Rajalakshmi Engineering College

Name: Yogeetha A
Email: 241801326@rajalakshmi.edu.in
Roll no: 241801326
Phone: 9790848844
Branch: REC
Department: I AI & DS FD
Batch: 2028
Degree: B.E - AI & DS

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

```
Node* insertNode(Node* root, int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->left = newNode->right = NULL;

    if (root == NULL) {
        return newNode;
    }

    Node* current = root;
    Node* parent = NULL;

    while (current != NULL) {
        parent = current;
        if (data < current->data) {
            current = current->left;
        } else if (data > current->data) {
            current = current->right;
        } else {
```

```
            // Data already exists, handle as needed (e.g., ignore, return root, update
count)
            free(newNode); // Prevent memory leak
            return root;  // Or some other appropriate action
        }
    }

    if (data < parent->data) {
        parent->left = newNode;
    } else {
        parent->right = newNode;
    }

    return root;
}


// Search

Node* searchNode(Node* root, int data) {
    if (root == NULL || root->data == data) {
        return root;
    }

    if (data < root->data) {
        return searchNode(root->left, data);
    } else {
        return searchNode(root->right, data);
    }
}
```

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