LAB 1: Manipulate fasta file in Python ASSIGNMENTS

First, take a look at the **LAB1_Tips** file and practice with each section described. We strongly believe that they could be helpful with the following assignments.

Assignment 1: Random fasta file generator

Write a python program that generates a fasta file containing reads with the following characteristics:
-Read id is in the following format: >read_id_X where X is a progressive number starting from 0.
-Sequences have length 50 bp
-Bases are randomly generated using a A,T,C,G alphabet, but probability of each base for each read should be given from the command line as a set of numbers (probA, probT, probC, probG)
- The number of reads should be passed as an argument from the command line
-The name of the fasta file should be passed as an argument from the command line
Example:

```
python read_generator simulatedfasta.fa 100 30 30 30 10
```

Assignment 2: Statistics extraction

Write a python program for extracting statistics from fasta files. The program must take as first argument from the command line the name of the input fasta file to be analyzed and write to an output text file (whose name is passed as second argument from the command line) a summary of the computed statistics.

The following are the expected output statistics:

-Statistics of single bases across all the reads: Number of A,T,C,G

-Number of reads having at least one low complexity sequence: AAAAAA, TTTTTT, CCCCCC or GGGGGG.

- Number of reads having a number of GC couples (so called **GC content**) higher than a threshold GC_THRESHOLD passed as third argument from the command line

-For each read having a GC content higher than GC_THRESHOLD, report its read_id and the number of GC couples

Assignment 3: Fasta comparison

Write a python program to compare two fasta files. The two fasta files are passed as first and second argument from the command line.

The two fasta files have the following characteristics:

-The fasta format of the two files is correct (no need to check the format)

-Each read takes up a single line

-Each input file does not contain duplicated reads (i.e. identical reads)

The program must write as output a third fasta file containing only the reads that are in common between the input files. The read ids in the output file should be composed by the read id of the first file concatenated with the read id of the second file.

Assignment 4: Consensus Region

Write a python program that reconstructs the consensus regions on a specific chromosome starting from a tab-separated file called *alignments.txt* made up of three columns: the read ID, the sequence of the read and the alignment position of the read onto the reference genome. An example of *alignments.txt* is available in the following.

Exploiting the sequence and the alignment position of each read, build the consensus regions on the selected chromosome. Please note that all reads have the same length and that multiple consensus regions are allowed for the same chromosome.

***alignments.txt* example**

```
read_0    CAGCCATGACACTAAGCACG      15
read_1    TTTAAAAAATCCGTGGACAC      40
read_2    GCATTTAAAAAATCCTTGGA      37
read_3    ATTTCGGCGGCGACACCCCG      0
read_4    TTCGGCGGCGACACACCGAT      2
read_5    ATATTTGGACACAAATGCAT      48
```

Logic to build the Consensus regions

Reference genome:

```
ATTTCGGCGGCGACACAGGGATGACACAGGGCACGCAGCATTTAAAAAATTTTTGGACACAGCAGCAT
0    5    10   15   20   25   30   35   40   45   50   55   60   65
```

Reads:

```
ATTTCGGCGGCGACACCCCG                    TTTAAAAAATCCGTGGACAC
  TTCGGCGGCGACACACCGAT               GCATTTAAAAAATCCTTGGA
               CAGCCATGACACTAAGCACG             ATATTTGGACACAAATGCAT
```

Consensus regions:

```
ATTTCGGCGGCGACACACCGATGACACTAAGCACG
GCATTTAAAAAATCCTTGGACACAAATGCAT
```