Binary Search Tree Insertion

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

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

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

Node* insert(Node* root, int key) {
    if (root == NULL) {
        return new Node(key);
    }
    if (key < root->key) {
        root->left = insert(root->left, key);
    } else {
        root->right = insert(root->right, key);
    }
    return root;
}

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

```
        return search(root->right, key);
    }
}

void inorder(Node* root) {
    if (root == NULL) {
        return;
    }
    inorder(root->left);
    cout << root->key << " ";
    inorder(root->right);
}

int Min(Node* root) {
    if (root == NULL) {
        cout << "BST is empty";
        return -1;
    }
    while (root->left != NULL) {
        root = root->left;
    }
    return root->key;
}

int Max(Node* root) {
    if (root == NULL) {
        cout << "BST is empty";
        return -1;
    }
    while (root->right != NULL) {
        root = root->right;
    }
    return root->key;
}

int main() {
    Node* root = NULL;
    root = insert(root, 50);
    insert(root, 30);
    insert(root, 70);
```

```
    insert(root, 20);
    insert(root, 40);
    insert(root, 60);
    insert(root, 80);

    cout << "Binary Search Tree: ";
    inorder(root);
    cout << endl;

    int val = 90;
    if (search(root, val)) {
        cout << val << " found in the BST" << endl;
    } else {
        cout << val << " not found in the BST" << endl;
    }

    cout << "Minimum value in BST: " << Min(root) << endl;
    cout << "Maximum value in BST: " << Max(root) << endl;

    return 0;
}
```

Output-

Binary Search Tree: 20 30 40 50 60 70 80
90 not found in the BST
Minimum value in BST: 20
Maximum value in BST: 80