BIOI 3500, *Advanced Bioinformatics Programming*
Assignment 2

Assignment 2 :: 50 points

Objective

This assignment will give you the opportunity to continue learning Python by solving straightforward bioinformatics problems.

Program A

Write a (procedural) Python (version 3) program that generates random coding DNA; i.e., DNA that starts with the start codon, ends with a stop codon, and does not contain any internal stop codons. The total length of the DNA (in codons) and the specific stop codon to use should be specified using command-line arguments, option letters **`c`** and **`s`**, respectively. The DNA should be displayed in all uppercase unless the **`l`** ("el") argument is present in which case it should be displayed in all lowercase. Use the Python **`getopt()`** module to process command-line arguments. Information about random numbers in Python can be found here: https://docs.python.org/3/library/random.html.

Note that none of the three command-line arguments (option letters **`c`**, **`s`**, and **`l`**) is required: if **`c`** is not present, a DNA sequence of random length between 25 and 50 codons (both inclusive) should be generated; if **`s`** is not present, a randomly-selected stop codon should be used; and **`l`** may or may not be present as described above. If a bad stop codon is given, a randomly-selected one should be used.

Name your program **`netID_Assign2A.py`** where **`netID`** is your UNO NetID.

Here are several examples.

```
$ python3 ./CodingSeq.py
ATGCCGAGGCCAATGCAGAGATCATGTCCATATTTGCGCGTTACACAACGCACACGGCGTGTTCCGCAGCTCACATAA

$ python3 ./CodingSeq.py -c 20
ATGGTTTCTTCCTCCTATCATTCACCCAATCATCAGACGGAATATCCAACCATCGGCTGA

$ python3 ./CodingSeq.py -c 20 -l
atgacatgtgaatctacaatggccagaggggactcgcccgtcgactttctatgtacttag

$ python3 ./CodingSeq.py -c 15 -s TAG
ATGTCTATCCTACGACAGACGGCTTGTCATTGTGAGTTGATCTAG

$ python3 ./CodingSeq.py -c 25 -s AAG
ATGGATCTTGCCGGTTTTTCATCGTTGCGCCCAAAATTAACCACCATGGGTGGCCATGAGACTACGGCATATTAG
```

Your program must be written in a consistent style and be appropriately documented. (See the *Python Coding Style Guide.*)

Program B

The genetic code (see the table on the next page) provides the translation between the nucleotides of mRNA and the amino acids of proteins. Recall that the genetic code is non-overlapping and that a given strand of DNA can be read in any one of six reading frames—three on the direct strand and three on the complementary strand. For example, the translation of fragment

`GACCGGCTCGAGTGCTACGCGCCACCCTCTCTACTACGACTAATT`

in all six reading frames is given below. (**`*`** is used to represent stop codons.)

+1: **`DRLECYAPPSLLRLI`**
+2: **`TGSSATRHPLYYD*`**
+3: **`PARVLRATLSTTTN`**
-1: **`N*S**RGWRVALEPV`**
-2: **`ISRSREGGA*HSSR`**
-3: **`LVVVERVARSTRAG`**

Write a (procedural) Python (version 3) program that displays the amino acid sequence (one-letter codes) for a fragment of DNA entered as a command-line argument (option letter **`s`**). The reading frame should be specified using a command-line argument (option letter **`f`**). Treat out-of-bounds reading frames as frame 1 on the given strand; e.g., frame 5 would be treated as frame +1; frame -4 would be treated as frame -1. Interpret reading frame 0 as reading frame +1. Use the Python **`getopt()`** module to process command-line arguments. Use an asterisk (**`*`**) to represent stop codons. Codons with one or more non-base characters (i.e., characters other than **`G`**, **`C`**, **`A`** and **`T`**) should be translated as **`X`**.

Name your program **`netID_Assign2B.py`** where **`netID`** is your UNO NetID.

First Base	Second Base								Third Base
	T		C		A		G		
T	TTT	Phe (F)	TCT	Ser (S)	TAT	Tyr (Y)	TGT	Cys (C)	T
	TTC	Phe (F)	TCC	Ser (S)	TAC	Tyr (Y)	TGC	Cys (C)	C
	TTA	Leu (L)	TCA	Ser (S)	TAA	STOP	TGA	STOP	A
	TTG	Leu (L)	TCG	Ser (S)	TAG	STOP	TGG	Trp (W)	G
C	CTT	Leu (L)	CCT	Pro (P)	CAT	His (H)	CGT	Arg (R)	T
	CTC	Leu (L)	CCC	Pro (P)	CAC	His (H)	CGC	Arg (R)	C
	CTA	Leu (L)	CCA	Pro (P)	CAA	Gln (Q)	CGA	Arg (R)	A
	CTG	Leu (L)	CCG	Pro (P)	CAG	Gln (Q)	CGG	Arg (R)	G
A	ATT	Ile (I)	ACT	Thr (T)	AAT	Asn (N)	AGT	Ser (S)	T
	ATC	Ile (I)	ACC	Thr (T)	AAC	Asn (N)	AGC	Ser (S)	C
	ATA	Ile (I)	ACA	Thr (T)	AAA	Lys (K)	AGA	Arg (R)	A
	ATG	Met (M) (START)	ACG	Thr (T)	AAG	Lys (K)	AGG	Arg (R)	G
G	GTT	Val (V)	GCT	Ala (A)	GAT	Asp (D)	GGT	Gly (G)	T
	GTC	Val (V)	GCC	Ala (A)	GAC	Asp (D)	GGC	Gly (G)	C
	GTA	Val (V)	GCA	Ala (A)	GAA	Glu (E)	GGA	Gly (G)	A
	GTG	Val (V)	GCG	Ala (A)	GAG	Glu (E)	GGG	Gly (G)	G

The output of your program should mimic—i.e., for the given input look exactly the same as—the following.

```
$ python3 Translate.py -f 2 -s CCATGGAGACGCAGGGGGTGATCCTCCGAGGGGC
HGDAGG*SSEG
```

Note the command is entered on a single line. Here's another sample run to illustrate:

```
$ python3 Translate.py -f -3 -s CCATGGAGACGCAGGGGGbTGATCCTCCqAGGGGC
PXEDXPPASPW
```

Your program must be written in a consistent style and be appropriately documented. (See the *Python Coding Style Guide*.)

Please upload both programs to Canvas in a single archive file (using tar or zip).