ASSIGNMENT#3

***Data Structures & Algorithms***

***WAQAS ASHIQ***

***BCS-F11-201***

***SECTION C***

***SUBMITTED TO: Sir Usman Ashraf***

# 

# /\*Binary seach tree\*/

# Code:

#include<stdio.h>  
#include<stdlib.h>  
  
typedef struct treeNode  
{  
        int data;  
        struct treeNode \*left;  
        struct treeNode \*right;  
  
}treeNode;  
  
treeNode\* FindMin(treeNode \*node)  
{  
        if(node==NULL)  
        {  
                 
                return NULL;  
        }  
        if(node->left)   
                return FindMin(node->left);  
        else   
                return node;  
}  
treeNode\* FindMax(treeNode \*node)  
{  
        if(node==NULL)  
        {  
                  
                return NULL;  
        }  
        if(node->right)   
                FindMax(node->right);  
        else   
                return node;  
}  
  
treeNode \* Insert(treeNode \*node,int data)  
{  
        if(node==NULL)  
        {  
                treeNode \*temp;  
                temp = (treeNode \*)malloc(sizeof(treeNode));  
                temp -> data = data;  
                temp -> left = temp -> right = NULL;  
                return temp;  
        }  
  
        if(data >(node->data))  
        {  
                node->right = Insert(node->right,data);  
        }  
        else if(data < (node->data))  
        {  
                node->left = Insert(node->left,data);  
        }  
          
        return node;  
  
}  
  
treeNode \* Delete(treeNode \*node, int data)  
{  
        treeNode \*temp;  
        if(node==NULL)  
        {  
                printf("Element Not Found");  
        }  
        else if(data < node->data)  
        {  
                node->left = Delete(node->left, data);  
        }  
        else if(data > node->data)  
        {  
                node->right = Delete(node->right, data);  
        }  
        else  
        {  
                 
                if(node->right && node->left)  
                {  
                          
                        temp = FindMin(node->right);  
                        node -> data = temp->data;   
                         
                        node -> right = Delete(node->right,temp->data);  
                }  
                else  
                {  
                          
                        temp = node;  
                        if(node->left == NULL)  
                                node = node->right;  
                        else if(node->right == NULL)  
                                node = node->left;  
                        free(temp);   
                }  
        }  
        return node;  
  
}  
  
treeNode \* Find(treeNode \*node, int data)  
{  
        if(node==NULL)  
        {  
                  
                return NULL;  
        }  
        if(data > node->data)  
        {  
                 
                return Find(node->right,data);  
        }  
        else if(data < node->data)  
        {  
                 
                return Find(node->left,data);  
        }  
        else  
        {  
                 
                return node;  
        }  
}  
  
void PrintInorder(treeNode \*node)  
{  
        if(node==NULL)  
        {  
                return;  
        }  
        PrintInorder(node->left);  
        printf("%d ",node->data);  
        PrintInorder(node->right);  
}  
  
void PrintPreorder(treeNode \*node)  
{  
        if(node==NULL)  
        {  
                return;  
        }  
        printf("%d ",node->data);  
        PrintPreorder(node->left);  
        PrintPreorder(node->right);  
}  
  
void PrintPostorder(treeNode \*node)  
{  
        if(node==NULL)  
        {  
                return;  
        }  
        PrintPostorder(node->left);  
        PrintPostorder(node->right);  
        printf("%d ",node->data);  
}  
  
int main(void)  
{  
                
              int choise;  
              int x;  
              printf("1)for insertion'\n2 for delete\n3)inorder\n4)preorder\n5)postorder\n6)to find min\n7)to find max\n8)find\n0)exit\n");  
              scanf("%d",&choise);  
              treeNode \*root = NULL;  
              treeNode \* temp;  
              while(choise!=0){  
                               switch(choise){  
                                              case 1:  
                                                   printf("\nEnter the value you tree:");  
                                                   scanf("%d",&x);  
                                                   root = Insert(root, x);  
                                                   printf("1)for insertion'\n2 for delete\n3)inorder\n4)preorder\n5)postorder\n6)to find min\n7)to find max\n8)find\n0)exit\n");  
                                                   scanf("%d",&choise);  
                                                     
