Programming Assignment #7
CS253, Fall 2017
Due: Thursday, Nov. 9th

"*Reading Level*"

Motivation & Description

We are getting to the end of the semester. I can now tell you what the final form of your project is. You are writing a system that takes in a query document and a set of background documents. The goal is the find a document among the set of background documents that is (1) on a similar subject to the query document and (2) and a desired reading level. For example, imagine that you have a document on a subject that you have to talk to your child about. Then you might want to find a similar document, but at a lower reading level. Alternatively, you might have a simple, introductory document, and you want to find a more sophisticated text on the same document. This is what the tool you are building is for. (Granted, it would be nicer with a web interface….)

So that's PA8. For PA7, you will build the one piece we are still missing: the code to estimate the reading level of a document. This can be done using what's known as the Coleman-Liau (CL) index. The CL index is quite simple CL(doc) = 0.0588L(doc) – 0.296S(doc) – 15.8, where L(doc) is the average number of letters per 100 words, and S(doc) is the average number of sentences per 100 words. Thus, to do this assignment you will go back to the early parts of your assignment, before you have stemmed any words, to the point where you have the document represented as a sequence of word strings and punctuation strings. You will count the number of letters per 100 words (ignoring punctuation strings, they are not words), and the number of sentences per 100 words (here you will need the punctuation strings to define sentence boundaries).

Task

Your program will take a single filename as its argument, and will write a single number -- the reading level of the document -- to std::cout. (Unless, of course, there is an error, in which case it will print an error message to std::cerr and return -1.)

The reading level of the input file is determined by the CL index. You will parse the input file into a sequence of word strings and punctuation strings as in PA3. You will calculate the term L(doc) by counting the total number of letters in the word strings in the file (i.e. not including punctuation or spaces). You will also count the number of word strings in the document, and divide the number of letters by 100 times the number of word strings. Note that you should not assume that there are 100 words in the document.

To compute S(doc) you need to know the number of sentences, which is the same as the number of punctuation strings that end sentences (i.e. that contain a period, question mark, or exclamation point). You then divide this number by 100 times the number of words.

Finally, you compute the CL index as CL(doc) = 0.0588L(doc) – 0.296S(doc) – 15.8 and print it to std::cout.

Hints

- Look at PA8 as described in the motivation. You will need all your PA4 through PA6 code for PA8. Don't throw it away! Make sure it will merge with your new PA7 code.
- We are still grading on accuracy, not efficiency, but PA9 is coming soon. Now is a good time to run valgrind on your code, locate the inefficiencies, and get rid of the most egregious one or two.

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 PA7 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 PA7 in a directory called PA7. 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.

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, Nov. 9th. 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.