Rajalakshmi Engineering College

Name: shiloh .s
Email: 240701498@rajalakshmi.edu.in
Roll no: 240701498
Phone: 9488883273
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>
struct node
{
   int data;
   struct node*left;
   struct node*right;
};
struct node*insert(struct node*root,int a)
{
   if(root==NULL)
   {
      struct node*nn=(struct node*)malloc(sizeof(struct node));
      nn->data=a;
      nn->left=NULL;
      nn->right=NULL;
      root=nn;
      return root;
   }
   else
   if(a<root->data)
   {
      root->left=insert(root->left,a);
   }
```

```
    else
    if(a>root->data)
    {
        root->right=insert(root->right,a);
    }
    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 a)
{
    if(root==NULL)
    {
        return root;
    }
    else
    if(a<root->data)
    {
        root->left=deletenode(root->left,a);
    }
    else
    if(a>root->data)
    {
        root->right=deletenode(root->right,a);
    }
    else
    {
        if(root->left==NULL)
        {
            struct node*temp=root->right;
            free(root);
            return temp;
        }
```

```
        else
        if(root->right==NULL)
        {
            struct node*temp=root->left;
            free(root);
            return temp;
        }
        struct node*temp=findmin(root->right);
        root->data=temp->data;
        root->right=deletenode(root->right,temp->data);
    }
    return root;
}
void display(struct node*root)
{
    if(root!=NULL)
    {
        display(root->left);
        printf("%d ",root->data);
        display(root->right);
    }
}
int findmin1(struct node*root)
{
    if(root!=NULL)
    {
        while(root->left!=NULL)
        {
            root=root->left;
        }
    }
    return root->data;
}
int main()
{
    struct node*root=NULL;
    int a;
    scanf("%d",&a);
    for(int i=0;i<a;i++)
    {
        int b;
        scanf("%d",&b);
```

```
        root=insert(root,b);
    }
    int c;
    c=findmin1(root);
    root=deletenode(root,c);
    display(root);
    return 0;
}
```

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;
};
struct node*insert(struct node*root,int a)
{
   if(root==NULL)
   {
      struct node*nn=(struct node*)malloc(sizeof(struct node));
      nn->data=a;
      nn->left=NULL;
      nn->right=NULL;
      root=nn;
      return root;
   }
   else
   if(a<root->data)
   {
      root->left=insert(root->left,a);
   }
   else
   if(a>root->data)
   {
      root->right=insert(root->right,a);
   }
   return root;
}
void display(struct node*root,int a,int b)
{
   if(root!=NULL)
   {
      display(root->left,a,b);
      if(root->data>=a&&root->data<=b)
      {
```

```
        printf("%d ",root->data);
        }
        display(root->right,a,b);
    }
}
int main()
{
    struct node*root=NULL;
    int a;
    scanf("%d",&a);
    for(int i=0;i<a;i++)
    {
        int b;
        scanf("%d",&b);
        root=insert(root,b);
    }
    int c,d;
    scanf("%d %d",&c,&d);
    display(root,c,d);
    return 0;
}
```

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

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

struct Node {
    int key;
    struct Node* left;
    struct Node* right;
};

struct Node* root = NULL;

struct Node* newNode(int item) {
    struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
    temp->key = item;
    temp->left = NULL;
    temp->right = NULL;
    return temp;
}

struct Node* insert(struct Node* node, int key) {
    if (node == NULL)
        return newNode(key);
    if (key < node->key)
        node->left = insert(node->left, key);
    else
        node->right = insert(node->right, key);
    return node;
}

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

```
void preorder(struct Node* root) {
    if (root != NULL) {
        printf("%d ", root->key);
        preorder(root->left);
        preorder(root->right);
    }
}

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

void freeTree(struct Node* root) {
    if (root != NULL) {
        freeTree(root->left);
        freeTree(root->right);
        free(root);
    }
}

int main() {
    int choice, n, i, val;

    while (1) {
        scanf("%d", &choice);

        if (choice == 1) {
            freeTree(root);
            root = NULL;
            scanf("%d", &n);
            for (i = 0; i < n; i++) {
                scanf("%d", &val);
                root = insert(root, val);
            }
            printf("BST with %d nodes is ready to use\n", n);
        } else if (choice == 2) {
            printf("BST Traversal in INORDER\n");
            inorder(root);
```

```
            printf("\n");
        } else if (choice == 3) {
            printf("BST Traversal in PREORDER\n");
            preorder(root);
            printf("\n");
        } else if (choice == 4) {
            printf("BST Traversal in POSTORDER\n");
            postorder(root);
            printf("\n");
        } else if (choice == 5) {
            break;
        } else {
            printf("Wrong choice\n");
        }
    }

    freeTree(root);
    return 0;
}
```

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