General Assignment Information

- Submit each part of the assignment accordingly. 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 $\lceil 5.25 \rceil \times 5\% = 30\%$.
- Submissions are not accepted after 72 hours past the due date.
- For each Question 1 and 2, submit your code to Marmoset; for Question 3 submit a `.pdf` report describing your solution and the files necessary to run your Jupyter notebook.
- The following python libraries are allowed: `numpy`, `scipy`, `torch`, `tensorflow`, `keras`, `scikit-learn`, `pandas`, `matplotlib`. If there is a library that you really need to use and is not listed here, please discuss it with the TA.
- The filename corresponding to Q3 must include your username and/or student ID.

Problem Statements

[30] 1. **Multiple Pattern Matching Problem**.

During lectures, we have studied various algorithms for string matching and discussed their applications in genome assembly and variant calling. In this problem, you are asked to implement an efficient algorithm to locate all occurrences of all the reads from a collection within a given reference genome.

Input: A `.txt` file containing:

- A reference genome S (a string).
- A collection R of read sequences.

```
<reference genome string S>
<read pattern 1>
<read pattern 2>
...
<read pattern n>
```

Output:
A text file `indices.txt` containing the starting indices (0-indexed) of every occurrence of the reads from R in S, separated into a new line and in any order.

Constraints:

- $10 \leq \text{len}(S) \leq 10000$
- $|R| \leq 16000$

Program Specifications:

- Your solution **must be implemented in Python** and saved in a file named `Q1_MultiplePatternMatching.py`. Your program will be executed as follows:

```
python Q1_MultiplePatternMatching.py input_file.txt
```

[30] 2. **Natural Peptide Classifier**.

Definition: A peptide is a short chain of amino acids.

Write a program that computes the probability that a given peptide sequence is natural. Here, a peptide is considered "natural" if it appears as a contiguous substring in a protein produced by a real organism.

Input:

A single FASTA file containing a list of peptide sequences.

```
>seq_1
LLLSLYYPNDRKLLDYKE
>seq_2
VSRVSSDADPAGGWCRKWYSAHRGPDQDAALG
```

Output:
A CSV file where each line corresponds to a peptide from the input file (in the same order). Each line should contain two fields separated by a comma and a space:
(a) The sequence identifier.
(b) The probability (a value between 0 and 1) that the peptide is natural.

```
seq_1, 0.9
seq_2, 0.04
```

Program Specifications:

- The test dataset will contain a mixture of an equal number of natural and random peptide sequences. The natural peptides will be sampled from a real protein sequence database. We will use the uniprot/swissprot protein sequence database to sample natural peptides.
- Each peptide Has a length uniformly distributed between 20 and 40 (inclusive).
- Your solution **must be implemented in Python** and saved in a file named `Q2_PeptideClassifier.py`
- Your program will be executed as follows:

  ```
  python Q2_PeptideClassifier.py inputFile
  ```

- Your total submission size should not exceed 2M bytes, and submissions that are significantly larger than 2M bytes may lead to penalty or rejection of the submission.

Hints:

- If you are stuck, consider computing the probability based on an empirical k-mer frequency score. For each k-mer in a peptide, let p be its frequency in real protein sequences and q be its frequency in a shuffled version of the peptide. You might use log(p/q) as a score for each k-mer, summing these scores to obtain an overall score. Experiment with different values of k (e.g., k $\in \{2, 5\}$) and devise a method to convert the score into a probability.
- You are free to use the uniprot/swissprot protein sequence database to train a classifier. If you do that, note that there may be IUPAC symbols in the sequences that do not code one of the 20 common amino acids. **Our test cases will not include those peptides.** The random peptides will be obtained by a process similar to the natural peptides, except that after the sampling, the amino acids in the sequence of each peptide are shuffled randomly. Notice that the random and natural peptides in a test file are sampled independently of each other.

[40] 3. **Viral Genome Sequence Classification using Neural Networks**.

Download the files `DNA_Sequence_Classification_Complete.ipynb` and `HIV_data` from the Learn website (you may consult the additional tutorial files if necessary). In this assignment, you will:

- Follow the annotations in the provided Jupyter Notebook.
- Preprocess a dataset of viral genome sequences.
- Train two neural network classifiers: a multilayer perceptron (MLP) and a convolutional neural network (CNN).
- Evaluate and compare the performance of these models.

Task:
Your grade will be determined by the average test accuracy of the CNN and the MLP classifiers, normalized as per the class grading scheme.

Submission:
Submit your completed Jupyter Notebook along with all necessary files to run the notebook and a PDF report explaining the selection of your architectures and hyperparameters. Ensure that your filenames include your username and/or student ID.