CSCI-15 Assignment #2, C-string processing. (60 points) Due 9/20/18

**You MAY NOT use C++ string objects for any purpose in this program.** If you use a C++ `string` object for this assignment **at all**, I will give you a score of **zero**.

Write a C++ program that reads lines of text from a file using the ifstream getline() method, tokenizes the lines into words ("tokens") using strtok(), and keeps statistics on the data in the file. Your input and output file names will be supplied to your program on the command line, which you will access using argc and argv[].

You need to count the total number of words seen, the number of unique words, the count of each individual word, and the number of lines. By "unique words", I just mean words that have not been seen yet in the program. Once a word has been seen, it is no longer unique if or when you see it again.

Also, remember and print the longest and shortest words in the file. If there is a tie for longest or shortest word, you may resolve the tie in any consistent manner (e.g., use either the first one or the last one found, but use the same method for both longest and shortest).

You may assume the lines comprise words (contiguous lower-case letters [a-z]) separated by spaces, terminated with a period. You may ignore the possibility of other punctuation marks, including possessives or contractions, like in "Jim's house". Lines before the last one in the file will have a newline ('\n') after the period. In your data files, omit the '\n' on the last line. You may assume that a line will be no longer than 100 characters, an individual word will be no longer than 15 letters and there will be no more than 100 unique words in the file. **However, there may be more than a hundred total words or lines in the file.**

Read the lines from the input file, and echo-print them to the output file. After reaching end-of-file on the input file (or reading a line of length zero, which you should treat as the end of the input data), print the words with their occurrence counts, one word/count pair per line, and the collected statistics to the output file. You will also need to create other test files of your own. Also, your program must work correctly with an EMPTY input file – which has NO statistics.

My test file looks like this (exactly four lines, with NO NEWLINE on the last line):

```
the quick brown fox jumps over the lazy dog.
now is the time for all good men to come to the aid of their party.
all i want for christmas is my two front teeth.
the quick brown fox jumps over a lazy dog.
```

Copy and paste this into a file for one of your tests.

Hints:

Use a 2-dimensional array of char, 100 rows by 16 columns (why not 15?), to hold the unique words, and a 1-dimensional array of ints with 100 elements to hold the associated counts. For each word, scan through the occupied lines in the array for a match (use strcmp()), and if you find a match, increment the associated count, otherwise (you got past the last word in the occupied part of the table), add the word to the table and set its count to 1.

The separate longest word and the shortest word need to be saved off in their own C-strings. (Why can't you just keep a pointer to them in the tokenized data?)

Remember – put NO NEWLINE at the end of the last line, or your test for end-of-file might not work correctly. (This may cause the program to read a zero-length line or seem to read the last line twice before seeing end-of-file.)

This is not a long program – no more than about 2 pages of code.