HW-3 CSL2020 (Dec 2020 - Jan 2021)

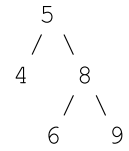
Assigned: 28 Dec 2020 Submission deadline: 05 Jan 2021 (Tuesday) 2 pm Marks: 6

Submit only one file with name: hw3_roll_number.c

A binary search tree (BST) is a special kind of binary tree, in which the following is true:

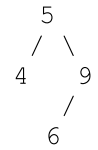
For each node, data at the left node is smaller than the data at the right node. (Recall that a node contains a data field, and two pointers).

Assume that we construct a BST with values 5,8,9,4,6 in this order. Then the resultant tree will look like



On the other hand, if the input data sequence was 8,9,5 (or 8,5,9) then a balanced binary tree with height 1 will be obtained. (We always start with the root and go left or right, recursively, depending on the input value being smaller or larger).

To delete a node from the binary tree, one needs to readjust the other nodes so that the BST property still holds. For example, to delete 9 from the above tree, simply delete the node containing value 9. However, to delete 8, we will need to move 9 up. The resultant tree will look like:



1. Write a C program to do the following: (Maintain a global pointer to the root of the BST. Display a menu after each operation. And depending on user input, call appropriate function with inputs provided by the user. Assume that all the values in the BST are distinct and positive).

(i) On menu input 1, call a function to insert data into a BST (which is initially empty). The input will be given as a sequence of positive integers. The length of the sequence is not pre-specified, and hence you should allocate the necessary memory dynamically. Ex. We can say, 2,4,3,7,5,1 and the appropriate BST with 6 nodes should be constructed.

(ii) On menu input 2, call a function to delete a node from the BST. The node value will be input by the user. (If there is no node with such a value then print 'error'). Ex. In the BST constructed by you in step (i), we may ask you to delete 4 (i.e. the node containing the value 4). At one time, we will ask you to

delete only one node, but we can call the same function multiple times in succession. You should take care of the BST being empty when node deletion function is called. (Take care to handle deletion of an internal node with one child or with two children; a leaf; or the root itself).

(iii) On menu input 3, display the number of leaves, the number of internal nodes, and the height of the tree. (Assume that the root is at height 0. That is, the maximum level number and the height are equal.)

(iv) On menu input 4, display the in- order traversal of the tree. (Non-recursive code. No marks for the recursive version. However, you should verify your code by checking that the output matches with the recursive version in every case.)

(v) On menu input 5, display the pre-order traversal of the tree. (Non-recursive code. Same comment as in (iv) above.)

(vi) On menu input 5, display the post-order traversal of the tree. (Non-recursive code. Same comment as in (iv) above.)