Homework #5

Due December 3rd, 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 *subject dataset* (human genome assembly that you used in HW#1). Recall that it is located at: **/common/contrib/classroom/inf503/genomes/human.txt**

- This file contains multiple scaffolds that comprise the human genome
- The genome is in FASTA format (see insert)
  - The headers are unique and always begin with the ">" character. These can be discarded for this homework. Each line of genome file is exactly 80 characters long (plus carriage return character)
  - The genomic sequences consist of the following alphabet {A, C, G, T, N}

```
>568815281-9606
CTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCT
AACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAA
CCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACC
CTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCT
AACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTTAACCCTAACCCTAACCCTAACCCTA
ACCCTAACCCTAACCCTAACCCTCACCCTTCTAACTGGACTCTGACCCTGATTGTTGAGGGCTGCAAAGAGGAA
GAATTTTATTTACCGTCGCTGTGGCCCCGAGTTGTCCCAAAGCGAGGTAATGCCCGCAAGGTCTGTGCTGATCAGGACGC
AGCTCTGCCTTCGGGGTGCCCCTGGACTGCCCGCCCGCCCGGGTCTGTGCTGAGGAGAACGCTGCTCCGCCTCCGCGGTA
CTCCGGACATATGTGCAGAGAAGAACGCAGCTGCGCCCTCGCCATGCTCTGCGAGTCTCTGCTGATGAGAACACAGCTTC
ACTTTCGCAAAGGCGCAGCGCCGGCGCAGGCGCGGAGGGGCGCGCAGCGCCGGCGCAGGCGCGGAGGGGCGCGCCCGAAC
CCGAACCCTAATGCCGTCATAAGAGCCCTAGGGAGACCTTAGGGAACAAGCATTAAACTGACACTCGATTCTGTAGCCGG
CTCTGCCAAGAGACATGGCGTTGCGGTGATATGAGGGCAGGGGTCATGGAAGAAAGCCTTCTGGTTTTAGACCCACAGGA
AGATCTGTGACGCGCTCTTGGGTAGAGCACACGTTGCTGGGCGTGCGCTTGAAAAGAGCCTAAGAAGAGGGGGCGTCTGG
AAGGAACCGCAACGCCAAGGGAGGGTGTCCAGCCTTCCCGCTTCAACACCTGGACACATTCTGGAAAGTTTCCTAAGAAA
```

Problem #1 (of 1): Prefix Trie

Create a class called ***Prefix_Trie.*** The purpose of the class will be to contain a dataset of genomic sequences (queries) and all of the functions needed to operate on this set. Use the **prefix trie** data-structure to store the genomic fragments of a given size. Here you will be performing fuzzy matching, tolerating up to 1 mismatch.

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

- A default constructor
- At least one custom constructor to build a trie from a set of queries (of size n)
- A function to traverse (*search*) the trie using a genome of size G. Note that you can assume that G >> n. You will need to implement a ***fuzzy search tolerating up to 1 mismatch*** **(substitutions only)**. Hint: use a stack to keep track of branches in the tree that need to be explored.
- A destructor
- A copy constructor

A. (**25pts) Basic prefix trie**: Pick a random 50K long segment from the human genome assembly. Generate 5K, 50K, 100K, and 1M random 36-mers this segment and store them in the prefix trie. Hint: generate a random starting position somewhere in the segment and read 36 characters starting from that position.

   - For each of the 36-mer datasets, what are the sizes of the trie (# of nodes)? Explain the pattern that you observed.
   - Iterate through all possible 36-mers in the segment, using each to search / traverse the prefix trie with up to 1 mismatch. How many of your 36-mers had a match? Does it make sense? Explain why.

B. **(25pts) Impact of error rate on trie structure:** Use the same random 50K long segment from the human genome assembly that you used in part A. Generate 5K, 50K, 100K, and 1M random 36-mers from this segment with **5% per-base error rate** and store them in the prefix trie. Hint: repeat the process from part A, except each base of 36-mer has a 5% chance of mutation/error.

   - For each of the 36-mer datasets, what are the sizes of the trie (# of nodes)? Explain differences (if any) between the trie sizes in partA and part B.
   - Iterate through all possible 36-mers in segment, using each to search / traverse the prefix trie with up to 1 mismatch. How many of your 36-mers had a match? Does it make sense? Explain why.