ICS 46
Homework Assignment 3

Programming Assignment: (Important differences from HW2 are highlighted in RED)
(40 points - 20 points each) Note this part is different than you did in HW2. Implement abstract base class SortedList which implements a sorted list of string objects. It should have pure virtual functions insert(), find(), and remove(). Methods will be void insert(string word), bool find(string word), void remove(string word). You should also write isEmpty and isFull as well as the contructor and destructor for each class.

Allow duplicate entries in your SortedLists, but our tests will not test this because each word is in the input file only once.

You will implement SortedList two ways (two more classes, each derived from SortedList): 1) as SortedArrayList (with three data members **buf**, **capacity**, and **size**) and 2) as SortedLinkedList (with one data member **head**). DO NOT ADD ANY OTHER DATA MEMBERS. Note, you must use binary search for SortedArrayList::find(). Use short-circuited sequential search for SortedLinkedList::find().

(40 points – 20 each) (This part is also different from HW2 in that you are reducing the 6 test functions down to three that are shared for both implementations of SortedList.) Test both of your SortedList implementations by inserting all 45,392 words from the file random.txt (at the link below) then looking up all words from random.txt (they will all be there because you just inserted them all). Pass 45,500 as the initial size of your SortedArrayList so it is plenty large to handle all these words without need for growing. Instrument your program using the provided Timer class in Timer.h (at the link below) so that it measures each of the following times, for each of your two implementations of SortedList: 1) insert all words (write a SINGLE plain function (not a method) called insert_all_words that takes a referece to a SortedList), then 2) find all words (write a SINGLE plain function called find_all_words), and finally, 3) remove all the words (write a SINGLE plain function (not a method) called remove_all_words). Note, these will each make a pass over the input file to get the word to insert, find, or remove. Note, you can pass in either a SortedLinkedList or a SortedArrayList to each of these three functions since they operate on any kind of SortedList. DO NOT TAKE SHORTCUTS FOR remove_all_words - ever in this class. You must remove them one at a time by calling remove(word) for each of the 45,392 words. Be sure to write the time complexity next to every method and every function in your program.

You can find random.txt (shuffled) and words.txt (sorted) at this URL. You can also find a timer class Timer.h which includes a sample of how to use it at the bottom.

Output of your program will be three times for each SortedList: insertAllWords time, findAllWordsTime, and removeAllWords time. Because there are two implementations of SortedList (SortedArrayList and SortedLinkedList), you will print 6 times.

(20 points - minus 5 points for each one missing or incorrect) Write the time complexity next to each method in SortedArrayList, and SortedLinkedList, and your testing functions (e.g., insertAllWords, findAllWords, etc).

There is another file, words.txt, which is the file in sorted order, but I used a different program to sort it. Use operator < for std::string to order the words. Don't worry if your order is a little different due to capitalization or punctuation.

I suggest you create a smaller random.txt for initial testing. Once your program seems to be working correctly, you can run in on the full random.txt file.

- Submit two things: 1) a single report in .pdf format that shows your program works correctly when run under valgrind. Your report must follow the specified format, then submitted to GradeScope. 2) You must also submit a zip of your program which includes Makefile that builds your program called testSortedList by compiling testSortedList.cpp with the required warning flags.
- Copy/Paste your code for the functions (with their time complexity) related to insertAllWords for each of the two different implementations of SortedList, (should be about 6 functions), e.g.,
    - SortedArrayList::insert(string word); // O(N^2)
    - binary_search(); // O(?)
    - copy_down(); // O(?)
    - insertAllWords(SortedArrayList & L); // O(?)
    - SortedLinkedList::insert(); // O(?)
    - insertAllWords(SortedLinkedList & L); // O(?)

Homeworks that are not formatted nicely will not be accepted. Pay careful attention to memory management. Do not share memory across objects. Obvious code copying, from a textbook, from another student, or within your own program, will be down-graded (write everything once (perhaps by writing auxillary functions), then call the function or method each time that operation is required). Feel free to add additional functions or methods, but please do not add any more data members to class MyString (such as an int to store the logical length of the character string).