Aim: In this assignment, you are asked to write a program that uses **Bloom Filter** to perform approximate membership test for DNA sequences. Your program will calculate the overlaps between two FASTA files. You will take a two separate FASTA files (called as *reference*), one to build the Bloom filter, and the other as the *query*. Both FASTA files will include **arbitrary** number of DNA sequences of **arbitrary** length. Your program will also take in the k value, and the bitvector size in **bytes**. You can choose and implement **at least three** any hash functions, but you may implement additional ones.
Your program will scan the *reference* FASTA file, generate all possible k-mers, and index them in the Bloom filter. Then, it will do generate all k-mers from the *query* FASTA file, and search whether they were previously seen. If a sequence in either *reference* or *query* FASTA files is shorter than k, omit that sequence. **Do not forget that DNA is double-stranded, and use only a canonical representation of equivalent k-mers.** Do **NOT** give the intermediate steps as output.

Command line examples: Be sure that your code works using the following command **(NOT PARSING the arguments will cost -15 points)**:

```
bloomFilter --ref reference.fasta --query query.fasta --kmer $kmerLength --bloomsize $size
```

Parameters:

- −−**ref** FASTA-formatted file that contains the sequences to index.
- −−**query** FASTA-formatted file that contains the sequences to search in *reference*.
- −−**kmer** k-mer length.
- −−**bloomsize** Size of the Bloom filter bit vector in **bytes**.

Output:
Print the following to standard output:

(1) Number of (not necessarily distinct) k-mers indexed in *reference*.
(2) Number of (not necessarily distinct) k-mers scanned in *query*.
(3) Number of (not necessarily distinct) k-mers from *query* found in the *reference*.

Notes:

- You must write your code yourself. Sufficient evidence of plagiarism will be treated the same as for plagiarism or cheating.
- Non-compiling submissions will not be evaluated.
- Your code will be compiled into a **single binary** using the Makefile. If scripting languages are used, a single wrapper script should be provided.
- Do not submit the program binary. You must submit the following items:
  - All **"source"** files.
  - A script to compile the source code and produce the binary (Makefile), **if required**.
  - A README.txt file that describes how the compilation progress works, **if required**.
- Create a directory, of which format is "surname_name_hw6", put all required files into that directory, and then zip it. You will have a give a single zipped file, 'surname_name_hw6.zip'.
- **Submit your code through the Moodle page. DO NOT EMAIL.**
- C / C++, Python, or Java will be used as programming language. STL is allowed. **You may reuse code(s) from your previous homework assignments.** Make sure your code compiles and works in Linux systems (gcc compiler for C/C++).
- All submissions must be made by 23:59, December 30, 2019. Late submissions will lose 20 points for each additional day. Three or more days of delay will result in a zero grade.