BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
Munshi nagar, Andheri (W) ,Mumbai - 400058
DEPARTMENT OF MASTER OF COMPUTER APPLICATION

**CLASS: F.Y. MCA SEM: I**

**COURSE CODE: MC504 SUBJECT NAME: DATA STRUCTURES**

**ROLL NO. : 2023510001 BATCH: D**

**NAME: Vaibhav Agarwal**

**EXPERIMENT NO: 05**

**EXPERIMENT TITLE:**

**Implement BST with following operations**

1. **search**
2. **insert**
3. **traversals**

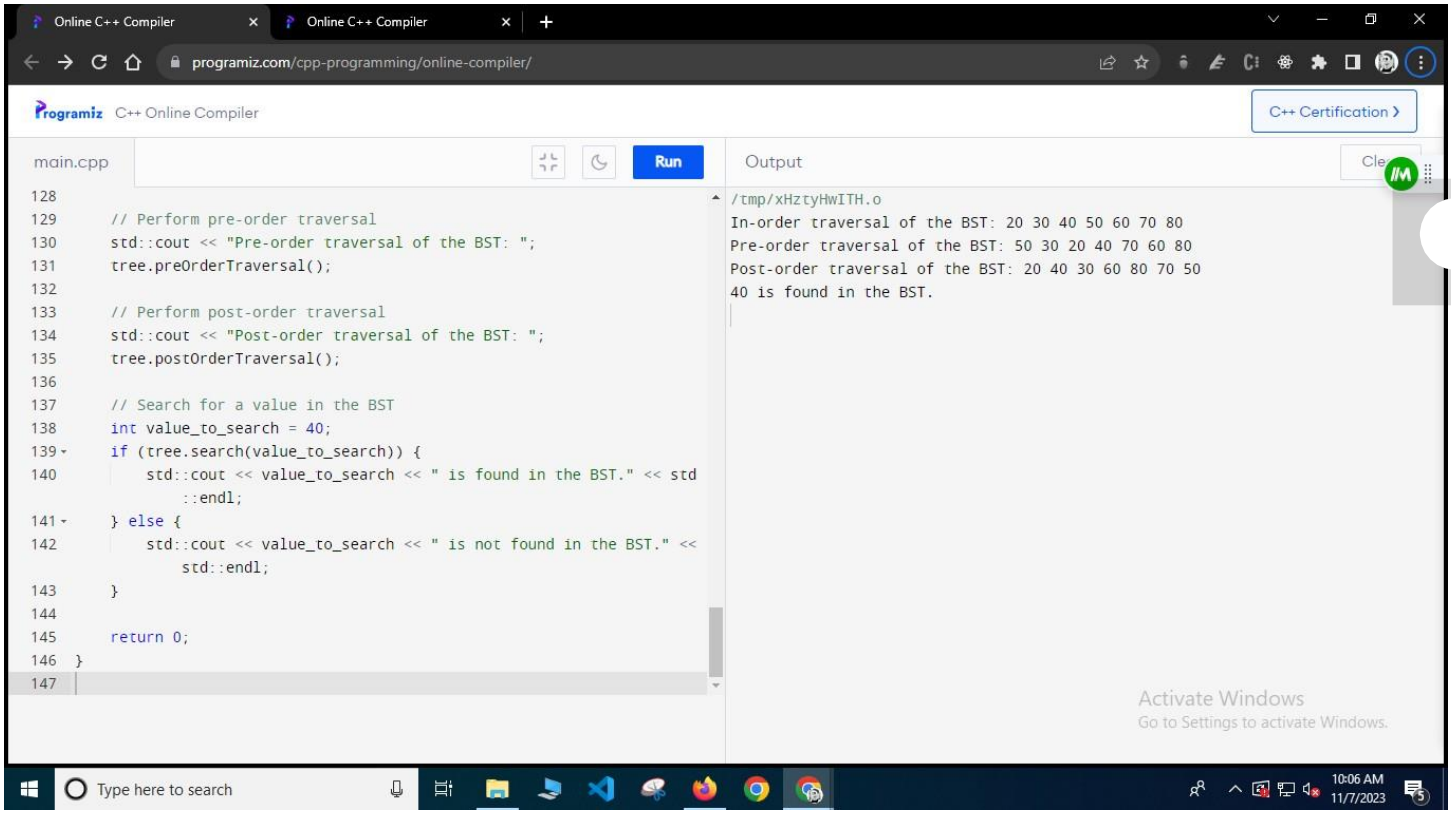
Code<>:

```
#include <iostream>

// Binary Search Tree Node
class Node {
public:
    int data;
    Node* left;
    Node* right;

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

// Binary Search Tree class
class BST {
private:
    Node* root;
```

```
// Helper function to insert a new node into the BSTNode*
insertRecursive(Node* current, int value) {
    if (current == nullptr) {
        return new Node(value);
    }

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

    return current;
}

// Helper function to perform an in-order traversal
void inOrderRecursive(Node* current) {
    if (current != nullptr) {
        inOrderRecursive(current->left);
        std::cout << current->data << " ";
        inOrderRecursive(current->right);
    }
}

// Helper function to perform a pre-order traversal
void preOrderRecursive(Node* current) {
    if (current != nullptr) {
        std::cout << current->data << " ";
        preOrderRecursive(current->left);
        preOrderRecursive(current->right);
    }
}

// Helper function to perform a post-order traversal
void postOrderRecursive(Node* current) {
    if (current != nullptr) {
        postOrderRecursive(current->left);
        postOrderRecursive(current->right);
        std::cout << current->data << " ";
    }
}

// Helper function to search for a value in the BST
```

```
    bool searchRecursive(Node* current, int value) {
        if (current == nullptr) {
            return false;
        }

        if (current->data == value) {
            return true;
        } else if (value < current->data) {
            return searchRecursive(current->left, value);
        } else {
            return searchRecursive(current->right, value);
        }
    }

public:
    BST() {
        root = nullptr;
    }

    // Public function to insert a value into the BST
    void insert(int value) {
        root = insertRecursive(root, value);
    }

    // Public function to perform in-order traversal of the BST
