CS 218 DATA STRUCTURES
ASSIGNMENT 3
SECTION A, B, C and D
Fall 2019

DUE: 24th October 2019

NOTE: Late submissions will not be accepted

TO SUBMIT: Documented and well written structured code in C++ on classroom. Undocumented code will be assigned a zero. The name of your file should be your NUCES roll number. If you have more than one files then zip them into a single file and then submit.

PROBLEM BACKGROUND – SORTED STACKLESS BST

Binary Search Trees are mainly used to store ordered data but to access the data in sorted order one would need stack. In order to access the data in sorted order without stack we use an additional data member **nextInOrder** in the node (a pointer to node for the successor or predecessor). We may want to access the data in ascending order or in descending order. So the **nextInOrder** will either point to the successor node or predecessor node. A Boolean variable **IsSuccessor** will be added in the class BST. If **IsSuccessor** is true then all the **nextInOrder** pointers will be pointing towards their successor nodes and maximum data node's pointer will be nullptr. Otherwise these pointers will point to their predecessor nodes and minimum data node's pointer will be nullptr.

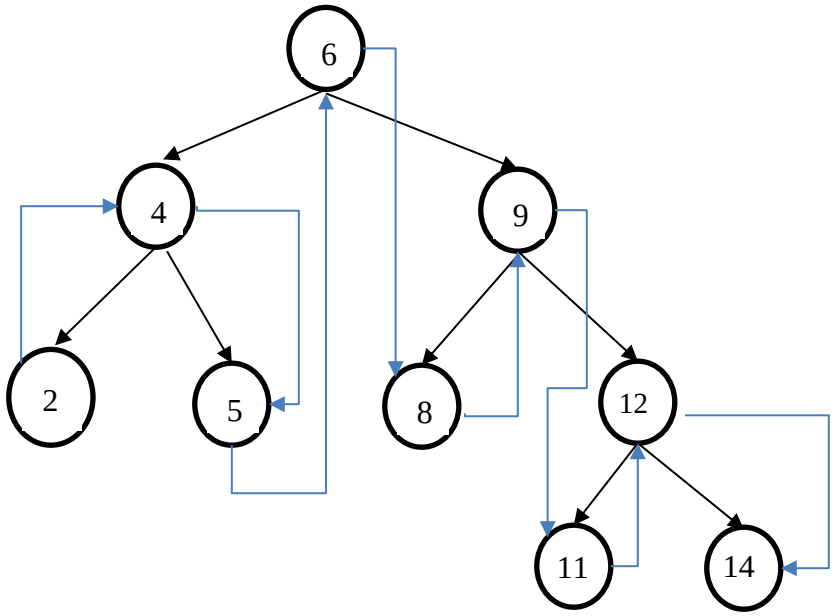

Figure1: Sorted Stackless BST with successor pointers

ASSIGNMENT DETAILS

In this assignment, your task is to implement template class of node and SortedStacklessBST. Implement the following functions in SortedStacklessBST:

Constructor: The default constructor must initialize the root pointer to nullptr and IsSuccessor to true.

Insert: Takes a parameter **d** to be inserted in sorted stackless BST. If d is already present then return false. Otherwise insert d as anew node in the tree, set all three pointers (left, right and nextInOrder) and return true. The nextInOrder pointer will point to its successor node if the IsSuccessor is true. Otherwise it will point to its predecessor node. Implement this function without recursion. Below is an example of insertion of data 3 in Figure1.

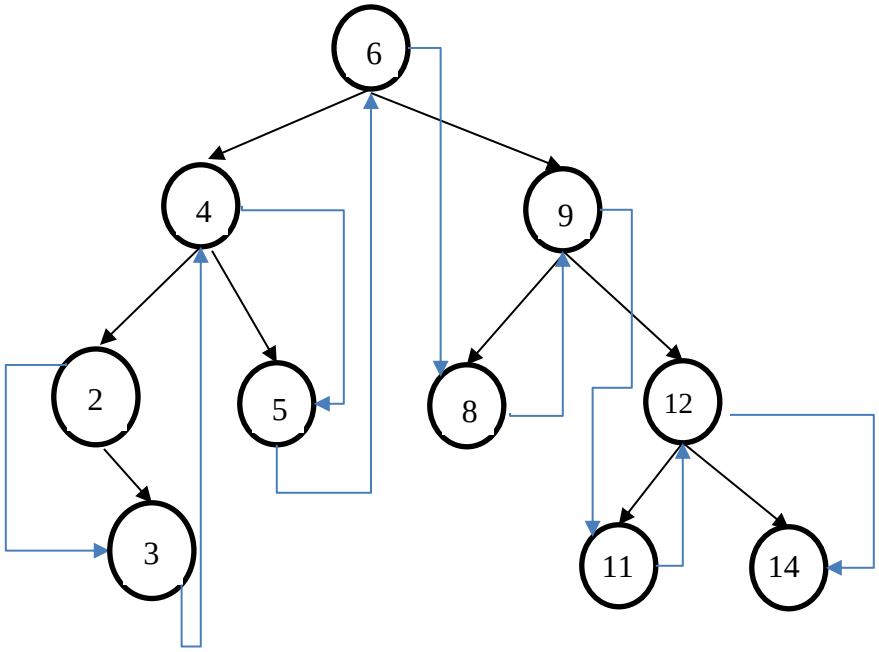

Figure2: Modified tree in Figure1 after insertion of 3

Remove: Takes a parameter **d** to be removed from sorted stackless BST. If d is not present return false. Otherwise remove d from the tree and return true. All rules of BST deletion will apply. You have to adjust the nextInOrder as well. Below is an example of deletion of data 12 in Figure2. Implement this function using recursion.

