

## Homework #3

Due March 22<sup>nd</sup>, 11:59pm

---

Each homework submission must include:

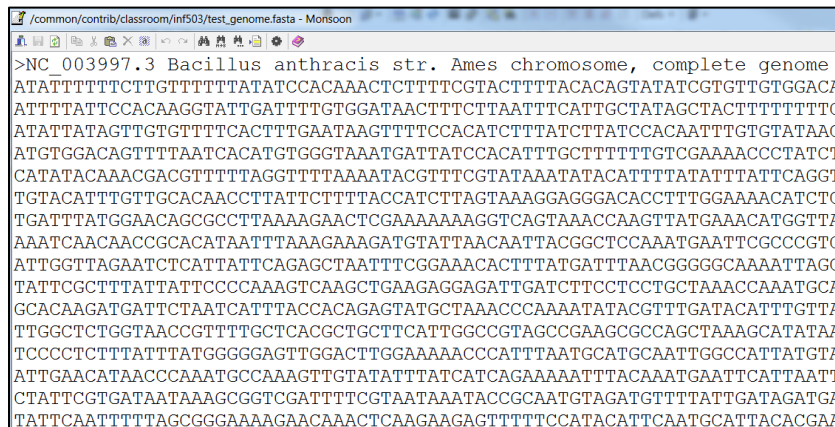
- An archive (.zip or .gz) file of the source code containing:
  - The makefile used to compile the code on Monsoon (**5pts**)
  - All .cpp and .h files (**5pts**)
- A full write-up (.pdf or .doc) file containing answers to homework's questions (**5pts**), including the exact command line needed to execute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



```
/common/contrib/classroom/inf503/test_genome.fasta - Monsoon
>NC_003997.3 Bacillus anthracis str. Ames chromosome, complete genome
ATATTTTTCTTGTTTTTATATCCACAACTCTTTTCGTACTTTTACACAGTATATCGTGTGTGGACA
ATTTTATCCACAAGGTATTGATTTTGTGGATAACTTTCTAATTTTCATTGCTATAGCTACTTTTTTTG
ATATTATAGTTGTGTTTCACCTTTGAATAAGTTTCCACATCTTTATCTTATCCACAATTTGTGTATAAC
ATGTGGACAGTTTAAATCACATGTGGGTAAATGATTATCCACATTTGCTTTTTTGTGCGAAAACCTATCT
CATATACAAACGACGTTTTTAGGTTTTAAAATACGTTTCGTATAAAATATACATTTTATATTTATTCAGGT
TGTACATTTGTTGCACAACCTTATCTTTTACCATCTTAGTAAAGGAGGGACACCTTTGGAAAACATCTC
TGATTTATGGAACAGCGCCTTAAAGAACTCGAAAAAAGGTCAGTAAACCAAGTTATGAAACATGGTTA
AAATCAACAACCGCACATAATTTAAAGAAAGATGTATTAACAATTACGGCTCCAAATGAATTCGCCCGTG
ATTGGTTAGAATCTCATTTATTCAGAGCTAATTTTCGGAACACCTTTATGATTTAACGGGGGCAAAATTAGC
TATTCGCTTTATTTATCCCAAAGTCAAGCTGAAGAGGAGATTGATCTTCTCCTGCTAAACCAATGCA
GCACAAGATGATCTAATCATTTACCACAGAGTATGCTAAACCAAAATATACGTTTGATACATTTGTTA
TTGGCTCTGGTAACCGTTTTGCTCACGCTGCTTCATTGGCCGTAGCCGAAGCGCCAGCTAAAGCATATAA
TCCCTCTTTATTTATGGGGGAGTTGGACTTGGAAAAACCATTTAATGCATGCAATTGGCCATTATGTA
ATTGAACATAACCCAAATGCCAAGTTGTATATTTATCATCAGAAAAATTTACAAATGAATTCATTAATT
CTATTCGTGATAATAAAGCGGTCGATTTTCGTAATAAAATACCGCAATGTAGATGTTTTATTGATAGATGA
TATTCATTTTGTAGCGGAAAAGAACAACTCAAGAAGAGTTTTTCCATACATTCAATGCATTACACGAA
```

### Problem #1: The hash table with chaining

Create a class called **FASTAreadset\_HT**. Use the hash table data-structure to store the genomic fragments of a given size. The class must include the size of the hash table (*m*) as one of its configurable parameters. If you have a duplicate sequence fragment or a duplicate hash value, use chaining method to resolve collisions. Resizing is optional - you can hard-code the proper hash table size through the constructor. Use Radix / division scheme for hash function implementation.

At minimum, the class must contain (15pts):

- A constructor
- A destructor
- A function to search the hash table for a given n-mer sequence (returning a presence / absence Boolean value should be sufficient)
- A function to insert a given n-mer sequence into the hash table
- A function to convert a given sequence to a Radix notation (use double or unsigned int data type to store the radix value)

A. (20 pts) **Assess the impact of the hash table size.** For this you will need to set the hash table to a fixed value (*m*, see below). Set the size of your hash table (*m*) to 10 thousand, 100 thousand, 1 million, and 10 million elements. Populate the hash table with every possible 16-mer that you can generate from the *B. anthracis* genome.

- For each of your 4 hash table sizes, how many collisions did you observe while populating the hash?
- For each of your 4 hash table sizes, how long did it take you to populate the hash table? Do the timing results make sense? Explain.

B. (20 pts) **Searching speed:** Set the hash table size to 10 million. Populate the hash table with every possible 16-mer that you can generate from the *B. anthracis* genome.

- Generate 1 million random 16-mers from the *B. anthracis* genome (pick a random start somewhere in the genome and read 16 characters from that point). Search these within your hash table. How many of these fragments did you find and how long did it take?
- Generate 1 million completely random 16-mers (not from the genome): Search these within your hash table. How many of these fragments did you find and how long did it take? Do the timing results make sense? Explain.
- Without implementing anything, explain how would the timing for above 2 tasks change if *m* was equal to 10,000.

- C. **(20 pts) The trouble with hash tables.** Set the hash size to 10,000,000 and populate it using all 16-mers found in the *B. anthracis* genome.
- Iterate through the *B. anthracis* genome again, generating all possible 16-mers. How many of these 16-mer fragments were found in your hash table?
  - Iterate through the *B. anthracis* genome, introducing a 1% per-base error rate (every character in the 16-mer has a 1% chance to change to something else). **(10 pts** of extra credit if you use an appropriate distribution during simulation). How many of these fragments did you find?