Rajalakshmi Engineering College

Name: NISHANTH B
Email: 240701364@rajalakshmi.edu.in
Roll no: 240701364
Phone: 7904264876
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_CY_Updated

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Arun is working on a Binary Search Tree (BST) data structure. His goal is to implement a program that reads a series of integers and inserts them into a BST. Once the integers are inserted, he needs to add a given integer value to each node in the tree and find the maximum value in the BST.

Your task is to help Arun implement this program.

Input Format

The first line of input consists of an integer N, representing the number of elements to be inserted into the BST.

The second line consists of N space-separated integers, each representing an element to be inserted into the BST.

The third line consists of an integer add, representing the value to be added to each node in the BST.

Output Format

The output prints the maximum value in the BST after adding the add value.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 20 25
5
Output: 30

Answer

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


struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};


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


struct Node* insert(struct Node* root, int data) {
    if (root == NULL)
        return createNode(data);
    if (data < root->data)
```

```
        root->left = insert(root->left, data);
    else
        root->right = insert(root->right, data);
    return root;
}


void addToAllNodes(struct Node* root, int add) {
    if (root == NULL)
        return;
    root->data += add;
    addToAllNodes(root->left, add);
    addToAllNodes(root->right, add);
}


int findMax(struct Node* root) {
    while (root->right != NULL)
        root = root->right;
    return root->data;
}

int main() {
    int N, i, val, add;
    scanf("%d", &N);

    struct Node* root = NULL;

    for (i = 0; i < N; i++) {
        scanf("%d", &val);
        root = insert(root, val);
    }

    scanf("%d", &add);

    addToAllNodes(root, add);

    int max = findMax(root);
    printf("%d\n", max);

    return 0;
}
```

Status : Correct *Marks : 10/10*

2. Problem Statement

Jake is learning about binary search trees(BST) and their operations. He wants to implement a program that can delete a node from a BST based on the given key value and print the remaining nodes in an in-order traversal.

Assist Jake in the program.

Input Format

The first line of input consists of an integer n, representing the number of elements in BST.

The second line consists of n space-separated integers, representing the elements of the tree.

The third line consists of an integer x, representing the key value of the node to be deleted.

Output Format

The first line of output prints "Before deletion: " followed by the in-order traversal of the initial BST.

The second line prints "After deletion: " followed by the in-order traversal after the deletion of the key value.

If the key value is not present in the BST, print the original tree as it is.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
8 6 4 3 1
4

Output: Before deletion: 1 3 4 6 8
After deletion: 1 3 6 8

Answer

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

struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

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

struct Node* insert(struct Node* root, int data) {
    if (root == NULL)
        return createNode(data);
    if (data < root->data)
        root->left = insert(root->left, data);
    else
        root->right = insert(root->right, data);
    return root;
}

void inorder(struct Node* root) {
    if (root == NULL)
        return;
    inorder(root->left);
    printf("%d ", root->data);
    inorder(root->right);
}

struct Node* findMin(struct Node* root) {
    while (root && root->left != NULL)
        root = root->left;
```

```
    return root;
}

struct Node* deleteNode(struct Node* root, int key) {
    if (root == NULL)
        return NULL;
    if (key < root->data)
        root->left = deleteNode(root->left, key);
    else if (key > root->data)
        root->right = deleteNode(root->right, key);
    else {
        if (root->left == NULL) {
            struct Node* temp = root->right;
            free(root);
            return temp;
        } else if (root->right == NULL) {
            struct Node* temp = root->left;
            free(root);
            return temp;
        }
        struct Node* temp = findMin(root->right);
        root->data = temp->data;
        root->right = deleteNode(root->right, temp->data);
    }
    return root;
}

int search(struct Node* root, int key) {
    if (root == NULL)
        return 0;
    if (key == root->data)
        return 1;
    if (key < root->data)
        return search(root->left, key);
    else
        return search(root->right, key);
}

int main() {
    int n, i, x, val;
    scanf("%d", &n);
    struct Node* root = NULL;
```

```
    for (i = 0; i < n; i++) {
        scanf("%d", &val);
        root = insert(root, val);
    }
    scanf("%d", &x);
    printf("Before deletion: ");
    inorder(root);
    printf(" ");
    if (search(root, x))
        root = deleteNode(root, x);
    printf("\nAfter deletion: ");
    inorder(root);
    printf("\n");
    return 0;
}
```

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

3. Problem Statement

John is building a system to store and manage integers using a binary search tree (BST). He needs to add a feature that allows users to search for a specific integer key in the BST using recursion.

Implement functions to create the BST and perform a recursive search for an integer.

Input Format

The first line of input consists of an integer representing, the number of nodes.

The second line consists of integers representing, the values of nodes, separated by space.

The third line consists of an integer representing, the key to be searched.

Output Format

The output prints whether the given key is present in the binary search tree or not.

Refer to the sample output for the exact format.

Sample Test Case

Input: 7
10 5 15 3 7 12 20
12
Output: The key 12 is found in the binary search tree

Answer

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

struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

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

struct Node* insert(struct Node* root, int data) {
    if (root == NULL)
        return createNode(data);
    if (data < root->data)
        root->left = insert(root->left, data);
    else
        root->right = insert(root->right, data);
    return root;
}

int search(struct Node* root, int key) {
    if (root == NULL)
        return 0;
    if (root->data == key)
```

```
        return 1;
    if (key < root->data)
        return search(root->left, key);
    else
        return search(root->right, key);
}

int main() {
    int n, i, val, key;
    scanf("%d", &n);
    struct Node* root = NULL;
    for (i = 0; i < n; i++) {
        scanf("%d", &val);
        root = insert(root, val);
    }
    scanf("%d", &key);
    if (search(root, key))
        printf("The key %d is found in the binary search tree\n", key);
    else
        printf("The key %d is not found in the binary search tree\n", key);
    return 0;
}
```

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