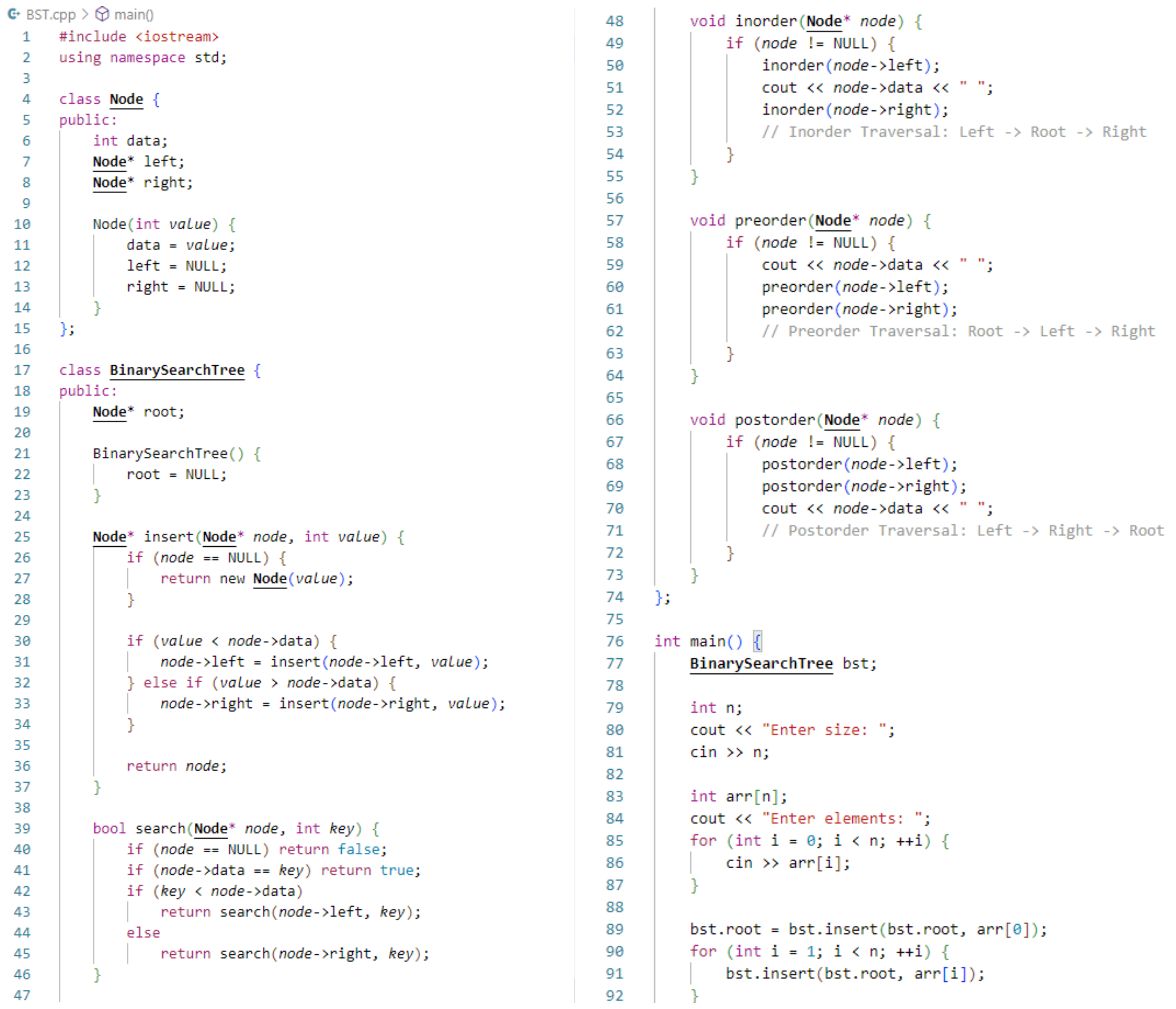
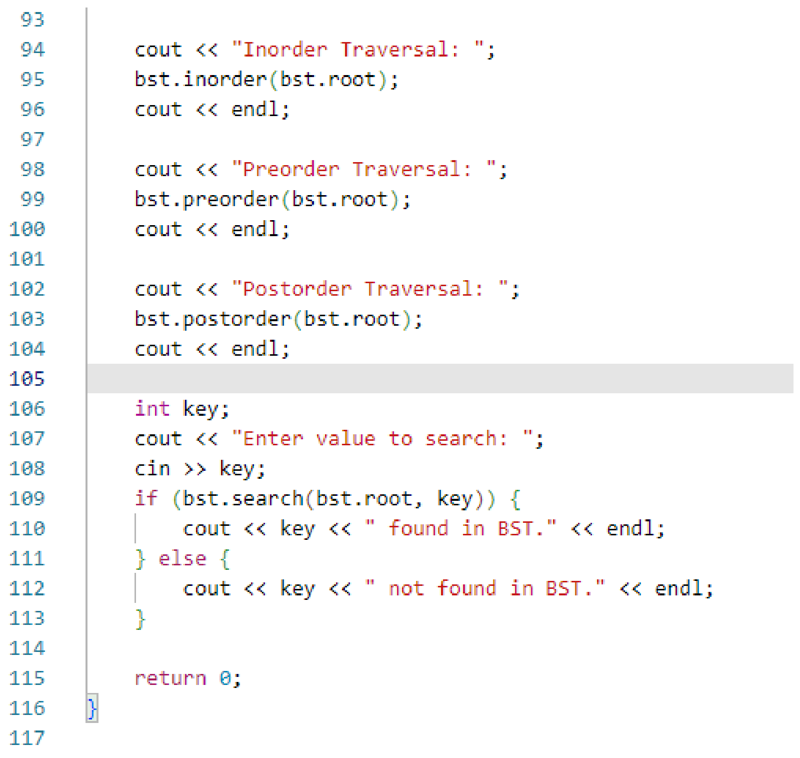
Problem:

