**CIRCULAR LINKED LIST:**

#include<conio.h>

#include<process.h>

#define NULL 0

struct node

{

int info;

struct node \*link;

};

typedef struct node \*NODE;

NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));

if(x==NULL)

{

printf("mem full\n");

exit(0);

}

return x;

}

void freenode(NODE x)

{

free(x);

}

NODE cir\_insert\_front(NODE last,int item)

{

NODE temp;

temp=getnode();

temp->info=item;

if(last==NULL)

last=temp;

temp->link=last->link;

last->link=temp;

return last;

}

NODE cir\_insert\_rear(NODE last,int item)

{

NODE temp;

temp=getnode();

temp->info=item;

if(last==NULL)

last=temp;

else

temp->link=last->link;

last->link=temp;

return temp;

}

NODE cir\_delete\_front(NODE last)

{

NODE temp,first;

if(last==NULL)

{

printf("list empty\n");

return NULL;

}

if(last->link==last)

{

printf("item deleted is %d\n",last->info);

freenode(last);

return NULL;

}

first=last->link;

last->link=first->link;

printf("item deleted at front end is %d\n",first->info);

freenode(first);

return last;

}

NODE cir\_delete\_rear(NODE last)

{

NODE prev;

if(last==NULL)

{

printf("list empty\n");

return NULL;

}

if(last->link==last)

{

printf("item deleted is %d\n",last->info);

freenode(last);

return NULL;

}

prev=last->link;

while(prev->link!=last)

{

prev=prev->link;

}

prev->link=last->link;

printf("item deleted at rear end is %d\n",last->info);

freenode(last);

return prev;

}

void display(NODE last)

{

NODE temp;

if(last==NULL)

{

printf("list empty\n");

return;

}

printf("contents of circular list are:\n");

for(temp=last->link;temp!=last;temp=temp->link)

{

printf("%d\n",temp->info);

}

printf("%d\n",temp->info);

}

void main()

{

int item,choice;

NODE last=NULL;

for(;;)

{

printf("\n1:cir\_insert\_front\t2:cir\_insert\_rear\t3:cir\_delete\_front\t4:cir\_delete\_rear\t5:display\t6:exit\n");

printf("enter the choice: \t");

scanf("%d",&choice);

switch(choice)

{

case 1:printf("enter the item at front end:\t");

scanf("%d",&item);

last=cir\_insert\_front(last,item);

break;

case 2:printf("enter the item at rear end:\t");

scanf("%d",&item);

last=cir\_insert\_rear(last,item);

break;

case 3:last=cir\_delete\_front(last);

break;

case 4:last=cir\_delete\_rear(last);

break;

case 5:display(last);

break;

default:exit(0);

}

}

}