Programming Assignment #5

Linked List of string

CS 2308.255 + CS5301 Spring 2019
Instructor: Jill Seaman

Due: Wednesday, 4/10/2019: upload electronic copy by 11:55pm.

Problem: Implement an interface that manipulates a list of strings. You will be provided with the following files on the class website:

- **StringList.h** containing a class declaration, set up for a linked list representation.
- **Driver.cpp** containing a main function you can use to test your implementation.

You will be responsible for providing the StringList.cpp file, including the implementation of the StringList member functions (described below):

StringList and **~StringList**: creates an empty list, and deallocates all the nodes in the list, respectively.

count: returns the total number of strings (nodes) in the list. Any duplicate strings should be counted.

push_front(string) Adds a new node containing the string to the front of the list.

push_back(string) Adds a new node containing the string to the end of the list.

pop_front() removes the first node, if the list is not empty (else does nothing).

display(): displays the strings in the list to the screen, one string per line.

remove(StringNode *m) removes the node that m points to from the linked list. This function must be private. Does nothing if m is not pointing to a node in the list.

maximum(): returns a pointer to the node containing the string that would come last in alphabetical (ascii) ordering. Does not change the list! Returns NULL if the list is empty. This function must be private.

sort(): Here is the algorithm you **must** use for implementing the sort function:

1. Define a StringNode * to be the head of a new list (make it the empty list). This should be a local variable (not a class member).
2. Repeat until the original list is empty:

   a. Find the maximum string in the **original** list and remove that node from the original list. Call functions you have already defined to do this.
   b. Insert this node into the proper position in the **new** list (at the front).
3. make the old head pointer (now empty) point to the new list!

Input/Output:

Use the provided Driver.cpp file to test your code. I recommend trying to implement one or two functions at a time, and testing them, rather than implementing all the functions and then trying to debug them all at once.

NOTES:

- This program must be done in a **Linux or Unix** environment, using a command line compiler like g++. Do not use codeblocks, eclipse, or Xcode to compile.
- Put your code in a file named **StringList.cpp.**
- Your StringList.cpp file **must compile** with the (unchanged) provided files, otherwise you may receive a score of 0.
- You may re-use code from the **NumberList** class (source: book/slides/website).

Logistics:

For this assignment you need to submit only the **StringList.cpp** file. You do not need a zip file, you do not need a makefile, you do not need to provide your driver.

There are two steps to the turn-in process:

1. Submit an electronic copy using the Assignments tool on the TRACS website for this class.
2. Submit a printout of the source file at the beginning of class, the next class day after the assignment is due. Please **print your name on top of the front page**, and staple if there is more than one page.

Note: Each member of a group must submit their own electronic copy and their own printout!! Make sure your name is written or circled on your printout.

See the assignment turn-in policy on the course website (cs.txstate.edu/~js236/cs2308) for more details.