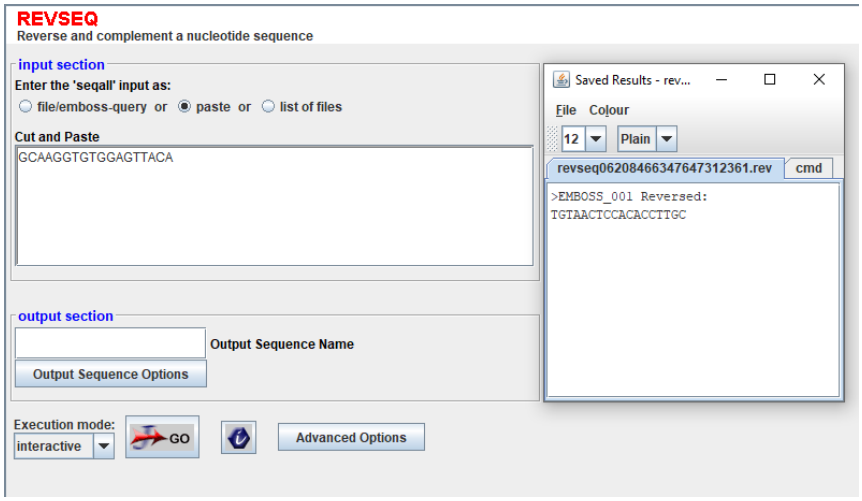
BT3040 – BIOINFORMATICS – Assignment 1

Submitted by Sahana (BE17B038)

Question 3



Question 4

Python code –

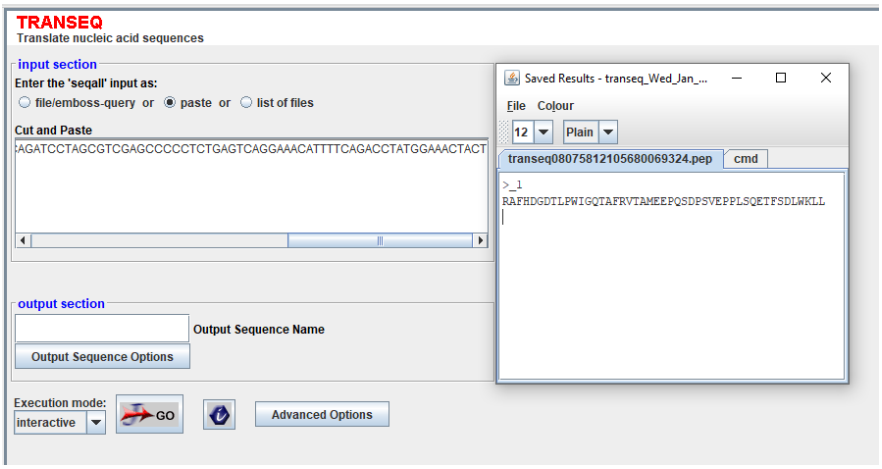
```
def compliment(string):
        COMPLEMENT = {'A':'T', 'T':'A', 'C':'G', 'G':'C'}
        # each character from strin (dna) is complemented. the entire
string is reversed and jooined without leaving any space.
        rstring=''.join(reversed([COMPLEMENT[i] for i in string]))
        return(rstring)
```

Implementing the code -

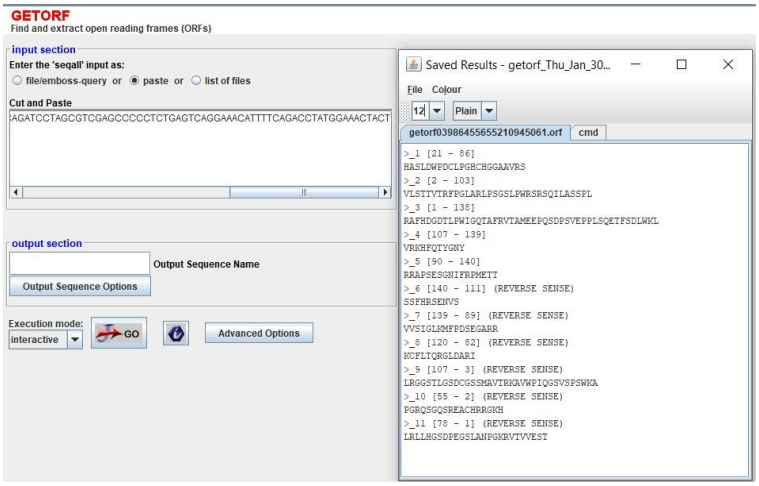
```
DNA = 'GCAAGGTGTGGAGTTACA'
compliment(DNA)
```

Question 5

(i)



(ii) 2nd ORF is where we can find the mentioned protein sequence.



