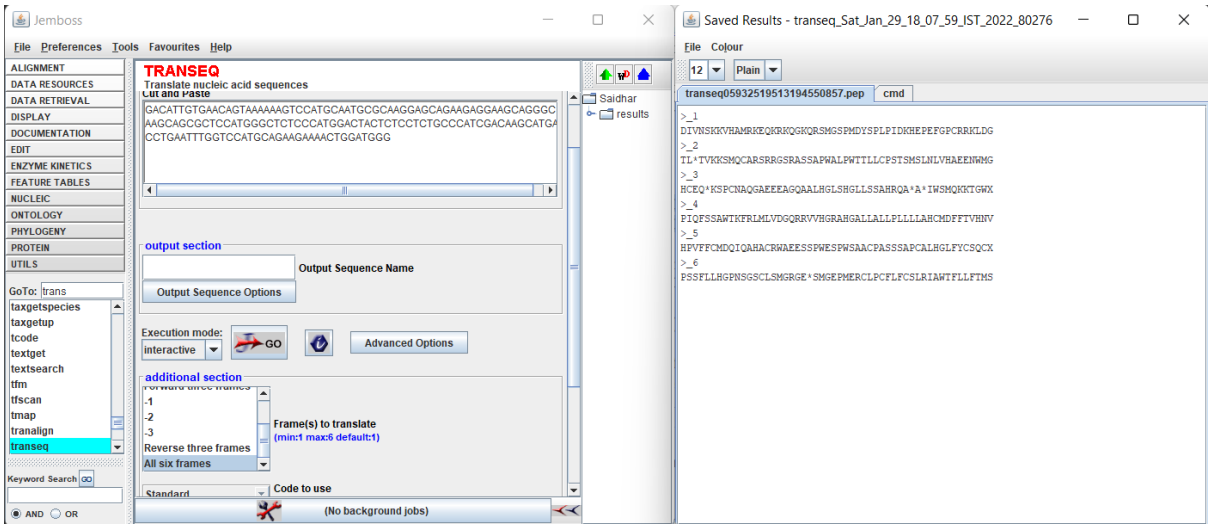


3. Code for Finding complementary sequence for the given DNA sequence:

```
string="CTCGGATTTGTAAAGATCATGATCTCATACATAGTACCTAGCCATTG"
string=string.replace("C", "P")
string=string.replace("G", "C")
string=string.replace("A", "Q")
string=string.replace("T", "A")
string=string.replace("P", "G")
string=string.replace("Q", "T")
```

```
string=string[::-1]
print(string)
```

4. Using the TRANSEQ function, we can find protein sequence for the given DNA sequence. (Make sure to select “All six reading Frames”)



(ii) From the above figure, we can say that the frame equivalent to the given sequence is the 4th protein sequence. That is the first reverse reading frame.

5. Program to translate DNA sequence to Protein sequence:

```
def match(a, b, c):
    if (a=='A'):
        if (b=='A'):
            if (c=='T' or c=='C'):
                return 'N'
            elif (c=='A' or c=='G'):
                return 'K'
        elif (b=='C'):
            return 'T'
        elif (b=='G'):
            if (c=='T'or c=='C'):
                return 'S'
            elif (c=='A'or c=='G'):
                return 'R'
        elif (b=='T'):
            if (c=='T'or c=='C'or c=='A'):
                return 'I'
            elif (c=='G'):
                return 'M'
    elif (a=='C'):
        if (b=='A'):
            if (c=='T'or c=='C'):
                return 'H'
            elif (c=='A'or c=='G'):
                return 'Q'
        elif (b=='C'):
            return 'P'
```

```
        elif (b=='G'):
            return 'R'
        elif (b=='T'):
            return 'L'
    elif (a=='G'):
        if (b=='A'):
            if (c=='T'or c=='C'):
                return 'D'
            elif (c=='A'or c=='G'):
                return 'E'
        elif (b=='C'):
            return 'A'
        elif (b=='G'):
            return 'G'
        elif (b=='T'):
            return 'V'
    elif (a=='T'):
        if (b=='A'):
            if (c=='T'or c=='C'):
                return 'Y'
        elif (b=='C'):
            return 'S'
        elif (b=='G'):
            if (c=='C'or c=='T'):
                return 'C'
            elif (c=='G'):
                return 'W'
        elif (b=='T'):
            if (c=='C'or c=='T'):
                return 'F'
            elif (c=='A'or c=='G'):
                return 'L'
string="GACATTGTGAACAGTAAAAAAGTCCATGCAATGCGCAAGGAGCAGAAGAGGAAG
CAGGGCAAGCAGCGCTCCATGGGCTCTCCCATGGACTACTCTCCTCTGCCCATCGACAAGCA
TGAGCCTGAATTTGGTCCATGCAGAAGAAAACTGGATGGG"
compstr=string[::-1]
compstr=compstr.replace("A", "P")
compstr=compstr.replace("C", "Q")
compstr=compstr.replace("G", "C")
compstr=compstr.replace("T", "A")
compstr=compstr.replace("P", "T")
compstr=compstr.replace("Q", "G")
s=[]
string=list(string)
compstr=list(compstr)
strstr=string.copy()
revstr=compstr.copy()
for j in range(0,6):
    print("Reading Frame no.: ", j+1)
    if (j<3):
        s=[]
```

```
        for i in range(int(len(string)/3)):
            s.append(match(strstr[3*i], strstr[3*i+1],
strstr[3*i+2]))
        s=['*' if i is None else i for i in s]
        if (j==2):
            s.pop(len(s)-1)
            s.append('X')
        strstr.pop(0)
        strstr.append(string[j])
        for i in range(len(s)):
            print(s[i], end='')
        print("")
    else:
        s=[]
        for i in range(int(len(string)/3)):
            s.append(match(revstr[3*i], revstr[3*i+1],
revstr[3*i+2]))
        s=['*' if i is None else i for i in s]
        if (j==5):
            s.pop(len(s)-1)
            s.append('X')
        revstr.pop(0)
        revstr.append(compstr[len(compstr)-1-j+3])
        for i in range(len(s)):
            print(s[i], end='')
        print("")
```

