**Program 21**

**To implement doubly linked list**

Algorithm

Step 1:-Create a node using malloc function.

Step 2:-[insertion]Insertion can be performed at both the location i.e at the beginning and the end

Step 3:-[Deletion]The deletion application can be performed at the beginning as well as the end of the node.

Step 4:-if the End of the Node does not exist then underflow condition can be applied.

Step 5:-End

Source code

#include<stdio.h>

struct circular

{

int i;

struct circular \*front;

struct circular \*back;

};

struct circular \*temp;

struct circular \*head;

struct circular \*p;

struct circular \*mid;

struct circular \*move;

int cnt=0;

void create(void);

void insert(void);

void display(void);

void del(void);

void main()

{

int ch=0;

while(ch!=5)

{

printf("\n1.CREATE");

printf("\n2.INSERT");

printf("\n3.DELETE");

printf("\n4.DISPLAY");

printf("\n5.EXIT");

scanf("%d",&ch);

if(ch==1)

{

create();

cnt++;

cnt++;

}

if(ch==2)

{

insert();

cnt++;

}

if(ch==3)

{

del();

cnt--;

}

if(ch==4)

{

display();

}

if(ch==5)

{

break;

}

}

}

void create()

{

head=(struct circular \*)malloc(sizeof(struct circular));

head->back=head;

head->front=head;

printf("ENETER THE DATA");

scanf("%d",&head->i);

temp=head;

temp->back=(struct circular \*)malloc(sizeof(struct circular));

temp=temp->back;

temp->back=head;

head->front=temp;

printf("ENETER THE DATA");

scanf("%d",&temp->i);

}

void insert()

{

int add,t;

printf("\n\t ENTER ANY NUMBER BETWEEN 1 AND %d",cnt);

scanf("%d",&add);

p=head;

t=1;

while(t<add)

{

p=p->back;

t++;

}

mid=(struct circular \*)malloc(sizeof(struct circular));

printf("\n\n\nENETER THE DATA");

scanf("%d",&mid->i);

mid->back=p->back;

p->back=mid;

p->back->front=mid;

mid->front=p;

}

void display()

{

p=head;

printf("%\nd-->",p->i);

p=p->back;

while(p!=head)

{

printf("\n%d-->",p->i);

p=p->back;

}

}

void del(void)

{

int add,t;

printf("\n\n\t ENTER ANY NUMBER BETWEEN 1 AND %d",cnt);

scanf("%d",&add);

p=head;

t=1;

while(t<add-1)

{

p=p->back;

t++;

}

mid=p->back;

p->back=mid->back;

mid->back->front=p;

free(mid);

}



