Homework #1

Due February 16th, 11:59pm

Each homework submission must include:

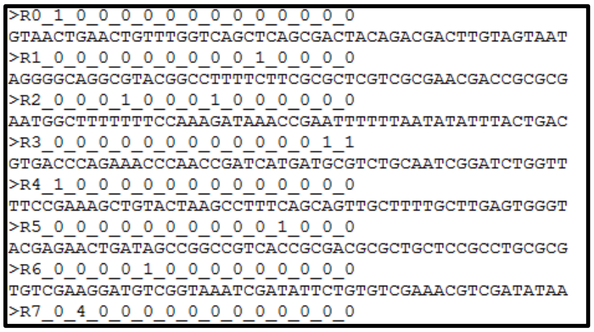
- An archive (.zip or .gz) file of the source code containing:
    - The makefile used to compile the code on Monsoon **(5pts)**
    - All .cpp and .h files **(5pts)**
    - A readme.txt file outlining all modules (if any) needed for the execution of the code and the exact command lines needed to answer homework's questions **(5pts)**
- A full write-up (.pdf of .doc) file containing answers to homework's questions **(5pts)** – screenshots of code output are ok.

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.
- 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 High Throughput Sequence reads dataset located on Monsoon: /common/contrib/classroom/inf503/hw_dataset.fa

- Dataset contains approximately 36 million 'reads' (genomic sequence fragments of equal length)
- Each read is exactly 50 nucleotides (characters) long
- The read set is in FASTA format (see insert)
    - The headers are unique and consist of the read ID number (e.g. R1) and a series of 'copy number' values for the number of times this read is present in sample 1, 2, … (separated by underscore "_")
    - The genomic sequences consist of the following alphabet {A, C, G, T, N}



```
>R0_1_0_0_0_0_0_0_0_0_0_0_0_0_0
GTAACTGAACTGTTTGGTCAGCTCAGCGACTACAGACGACTTGTAGTAAT
>R1_0_0_0_0_0_0_0_0_0_1_0_0_0_0
AGGGGCAGGCGTACGGCCTTTTCTTCGCGCTCGTCGCGAACGACCGCGCG
>R2_0_0_0_1_0_0_0_1_0_0_0_0_0_0
AATGGCTTTTTTTCCAAAGATAAACCGAATTTTTTAATATATTTACTGAC
>R3_0_0_0_0_0_0_0_0_0_0_0_0_1_1
GTGACCCAGAAACCCAACCGATCATGATGCGTCTGCAATCGGATCTGGTT
>R4_1_0_0_0_0_0_0_0_0_0_0_0_0_0
TTCCGAAAGCTGTACTAAGCCTTTCAGCAGTTGCTTTTGCTTGAGTGGGT
>R5_0_0_0_0_0_0_0_0_0_0_1_0_0_0
ACGAGAACTGATAGCCGGCCGTCACCGCGACGCGCTGCTCCGCCTGCGCG
>R6_0_0_0_0_1_0_0_0_0_0_0_0_0_0
TGTCGAAGGATGTCGGTAAATCGATATTCTGTGTCGAAACGTCGATATAA
>R7_0_4_0_0_0_0_0_0_0_0_0_0_0_0
```

Problem #1 (of 1): Arrays and Classes

Create a class called ***FASTA_readset***. The purpose of the class will be to contain a FASTA read set, all of the statistics associated with it, and all of the functions needed to operate on this set. Use an array data-structure to store the genomic sequence of the given read dataset. Use character arrays (char[ ]) to store the sequence, rather than 'string' object (you should have an array-of-arrays 36 million by 50). Use another set of arrays to store the copy number counters from the header string. At minimum, the class must contain:

- A default constructor
- At least one custom constructor (e.g. one taking a file path or ifstream as input)
- A function to read the FASTA fasta file
- A single function to compute all statistics for the Readset (see below)
- A destructor

A. Using the first 1 million reads, estimate and <u>**report**</u> the total CPU time and RAM it will take to initialize (fill up) the array data-structure with the entire 36 million reads. Note that this may mean creating custom constructor to read first ***X*** reads rather than to the End-Of-File.

B. Test your prediction using the entire 36 million read set – <u>**report**</u> actual RAM and CPU time used. Refer to Monsoon workshop notes for help in estimating actual runtime and RAM usage of your run. Were you accurate? If not, explain what you think caused the discrepancy.

C. Compute the following statistics for your read set
    - Total number of unique sequence fragments (here, safe to assume this is the total number of sequence fragments in the file).
    - Total number of reads for each 'data set' separately (recall there are 14 data sets in our example here). You will need 14 different totals.
    - Number of A, C, G, and T characters in the dataset.

D. Implement a destructor for your class to delete / deallocate your array data structure. How long did it take? Does this make sense to you?

E. Implement a function that would sort the genomic sequences (fragments not characters within a fragment) in your array in alphabetic order.
    - What is the 'big O' notation of your approach (linear / quadratic / cubic / etc)? Please note that depending on the efficiency of your algorithm, you may not be able to alphabetically sort the entire 36 million reads in a reasonable amount of time (24-36 CPU hours). If this happens, reduce the problem size (by using a smaller subset of the reads) and estimate the final run time.
    - Print the first 20 lines of the sorted output