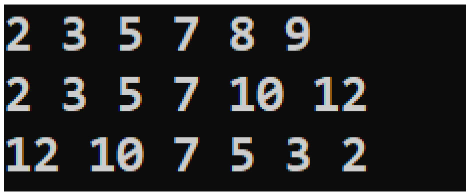
csc 326 lab 8

```
#include <iostream>

using namespace std;

class TreeNode{

public:

   int val;

   TreeNode* left = nullptr;

   TreeNode* right = nullptr;

   TreeNode(){}

   TreeNode(int val): val(val){}

};

// insert in binary search tree

TreeNode * insert(TreeNode * root,int val){

 if(not root)

  return new TreeNode(val);

 if(root->val > val)

  root->left = insert(root->left,val);

 else

  root->right = insert(root->right,val);

 return root;

}

// to build tree

TreeNode *  buildTree(int * arr,int size){

 TreeNode * ans = nullptr;

 for (int i = 0; i < size; ++i)
```

```
  ans = insert(ans,arr[i]);
 return ans;
}
// print the tree
void inorder(TreeNode * root,int level = 0){
 if(not root){
  return;
 }
 inorder(root->left,level + 1);
 cout<<root->val<<" ";
 inorder(root->right,level + 1);
}
// swapSubtrees function
void swapSubtrees(TreeNode * root,int val){
 if(not root) // if root is null
  return;
 // if value found or val is set to flag
 if(root->val == val or val == -999){
  // call for left and write with flag
  swapSubtrees(root->left,-999);
  swapSubtrees(root->right,-999);
  // swap the left and right node
  swap(root->left,root->right);
  return;
 }

 // if not found keep search in left and right

 swapSubtrees(root->left,val);

 swapSubtrees(root->right,val);

}

int main(int argc, char const *argv[])

{

 // check build tree method

 int arr[] = {5,2,7,3,9,8};

 TreeNode * root = buildTree(arr,6);

 inorder(root);

 cout<<"\n";

 // create tree

 /*

     7

      /  \

     /    \

    3      10

      / \      \

     /   \      \

    2     5     12

 */
 TreeNode * root1 = new TreeNode(7);
 root1->left = new TreeNode(3);
 root1->left->left = new TreeNode(2);
 root1->left->right = new TreeNode(5);
 root1->right = new TreeNode(10);
 root1->right->right = new TreeNode(12);
 inorder(root1);
 cout<<"\n";
 // swap the all subtree of root
 swapSubtrees(root1,7);
 inorder(root1);
 return 0;
}
```

output



```
2 3 5 7 8 9
2 3 5 7 10 12
12 10 7 5 3 2
```