Homework #4

Due April 12th, 11:59pm

Each homework submission must include:

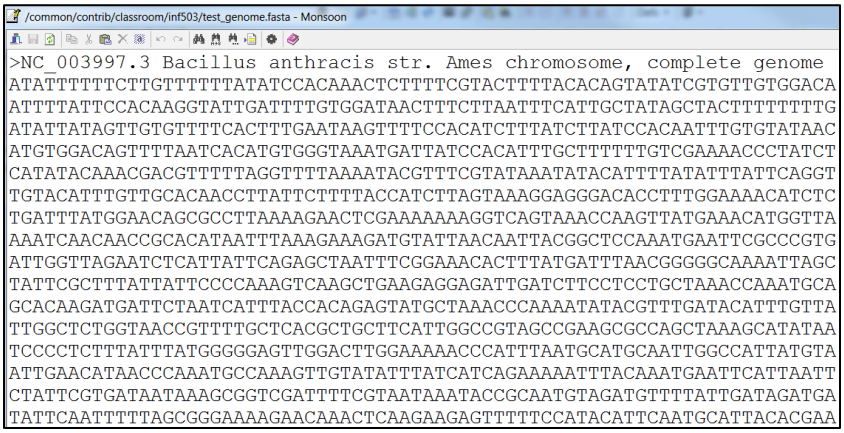
- An archive (.zip or .gz) file of the source code containing:
  - The makefile used to compile the code on Monsoon **(5pts)**
  - All .cpp and .h files **(5pts)**
- A full write-up (.pdf of .doc) file containing answers to homework's questions **(5pts)**, including the exact command line needed to execute every subproblem of the homework

The source code must follow the following guidelines:

- No external libraries that implement data structures discussed in class are allowed, unless specifically stated as part of the problem definition. Standard input/output and utilities libraries (e.g. math.h) are ok.
- All external data sources (e.g. input data) must be passed in as a command line argument (no hardcoded paths within the source code **(5pts)**.
- Solutions to sub-problems must be executable separately from each other. For example, via a special flag passed as command line argument **(5pts)**

For this homework, you will need to use the sequence fragments you used as part of Homework #1 and #2, located at **/common/contrib/classroom/inf503/hw_dataset.fa** . You will also need the genome sequence for B. anthracis bacterium located at: **common/contrib/classroom/inf503/test_genome.fasta**

- This genome file contains a header (denoted by '>') followed by ~5.2 million characters of its genomic code (alphabet A, C, G, T)
- Please be aware that the genome is spread across multiple lines of the file (see insert)



Problem #1 (of 1): Having a BLAST

Create a class called ***BLAST_DB*,** which would facilitate seed-based searching. Use the **hash table** data-structure with chaining collision resolution as the primary data structure in the class. The class must be able to take a genome sequence, break it down into all possible seeds with a given word size (for this homework, **word size = 11**), and store these seeds (along with their location in the genome) in the hash table. Use Radix / division scheme for hash function implementation. Seed searching should be done via the hash table, seed extension should be done using a Needleman-Wunsch algorithm. Assume a gap penalty of **-1**, mismatch penalty of **-1**, and match score of **+2**.

At minimum, the class must contain (**25pts**):

- A constructor
- A destructor
- A function to disassemble the genome into seeds of a given size and store them into the hash table data structure.
- A function to implement a Needleman Wunsch algorithm.
- A function to search a given query withing the BLAST_DB class. Your function must return the genome location of the best hit and score for the best hit.

A. **(50 pts) Basic seed-based searching**. Implement seed-based Needleman-Wunsch searching algorithm. For this you will need to (a) precompute and store all possible 11-mer seeds of the *B. anthracis* genome in the BLAST_DB's hash-table data structure – use m = 10,000,000 to ensure good performance and do not forget to store seed's location in the genome; (b) disassemble each query into all possible 11-mers, storing the location of each 11-mer in the original query; (c) search the query's 11-mers in the seeds stored in BLAST_DB and store each seed-hit in a stack or queue data structure (d) expand each seed into a full alignment by selecting a proper portion of the genome sequence and calling your Needleman-Wunsch function; and (e) compare scores for all seed extensions to find the best alignment.

- Randomly select 10K, 100K, and 1M (million) 50-mer fragments from the *B. anthracis* genome and use these fragments to query the BLAST_DB containing *B. anthracis*. How many fragments were you able to find and how long did it take?
- Randomly select 10K, 100K, and 1M (million) 50-mer fragments from the *B. anthracis* genome and introduce a 5% per-base error rate (every character has a 5% change of being changed to some other random character). Use these error-filled fragments to query the BLAST_DB containing *B. anthracis*. How many fragments were you able to find?
- Read the High Throughput Sequence reads dataset you used for homework #1 and #2 (located at **/common/contrib/classroom/inf503/hw_dataset.fa**). For this assignment, you can completely disregard the headers of the sequence fragments (i.e. R0_0_1…). Search the entire contents of this dataset in the BLAST_DB. How many perfect hits did you find? (hint: perfect hit's score = 100) Please note that depending on the efficiency of your algorithm, this step may take a long time. First estimate the total time using 1,000, 10,000, and 100,000 queries – if total time estimate is greater than 24 CPU hours, provide estimate rather than exact number.

B. (**15 points extra credit**) **Making it pretty.** Adapt your Needleman-Wunsch function to also return a human-friendly alignment (see below). You will need to actually perform a traceback and identify matching characters and gaps (if any). Hint: use a struct of 3 character arrays, 2 for sequences one for alignment codes {x, I , } – use whitespace for gaps).

```
GATTA
| x||
G_CTA
```