                                                   break;  
                                                   case 2:  
                                                   printf("\nEnter the value you want to delete tree:");  
                                                   scanf("%d",&x);  
                                                   root = Delete(root,x);  
                                                   printf("1)for insertion'\n2 for delete\n3)inorder\n4)preorder\n5)postorder\n6)to find min\n7)to find max\n8)find\n0)exit\n");  
                                                   scanf("%d",&choise);  
                                                        break;  
                                                         case 3:  
                                                              PrintInorder(root);  
                                                              printf("\n");  
                                                              printf("1)for insertion'\n2 for delete\n3)inorder\n4)preorder\n5)postorder\n6)to find min\n7)to find max\n8)find\n0)exit\n");  
                                                              scanf("%d",&choise);  
                                                              break;  
                                                                
                                                              case 4:  
                                                                   PrintPreorder(root);  
                                                                   printf("\n");  
                                                                   printf("1)for insertion'\n2 for delete\n3)inorder\n4)preorder\n5)postorder\n6)to find min\n7)to find max\n8)find\n0)exit\n");  
                                                                   scanf("%d",&choise);  
                                                                   break;  
                                                                     
                                                                   case 5:  
                                                                        PrintPostorder(root);  
                                                                        printf("\n");  
                                                                        printf("1)for insertion'\n2 for delete\n3)inorder\n4)preorder\n5)postorder\n6)to find min\n7)to find max\n8)find\n0)exit\n");  
                                                                        scanf("%d",&choise);  
                                                                        break;  
                                                                        case 6:  
                                                                             temp = FindMin(root);  
                                                                             printf("Minimum element is %d\n",temp->data);  
                                                                             printf("1)for insertion'\n2 for delete\n3)inorder\n4)preorder\n5)postorder\n6)to find min\n7)to find max\n8)find\n0)exit\n");  
                                                                             scanf("%d",&choise);  
                                                                             break;  
                                                                             case 7:  
                                                                                  temp = FindMax(root);  
                                                                                  printf("Maximum element is %d\n",temp->data);  
                                                                                  printf("1)for insertion'\n2 for delete\n3)inorder\n4)preorder\n5)postorder\n6)to find min\n7)to find max\n8)find\n0)exit\n");  
                                                                                  scanf("%d",&choise);  
                                                                                  break;  
                                                                                  case 8:  
                                                                                       printf("\nEnter the value you to find in tree:");  
                                                                                       scanf("%d",&x);  
                                                                                       temp = Find(root,x);  
                                                                                       if(temp==NULL)  
                                                                                       {  
                                                                                       printf("Element not found\n");  
                                                                                       }  
                                                                                       else  
                                                                                       {  
                                                                                       printf("Element 8 Found\n");  
                                                                                       }  
                                                                                       printf("1)for insertion'\n2 for delete\n3)inorder\n4)preorder\n5)postorder\n6)to find min\n7)to find max\n8)find\n0)exit\n");  
                                                                                       scanf("%d",&choise);  
                                                                                       break;  
                                                                                         
                                              }  
                               }  
          
          
          
}

# Output:

# 

# /\*[Josephus Problem](http://cs-study.blogspot.com/2012/10/josephus-problem-in-data-structure.html)\*/

**Code:**

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

struct node{

int place;

struct node \*next;

};

void insert(struct node \*ptr,int data){

struct node \*q=ptr;

while(ptr->next!=q){

ptr=ptr->next;

}

ptr->next=(node \*)malloc(sizeof(node));

ptr=ptr->next;

ptr->place=data;

ptr->next=q;

}

void print(struct node \*ptr){

struct node \*p=ptr;

while(ptr->next!=p){

getch();

printf("%d\t",ptr->place);

ptr=ptr->next;

}

if(ptr->next==p){

if(ptr->place==5){

printf("\nThe number 5 is the leader");

}

getch();

printf("%d\t",ptr->place);

}

}

void del(struct node \*ptr){

int count=10;

struct node \*k;

struct node \*m;

while(count!=1){

for(int x=1;x<=3;x++){

k=ptr;

ptr=ptr->next;

}

m=ptr->next;

k->next=m;

free(ptr);

ptr=m;

count--;

}

}

int main(void){

int count=10;

struct node \*head;

head=(node \*)malloc(sizeof(node));

head->next=head;

head->place=1;

printf("\n10 persons are sitting on a round table:\n");

for(int i=2;i<=10;i++){

insert(head,i);

}

print(head);

del(head);

print(head);

}

**Output:**

