Problem 6: Problems on Binary Trees

Data Structures Lab (CS111)

In this practice problem, you need to write C programs to address the following problems on binary trees:

1. User provides *postorder* and *inorder* traversals of a binary tree. Construct the binary tree from these given traversal orders. Assume that the keys stored in each node of a tree is an integer. Keep one more field *streeSum* to store subtree sum in each tree node a where subtree sum of a means the sum of the keys stored in the nodes present in the subtree rooted at node a. Your program should populate this *streeSum* field on each node. **[5 marks]**

2. Next, you need to write *insertChild* function. The user provides two integer as key values *key1* and *key2*. If *key1* is not present in the binary tree constructed in the problem 1, you need to insert a new node with key value *key1* as the child of the node with key value *key2*. If *key2* is not present in the tree, you need to show one error message. If the node with key value *key2* is present in the tree, you need to insert the new node into the left subtree of that node when subtree sum of left subtree is less than or equal to sub tree sum of the right subtree. Insert the new node into the right subtree otherwise. Update the *streeSum* field accordingly. **[5 marks]**

3. Write *deleteNode* function to delete the node with the key value given by the user. While deleting the node you need to find the leaf node with the maximum key value in the left subtree of the specified node, exchange the key values and delete the leaf node when subtree sum of the left subtree is greater than the right subtree. You need to do the reverse otherwise. You need to again update the *streeSum* field accordingly. **[3 marks]**

4. Consider the scenario given in problem 2 again. If there is no node with *key2* in the tree constructed in problem 1, then create a separate tree with two nodes, where node with *key2* is the root node. Now keep on inserting nodes in this 2nd tree for all the insertion requests where *key2* is not present in the tree. While inserting nodes make sure that this second tree is an almost complete/ complete binary tree always. When this 2nd binary tree becomes a complete binary tree with 7 nodes, insert the whole tree into the subtree of the node for which the difference between *streeSum* of left subtree and *streeSum* of right subtree is minimum. If subtree sum of the right subtree of that particular node is more than the subtree sum of the left subtree, insert into the left subtree. Insert into the right subtree otherwise. **[7 marks]**