Rajalakshmi Engineering College

Name: RANNESH KHUMAR B R
Email: 240701422@rajalakshmi.edu.in
Roll no: 2116240701
Phone: 9042350670
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 : 2
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>
struct node{
    int data;
    struct node*left;
    struct node*right;
};
typedef struct node tree;

tree *create(int key){
    tree*newnode=(tree*)malloc(sizeof(tree));
    newnode->data=key;
    newnode->left=newnode->right=NULL;
    return newnode;
}
```

```
tree *insert(tree*root,int key){
   if(root==NULL)
      return create(key);
   else if(key<root->data)
      root->left=insert(root->left,key);
   else if(key>root->data)
      root->right=insert(root->right,key);
   return root;
}
tree *findmin(tree*root){
   while(root->left!=NULL)
      root=root->left;
   return root;
}
tree *del(tree*root,int target){
   if(root==NULL)
      return NULL;
   tree*temp=(tree*)malloc(sizeof(tree));
   if(target<root->data)
      root->left=del(root->left,target);
   else if(target>root->data)
      root->right=del(root->right,target);
   else if(root->left && root->right){
      temp=findmin(root->right);
      root->data=temp->data;
      root->right=del(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(tree*root){
   if(root!=NULL){
      inorder(root->left);
```

```
        printf("%d ",root->data);
        inorder(root->right);

    }
}

int main(){
    tree*root=NULL;
    int n;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        int key;
        scanf("%d",&key);
        root=insert(root,key);
    }
    printf("Before deletion: ");
    inorder(root);
    int target;
    scanf("%d",&target);
    del(root,target);
    printf("\nAfter deletion: ");
    inorder(root);
}
```

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

2. Problem Statement

You are given a series of magic levels (integers) and need to construct a Binary Search Tree (BST) from them. After constructing the BST, your task is to perform a range search, which involves finding and printing all the magic levels within a specified range [L, R].

Input Format

The first line of input consists of an integer N, the number of magic levels to insert into the BST.

The second line consists of N space-separated integers, representing the magic levels to insert.

The third line consists of two integers, L and R, which define the range for the search.

Output Format

The output prints all the magic levels within the range [L, R] in ascending order, separated by spaces.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 3 7
2 20
Output: 3 5 7 10 15

Answer

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

struct node{
   int data;
   struct node*left;
   struct node*right;
};
typedef struct node tree;

tree*create(int key){
   tree*newnode=(tree*)malloc(sizeof(tree));
   newnode->data=key;
   newnode->left=newnode->right=NULL;
   return newnode;
}

tree*insert(tree*root,int key){
   if(root==NULL)
      return create(key);
   else if(key<root->data)
      root->left=insert(root->left,key);
   else if(key>root->data)
```

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

void print(tree*root,int l,int r){
    if(root==NULL)
        return;
    if(root->data>l)
        print(root->left,l,r);
    if(root->data>=l && root->data <=r)
        printf("%d ",root->data);
    if(root->data<r)
        print(root->right,l,r);
}

int main(){
    int n;
    scanf("%d",&n);
    tree*root=NULL;
    for(int i=0;i<n;i++){
        int value;
        scanf("%d",&value);
        root=insert(root,value);
    }
    int l,r;
    scanf("%d %d",&l,&r);
    print(root,l,r);
}
```

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

3. Problem Statement

Dhruv is working on a project where he needs to implement a Binary Search Tree (BST) data structure and perform various operations on it.

He wants to create a program that allows him to build a BST, traverse it in different orders (inorder, preorder, postorder), and exit the program when needed.

Help Dhruv by designing a program that fulfils his requirements.

Input Format

The first input consists of the choice.

If the choice is 1, enter the number of elements N and the elements inserted into the tree, separated by a space in a new line.

If the choice is 2, print the in-order traversal.

If the choice is 3, print the pre-order traversal.

If the choice is 4, print the post-order traversal.

If the choice is 5, exit.

Output Format

The output prints the results based on the choice.

For choice 1, print "BST with N nodes is ready to use" where N is the number of nodes inserted.

For choice 2, print the in-order traversal of the BST.

For choice 3, print the pre-order traversal of the BST.

For choice 4, print the post-order traversal of the BST.

For choice 5, the program exits.

If the choice is greater than 5, print "Wrong choice".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1
5
12 78 96 34 55
2
3
4
5

Output: BST with 5 nodes is ready to use
BST Traversal in INORDER
12 34 55 78 96
BST Traversal in PREORDER
12 78 34 55 96
BST Traversal in POSTORDER
55 34 96 78 12

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node*left;
    struct node*right;
};
typedef struct node tree;

tree*create(int key){
    tree*newnode=(tree*)malloc(sizeof(tree));
    newnode->data=key;
    newnode->left=newnode->right=NULL;
    return newnode;
}

tree*insert(tree*root,int key){
    if(root==NULL)
        return create(key);

    else if(key<root->data)
        root->left=insert(root->left,key);
    else if(key>root->data)
```

```
        root->right=insert(root->right,key);
    return root;
}
void inorder(tree*root){
    if(root!=NULL){
        inorder(root->left);
        printf("%d ",root->data);
        inorder(root->right);
    }
}

void preorder(tree*root){
    if(root!=NULL){
        printf("%d ",root->data);
        preorder(root->left);
        preorder(root->right);

    }
}

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

int main(){
    int ch;
    int key;
    int n;
    tree*root=NULL;
    do{
        scanf("%d",&ch);
        if(ch==1){
            root=NULL;
            scanf("%d",&n);
            for(int i=0;i<n;i++){
                scanf("%d",&key);
                root=insert(root,key);
            }
```

```
            printf("BST with %d nodes is ready to use\n",n);
        }
        if(ch==2){
            printf("BST Traversal in INORDER\n");
            inorder(root);
            printf("\n");
        }
        if(ch==3){
            printf("BST Traversal in PREORDER\n");
            preorder(root);
            printf("\n");
        }
        if(ch==4){
            printf("BST Traversal in POSTORDER\n");
            postorder(root);
            printf("\n");
        }
        if(ch==5){
            break;
        }
        if(ch>5){
            printf("Wrong choice\n");
        }

    }while(ch<=5);

}
```

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