Rajalakshmi Engineering College

Name: Priyan S
Email: 240701402@rajalakshmi.edu.in
Roll no: 240701402
Phone: 9150170939
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 : 27.5

Section 1 : Coding

1. 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

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

struct Node{
   struct Node* left;
   int data;
   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);
   else if(root->data==data)
      return root;
   else if(root->data>data)
      root->left=insert(root->left,data);
   else
```

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

void kthMax(struct Node* root,int* k,int* result){
    if(root==NULL || *k==0)
        return;
    kthMax(root->right,k,result);
    (*k)--;
    if(*k==0){
        *result=root->data;
        return;
    }
    kthMax(root->left,k,result);
}

int main(){
    int n,k,data;
    scanf("%d", &n);
    struct Node* root=NULL;
    for(int i=0;i<n;i++){
        scanf("%d",&data);
        root=insert(root,data);
    }
    scanf("%d",&k);
    if((k<=0)|| (k>n)){
        printf("Invalid value of k\n");
    }else{
        int result=-1;
        kthMax(root,&k,&result);
        printf("%d\n",result);
    }
    return 0;
}
```

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

2. Problem Statement

Kishore is studying data structures, and he is currently working on implementing a binary search tree (BST) and exploring its basic operations. He wants to practice creating a BST, inserting elements into it, and performing a specific operation, which is deleting the minimum element from the tree.

Write a program to help him perform the delete operation.

Input Format

The first line of input consists of an integer N, representing the number of elements Kishore wants to insert into the BST.

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

Output Format

The output prints the remaining elements of the BST in ascending order (in-order traversal) after deleting the minimum element.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6
5 3 8 2 4 6
Output: 3 4 5 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 value){
   struct node* newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=value;
   newnode->left=newnode->right=NULL;
   return newnode;
}

struct node* insert(struct node* root,int value){
   if(root==NULL)
      return createnode(value);
   else if(root->data==value)
      return root;
   else if(root->data>value)
      root->left=insert(root->left,value);
   else
      root->right=insert(root->right,value);
   return root;
}

struct node* findmin(struct node* root){
   if(root!=NULL)
      while(root->left!=NULL)
         root=root->left;
   return root;
}

struct node* deletenode(struct node* root,int value){
   struct node* temp=(struct node*)malloc(sizeof(struct node));
   if(value<root->data){
      root->left=deletenode(root->left,value);
   }
   else if(value>root->data){
      root->right=deletenode(root->right,value);
   }
   else if(root->right && root->left){
      temp=findmin(root->right);
```

```
        root->data=temp->data;
        root->right=deletenode(root->right,root->data);
    }
    else{
        temp=root;
        if(root->left==NULL)
            root=root->right;
        else if(root->right==NULL)
            root=root->left;
        free(temp);
    }
    return root;
}

void Inorder(struct node* root){
    if(root!=NULL){
        Inorder(root->left);
        printf("%d ",root->data);
        Inorder(root->right);
    }
}
int main(){
    int n,value;
    scanf("%d",&n);
    struct node* root=NULL;
    for(int i=0;i<n;i++){
        scanf("%d",&value);
        root=insert(root,value);
    }
    struct node* minnode=findmin(root);
    root=deletenode(root,minnode->data);
    Inorder(root);
    return 0;
}
```

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

3. 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* right;
    struct node* left;
};

struct node* create( 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 create(data);

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

struct node*  findmin(struct node* root){
     if(root!=NULL)
        while(root->left!=NULL)
                root=root->left;
    return root;
}

struct node* deletenode(struct node* root,int data ){
     struct node* temp=(struct node*)malloc(sizeof(struct node));
     if(data<root->data){
          root->left=deletenode(root->left,data);
    }
    else if(data>root->data){
        root->right=deletenode(root->right,data);
    }
    else if(root->right && root->left){
        temp=findmin(root->right);
        root->right=deletenode(root->right,root->data);
    }
    else{
       temp=root;
       if(root->left==NULL){
           root=root->right;
        }
        else if(root->right==NULL){
           root=root->left;
           free(temp);
       }
    }
     return root;
}


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

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



int main(){
    int n,data,x;
    scanf("%d",&n);
    struct node* root=NULL;
    for(int i=0;i<n;i++){
        scanf("%d ",&data);
        root=insert(root,data);
    }
    printf("Before deletion: ");
    inorder(root);
    scanf("%d",&x);
    printf("\n");
    if(exists(root,x)){
        root=deletenode(root,x);
    }
    printf("After deletion: ");
    inorder(root);
    return 0;
}
```

Status : Partially correct *Marks : 7.5/10*