Programming Assignment #5
CS253, Fall 2017
Due: Tuesday, Oct. 17th

"*Exceptions and Capitalization*"

Motivation & Description

You have spent the last 4 assignments (5 weeks) writing code that reads a text file, breaks it up into word strings and punctuation strings, stems the word strings, and marks capitalization. Why? We're going to analyze text documents, obviously, but first we need to take care of two more loose ends.

One is capitalization. We currently have capitalized words, uncapitalized words, and "ambiguous" words marked with a "+". We need to resolve the ambiguous words. Our strategy for doing this is relatively easy, although it will make some errors. For every word that is ambiguously capitalized – i.e. every word marked with a "+" in PA3 – we will look through the rest of the words in the document. If the same word appears unambiguously capitalized in the same document, then we assume the word should be capitalized and remove the "+" while keeping the first letter capitalized. Otherwise, if the word does not appear unambiguously capitalized within the same document, we remove the "+" and convert the first letter to lowercase.

Significantly, in PA5 you will either (1) resolve ambiguous capitalizations before word stemming (so that words that become lowercase get stemmed) or (2) stem words if and when they are determined to be lowercase.

The second loose end is words that are not correctly stemmed by the Porter word stemming algorithm. The Porter algorithm stems words correctly *most* of the time – more than other approaches. But it makes mistakes, so your PA5 algorithm will take two filenames as arguments. The second is the text file to be analyzed. The first, however, is now a file of stemming exceptions. Each line of the exception file contains two words, separated by a space. The first is a word to be stemmed. The second is what the word should be stemmed to. For example, two lines in the exception file might look like:

```
proceed proceed
proceeds proceed
```

In the example above, the word *proceed* should be replaced by *proceed* (in other words, it remains unchanged), instead of going through the Porter stemming algorithm. The word *proceeds* should be shortened to *proceed* instead of being Porter stemmed. Any word that is not in the exception file should be stemmed as in PA4.

Task

Your program will take two file names as arguments. The first file is the stemming exceptions file. It contains two words per line, separated by a space. The first is a word that does not stem correctly according to the Porter algorithm. The second word is what the first word should stem to. The first file could be, but usually isn't, empty. The second file is the text file. It contains strings separated by whitespace. These strings should be divided into word strings and punctuation strings as in PA3. Word strings with ambiguous capitalization should be resolved to begin with either an uppercase or lowercase letter, as described above. Then, word strings that are eligible to be stemmed – in other words, word strings longer than two characters and contain no capital letters or numbers, as specified in PA4 – are compared to the words that appear at the beginnings of lines in the exception file. If the word string matches the first word in a line, the word string is replaced by the second word. Otherwise, the word is stemmed as in PA4. Your program will then output the lexicographically sorted and in some cases replaced or stemmed strings from the text file and their frequencies, in the same format as in previous assignments (PA2 through PA4).

Submitting Your Work

To submit your program, first create a single tar file containing all of your source files (i.e. your .cpp and .h files) and your makefile, but not your compiled files (no executable or .o files, please). Then submit the tar file as PA5 on Canvas.

Grading Your Homework

To grade your assignment, the GTAs will unpack your archive in an empty directory. They will compile your code by typing 'make'. This command must compile your code from scratch, and it must produce an executable called PA5 in a directory called PA5. If your code does not compile using make, you will receive a zero for the assignment. Compiling your code should not produce any warning messages. Although we will not deduct points for warning messages, we may in future assignments so get in the habit of making sure you code compiles cleanly. Assuming your program compiles, it will be graded by how it performs on test cases, including test cases with errors in one or both of the input files. If the input file contains errors, you will receive full credit if your program returns -1 to main. (In case of an error, it should also print a meaningful error message to std::cerr.) If the input does not contain errors, your program should return 0 to main.

Hints

- We are grading for correctness, not efficiency.
- If there are any errors in your PA4, fix those first. Then work on PA5.
- Use valgrind to spot memory errors, and fix them. Any time you expand your program (e.g. in future assignments), old memory errors can lead to new bugs, even if they did not create problems in previous assignments.

Policies

All work you submit must be your own. You may not submit code written by a peer, a former student, or anyone else. You may not copy or buy code from the web. The department academic integrity policies apply.

You may not submit your program late. To receive credit, it must be submitted on Tuesday, Oct. 17th. There is no late period. The exception is an unforeseeable emergency, for example a medical crisis or a death in the immediate family. If an unforeseeable emergency arises, talk to the instructor.