Rajalakshmi Engineering College

Name: akas prabhu
Email: 240801017@rajalakshmi.edu.in
Roll no: 240801017
Phone: 9360484615
Branch: REC
Department: I ECE FA
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

```
#include <stdio.h>
#include <stdlib.h>

// Definition of the Node structure
struct Node {
   int data;
   struct Node* left;
   struct Node* right;
};

// Function to create a new node
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 value into the BST
struct Node* insert(struct Node* root, int value) {
   if (root == NULL) {
      return createNode(value);
   }

   if (value < root->data) {
      root->left = insert(root->left, value);
   } else {
      root->right = insert(root->right, value);
   }

   return root;
}

// Function to search for a value in the BST
int search(struct Node* root, int value) {
   // If root is NULL or the value is present at the root
   if (root == NULL) {
      return 0; // Not found
   }

   // If the value is present at the current node
   if (root->data == value) {
      return 1; // Found
   }

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

   // Otherwise, search the right subtree
   return search(root->right, value);
}

int main() {
   int n, searchValue;

   // Read number of nodes in the BST
   scanf("%d", &n);
```

```
    struct Node* root = NULL;
    int value;

    // Read the values and insert them into the BST
    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        root = insert(root, value);
    }

    // Read the value to search in the BST
    scanf("%d", &searchValue);

    // Perform the search
    if (search(root, searchValue)) {
        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 : Correct ***Marks : 10/10***