5. Write a C++ program to perform the following operations:

a) Insert an element into a binary search tree.

b) Delete an element from a binary search tree. Give code in short

Program:

```
#include <iostream>
using namespace std;
// Node structure for BST
struct Node {
    int data;
    Node* left;
    Node* right;

    Node(int value) : data(value), left(nullptr), right(nullptr) {}
};

// Insert an element into BST
Node* insert(Node* root, int value) {
    if (!root) return new Node(value);
    if (value < root->data)
        root->left = insert(root->left, value);
    else
        root->right = insert(root->right, value);
    return root;
}

// Find minimum value node in the right subtree (used in deletion)
Node* findMin(Node* root) {
    while (root && root->left)
```

```
        root = root->left;
    return root;
}

// Delete an element from BST
Node* deleteNode(Node* root, int value) {
    if (!root) return root;
    if (value < root->data)
        root->left = deleteNode(root->left, value);
    else if (value > root->data)
        root->right = deleteNode(root->right, value);
    else { // Node to be deleted found
        if (!root->left) {
            Node* temp = root->right;
            delete root;
            return temp;
        } else if (!root->right) {
            Node* temp = root->left;
            delete root;
            return temp;
        }
        Node* temp = findMin(root->right); // Find min in right subtree
        root->data = temp->data; // Replace with inorder successor
        root->right = deleteNode(root->right, temp->data); // Delete successor
    }
    return root;
}

// Inorder traversal for testing
void inorder(Node* root) {
    if (root) {
```

```
        inorder(root->left);
        cout << root->data << " ";
        inorder(root->right);
    }
}

int main() {
    Node* root = nullptr;
    root = insert(root, 50);
    insert(root, 30);
    insert(root, 70);
    insert(root, 20);
    insert(root, 40);
    insert(root, 60);
    insert(root, 80);

    cout << "Inorder after insertion: ";
    inorder(root);
    cout << endl;
    root = deleteNode(root, 50); // Deleting node 50
    cout << "Inorder after deletion: ";
    inorder(root);
    cout << endl;
    return 0;
}
```