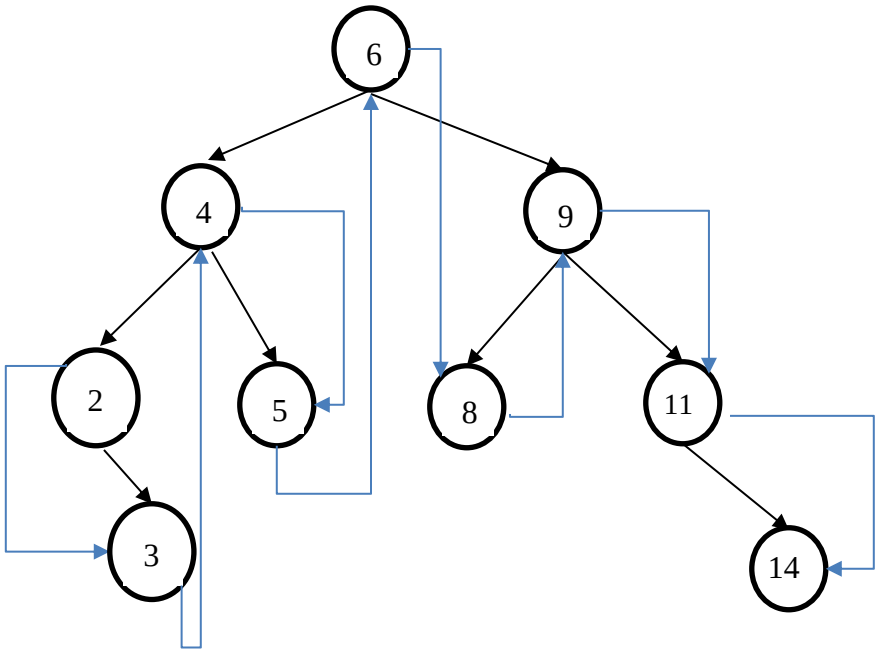

Figure3: Updated tree after removal of 12 in Figure2

Search: Takes a parameter d and returns true if d is available in the tree and false otherwise. Implement this function without recursion.
Print: If IsSuccessor is true, print all the data items in the tree in ascending order without using stack or recursion. Otherwise print data in descending order.
Print(T Low, T High): Takes two parameters Low and High and print all the data items in the range of Low and High in sorted order (Ascending or Descending depending on the value of IsSuccessor) without using recursion or stack.
IsLeafBalance: returns true if all the leaf nodes are on the same level and false otherwise. Use recursion to implement this function.
ReverseOder: This function should toggle the value of IsSuccessor (if successor is true make it false and vice versa) and change the direction of nextInOrder pointers in all the nodes. Use recursion to implement this function. Below is an example of the tree shown in Figure1 after call to ReverseOrder.

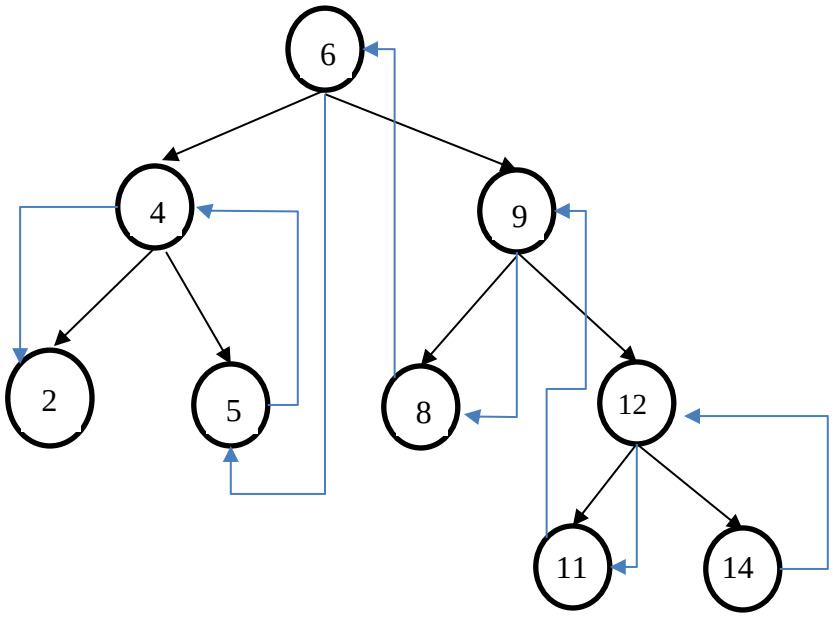

Figure4: Updated tree of Figure1 after call to ReverseOrder

Destructor: It should delete all the nodes in the tree without using recursion/stack.
Copy Constructor: Performs deep copy of the tree recursively.
Overloaded assignment operator: Performs deep copy of the tree recursively.
VERY IMPORTANT

- All the operations in your assignment must be **efficient** solutions. Less credit will be awarded to less efficient implementations.
- Academic integrity is expected of all the students. Plagiarism or cheating in any assessment will result in negative marking or an **F** grade in the course, and possibly more severe penalties.