

1. **Problem Name:** Write a C++ program to implement a Binary Search Tree with Insertion, Traversal(In-Order, Pre-Order, Post-Order) and Search operation.

```
Start here X Untitled1.cpp X
1  #include <iostream>
2  using namespace std;
3
4  class Node {
5  public:
6      int data;
7      Node* left;
8      Node* right;
9
10     Node(int value) {
11         data = value;
12         left = right = nullptr;
13     }
14 };
15
16 class BST {
17 public:
18     Node* root;
19
20     BST() {
21         root = nullptr;
22     }
23
24     // Insert function
25     Node* insert(Node* node, int value) {
26         if (node == nullptr) {
27             return new Node(value);
28         }
29
30         if (value < node->data) {
31             node->left = insert(node->left, value);
32         } else if (value > node->data) {
33             node->right = insert(node->right, value);
34         }
35
36         return node;
37     }
38
39     // In-order traversal (LNR)
40     void inOrder(Node* node) {
41         if (node == nullptr) return;
42         inOrder(node->left);
43         cout << node->data << " ";
44         inOrder(node->right);
45     }
46
47     // Pre-order traversal (NLR)
48     void preOrder(Node* node) {
49         if (node == nullptr) return;
50         cout << node->data << " ";
51         preOrder(node->left);
52         preOrder(node->right);
53     }
54
55     // Post-order traversal (LRN)
56     void postOrder(Node* node) {
57         if (node == nullptr) return;
58         postOrder(node->left);
59         postOrder(node->right);
60         cout << node->data << " ";
61     }
62
63     // Search function
64     bool search(Node* node, int key) {
65         if (node == nullptr) return false;
66         if (node->data == key) return true;
67         if (key < node->data) return search(node->left, key);
68         return search(node->right, key);
69     }
70 };
71
```

```

72 int main() {
73     BST tree;
74     int n;
75
76     cout << "Enter number of values to insert: ";
77     cin >> n;
78
79     int* values = new int[n];
80     cout << "Enter the values: ";
81     for (int i = 0; i < n; ++i) {
82         cin >> values[i];
83         tree.root = tree.insert(tree.root, values[i]);
84     }
85
86     cout << "\nIn-order Traversal: ";
87     tree.inOrder(tree.root);
88     cout << "\nPre-order Traversal: ";
89     tree.preOrder(tree.root);
90     cout << "\nPost-order Traversal: ";
91     tree.postOrder(tree.root);
92     cout << endl;
93
94     int key;
95     cout << "\nEnter value to search: ";
96     cin >> key;
97
98     if (tree.search(tree.root, key)) {
99         cout << "Found" << endl;
100     } else {
101         cout << "Not Found" << endl;
102     }
103
104     delete[] values;
105     return 0;
106 }
107

```

## Output:

```

E:\@CODES\Academic\DSA\I >
Enter number of values to insert: 5
Enter the values: 2 4 6 8 10

In-order Traversal: 2 4 6 8 10
Pre-order Traversal: 2 4 6 8 10
Post-order Traversal: 10 8 6 4 2

Enter value to search: 5
Not Found

Process returned 0 (0x0)   execution time : 12.698 s
Press any key to continue.

```