Rajalakshmi Engineering College

Name: Sajine Santhakumar
Email: 240701459@rajalakshmi.edu.in
Roll no: 240701459
Phone: 9952076750
Branch: REC
Department: I CSE FE
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

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

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

Node*FindMin(Node*root){
   if(root==NULL){return NULL;}
   else if(root->left==NULL){
        return root;
    }
    else{
        return FindMin(root->left);
    }
}

Node*FindMax(Node*root){
    if(root==NULL){return NULL;}
    else if(root->right==NULL){
        return root;
    }
    else{
        return FindMax(root->right);}
}

int main(){
    Node*root=NULL;
    int n;
    char e;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf(" %c",&e);
        root=insert(root,e);
    }
    Node*min=FindMin(root);
    Node*max=FindMax(root);
    printf("Minimum value: %c\n",min->data);
    printf("Maximum value: %c",max->data);
    return 0;
}
```

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

2. Problem Statement

Edward has a Binary Search Tree (BST) and needs to find the k-th largest element in it.

Given the root of the BST and an integer k, help Edward determine the k-th largest element in the tree. If k exceeds the number of nodes in the BST, return an appropriate message.

Input Format

The first line of input consists of integer n, the number of nodes in the BST.

The second line consists of the n elements, separated by space.

The third line consists of the value of k.

Output Format

The output prints the kth largest element in the binary search tree.

For invalid inputs, print "Invalid value of k".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
8 4 12 2 6 10 14
1

Output: 14

Answer

```
#include<stdio.h>
#include<stdlib.h>
int count=0,result=-1;
typedef struct node{
   int data;
   struct node *left,*right;
}Node;

Node*insert(Node*root,int e){
   Node*Newnode=(Node*)malloc(sizeof(Node));
   if(root==NULL){
      Newnode->data=e;
      Newnode->left=Newnode->right=NULL;
      root=Newnode;
```

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

void kthlargest(Node *root,int k){
    if(root==NULL || count>=k) return;
    kthlargest(root->right,k);
    count++;
    if(count==k){
        result=root->data;
        return;
    }
    kthlargest(root->left,k);
}

int main(){
    Node*root=NULL;
    int n,e,k;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&e);
        root=insert(root,e);
    }
    scanf("%d",&k);
    if(k>n||k<=0){
        printf("Invalid value of k\n");
    }
    else{
        kthlargest(root,k);
        printf("%d\n",result);
    }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

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
   int data;
   struct node *left,*right;
}Node;
```

```
Node*insert(Node*root,int e){
   Node*Newnode=(Node*)malloc(sizeof(Node));
   if(root==NULL){
      Newnode->data=e;
      Newnode->left=Newnode->right=NULL;
      root=Newnode;
   }
   else if(e<root->data){
      root->left=insert(root->left,e);
   }
   else if(e>root->data){
      root->right=insert(root->right,e);
   }
   return root;
}

Node*search(Node*root,int k){
   if(root==NULL||root->data==k){
      return root;
   }
   else if(k<root->data){
      return search(root->left,k);
   }
   else if(k>root->data){
      return search(root->right,k);
   }
}
int main(){
   Node*root=NULL;
   int n,e,k;
   scanf("%d",&n);
   for(int i=0;i<n;i++){
      scanf("%d",&e);
      root=insert(root,e);
   }scanf("%d",&k);
   root=search(root,k);
   if(root!=NULL){
      printf("The key %d is found in the binary search tree\n",k);
   }
   else{
      printf("The key %d is not found in the binary search tree\n",k);
   }return 0;
}
```

Status : Correct *Marks : 10/10*