Experiment no. 7

Objective: Implementation of Binary tree using Linked list

Code:

```
#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node *left,*right;

};

struct node *root;

void insert(int x)

{

   struct node *p,*previous,*current;

   p=(struct node *)malloc(sizeof(struct node));

   if(p==NULL)

   {

     printf("\n Out of memory");

   }

   p->data=x;

   p->left=NULL;

   p->right=NULL;

   if(root=NULL)

   {
     root=p;

     return;

   }

   previous=NULL;

   current=root;

   while(current!=NULL)

   {

       previous=current;

       if(p->data<current->data)

            current=current->left;

       else

            current=current->right;

            }

     if(p->data<previous->data)

          previous->left=p;

     else

 previous->right=p;

}

void inorder(struct node *t)

{

  if (t!=NULL)

  {

  inorder(t->left);

  printf("\n %5d",t->data);

  inorder (t->right);

  }

}
```

```
void del(int x)

{

   int tright=0,tleft=0;

   struct node *ptr=root;

   struct node *parent=root;

   struct node *t1=root;

   struct node *temp=root;

   while(ptr!=NULL&& ptr->data!=x)

   {

      parent=ptr;

      if (x<ptr->data)

            ptr=ptr->left;

      else

           ptr=ptr->right;

   }

   if (ptr==NULL)

   {

     printf("\n Delete element not found");

     return ;

   }

   else if(t1->data==x && (t1->left ==NULL || t1->right==NULL))

          if(t1->left==NULL)

             t1=t1->right;

          else

             t1=t1->left;

   else if (ptr->left==NULL)

     if (x<parent->data)
```

```
    parent->left=ptr->right;
  else
    parent->right=ptr->right;
else if (ptr->right==NULL)
  if (x<parent->data)
    parent->left=ptr->left;
  else
    parent->right=ptr->left;
else
{
temp=ptr;
parent=ptr;
if((ptr->left)>=(ptr->right))
{
   ptr=ptr->left;
   while(ptr->right!=NULL)
   {
        tright=1;
        parent=ptr;
        ptr=ptr->right;
   }
   temp->data=ptr->data;
   if(tright)
        parent->right=ptr->left;
   else
        parent->left=ptr->left;
 }
 else
    {
     ptr=ptr->right;
     while (ptr->left!=NULL)
     {
           tleft=1;
           parent=ptr;
           ptr=ptr->left;
      }
      temp->data=ptr->data;
      if(tleft)
           parent->left=ptr->right;
      else
           parent->right=ptr->right;
    }
    free(ptr);
  }
}

void main()
{
int op,n,srchno;
root=(struct node *)malloc(sizeof(struct node));
root->data=30;
root->right=root->left=NULL;
clrscr();
do
```

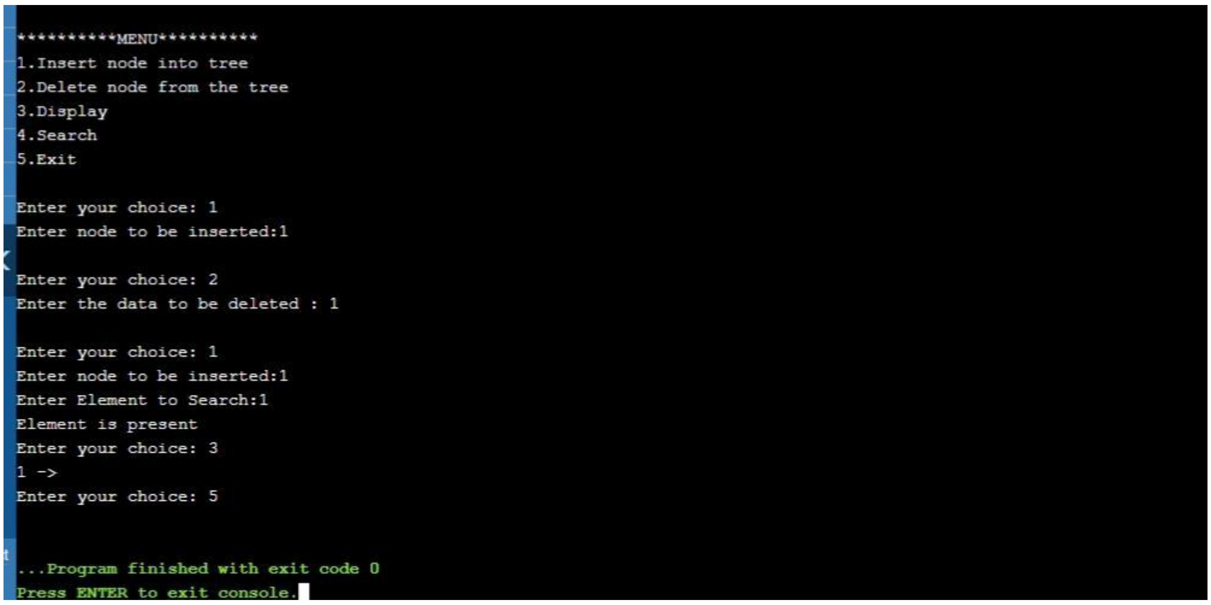
```
{
   printf(“\n***********MENU**********”);
   printf("\n 1.Insert node into tree");
   printf("\n 2.Delete node from tree");
   printf("\n 3.Display");
   printf("\n 4.Search");
   printf(“\n5.Exit");
   printf("\n Enter your choice\n");
   scanf("%d",&op);

  switch (op)
   {
   case 1: printf("\n Enter node to be inserted\n");
           scanf("%d",&n);
           insert(n);
           break;
   case 2: printf("\n Enter the data to be deleted\n");
           scanf("%d",&srchno);
           del(srchno);
           break;
   case 3:inorder(root);
           getch();
           break;
   case 4:printf(“\nElememt is present”);
   default: exit(0);
   }
  }while(op<4);
```

```
getch();

}
```

Output:



```
**********MENU**********
1.Insert node into tree
2.Delete node from the tree
3.Display
4.Search
5.Exit

Enter your choice: 1
Enter node to be inserted:1

Enter your choice: 2
Enter the data to be deleted : 1

Enter your choice: 1
Enter node to be inserted:1
Enter Element to Search:1
Element is present
Enter your choice: 3
1 ->
Enter your choice: 5


...Program finished with exit code 0
Press ENTER to exit console.
```

Presented by: Gelle Hruthesh reddy (20BCB7031)