Rajalakshmi Engineering College

Name: Sankara Gomathi R
Email: 240701470@rajalakshmi.edu.in
Roll no: 240701470
Phone: 7530026101
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

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

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

node*insert(node*root,int e){
   node*newn=(node*)malloc(sizeof(node));
   if(root==NULL){
      newn->data=e;
      newn->left=NULL;
      newn->right=NULL;
      root=newn;
   }
   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);
    return 0;
}
```

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

2. 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

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

node*insert(node*root,int e){
   node*newn=(node*)malloc(sizeof(node));
   if(root==NULL){
      newn->data=e;
      newn->left=NULL;
      newn->right=NULL;
      root=newn;
   }
   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);
   }
}

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

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

int main(){
    node*root=NULL;
    int n,e,choice;
    do{
        scanf("%d",&choice);
        switch(choice){
            case 1:
            root=NULL;
            scanf("%d",&n);
            for(int i=0;i<n;i++){
                scanf("%d",&e);
                root=insert(root,e);
            }
            printf("BST with %d nodes is ready to use\n",n);
            break;
            case 2:
            printf("BST Traversal in INORDER\n");
            inorder(root);
            printf("\n");
            break;
            case 3:
            printf("BST Traversal in PREORDER\n");
            preorder(root);
            printf("\n");
            break;
            case 4:
            printf("BST Traversal in POSTORDER\n");
            postorder(root);
            printf("\n");
            break;
            case 5:
            return 0;
            break;
            default:
            printf("Wrong choice\n");
        break;
      }
   }while(true);
   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*newn=(node*)malloc(sizeof(node));
    if(root==NULL){
       newn->data=e;
       newn->left=NULL;
       newn->right=NULL;
       root=newn;
    }
    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***