CS5403 Data Structures and Algorithms

Assignment 6

Assigned date: 11/19/2015

Due date: 12/02/2015

Please include your name in your source file and the answer document.

Do not use a separate header file, application file and implementation file. Instead, put all your code in a single file called hw6.cpp and submit it via the Blackboard website.

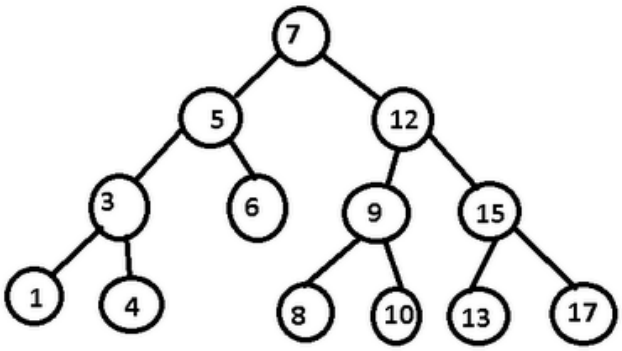
This assignment has two parts: Coding and Questions.
Include your answers to questions in a separate document and submit via blackboard.

Please include the return 0; statement in main() in this assignment and all future assignments.

Do not use global variables.
Do not overload operators.
Do not use friend functions.
Do not use STL.
Do not use the <vector> library.

Problem:

In this assignment you will see how a Binary Search Tree actually speeds up search operation. Binary search tree is a binary tree where the left child is smaller than the parent and right child is larger than the parent. For instance:



For this assignment you will implement a Binary Search Tree, which supports Insert and Search operations. Both insert and search should be recursive functions. Your tree should be designed to store integer values. Your search operation should start from the root and search the tree by either hopping to right child or left child at each node. Your search operation should keep track of how many nodes are visited during the search and report this number along with the information that whether the element is found or not.

For comparison, implement the same insert and search functionality using a simple linked-list. Your search here should report how many nodes are visited as well. We will compare these numbers for BST and Linked-Lists.

Your main function should prompt list of integers from the user. User provides this list in one line. You should store these integers both in your BST and Linked-List. Then your program should ask for an integer to search. After user inputs an integer, your program should search for the integer both in BST and Linked-List and report how many nodes are visited in each case. For instance, a sample interaction would be:

>> Please enter list of integers:
>> 3 2 6 4 12
>> Please enter an integer to search:
>> 12
>> 12 is found. BST visited 3 nodes. Linked-List Visited 5 nodes.

Questions:

After writing your program, do the following experiments and answer the questions:

1. Input 1024 distinct random integers into your program. (You don't need to type in 1024 integers. You can do this within your main function without any user interaction). And then search for an integer, which is not one of these 1024 integers. (i.e. your search function will not find the element). How many nodes does BST search visit? How many nodes does Linked-List visit? Repeat

this experiment 10 times, each time with a different 1024 random integers. Report your results in a table. On average BST or Linked-List visits fewer nodes?

2. Repeat the same experiment in part 1, but this time sort the 1024 random integers before inserting into BST and Linked-List. You can use std::sort() function for sorting. Report your results in a table. What difference do you observe in your results as opposed to your results in part 1? Why do you think this is happening? Briefly explain.