6. Program to find the positions and total number of matches for a user defined string in a given DNA sequence:

```
def find_motif(ms, ss):
    arr=[]
    for i in range(len(ms)-len(ss)):
        if (ms[i:i+len(ss)]==ss):
            arr.append(i+1)
    return arr
mainstring="GACATTGTGAACAGTAAAAAAGTCCATGCAATGCGCAAGGAGCAGAAGAG
GAAGCAGGGCAAGCAGCGCTCCATGGGCTCTCCCATGGACTACTCTCCTCTGCCCATCGACA
AGCATGAGCCTGAATTTGGTCCATGCAGAAGAAAACTGGATGGG"
substring=input()
arr=find_motif(mainstring, substring)
print("total number of matches: ", len(arr))
print("Position of matches:")
for i in range(len(arr)):
    print(arr[i], end=' ')
```

7. I've explored EMBOSS software and discovered some cool features. Some of them are:

Description	Program Name
Calculating GC Content	GEECEE
Protein to Nucleotide Seq.	BACKTRANSEQ

Plot Bending & Curvature of B-DNA Twisting of B-DNA	BANANA BTWISTED

8. Program to Calculate Average Base Stacking Energy for the given sequence:

```
def database(string):
    if (string[0]=="A"):
        if (string[1]=="A"):
            return -4
        elif (string[1]=="T"):
            return -7
        elif (string[1]=="C"):
            return -5
        elif (string[1]=="G"):
            return -11
    elif (string[0]=="T"):
        if (string[1]=="A"):
            return -7
        elif (string[1]=="T"):
            return -2
        elif (string[1]=="C"):
            return -3
        elif (string[1]=="G"):
            return -4
    elif (string[0]=="C"):
        if (string[1]=="A"):
            return -9
        elif (string[1]=="T"):
            return -5
        elif (string[1]=="C"):
            return -6
        elif (string[1]=="G"):
            return -7
    elif (string[0]=="G"):
        if (string[1]=="A"):
            return -9
        elif (string[1]=="T"):
            return -6
        elif (string[1]=="C"):
            return -4
        elif (string[1]=="G"):
            return 11
def avg_base_stack_egy(string):
    arr=[]
    summ=0
    for i in range(len(string)-1):
        arr.append(string[i:i+2])
        summ+=database(arr[i])
    avg=summ/(len(arr))
    return avg

string="CTCGGATTTGTAAAGATCATGATCTCATACATAGTACCTAGCCATTG"
```

```
print(avg_base_stack_egy(string))
```

9. Using the Seq2Feature tool, to compute the average melting temperature,
    1. Visit the site, click compute, click Average Property and enter the sequence in FASTA format. Then check the Physiochemical Properties check box and submit it. The results are as follows:

        (i) `Melting Temperature: 48.0022`

        (ii) `Melting Temperature: 107.867`

        And the reason why the Melting temperature of the 2nd sequence is higher than the first is the fact that the bond strength between the main and complementary strand. For the first sequence, i.e., for ATATATATAT, the complementary strand would be TATATATATA and the bonds between A and T are only two hydrogen bonds when compared to three hydrogen bonds of G and C which were present in 2nd sequence.

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 AA^-2	2.53
Flexibility_slide	kJ mol^-1 AA^-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

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 AA^-2	6.49111
Flexibility_slide	kJ mol^-1 AA^-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

10. By checking the Nucleotide Content Checkbox in the Seq2Feature tool, we can find the AT and GC content of the given sequence.

    `AT_Content: 58.333333`

    `GC_Content: 41.666667`

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