TCSS 333 HW5 Spring 2015

Due: by midnight Friday May 22 (or with 10% late penalty by midnight Saturday May 23)

Two files are posted on Canvas for use with this assignment: RedBadge.txt and LittleRegiment.txt. They are both works of fiction written by the author Stephen Crane. Make up a single linked list of words that appear in these files. For each word, keep track of the number of times it appears in each file (use two separate counters). We are especially interested in words appearing more frequently in one file than the other. After reading both input files, you should sort the word list in decreasing order of the difference between the counts. If the word "colonel" appears 80 times in one file and 15 times in the other, then it will be sorted based on the number 65 (the difference between the counts). All differences are positive. After this sort is complete, print out a numbered list of the first 50 words in the list, one word per line. For each word, display the two counts for that word. This list should include words that are used much more in one book than the other.

In addition to main, you should write a number of other functions to decompose your program. You should especially isolate your linked list operations from the rest of your program and put these linked list operations into a separate source code file. Your program will then consist of 3 files:

main.c list.c list.h

The entire program (including the sorting) must be done using the linked list. Putting all the data into an array is not allowed. Bubble sort is fine. When you see two nodes that are out of order, you should choose carefully between swapping the nodes (which requires changing several pointers) and swapping just the data in the nodes (no pointer changes required). Either approach is OK; one is less risky!

Finding words can be a little tricky because of punctuation. To standardize our ideas about what constitutes a word, we will follow this procedure:

- first use %s to scan in a string
- all letters should be converted to lower case as needed
- the word in the string may contain alphabetic characters, hyphen, and apostrophe only

(These are the "acceptable" characters.)

start at the left end of the string with the first acceptable character
scan to the right
stop when you find a character that is not acceptable or the string ends

For example, one of the files contains this line:
"Shucks!"
Using the method outlined above, the word in this string is shucks
(We have uncapitalized the S, jumped over the initial " and stopped on seeing the !)

As always, this is a programming assignment. All the code should be written by you.

You are welcome to discuss the correct answer on Canvas, but I will not post a solution.