```
//DOUBLY LINKED LIST
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<stdlib.h>
struct node
{
        int info;
        struct node *llink;
        struct node *rlink;
        };
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 dinsert_front(int item,NODE head)
{
NODE temp,cur;
temp=getnode();
```

```
temp->info=item;
cur=head->rlink;
head->rlink=temp;
temp->llink=head;
temp->rlink=cur;
cur->llink=temp;
return head;
}
NODE dinsert_leftpos(int item,NODE head ,int pos){
  NODE temp,cur,perv;temp=getnode();temp->info=item;
  int i=1;
  cur=head->rlink;
  perv=NULL;
  while(i<pos && cur!=head){
    perv =cur;
  cur=cur->rlink;i++;
  }
  if(cur==head)
  {
   printf("POSITION not found\n");
   return head;
   }
  perv ->rlink=temp;
  temp->rlink=cur;
  temp->llink=perv;
  cur->llink =temp;
  return head;
}
NODE dinsert_rear(int item,NODE head)
{
NODE temp,cur;
```

```
temp=getnode();
temp->info=item;
cur=head->llink;
head->llink=temp;
temp->rlink=head;
temp->llink=cur;
cur->rlink=temp;
return head;
}
NODE ddelete_front(NODE head)
{
NODE cur,next;
if(head->rlink==head)
{
printf("dq empty\n");
return head;
}
cur=head->rlink;
next=cur->rlink;
head->rlink=next;
next->llink=head;
printf("the node deleted is %d",cur->info);
freenode(cur);
return head;
}
NODE ddelete_rear(NODE head)
{
NODE cur,prev;
if(head->rlink==head)
{
printf("dq empty\n");
```

```
return head;
}
cur=head->llink;
prev=cur->llink;
head->llink=prev;
prev->rlink=head;
printf("the node deleted is %d",cur->info);
freenode(cur);
return head;
}
void display(NODE head)
{
NODE temp;
if(head->rlink==head)
{
printf("dq empty\n");
return;
}
printf("contents of dq\n");
temp=head->rlink;
while(temp!=head)
{
printf("%d \t",temp->info);
temp=temp->rlink;
}
printf("\n");
}
void main()
{
NODE head,last;
int item,pos, choice;
```

```
head=getnode();
head->rlink=head;
head->llink=head;

for(;;)
{
        printf("\n1:insert front\t2:insert rear\t3:delete front\t4:delete rear\t5:display\t6:left-side-insert\t7:exit\n");
        printf("enter the choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
                case 1: printf("enter the item at front end\n");
                        scanf("%d",&item);
                        last=dinsert_front(item,head);
                        break;
                case 2: printf("enter the item at rear end\n");
                        scanf("%d",&item);
                        last=dinsert_rear(item,head);
                        break;
                case 3:last=ddelete_front(head);
                        break;
                case 4: last=ddelete_rear(head);
                        break;
                case 5: display(head);
                        break;
    case 6:  printf("enter the item at left side pos to entered\n");
                        scanf("%d",&item);
     printf("POSITION\t");
                        scanf("%d",&pos);
                        last=dinsert_leftpos(item,head,pos);
```

```
                    break;
                default:exit(0);
                }
        }
getch();
}
```