```
/*INSERTION AND DELETION IN Doubly LINKED LIST*/

#include<stdio.h>

#include<conio.h>

void Insertion_at_beg();

void Insertion_at_end();

void Insertion_at_specific();

void Deletion_from_beg();

void Deletion_from_end();

void Deletion_from_specific();

void display();


struct node

{

int data;

struct node *next;

struct node *previous

}

struct node *first=NULL;

Struct node *last=NULL;

void main()

{

int ch;

clrscr();

printf("1.Insertion at beg \n

        2.Insertion at end \n
```

```
        3.Insertion at specific position
        4.Deletion from beg
        5.Deletion from end
        6.Deletion from specific
        7.Display\n
        8.Exit");
do
{
printf("\nEnter ur choice: ");
scanf("%d",&ch);
switch(ch)
{
case 1:Insertion_at_beg();break;
case 2:Insertion_at_end();break;
case 3:Insertion_at_specific();break;
case 4:Deletion_from_beg();break;
case 5:Deletion_from_end();break;
case 6:Deletion_from_specific;break;
case 7:Display();break;
case 8:printf("Program Exited");break;
default:printf("Invalid choice");
}
}while(ch!=8);
getch();
}
```

```
void Insertion_at_beg()
{
struct node *ptr;
printf(“Enter the item”);
scanf(“%d”,&item);
ptr=(struct node *)malloc(sizeof(struct node));
ptr->data=item;
ptr->next=NULL;
ptr->previous=NULL;

if(first==NULL)
first=Last=ptr;

else
{first->previous=ptr;
ptr->next=first;
first=ptr;
}
printf(“Inserted Element is %d”,item);
}


void Insertion_at_end()
{
```

```
struct node *ptr,*temp;
printf(“Enter the item”);
scanf(“%d”,&item);
ptr=(struct node *)malloc(sizeof(struct node));
ptr->data=item;
ptr->next=NULL;
if(first==NULL)
first=Last=ptr;
else
{
last->next=ptr;
ptr->previous=last;
last=ptr;
}

void insert_specific()
 {
        int n;
        struct node *nw, *ptr;
if (first == NULL)
 printf("\n\nLinked list is empty.
It must have at least one node.\n");
 else
 {
        printf("\n\nEnter INFO after which new node
```

```
is to be inserted: ");
            scanf("%d", &n);
            printf("\n\nEnter ITEM: ");
            scanf("%d", &item);
            ptr = first;
    nw = (struct node *)malloc(sizeof(struct node));
                        nw->data=item;
            nw->next=null;
            nw->previous=null;

 while (ptr-> != NULL)
            {
            if (ptr->data == n)
            {
                        nw->next = ptr->next;
                        nw->previous=ptr;
                  ptr->next->previous=nw;
                        ptr->next = nw;
                        printf("\n\nItem inserted: %d", item);

            }
 else
 ptr = ptr->next;

}
```

```
}
}


void delete_beg()
{
 int x;
struct node*ptr;
if(first==NULL)
printf(“Linked is empty”);
else
{
 ptr=first;
x=ptr->data;
first=first->next;
first->previous=NULL;
}
printf(“The deleted element is  %d”,x);
free(ptr);
}
}

void delete_end()
{
struct node *ptr;
```

```
int x;
if(last==NULL)
printf(“Linked list is empty”);
else
{
ptr=last;
x=ptr->data;
last=last->previous;
last->next=NULL;
}
printf(“Deleted element %d”,x);
free(ptr);
}
```

```
Void delete_any()
{
int key;
if(first->next==NULL)
    printf(“\n Empty Linked list. Deletion not possible.”);
else
{
printf(“Enter data of node to be deleted.”);
scanf(“%d”,&key);
```

```
ptr=first;
while((ptr->next !=NULL)||(prt->data !=key))
{
ptr1=ptr;
ptr=ptr->next;
}
if(ptr->data = = key)
{
ptr1->next=ptr->next;
ptr->next->previous=ptr->previous;
free(ptr);
printf(“\n node with data %d deleted.”,key);
}
else
printf(“no such data found”);
}
}
void displayforward()
{
struct node *ptr;

if(first==NULL)
printf("Double linked list is empty");
else
```

```
{
ptr=first;
while(ptr!=NULL)
{
printf(data is%d\n"ptr->data);
ptr=ptr->next;
}
}
}

void displaybackward()
{
struct node *ptr;
if(last==NULL)
printf("Double linked list is empty");
else
{
ptr=last;
while(ptr!=NULL)
{
printf(data is%d\n"ptr->data);
ptr=ptr->previous;
}
}
}
```