EECE.3220: Data Structures

Spring 2019

Programming Assignment #5: Ordered Word List

Due **Thursday, 5/9/19**, 11:59:59 PM

This assignment is strictly for extra credit—you may earn up to 4 points of extra credit on your final average. NO LATE SUBMISSIONS ACCEPTED

1. Introduction

This assignment provides you with an opportunity to work with a linked list, covered in Lectures 30-31. You will design a class, `WordList`, to hold an ordered linked list of words.

2. Deliverables

You should submit three files for this assignment:

- ***prog5_main.cpp:*** Source file containing your main function.
- ***WordList.h / WordList.cpp:*** Header/source files containing the `WordList` class definition and member function definitions.

Submit a single archive file (.zip only) containing all .h and .cpp files to the “Program 5” assignment on Blackboard.

3. Specifications

Input/Output: Your main program should implement a string-based command interface that responds to the following commands:

- **add:** Prompts the user to enter a word and stores that word in the list
- **delete:** Prompts the user to enter a word and removes that word from the list, if possible
- **print:** Prints the entire list in alphabetical order
- **first:** Prints only the first word (in alphabetical order) in the list
- **find:** Prompts the user to enter a word and tests to see if that word is in the list, printing an appropriate message whether it is found or not
- **exit:** End program and exit

4. Hints

A few notes on your list implementation:

- The word list should be implemented as a linked list. We went through some of the details in Lectures 30-31, and all linked data structures (like the stack and queue implementations we discussed) are very similar, but you've got to provide some slightly different functionality in each.
- Make sure your implementation includes a destructor that deletes all nodes.
- While your search function can involve a simple traversal that stops if you find a match, a more efficient version will stop as soon as you reach a point in the list where you can't possibly find the word you're looking for.

5. Test Cases

Your output should closely match these test cases exactly for the given input values, at least in terms of format and general functionality. I will use these test cases in grading of your assignments, but will also generate additional cases that will not be publicly available. Note that these test cases may not cover all possible program outcomes. You should create your own tests to help debug your code and ensure proper operation for all possible inputs.

In the test cases below, user input is underlined.

TEST CASES TO BE ADDED LATER