Project 3

Due August 31, 2018 at 11:59 PM

This project specification is subject to change at any time for clarification. For this project you will be working with a partner. You will be constructing a data structure (called CFrequentWordCounter) that will count the occurrences of words (both case sensitive and insensitive). Lines of text will be passed to the data structure which will need to be parsed into words. A word is considered any consecutive string of alpha numeric characters in a row. All punctuation and whitespace should be ignored, with one caveat. The single quote will be considered as part of a word, *iff* it is preceded and succeeded by alphanumeric characters. The CFrequentWordCounter has a GetFrequentWords function that must fill up the two vectors with the *n* most occurring words seen thusfar. Ties in frequency are broken by the ASCII sorting of the words (so "Three" would appear before "four"). As part of the project you are encouraged to add your own GoogleTests for testing any data structures that you will be building in order to implement the CFrequentWordCounter data structure.

A working example can be found on the CSIF in /home/cjnitta/ecs60/proj3. Your program is expected to run in approximately the same amount of time as the provided example. Extra credit may be available for solutions that significantly outperform the provided baseline solution. The use of the C++ STL containers is limited, you may use the string, tuple, vector, and list containers. Your program may not have memory leaks and will be tested using valgrind.

You can run the example program with the command:
```
./proj3 textfile.txt [num_words]
```

Your solution match the baseline because of the strict sorting order.

Several text files have been provided for testing. They can be found in the data directory of the given tgz as well as the /home/cjnitta/ecs60/proj3 directory on the CSIF.

You **must** submit the source file(s), a Makefile, README.txt file and interactive grading timeslot CSV file, in a tgz archive. You can tar gzip a directory with the command:
```
tar -zcvf archive-name.tgz directory-name
```

You should avoid using existing source code as a primer that is currently available on the Internet. You **must** specify in your readme file any sources of code that you have viewed to help you complete this project. All class projects will be submitted to MOSS to determine if students have excessively collaborated. Excessive collaboration, or failure to list external code sources will result in the matter being referred to Student Judicial Affairs.

A rough outline for approaching this project might be:

1. Write a function or build a structure to convert a line of text into individual words. You should write tests for this, as it should be to come up with a number of tests easily. Also write your tests before you code.

2. Build a structure to hold the words and the counts, so that the count can be incremented for every word. Keep in mind that you will later want to be able to track case sensitive and insensitive words.
3. Find the top n occurrences of case sensitive and insensitive words for the structure you built.