Rajalakshmi Engineering College

Name: Jaidev Arunachalam
Email: 241801099@rajalakshmi.edu.in
Roll no: 241801099
Phone: 9940254113
Branch: REC
Department: I AI & DS FB
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 : 0

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>


struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

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

```
    // If the value to be inserted is smaller, insert it in the left subtree
    if (value < root->data) {
        root->left = insertNode(root->left, value);
    }
    // If the value to be inserted is greater, insert it in the right subtree
    else if (value > root->data) {
        root->right = insertNode(root->right, value);
    }

    // Return the unchanged root pointer
    return root;
}

// 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 at the root
    if (root == NULL || root->data == value) {
        return root;
    }

    // If the value is smaller than the root's data, search in the left subtree
    if (value < root->data) {
        return searchNode(root->left, value);
    }

    // If the value is larger than the root's data, search in the right subtree
    return searchNode(root->right, value);
}

int main() {
    struct Node* root = NULL;
    int numNodes, value, searchValue;

    // Read the number of nodes
    scanf("%d", &numNodes);

    // Insert nodes into the BST
    for (int i = 0; i < numNodes; i++) {
        scanf("%d", &value);
        root = insertNode(root, value);
    }
```

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

    // Search for the value in the tree
    struct Node* result = searchNode(root, searchValue);

    // Print the result
    if (result != NULL) {
        printf("Value %d is found in the tree.\n", searchValue);
    } else {
        printf("Value %d is not found in the tree.\n", searchValue);
    }

    return 0;
}
```

Status : Wrong **Marks : 0/10**