Question 6 –

Python code –

```
def translation(string):

    l = len(string)

    rnaStrand = string.replace('T','U')

    protein=''

    Codon = {'UUU' : 'F', 'UUC' : 'F', 'UUA' : 'L', 'UUG' : 'L', 'CUU' :
'L', 'CUC' : 'L', 'CUA' : 'L', 'CUG' : 'L', 'AUU' : 'I', 'AUC' : 'I', 'AUA'
: 'I', 'AUG' : 'M','GUU' : 'V', 'GUC' : 'V', 'GUA' : 'V', 'GUG' : 'V','UCU'
: 'S', 'UCC' : 'S', 'UCA' : 'S', 'UCG' : 'S','AGU' : 'S', 'AGC' : 'S',
'CCA' : 'P', 'CCG' : 'P','CCU' : 'P', 'CCC' : 'P', 'ACA' : 'T', 'ACG' :
'T','ACU' : 'T', 'ACC' : 'T', 'GCA' : 'A', 'GCG' : 'A','GCU' : 'A', 'GCC' :
'A', 'UAU' : 'Y', 'UAC' : 'Y','CAU' : 'H', 'CAC' : 'H', 'CAA' : 'Q', 'CAG'
: 'Q','AAU' : 'N', 'AAC' : 'N', 'AAA' : 'K', 'AAG' : 'K','GAU' : 'D', 'GAC'
: 'D', 'GAA' : 'E', 'GAG' : 'E','UGU' : 'C', 'UGC' : 'C', 'UGG' : 'W',
'AGA' : 'R','CGU' : 'R', 'CGC' : 'R', 'CGA' : 'R', 'CGG' : 'R','AGG' : 'R',
'GGC' : 'G', 'GGA' : 'G', 'GGG' : 'G','GGU' : 'G', 'UAA' : '' , 'UAG' : '',
'UGA' : ''}

    for i in range(0,l-1,3):

        # rnaStrand is translated

        small=rnaStrand[i:i+3]

        protein+=Codon[small]

    return protein
```

Implementing the code-

```
DNA =
'CGTGCTTTCCACGACGGTGACACGCTTCCCTGGATTGGCCAGACTGCCTTCCGGGTCACTGCCATGGAGGAGCC
GCAGTCAGATCCTAGCGTCGAGCCCCCTCTGAGTCAGGAAACATTTTCAGACCTATGGAAACTACTA'

translation(DNA)
```

Question 7 –

Python code –

```
def string_search(string,DNA):
    m=len(string)
    n=len(DNA)
    ctr=0
    position=[]
    for i in range(n):
        if DNA[i:i+m]==string:
            ctr+=1
            position.append(i)
    print('Position of occurance = ')
    print(position)
    print('Total number of occurance = %d'%ctr)
```

Implementing the code-

```
DNA='GCATTAAGGTGTGGAGTTACACAGTTACCAGGTTA'
string='TTA'
string_search(string,DNA)
```

Output –

```
def string_search(string,DNA):
    m=len(string)
    n=len(DNA)
    ctr=0
    position=[]
    for i in range(n):
        if DNA[i:i+m]==string:
            ctr+=1
            position.append(i)
    print('Position of occurance = ')
    print(position)
    print('Total number of occurance = %d'%ctr)
DNA='GCATTAAGGTGTGGAGTTACACAGTTACCAGGTTA'
string='TTA'
string_search(string,DNA)
```

```
Position of occurance =
[3, 16, 24, 32]
Total number of occurance = 4
```

Question 8-

Function DAN calculates Nucleic acids' melting temperature (Tm) and reports answer in a graph (if specified).

Function BANANA plots bending and curvature data of normal (B) DNA double helix. The program calculates the magnitude of local bending and macroscopic curvature at each point along an arbitrary B-DNA sequence, using any desired bending model that specifies values of twist, roll and tilt as a function of sequence. The program outputs both a graphical display and a text file of the results.

Question 9 –

Python code –

```
def energy(string):
    BSE={'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}
    energy=0
    n = len(string)
    for i in range(n-1):
        base=string[i:i+2]
        energy+=BSE[base]
    avg_energy=energy/(n-1)
    print('Average Base Stacking Energy = %f'%avg_energy)

DNA = 'GCAAGGTGTGGAGTTACA'
energy(DNA)
```

Manual calculation (performed for sequence in question 3)–

	A	T	G	C
A	1	0	2	1
T	1	1	2	0
G	1	3	2	1
C	2	0	0	0

Sum of all the base-stacking-energies are = 1(-4) + 2(-11) + 1(-5) + 1(-7) + 1(-2) + 2(-4) + 1(-9) + 3(-6) + 2(+11) + 1(-4) + 2(-9) = -75

Average base stacking energy = -75/17 = -4.411765 units

Question 10 –

- **ls** – The ls command is a command-line utility for listing the contents of a directory or directories given to it via standard input.
- **cd** – Command is used to change the directory.
- **mkdir** – Allows users to create or make new directories. mkdir stands for "make directory."
- **cp** – Is a command-line utility for copying files and directories. It supports moving one or more files or folders with options for taking backups and preserving attributes.
- **mv** – Is a command line utility that moves files or directories from one place to another. It supports moving single files, multiple files and directories.
- **wc** – Is a command line utility for printing newline, word and byte counts for files. It can return the number of lines in a file, the number of characters in a file and the number of words in a file.

- **cat** – Short for “concatenate”. Allows us to create single or multiple files, view contain of file, concatenate files and redirect output in terminal or files
- **tail** – Is a command-line utility for outputting the last part of files given to it via standard input. It writes results to standard output. By default, tail returns the last ten lines of each file that it is given.
- **more** – Lets you view text files or other output in a scrollable manner. It displays the text one screenful at a time, and lets you scroll backwards and forwards through the text, and even lets you search the text.
- **ssh** – Provides a secure encrypted connection between two hosts over an insecure network. This connection can also be used for terminal access, file transfers, and for tunnelling other applications.
- **scp** –Stands for “secure copy”. Is used to copy file(s) between servers in a secure way.
    - scp –P port: Specifies the port to connect on the remote host.
    - scp –p: Preserves modification times, access times, and modes from the original file.
    - scp –q: Disables the progress meter.
    - scp –r: Recursively copy entire directories.
    - scp –S program: Name of program to use for the encrypted connection. The program must understand ssh options.
    - scp –v: Verbose mode. Causes *scp* and *ssh* to print debugging messages about their progress. This is helpful in debugging connection, authentication, and configuration problems.