```
#include <stdio.h>

#include <stdlib.h>


// Define the structure of the BST node

struct Node {

   int data;

   struct Node* left;

   struct Node* right;

};


// Function to create a new node

struct Node* createNode(int value) {

   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

   newNode->data = value;

   newNode->left = newNode->right = NULL;

   return newNode;

}


// Function to insert a new node in BST

struct Node* insert(struct Node* root, int value) {

   if (root == NULL) {

      return createNode(value);

   }

   if (value < root->data) {

      root->left = insert(root->left, value);
```

```
    } else if (value > root->data) {

        root->right = insert(root->right, value);

    }

    return root;

}


// Function for inorder traversal (Left-Root-Right)

void inorderTraversal(struct Node* root) {

    if (root != NULL) {

        inorderTraversal(root->left);

        printf("%d ", root->data);

        inorderTraversal(root->right);

    }

}


// Function to search for a value in the BST and print its location

int search(struct Node* root, int value) {

    if (root == NULL) {

        return 0; // Value not found

    }

    if (root->data == value) {

        printf("%d is found at the root.\n", value);

        return 1; // Value found at root

    }

    if (value < root->data) {

        if (search(root->left, value)) {

            printf("%d is found in the left subtree of %d.\n", value, root->data);
            return 1;

        }

    } else {

        if (search(root->right, value)) {

            printf("%d is found in the right subtree of %d.\n", value, root->data);

            return 1;

        }

    }

    return 0; // Value not found

}


int main() {

    struct Node* root = NULL;

    int values[] = {50, 30, 70, 20, 40, 60, 80};

    int n = sizeof(values) / sizeof(values[0]);


    // Inserting values into BST

    for (int i = 0; i < n; i++) {

        root = insert(root, values[i]);

    }


    // Display the inorder traversal of the BST

    printf("Inorder Traversal of BST: ");

    inorderTraversal(root);

    printf("\n");


    // Search for a value in the BST
    int searchValue;

    printf("Enter a value to search in the BST: ");

    scanf("%d", &searchValue);



    if (!search(root, searchValue)) {

        printf("%d is not found in the BST.\n", searchValue);

    }



    return 0;

}
```