```c
#include <stdio.h>
#include <stdlib.h>

struct node {
    int key;
    struct node *left, *right;
};

// A utility function to create a new BST node
struct node* newNode(int item)
{
    struct node* temp
        = (struct node*)malloc(sizeof(struct node));
    temp->key = item;
    temp->left = temp->right = NULL;
    return temp;
}

// A utility function to insert
// a new node with given key in BST
struct node* insert(struct node* node, int key)
{
    // If the tree is empty, return a new node
    if (node == NULL)
        return newNode(key);

    // Otherwise, recur down the tree
    if (key < node->key)
        node->left = insert(node->left, key);
    else if (key > node->key)
        node->right = insert(node->right, key);

    // Return the (unchanged) node pointer
    return node;
}

// Utility function to search a key in a BST
struct node* search(struct node* root, int key)
{
    // Base Cases: root is null or key is present at root
    if (root == NULL || root->key == key)
        return root;

    // Key is greater than root's key
    if (root->key < key)
        return search(root->right, key);

    // Key is smaller than root's key
    return search(root->left, key);
}

// Driver Code
int main()
{
    struct node* root = NULL;
    int key;
    root = insert(root, 50);
    insert(root, 30);
    insert(root, 20);
    insert(root, 40);
    insert(root, 70);
    insert(root, 60);
    insert(root, 80);
```

```
    printf("\n Enter the number which you want to search : ");
    scanf("%d",&key);

    // Searching in a BST
    if (search(root, key) == NULL)
        printf("%d is not found in Tree.\n", key);
    else
        printf("%d is found in Tree.\n", key);

        return 0;
}
```