

Rajalakshmi Engineering College

Name: NETHRA CHANDRAGANDHI T
Email: 240701357@rajalakshmi.edu.in
Roll no: 240701357
Phone: 9487531086
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

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>
```

```
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=NULL;
```

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

```

```

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

```

```

Node *deletee(Node *root,int d){
    Node *tempnode=(Node*)malloc(sizeof(Node));
    if(root==NULL){
        return NULL;
    }
    if(d<root->data){
        root->left=deletee(root->left,d);
    }
    else if(d>root->data){
        root->right=deletee(root->right,d);
    }
    else if(root->left && root->right){
        tempnode=findmin(root->right);
        root->data=tempnode->data;
        root->right=deletee(root->right,root->data);
    }
}

```

```

    }
    else{
        tempnode=root;
        if(root->left==NULL){
            root=root->right;
        }
        else if(root->right==NULL){
            root=root->left;
        }
        free(tempnode);
    }
    return root;
}

int main(){
    Node *root=NULL;
    int n,e,d;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&e);
        root=insert(root,e);
    }
    scanf("%d",&d);
    printf("Before deletion: ");
    inorder(root);
    printf("\n");
    root=deletee(root,d);
    printf("After deletion: ");
    inorder(root);
    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

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node*left,*right;
};
struct node*cn(int val){
    struct node*newn=(struct node*)malloc(sizeof(struct node));
    newn->data=val;
    newn->left=newn->right=NULL;
    return newn;
}
struct node*insert(struct node*root,int val){
```

```

    if(root==NULL)
        return cn(val);
    if(val<root->data)
        root->left=insert(root->left,val);
    else
        root->right=insert(root->right,val);
    return root;
}
void sortedarray(struct node*root,int* arr,int *idx){
    if(root==NULL)
        return;
    sortedarray(root->left,arr,idx);
    arr[*idx++]=root->data;
    sortedarray(root->right,arr,idx);
}
int main(){
    struct node*root=NULL;
    int n,val,arr[1000],ind=1,index;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&val);
        root=insert(root,val);
    }
    scanf("%d",&index);
    sortedarray(root,arr,&ind);
    if(index>n || index<=0)
        printf("Invalid value of k");
    else
        printf("%d",arr[(n+1)-index]);
}

```

Status : Correct

Marks : 10/10

3. 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>
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=NULL;
        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 inorder(Node *root){
    if(root!=NULL){
        inorder(root->left);
        printf("%d ",root->data);
        inorder(root->right);
    }
}

```

```

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

```

```

Node *deletee(Node *root,int min){
    Node *tempnode=(Node*)malloc(sizeof(Node));
    if(min<root->data){
        root->left=deletee(root->left,min);
    }
    else if(min>root->data){
        root->right=deletee(root->right,min);
    }
    else if(root->left && root->right){
        tempnode=findmin(root->right);
        root->data=tempnode->data;
        root->right=deletee(root->right,root->data);
    }
}

```



```

    }
    else{
        tempnode=root;
        if(root->left==NULL){
            root=root->right;
        }
        else if(root->right==NULL){
            root=root->left;
        }
        free(tempnode);
    }
    return root;
}

```

```

int main(){
    Node *root=NULL;
    int n,e;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&e);
        root=insert(root,e);
    }
    Node *min=findmin(root);
    root=deletee(root,min->data);
    inorder(root);
}

```

Status : Correct

Marks : 10/10