General Assignment Information

- Submit an electronic copy of your assignment via LEARN. If you run out of your allotted late submission time during the term (72 hours), your submission will incur a 5% penalty for every rounded-up hour past the deadline.
  For example, an assignment submitted 5 hours and 15 minutes late will receive a penalty of ceiling(5.25)*5% = 30%.
- Submissions are not accepted after 72 hours past the due date.
- For each question, submit a `.pdf` report describing your solution and the files necessary to run your program, including instructions to produce the desired output. Additionally, each question may ask you to submit extra information
- You will lose marks if your code is unreadable, sloppy, or inefficient.
- Submit all your files in a single compressed file (.zip, .tar etc.)
- You **must** include the specific Python version used to compile their program (e.g. the output of `python --version` command). **You are allowed to use external Python libraries in your solutions**. However, you should assume that the system used for testing will not run any package installation software other that `pip`.
- The filename should include your username and/or student ID.

Problem Statements

[10] 1. **k-mer Composition**. Given a value of $k \in \mathbb{Z}^+$, and DNA string v, compute v's k-mer frequency array F_s, where $F_s[m]$ denotes the number of times the m-th k-mer (in lexicographic order) appears in v. **Note:** Some bases in v may not belong to the nucleotide alphabet {A, C, G, T}. Justify your approach to handling such cases in your algorithm.

   Input: A FASTA file containing a DNA sequence s (length at most 100 kbp) and an integer $1 \leq k \leq 7$.

   Output: A text file `k-mers.txt` with the components of the k-mer frequency array A(s) in lexicographic order and separated by a space.

[20] 2. **de Bruijn Graph Construction**. In genome assembly, overlapping k-mers provide useful structural information. Given a set S of $(k+1)$-mers from an unknown DNA sequence, let S_{rc} denote the set of reverse complements of elements in S. Define the de Bruijn graph B_k of order k as:

- Nodes represent all k-mers appearing as substrings of $(k+1)$-mers in $S \cup S_{rc}$.
- Directed edges correspond to $(k+1)$-mers in $S \cup S_{rc}$, forming edges $(r_{[1:k]}, r_{[2:k+1]})$.

Input: A `reads.txt` file with up to 1000 (possibly repeating) DNA strings of equal length (not exceeding 50 bp), representing $(k+1)$-mers.

```
TGAT
CATG
TCAT
ATGC
CATC
CATC
```

Output: A `deBruijn.txt` file with the adjacency list of the de Bruijn graph for $S \cup S_{rc}$.

```
(ATC, TCA)
(ATG, TGA)
(ATG, TGC)
(CAT, ATC)
(CAT, ATG)
(GAT, ATG)
(GCA, CAT)
(TCA, CAT)
(TGA, GAT)
```

[35] 3. **Alignment-free analysis of viral phylogenies.** Given the text file `Dengue.txt` listing NCBI accession IDs write a Python program to:

(a) Download sequences from NCBI.
(b) Process sequences (convert to uppercase, and handle ambiguous base pairs).
(c) Compute normalized k-mer frequency vectors (L1 norm).
(d) Compute a valid distance matrix using the following measures of dissimilarity:
- Euclidean distance
- Cosine similarity
- Pearson correlation

(e) Construct a UPGMA tree for each measure.

Consider the candidate viral sequence in `candidate.fas`, add it to the set of total sequences and rerun the analysis.

What to submit: Three Newick-formatted trees: `Euclidean.nwk`, `Cosine.nwk`, `Pearson.nwk` including all the sequences and a report with your approach to the solution and the answers to the following questions:

- How well are subtypes clustered in the initial tree?
- How does the test sequence affect the tree?
- What is the predicted subtype of the test sequence?
- Which metric is preferable for alignment-free analysis?

[35] 4. **Genome assembly of simulated genomes** A critically ill patient arrives at the hospital with a rapidly deteriorating condition. Standard diagnostic tests fail to identify the pathogen. In a race against time, doctors turn to the computational biology team for a solution. Due to urgency, an old Illumina sequencing machine is used to generate short reads stored in a FASTA file. Implement a genome assembler to reconstruct the viral genome s from the given set of reads.
Input: The set of reads present in the `reads.fna` file.
Output: A FASTA file containing the assembled sequence.

This is an open-ended problem without a unique optimal solution. Although the quality of your assembled sequence will be evaluated using BLAST's percentage identity against the true genome, most of your grade will depend on the clarity and completeness of your explanations.

Bonus Question (5 points): Based on your assembled genome, hypothesize which virus caused the infection. Provide supporting evidence using computational analyses.