    void inOrderTraversal() {
        inOrderRecursive(root);
        std::cout << std::endl;
    }

    // Public function to perform pre-order traversal of the BST
    void preOrderTraversal() {
        preOrderRecursive(root);
        std::cout << std::endl;
    }

    // Public function to perform post-order traversal of the BST
    void postOrderTraversal() {
        postOrderRecursive(root);
        std::cout << std::endl;
    }

    // Public function to search for a value in the BST
```

```
    bool search(int value) {
        return searchRecursive(root, value);
    }
};

int main() {
    BST tree;

    // Insert values into the BST
    tree.insert(50);
    tree.insert(30);
    tree.insert(70);
    tree.insert(20);
    tree.insert(40);
    tree.insert(60);
    tree.insert(80);

    // Perform in-order traversal
    std::cout << "In-order traversal of the BST: ";
    tree.inOrderTraversal();

    // Perform pre-order traversal
    std::cout << "Pre-order traversal of the BST: ";
    tree.preOrderTraversal();

    // Perform post-order traversal
    std::cout << "Post-order traversal of the BST: ";
    tree.postOrderTraversal();

    // Search for a value in the BST
    int value_to_search = 40;
    if (tree.search(value_to_search)) {
        std::cout << value_to_search << " is found in the BST." << std::endl;
    } else {
        std::cout << value_to_search << " is not found in the BST." << std::endl;
    }

    return 0;
}
```


Online C++ Compiler
Online C++ Compiler
programiz.com/cpp-programming/online-compiler/
Programiz C++ Online Compiler
C++ Certification
main.cpp
Run
Output
Cle
128
129 // Perform pre-order traversal
130 std::cout << "Pre-order traversal of the BST: ";
131 tree.preOrderTraversal();
132
133 // Perform post-order traversal
134 std::cout << "Post-order traversal of the BST: ";
135 tree.postOrderTraversal();
136
137 // Search for a value in the BST
138 int value_to_search = 40;
139 if (tree.search(value_to_search)) {
140 std::cout << value_to_search << " is found in the BST." << std
::endl;
141 } else {
142 std::cout << value_to_search << " is not found in the BST." <<
std::endl;
143 }
144
145 return 0;
146 }
147
/tmp/xHztyHwITH.o
In-order traversal of the BST: 20 30 40 50 60 70 80
Pre-order traversal of the BST: 50 30 20 40 70 60 80
Post-order traversal of the BST: 20 40 30 60 80 70 50
40 is found in the BST.
Activate Windows
Go to Settings to activate Windows.
Type here to search
10:06 AM
11/7/2023