CS/SE 1337 – Homework 4 – String Processing

Dr. Doug DeGroot's C++ Class

Due: Sunday, March 28, 2021, by midnight

Purpose: to get more experience processing and manipulating C++ strings and vectors.

How to submit: Upload your .cpp file and copies of your entire output to the eLearning site. **The project must compile on CodeBlocks.**

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.

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 augment the code to read and process the others.

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 a number of tongue-twister sentences, each ending with a period, question mark, or other terminating punctuation symbol (and there may be intermediate punctuation symbols as well). You are to open the file, read every sentence into a vector of strings, and perform the following text manipulations on each sentence. 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. Store the sentences in a vector of strings. Then for each sentence, store the words of each sentence in a vector of words (strings). Thus you will want to use both a vector<string> (for the sentences) and a 2-dimensional vector<vector<string>> (for the words). This is because some of the processing you will do will be on the sentences and some will be on the words/characters. Process each of the sentences as follows and report ***on separate lines*** the following items:
3. First, simply print the sentence.
4. Then second, report the number of characters in the sentence (i.e., the string's length).
5. Third, report the number of words in each sentence (not unique words, just the number of actual words in the string).
6. Fourth, find the longest word in each sentence and report it.
7. Then fifth, count the number of vowels, the number of consonants, the number of spaces, and 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). **However**, create an array/vector of proper nouns, like Fritos, Friday, Spain, etc. When you are doing your conversion to Sentence Case, do NOT lowercase these proper nouns.
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. Convert the text in each line to Mocking (Spongebob) Case (each letter in the sentence should be converted to upper or lower case with a 50% probability, or whatever probability you prefer). This is similar to Toggle Case, but slightly different; you will have to use different logic for the two cases.
    Like this: My dOG haS fLEaS.
12. After doing all the above, proceed to the next sentence in your vector of sentences and repeat the process
13. And for a final bit of fun, you might try converting each sentence into Leet Speak. I've included with this homework assignment a text file called **LeetChars-Small.txt**. It contains 26 pairs of letter-leet combinations. Leet characters come in a whole slew of different forms, so I just selected one that I believe are the most commonly used. Note that leet chars aren't chars in the C++ sense; instead, they are short strings of chars. You should convert to Leet Speak in a manner similar to what you did with Mocking Case, that is, you should convert only somewhere like 50% of your characters to their leet equivalent. But feel free to do more or less depending on what you think looks the most interesting.
14. For each major goof, we will probably deduct 6 points, so don't make any goofs! And you must exercise good coding conventions and styles as before.

Annotations for the code:

1. 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.)
2. 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.
3. Add comments at the top of your main.cpp file to include your name, the name of the program, and notes on how your design works when executed. Point out any special features or techniques you added using a comment saying "// Special Features."
4. 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.
5. You absolutely MUST use consistent indentation and coding styles throughout the program. Failure to do so will result in a loss of 5 points.
6. If the program does not work at all or even in part, or works incorrectly, 10 points will be deducted.
7. LeAVe YoUr DEbugGING coDe iN uR program! (But turn it off by commenting it out before submitting the homework.)
8. No late submissions will be accepted, period. Please meet the deadline. Don't even ask.