EECE.3220: Data Structures

Spring 2017

Programming Assignment #6: Data Structures to Improve Search Times

Due **Wednesday, 5/3/17**, 11:59:59 PM

This assignment is strictly for extra credit

1. Introduction

This assignment provides you with an opportunity to work with one of the data structures we discussed in the last month of the semester: binary search trees, heaps, priority queues, and hash tables. You will design a class to hold a list of words in one of these data structures. Your `WordList` class should implement all appropriate operations for the data structure you use.

2. Deliverables

You should submit three files for this assignment:

- ***prog6_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 all three files by uploading them to your Dropbox folder. Ensure your file names match the names specified above. Do not place these files in a subdirectory—place them directly in your shared folder. Failure to meet this specification will reduce your grade, as described in the program grading guidelines.

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, preferably 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

Data structure-dependent hints:

- For data structures with statically allocated space, assume the maximum word list length is 20.
- Any data structure that dynamically allocates its space must include a destructor that deletes all dynamically allocated data.
- In a recursive structure like the binary search tree, recursive operations (for example, adding a new node) are best implemented with a recursive helper function that's first called by the public, non-recursive member function.
- Some of the data structures you can use make it much easier to track the order of words in the list and therefore print the list in order.

ADDITIONAL HINTS TO BE ADDED LATER

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