CSC 228-01 Data Structures and Algorithms, Spring 2020
Instructor: Dr. Natarajan Meghanathan

Assignment 2 (Programming): Design and Implementation of a Sorted List ADT

Due by: Feb. 11th, 11.59 PM

In this programming assignment, you will implement a type of List ADT called the Sorted List. A Sorted List will store its elements in the sorted order. That means, every element that is inserted into the list should be inserted at an appropriate location such that the list (which was sorted before the insertion) will also stay sorted after the insertion. Each insertion operation should take at most linear time (i.e., O(*n*) time, where *n* is the number of elements already in the list).

You are given the code for implementing a Sorted List using both a dynamic array and a singly linked list. The main functions in both these code files are already written for you to create a list, generate random numbers and insert them to the list as well as the timers are setup to measure the time to insert. As you can notice in the 'for' loop of the main function, the insertSortedOrder(...) function is called on to insert every random integer in the sorted order in the list (i.e., the list should remain sorted after each insertion).

Following is an example illustration:

	Contents of the SortedList after the Insertion
Initialization	{ } // empty
Insert 10	10
Insert 3	3 10
Insert 5	3 5 10
Insert 12	3 5 10 12
Insert 7	3 5 7 10 12
Insert 2	2 3 5 7 10 12
Insert 8	2 3 5 7 8 10 12

You would test your code with list size value of 10000 and the maximum value of 50000 for any integer in the list. The code will print the average time per insertion in nano seconds.

Submission:
Items 1-5 in a single word or PDF document
Items 6 and 7, each submitted as separate .cpp files

1 - 15 pts) Pseudo code of your algorithm to insert an element to a dynamic array-based SortedList.
2 - 5 pts) Analysis/Explanation of the time complexity of your algorithm (1).
3 - 15 pts) Pseudo code of your algorithm to insert an element to a singly linked list-based SortedList.
4 - 5 pts) Analysis/Explanation of the time complexity of your algorithm (2).
5 - 5 pts) Screenshots of the outputs showing the average time per insertion for both the implementations.
6 - 30 pts) The entire .cpp file for the dynamic array-based SortedList implementation.
7 - 25 pts) The entire .cpp file for the singly linked list-based SortedList implementation.