**Aim:- Write a program to implement following operations on the doubly linked list.**

**(a) Insert a node at the front of the linked list.**

**(b) Insert a node at the end of the linked list.**

**(c) Delete a first node of the linked list.**

**(d) Delete a node before specified position.**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int info;

struct node \*lptr,\*rptr;

}\*l,\*r;

void doubins(int);

void doubdel(int);

void display();

int chk;

void main()

{

int ch,x;

l=NULL;

r=NULL;

do

{

printf("\n Press:=>\n");

printf("\n 1.Insert Node");

printf("\n 2.Delete Node");

printf("\n 3.Display Doubly Linked List");

printf("\n 4.Exit");

printf("\n Enter Choice: ");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("\n\t Enter Element: ");

scanf("%d",&x);

doubins(x);

display();

break;

case 2:

printf("\n\t Enter Element which you want to Delete: ");

scanf("%d",&x);

chk=0;

if(l!=NULL) //if(r!=NULL)

{

printf("\n\n Before Deletion:=>");

printf("\n----------------");

display();

}

doubdel(x);

if(chk==0) //if(r!=NULL)

{

printf("\n\n\n\n\n After Deletion:=>");

printf("\n---------------");

display();

}

break;

case 3:

display();

break;

case 4:

exit(0);

default:

printf("\n\t Invalid Choice.\n\tTry Again.");

}

}while(ch!=4);

}

void doubins(int x)

{

struct node \*New,\*temp;

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

New->info=x;

if(l==NULL) //if(r==NULL)

{

New->lptr=NULL;

New->rptr=NULL;

l=New;

r=New;

return;

}

if(x<=l->info) //If the Node is less than all the Nodes

{

New->lptr=NULL;

New->rptr=l;

l->lptr=New;

l=New;

return;

}

temp=l; //If the Node is inserted in between two nodes

while(temp->info<x && temp!=NULL)

temp=temp->rptr;

if(temp!=NULL)

{

New->lptr=temp->lptr;

New->rptr=temp;

temp->lptr=New;

New->lptr->rptr=New;

return;

}

New->rptr=NULL; //Node is inserted at last

New->lptr=r;

r->rptr=New;

r=New;

}

void doubdel(int x)

{

struct node \*temp;

if(l==NULL) //if(r==NULL)

{

printf("\n\n\tDoubly Linked List Underflow on Delete.");

chk=1;

return;

}

if(x==l->info) //If First Node is tobe Deleted

{

temp=l;

l=l->rptr;

l->lptr=NULL;

free(temp);

return;

}

if(x==r->info) //If Last Node is tobe Deleted

{

temp=r;

r=r->lptr;

r->rptr=NULL;

free(temp);

return;

}

temp=l;

while(temp->info!=x && temp!=NULL)

temp=temp->rptr;

if(temp==NULL) //If Node not found in the List

{

/\* gotoxy(1,13);

delline();

gotoxy(1,13);

delline();

gotoxy(1,14);

delline();

gotoxy(1,16);

delline();

gotoxy(1,17);

delline();

gotoxy(1,18);

delline();

gotoxy(10,13);

printf("\n\tNode not found.");

chk=1;

return; \*/

}

temp->lptr->rptr=temp->rptr;

temp->rptr->lptr=temp->lptr;

free(temp);

return;

}

void display()

{

struct node \*temp;

if(l==NULL) //if(r==NULL)

printf("\n\n\tDoubly Linked List is Empty.");

else

{

printf("\n\nDoubly Linked List:\n\n\n");

printf("l = %u\n\n",l);

temp=l;

while(temp!=NULL)

{

printf("[%u|%u|%d|%u]->",temp,temp->lptr,temp->info,temp->rptr);

temp=temp->rptr;

}

printf("NULL");

printf("\n\nr = %u",r);

}

}