Rajalakshmi Engineering College

Name: NITHISH RAJ L
Email: 240701366@rajalakshmi.edu.in
Roll no: 2116240701366
Phone: 8072719523
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

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

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

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

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

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

```
    return root;
}

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

int main() {
    int N, key;
    scanf("%d", &N);

    struct Node* root = NULL;

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

    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***

2. 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

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

typedef struct Node
{
  int data;
```

```
    struct Node* right;
    struct Node* left;
}Node;

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

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

int findMax(Node* root)
{
    if(root==NULL)
        return -1;
    while(root->right!=NULL)
    {
        root=root->right;
    }
    return root->data;
}
```

```
int main()
{
   int n;
   scanf("%d",&n);
   Node* root=NULL;
   int value;
   for(int i=0;i<n;i++)
   {
      scanf("%d",&value);
      root=insert(root,value);
   }
   int add;
   scanf("%d",&add);
   addValueToNodes(root,add);
   int maxValue=findMax(root);
   printf("%d\n",maxValue);
   return 0;
}
```

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

3. Problem Statement

Emily is studying binary search trees (BST). She wants to write a program that inserts characters into a BST and then finds and prints the minimum and maximum values.

Guide her with the program.

Input Format

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

The second line consists of N space-separated characters.

Output Format

The first line of output prints "Minimum value: " followed by the minimum value of the given inputs.

The second line prints "Maximum value: " followed by the maximum value of the given inputs.

Refer to the sample outputs for formatting specifications.

Sample Test Case

Input: 5
Z E W T Y

Output: Minimum value: E
Maximum value: Z

Answer

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

typedef struct Node
{
   char data;
   struct Node* right;
   struct Node* left;
}Node;

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

Node* insert(Node* root,char 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;
}

char findMin(Node* root)
{
    if(root==NULL)
        return '\0';
    while(root->left!=NULL)
    {
        root=root->left;
    }
    return root->data;
}

char findMax(Node* root)
{
    if(root==NULL)
        return '\0';
    while(root->right!=NULL)
    {
        root=root->right;
    }
    return root->data;
}

int main()
{
    int n;
    scanf("%d",&n);
    Node* root=NULL;
    char value;
    for (int i=0;i<n;i++)
    {
        scanf(" %c",&value);
        root=insert(root,value);
    }
    char minValue=findMin(root);
    char maxValue=findMax(root);
    printf("Minimum value: %c\n",minValue);
    printf("Maximum value: %c\n",maxValue);
    return 0;
}
```

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