Rajalakshmi Engineering College

Name: srihari krishnakumar
Email: 241501093@rajalakshmi.edu.in
Roll no: 241501093
Phone: 7200067930
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

// Define structure for a BST Node
struct TreeNode {
    int data;
    struct TreeNode* left;
    struct TreeNode* right;
};

// Function to create a new node
struct TreeNode* createNode(int value) {
    struct TreeNode* newNode = (struct TreeNode*) malloc(sizeof(struct
TreeNode));
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}
```

```
// Function to insert a node into BST
struct TreeNode* insert(struct TreeNode* root, int value) {
    if (root == NULL) {
        return createNode(value); // If tree is empty, create root
    }

    if (value < root->data) {
        root->left = insert(root->left, value); // Go left
    } else if (value > root->data) {
        root->right = insert(root->right, value); // Go right
    }
    // If value is equal, do nothing (no duplicates in this case)

    return root;
}

// Function to search for a value in BST
int search(struct TreeNode* root, int key) {
    if (root == NULL) return 0;

    if (key == root->data) return 1;
    else if (key < root->data) return search(root->left, key);
    else return search(root->right, key);
}

int main() {
    int n, value, key;
    scanf("%d", &n);

    struct TreeNode* 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);
    }

    return 0;
}
```

Status : Correct **Marks : 10/10**