```
#include<stdio.h>
#include<stdlib.h>

struct node {
 int info;   struct node *rlink;
 struct node *llink; };

typedef struct node *NODE;

NODE ~~getn~~ getnode ( ) {
  NODE x;
  x=(NODE) malloc (sizeof(struct node));
  if (x==NULL){
   printf("memory full");
   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_rear(NODE head, int item){
{ 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("List empty");
    return head; }
cur = head->rlink;
next = cur->rlink;
head->rlink = next;
next->llink = head;
printf("node deleted is %d", cur->info);
freenode(cur);
  return head; }

NODE ddelete_rear(NODE head){
 NODE cur, prev;
 if(head->rlink == head)
   { printf("List empty"); return head; }
  cur = head->llink;
  prev = cur->llink;
  head->llink = prev;
  prev->rlink= head;
printf("Item deleted is %d", cur->info);
freenode(cur);
return head; }
```

```
void display(NODE head){
NODE temp; temp = head
if(head->rlink==head)
 printf("list empty");
for(temp=head->rlink; temp!=NULL; temp=temp->rlink)
   printf("\n %d", temp->info);
}

NODE dinsert_pos(int item, NODE head, int pos){
{ NODE temp, cur, prev;
  temp=getnode();
  temp->info=item;
  int i=1;
  cur=head->rlink;
  prev=NULL;
  while(i<pos && cur!=head){
  prev=cur;
   cur=cur->rlink;
    i++; }
  if(cur==head){ printf("return pos. not found");
     return head;}
     prev->rlink=temp;
     temp->rlink=cur;
     temp->llink=prev;
     cur->llink=temp;
     return head;
}
```

```
void main()
{
node *head = last;
int item, choice;
head = getnode();
head -> next = head;
head -> link = head;
for(;;)
{
printf("Enter your choice. 1->1. Insert front
2-> Insert rear 3-> delete front 4-> delete rear
5-> display 6-> insert left to");
scanf("%d", &choice);
switch(ch)
{
case 1: printf("Enter item to be inserted");
        scanf("item%d", &item);
        last = insert_front(head, item);
        break;
case 2: printf("Enter item to be inserted");
        scanf("%d", &item);
        last = insert_rear(head, item);
        break;
case 3: last = delete_front(head); break;
case 4: last = delete_rear(head); break;

case5: display(head); break;
case 6: printf("Enter the item and position");
        scanf("%d %d", &item, &pos);
        last = insert_pos(item, head);
        break;
}}}
```