

# Rajalakshmi Engineering College

Name: Rayvan Sanjai  
Email: 240701425@rajalakshmi.edu.in  
Roll no: 2116240701425  
Phone: 9380572043  
Branch: REC  
Department: I CSE FD  
Batch: 2028  
Degree: B.E - CSE

Scan to verify results



## NeoColab\_REC\_CS23231\_DATA STRUCTURES

### REC\_DS using C\_Week 5\_COD\_Question 3

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

You are required to implement basic operations on a Binary Search Tree (BST), like insertion and searching.

Insertion: Given a list of integers, construct a Binary Search Tree by repeatedly inserting each integer into the tree according to the rules of a BST.

Searching: Given an integer, search for its presence in the constructed Binary Search Tree. Print whether the integer is found or not.

Write a program to calculate this efficiently.

##### ***Input Format***

The first line of input consists of an integer n, representing the number of nodes

in the binary search tree.

The second line consists of the values of the nodes, separated by space as integers.

The third line consists of an integer representing, the value that is to be searched.

### **Output Format**

The output prints, "Value <value> is found in the tree." if the given value is present, otherwise it prints: "Value <value> is not found in the tree."

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 7

8 3 10 1 6 14 23

6

Output: Value 6 is found in the tree.

### **Answer**

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int key;
    struct Node *left, *right;
} Node;
Node* createNode(int key) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->key = key;
    newNode->left = newNode->right = NULL;
    return newNode;
}
Node* insert(Node* root, int key) {
    if (root == NULL) return createNode(key);

    if (key < root->key)
        root->left = insert(root->left, key);
    else if (key > root->key)
        root->right = insert(root->right, key);
```

```

        return root;
    }
    int search(Node* root, int key) {
        if (root == NULL) return 0;
        if (root->key == key) return 1;

        return (key < root->key) ? search(root->left, key) : search(root->right, key);
    }
    void freeBST(Node* root) {
        if (root) {
            freeBST(root->left);
            freeBST(root->right);
            free(root);
        }
    }
    int main() {
        int n, value, key;
        scanf("%d", &n);
        Node* root = NULL;
        for (int i = 0; i < n; i++) {
            scanf("%d", &value);
            root = insert(root, value);
        }
        scanf("%d", &key);
        if (search(root, key))
            printf("Value %d is found in the tree.\n", key);
        else
            printf("Value %d is not found in the tree.\n", key);
        freeBST(root);
        return 0;
    }

```

**Status :** Correct

**Marks :** 10/10