Rajalakshmi Engineering College

Name: Nishanth V C
Email: 240801227@rajalakshmi.edu.in
Roll no: 240801227
Phone: 9043313020
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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

```
struct Node* insertNode(struct Node* root, int value) {
    // If the tree is empty, create a new node
    if (root == NULL) {
        return createNode(value);
    }

    // Otherwise, insert the value recursively
    if (value < root->data) {
        root->left = insertNode(root->left, value);
    } else if (value > root->data) {
        root->right = insertNode(root->right, value);
    }

    return root;  // Return the root node
}

// Function to search for a value in the BST
struct Node* searchNode(struct Node* root, int value) {
    // If the root is NULL or the value is found, return the node
```

```
    if (root == NULL || root->data == value) {
        return root;
    }

    // If the value is smaller, search the left subtree
    if (value < root->data) {
        return searchNode(root->left, value);
    } else {
        // Otherwise, search the right subtree
        return searchNode(root->right, value);
    }
}
```

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