7. Write C++ programs that use recursive functions to traverse the given binary tree in

a) Preorder

b) in order

c) post order.

```
#include <iostream>
using namespace std;

// Definition for a binary tree node
struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;

    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
};
void preorderTraversal(TreeNode* node) {
    if (node == nullptr)
        return;

    // Print the root value
    cout << node->val << " ";

    // Traverse the left subtree
    preorderTraversal(node->left);

    // Traverse the right subtree
    preorderTraversal(node->right);
}
void inorderTraversal(TreeNode* node) {
    if (node == nullptr)
```

```
        return;

    // Traverse the left subtree
    inorderTraversal(node->left);

    // Print the root value
    cout << node->val << " ";

    // Traverse the right subtree
    inorderTraversal(node->right);
}
void postorderTraversal(TreeNode* node) {
    if (node == nullptr)
        return;

    // Traverse the left subtree
    postorderTraversal(node->left);

    // Traverse the right subtree
    postorderTraversal(node->right);

    // Print the root value
    cout << node->val << " ";
}
int main() {
    // Create a sample binary tree
    TreeNode* root = new TreeNode(1);
    root->left = new TreeNode(2);
    root->right = new TreeNode(3);
    root->left->left = new TreeNode(4);
    root->left->right = new TreeNode(5);
```

```
    root->right->left = new TreeNode(6);
    root->right->right = new TreeNode(7);

    // Preorder traversal
    cout << "Preorder Traversal: ";
    preorderTraversal(root);
    cout << endl;

    // Inorder traversal
    cout << "Inorder Traversal: ";
    inorderTraversal(root);
    cout << endl;

    // Postorder traversal
    cout << "Postorder Traversal: ";
    postorderTraversal(root);
    cout << endl;

    return 0;
}
```