Assignment 3

Due Wednesday, September 9 before midnight (California time)

1. Write a Python function that takes as input a DNA sequence string and returns its reverse complement string. For example, given an input string “CATGCCGGAATT”, the function returns the reverse complement string “AATTCCGGCATG” [2pt].

2. Write a Python function (or functions) to find all ORFs from six different reading frames (+1, +2, +3, -1, -2, -3) in a sequence string (you may need function #1 to get reverse complement string). Since we are only interested in relatively long ORFs, the function should only return those ORFs longer than an input threshold value. Compute the length of the ORFs as the number of codons (counting the start and stop codon). The start codon is “ATG” and the stop codons are “TAG”, “TAA”, and “TGA” [2pt].

   In the example below, there are two overlapping ORFs (the first three lines are the same sequence and the last three lines are the same reverse complement sequence, with the commas in different places to make it easier to read). The length of the ORF on the first line is seven. The length of the ORF on the second line is six. Each ORF starts with an “ATG”, ends up with a stop codon, and there is no in-frame stop codon in the sequence.

   For an input sequence: AATGCCCAAATGCTTTTAAAACCCATGTAGC

   Frame 1: AAT,GCC,CAA,ATG,CTT,TTA,AAA,CCC,ATG,TAG,C
   Frame 2: A,ATG,CCC,AAA,TGC,TTT,TAA,AAC,CCA,TGT,AGC
   Frame 3: AA,TGC,CCA,AAT,GCT,TTT,AAA,ACC,CAT,GTA,GC
   Frame -1: GCT,ACA,TGG,GTT,TTA,AAA,GCA,TTT,GGG,CAT,T
   Frame -2: G,CTA,CAT,GGG,TTT,TAA,AAG,CAT,TTG,GGC,ATT
   Frame -3: GC,TAC,ATG,GGT,TTT,AAA,AGC,ATT,TGG,GCA,TT

   The inputs to the function should be the sequence string and the threshold value. The output should be a list of lists. The length of the list should be the number of ORFs. Each element of the list should be another list with three numbers describing an individual ORF: the frame number, the nucleotide position of the first position of the start codon, and the nucleotide position of the first position of the in-frame stop codon. For the example above, with threshold 5, the output should be: [[1,9,27],[2,1,16]].

3. Download the FASTA file “ACE2.fasta” from Blackboard. Use function #2, with threshold 700, to find all ORFs. [2pt].

4. Write a Python function that takes two inputs: a float *gc* and an integer *l*. This function returns the expected value E[X] and the probability P(X>L), where X is a geometric random variable representing the number of codons until getting a stop codon [2pt].

   For example, with the inputs (*gc*=0.4, *l*=50):

   First, the function calculates *p*, the probability of getting a stop codon:

   $$p = P(TAG\ or\ TAA\ or\ TGA) = (0.3 \times 0.3 \times 0.2) + (0.3 \times 0.3 \times 0.3) + (0.3 \times 0.2 \times 0.3) = 0.063$$

   Second, the function calculates $E[X] = 1/p = 15.873$

   Third, the function calculates $P(X > L) = 1 - P(X \leq 50) = 1 - 0.961 = 0.039$

   Finally, the function returns 15.873 and 0.039

5. If you try to optimize your Python code, describe how you improve the performance [Bonus 1pt].

Turn in the code for the Python functions and the answers into one file in Jupyter Notebook format (.ipynb). Use the “Turnitin” link on Blackboard/Assignments/Assignment 3 to submit this file.