1:Doubly Linked List :

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*left;

struct node \*right;

};

struct node \*head=NULL;

void insertf()

{

int x;

struct node \*ne;

ne=(struct node \*)malloc(sizeof(struct node));

if(ne==NULL)

{

printf("\nList in Full\n");

return;

}

printf("\nEnter the Element to insert:");

scanf("%d",&x);

ne->data=x;

ne->left=NULL;

ne->right=NULL;

if(head==NULL)

{

head=ne;

}

else

{

ne->right=head;

head->left=ne;

head=ne;

}

}

void insertl()

{

int x;

struct node \*ne;

ne=(struct node \*)malloc(sizeof(struct node));

if(ne==NULL)

{

printf("\nList in Full\n");

return;

}

printf("\nEnter the Element to insert:");

scanf("%d",&x);

ne->data=x;

ne->left=NULL;

ne->right=NULL;

if(head==NULL)

{

head=ne;

}

else

{

struct node \*ptr=head;

ptr=head;

while(ptr->right!=NULL)

ptr=ptr->right;

ne->left=ptr;

ptr->right=ne;

}

}

void display()

{

struct node \*ptr1=head;

if(head==NULL)

{

printf("\nList is Empty!\n");

return;

}

else

{

while(ptr1!=NULL)

{

printf("%d ",ptr1->data);

ptr1=ptr1->right;

}

}

}

void search()

{

struct node \*ptr1=head;

if(head==NULL)

{

printf("\nList is Empty!\n");

return;

}

else

{

int s,c=1;

printf("\nEnter the Element to Search:");

scanf("%d",&s);

while(ptr1!=NULL)

{

if(ptr1->data==s)

{

printf("\nElement Found!! at position:%d",c);

return;

}

c++;

ptr1=ptr1->right;

}

if(ptr1==NULL)

printf("\nElement not Found!.");

}

}

void deletef()

{

if(head==NULL)

{

printf("\nList is Empty!.");

return;

}

struct node \*temp=head;

printf("\n%d is Deleted!!",head->data);

head=head->right;

if(head!=NULL)

head->left=NULL;

free(temp);

}

void deletel()

{

if(head==NULL)

{

printf("\nList is Empty!.");

return;

}

struct node \*ptr=head;

while(ptr->right!=NULL)

{

ptr=ptr->right;

}

struct node \*temp=ptr;

printf("\n%d is Deleted!!",ptr->data);

if(ptr->left==NULL)

head=NULL;

else

{

ptr=ptr->left;

ptr->right=NULL;

}

free(temp);

}

void deletes()

{

struct node \*ptr=head;

struct node \*temp,\*r,\*l;

if(head==NULL)

{

printf("\nList is Empty!.");

return;

}

int x;

printf("\nEnter the Element to Delete:");

scanf("%d",&x);

if(ptr->data==x)

{

printf("\n%d is Deleted!!",ptr->data);

temp=ptr;

head=ptr->right;

free(temp);

}

else

{

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

ptr=ptr->right;

if(ptr!=NULL)

{

printf("\n%d is Deleted!!",ptr->data);

l=ptr->left;

r=ptr->right;

l->right=ptr->right;

if(l->right!=NULL)

{

r->left=ptr->left;

}

free(ptr);

}

else

printf("\nElement not Found!");

}

}

void inserts()

{

struct node \*ne,\*ptr,\*ptr1;

ne=(struct node \*)malloc(sizeof(struct node));

ne->right=NULL;

ne->left=NULL;

if(ne==NULL)

{

printf("\nList is Full!");

return;

}

int x,key;

printf("\nEnter the new element:");

scanf("%d",&x);

ne->data=x;

ptr=head;

if(ptr==NULL)

{

head=ne;

return;

}

printf("\nEnter the preivous node:");

scanf("%d",&key);

while(ptr->data!=key&&ptr->right!=NULL)

{

ptr=ptr->right;

}

if(ptr->right==NULL)

{

ptr->right=ne;

ne->left=ptr;

}

else

{

ne->left=ptr;

ne->right=ptr->right;

ptr1=ptr->right;

ptr1->left=ne;

ptr->right=ne;

}

}

void main()

{

int c=1;

while(c>0)

{

printf("\nSelect:\n1:Insert At Front\n2:Insert A Last\n3:Insert at specific position\n4:Display\n5:Search\n6:Delete from Front\n7:Delete from last\n8:Deleting Specific Element\n9:Exit\n");

scanf("%d",&c);

switch(c)

{

case 1:insertf();

break;

case 2:insertl();

break;

case 3:inserts();

break;

case 4:display();

break;

case 5:search();

break;

case 6:deletef();

break;

case 7:deletel();

break;

case 8:deletes();

break;

case 9:c=-1;

break;

default:c=-1;

}

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

}

OUTPUT:

