Rajalakshmi Engineering College

Name: Jhanani shree
Email: 240701215@rajalakshmi.edu.in
Roll no: 240701215
Phone: 7373333511
Branch: REC
Department: I CSE AH
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

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

// Define the structure for a tree node
struct Node {
    int value;
    struct Node* left;
    struct Node* right;
};

// Function to insert a value into the BST
struct Node* insert(struct Node* root, int value) {
    // If the tree is empty, create a new node and return it
    if (root == NULL) {
        struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->value = value;
        newNode->left = NULL;
        newNode->right = NULL;
```

```
        return newNode;
    }

    // Otherwise, recur down the tree
    if (value < root->value) {
        root->left = insert(root->left, value); // Insert in the left subtree
    } else {
        root->right = insert(root->right, value); // Insert in the right subtree
    }

    return root; // Return the unchanged root pointer
}

// Function to search for a value in the BST
int search(struct Node* root, int value) {
    // Base cases: root is null or value is present at root
    if (root == NULL) {
        return 0; // Value not found
    }
    if (root->value == value) {
        return 1; // Value found
    }

    // Value is greater than root's value
    if (value > root->value) {
        return search(root->right, value); // Search in the right subtree
    }

    // Value is smaller than root's value
    return search(root->left, value); // Search in the left subtree
}

// Main function to demonstrate BST operations
int main() {
    struct Node* root = NULL; // Initialize the root of the BST
    int n;

    // Read the number of nodes
    scanf("%d", &n);
    int values[n];

    // Read the values for the nodes
```

```
    for (int i = 0; i < n; i++) {
        scanf("%d", &values[i]);
        root = insert(root, values[i]); // Insert each value into the BST
    }

    // Read the value to be searched
    int key;
    scanf("%d", &key);

    // Search for the value in the BST
    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***