Write a C++ program to implement a Binary Search Tree with Insertion, Traversal(In-Order, Pre-Order, Post-Order) and Search operation

Code:



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

class Node {
public:
    int data;
    Node* left;
    Node* right;

    Node(int value) {
        data = value;
        left = NULL;
        right = NULL;
    }
};

class BinarySearchTree {
public:
    Node* root;

    BinarySearchTree() {
        root = NULL;
    }

    Node* insert(Node* node, int value) {
        if (node == NULL) {
            return new Node(value);
        }

        if (value < node->data) {
            node->left = insert(node->left, value);
        } else if (value > node->data) {
            node->right = insert(node->right, value);
        }

        return node;
    }

    bool search(Node* node, int key) {
        if (node == NULL) return false;
        if (node->data == key) return true;
        if (key < node->data)
            return search(node->left, key);
        else
            return search(node->right, key);
    }

    void inorder(Node* node) {
        if (node != NULL) {
            inorder(node->left);
            cout << node->data << " ";
            inorder(node->right);
            // Inorder Traversal: Left -> Root -> Right
        }
    }

    void preorder(Node* node) {
        if (node != NULL) {
            cout << node->data << " ";
            preorder(node->left);
            preorder(node->right);
            // Preorder Traversal: Root -> Left -> Right
        }
    }

    void postorder(Node* node) {
        if (node != NULL) {
            postorder(node->left);
            postorder(node->right);
            cout << node->data << " ";
            // Postorder Traversal: Left -> Right -> Root
        }
    }
};

int main() {
    BinarySearchTree bst;

    int n;
    cout << "Enter size: ";
    cin >> n;

    int arr[n];
    cout << "Enter elements: ";
    for (int i = 0; i < n; ++i) {
        cin >> arr[i];
    }

    bst.root = bst.insert(bst.root, arr[0]);
    for (int i = 1; i < n; ++i) {
        bst.insert(bst.root, arr[i]);
    }
```



```cpp

    cout << "Inorder Traversal: ";
    bst.inorder(bst.root);
    cout << endl;

    cout << "Preorder Traversal: ";
    bst.preorder(bst.root);
    cout << endl;

    cout << "Postorder Traversal: ";
    bst.postorder(bst.root);
    cout << endl;

    int key;
    cout << "Enter value to search: ";
    cin >> key;
    if (bst.search(bst.root, key)) {
        cout << key << " found in BST." << endl;
    } else {
        cout << key << " not found in BST." << endl;
    }

    return 0;
}

```

output"

```
d:\GitHub002\03 Third Semester\CSE 2104_Data Structures Lab\Practice lab\output>.\"BST.exe"
Enter size: 5
Enter elements: 50 60 20 80 10
Inorder Traversal: 10 20 50 60 80
Preorder Traversal: 50 20 10 60 80
Postorder Traversal: 10 20 80 60 50
Enter value to search: 10
10 found in BST.

d:\GitHub002\03 Third Semester\CSE 2104_Data Structures Lab\Practice lab\output>
```