CS/SE 1337 – Homework 4 –Tongue Twisters (String Processing)

Dr. Doug DeGroot's C++ Class

Due: Sunday, March 12, 2023, by midnight

Purpose: to get experience processing and manipulating text strings and vectors

How to submit: Upload your entire project (code, PDF copies of output files, comments, etc.) to eLearning. The project must compile on either Codeblocks or MS Visual Studio.

Maximum Number of Points: 100

References: Chapter 10 of your text, the C++ Reference site (https://bit.ly/2pLQFzj) , and the PDF file on strings that I uploaded to eLearning (in case you don't have the newest version of our textbook).

Note: The following homework description may look like a lot of work, but it shouldn't be. Most of the requirements are simple variations of each other. Just write the code to process one sentence, and then expand the code to read and process the others in a loop.

Objective:

Write a program that performs the following tasks:

1. There is a data file provided with this homework called Tongue-Twisters.txt. It contains several tongue-twister sentences, each ending with a newline. You are to open the file, read each sentence, and perform the following text manipulations on each sentence—***one at a time.*** The text in the file will be something like the following:

    The rain in Spain falls mainly on the plains.
    How much wood would a woodchuck chuck if a woodchuck could chuck wood?
    Suzy sells seashells by the seashore.
    Becky's beagle barked and bayed, becoming bothersome for Billy.
    etc.
2. As you read a sentence, store the words of the sentence in a vector of words (strings).
3. Process each of the sentences as follows, one by one (i.e., don't let your program proceed to the next sentence until after performing and printing the results of all the procedures below on your current sentence and reporting the results).
4. First, simply print the sentence. Then report the number of characters in the sentence (i.e., the sentence's length).
5. Then, report the number of words in each sentence (not unique words, just the number of actual words in the string).
6. Find the longest word in each sentence and report it.
7. Then, count the number of vowels, the number of consonants, the number of spaces, the number of punctuation characters, and the number of *other* characters in the string. (Don't use a switch/case statement to test for vowels; use string's *find(…)* member function.)
8. Convert the text in each line/string to Title Case and print the result (first letter of each word is capitalized; all others aren't). Like this: My Dog Has Fleas
9. Convert the text in each line to Sentence Case (capitalize the first letter of the first word in the entire string but lowercase all the remaining words).

10. Convert the text in each line to Toggle Case (the first letter of each word is to be lowercase while the rest of the letters in each word are to be uppercase).
    Like this: mY dOG hAS fLEAS
11. Create a lookup table (vector) of words that should never be converted to lower-case when using sentence case (like Suzy, Fritos, Spain, etc.). Fill the table with words of your own choosing, but get them from the data file for this program. You don't have to include them all, but you should include most of them. And then never convert them to lower case when you're converting a phrase to sentence case text.
12. After doing all the above, proceed to the next sentence in your list.

Annotations for the code:

1. From now on, code with little or no debugging code support will lose 10 points or more, depending on the amount of debugging support you have included. Make your program talk to you, but be able to tell it to be quiet once you're ready to turn your code in. And *do* turn it off before submitting it to eLearning.
2. The main function can be at either the beginning or the end of the program. I don't care which. (There are pros and cons for each decision.)
3. Separate the output for each sentence from the next sentence, using one or more blank lines or something like the *line()* function we've seen several times in class.
4. Add comments at the top of your main.cpp file to include your name, the name of the program, the date you created the file, and notes on how your design works when executed.
5. Point out any special features or techniques you added using a comment saying "// Special Features."
6. Comment your code effectively, as we discussed in class. Use descriptive variable names everywhere so that the code becomes as self-documenting as possible. Use additional commentary to improve readability and comprehensibility by other people. You will lose 5 points if you code is cryptic, hard to read, not self-documenting, etc.
7. You absolutely MUST use consistent indentation and coding styles throughout the program. Ust the Google formatting style! Failing to do so will result in a loss of 5 points.
8. If the program does not work at all or even in part, or works incorrectly, at least points will be deducted.
9. LeAVE YoUr DEbugGING coDe iN uR program! (But turn it off by using a control flag before submitting the homework. Do not just comment it out.)
10. Print all your program's output to a file. (You might want to print to both a file and the console, but turn in the file version of your output.)
11. No late submissions will be accepted, period. Please meet the deadline. Don't even ask.
12. If you have questions, approach your graders first.