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 *preorder* and *inorder* traversals of a binary tree. Construct the binary tree from these given traversal orders. You can assume the keys stored in the tree are integers. Keep one more field *hd* in each tree node to store the height difference between the left subtree and right subtree of that node. Store it as negative integer if height of right subtree is more than height of the left subtree. **[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* has both the children, you need to insert the new node into the left subtree of that node if height of the left subtree is less than the height of the right subtree and into the right subtree otherwise. **[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 a leaf node in the left subtree of the specified node, exchange the key values and delete the leaf node when height of the left subtree is greater than the right subtree. You need to do the reverse when height of the right subtree is more than the height of the left subtree. **[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 always a almost complete binary tree. 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 $|hd|$ is maximum. If *hd* is positive, insert into the left subtree. Insert into the right subtree otherwise. **[7 marks]**