Double ended queue :

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#define qsize 5
int f=0, r=-1, ch;
int item, q[10];
int full(){
    return (r==qsize-1)?1:0;}
int empty(){
    return(f>r)?1:0;}
void insert_rear(){
    if(full()){
        printf("q overflow");
        return;}
    r=r+1;
    q[r]=item;}
void delet_front(){
    if(empty()){
        printf("empty");
        return;}
    printf("deleted %d", q[(f)++]);
    if(f>r){
        f=0;
        r=-1;}}
```

```
void insert_front(){
  if (f != 0) {
    f = f-1;
    q[f] = item;
    return; }
  else if ((f == 0) && (r == -1)){
    q[f+r] = item;
    return; }
  else
    Printf("Insertion not possible"); }
```

```
void delete_rear(){
  if (empty()){
    printf("Empty");
    return; }
  for(i=f; i<=r; i++)
    printf("%d\n", q[i]); }
```

```
void delete_rear() {
  if (empty()) {
    printf("Empty");
    return; }
  Printf("deleted %d", q[r]);
  if (f > r) {
    f = 0;
    r = -1; } }
```

```
void display(){
  int i;
  if (empty()){
    printf("Empty");
    return; }
  for(i=f; i<=r; i++)
    Printf("%d ", q[
```

```
void main(){
  clrscr();
  for(;;){
    printf("1.ins_r\n2.ins_f\n3.del_r\n4.del_f\n5.dis
    scanf("%d", &ch);
  switch(ch){
    case1: printf("Enter item");
           scanf("%d", &item);
           insert_rear(); break;
    case2: printf("Enter item");
           scanf("%d", &item);
           insert_front(); break;
    case3: delete_rear(); break;
    case4: delete_front(); break;
    case5: display(); break;
    default: exit(0); } }
```

Circular queue

```
#include<stdio.h>
#define MAX 5
int queue[MAX];
int front=-1, rear=-1;
void insert();
int delete();
void display();
int main(){
 int option, val;
 printf("1.Insert\n2.Delete\n3.Display\nEnter\n");
 scanf("%d",&op);
 switch(op){
   case 1: insert(); break;
   case 2: delete(); break;
   case 3: display(); break;}
 while(op!=5);
                return 0;
}
void insert(){
  int num;
  printf("Enter no. to be added in");
  scanf("%d",&num);
  if(front==0 && rear==MAX-1)
        printf("Overflow\n");
  else if(front==-1 && rear==-1){
     front=0; rear=0;
      queue[rear]=num;}
  else if(rear==MAX-1 && front!=0){
     rear=0; queue[rear]=num;}
  else{
     rear++; queue[rear]=num;}}
```

```
int delete() {
  int val;
  if (front == -1 && rear == -1) {
    printf("underflow");
    return -1; }
  val = queue[front];
  if (front == rear) {
      front = -1; rear = -1; }
  else {
      if (front = max-1)
            front = 0;
      else
            front++ }
  return val; }
void display() {
  int i; printf("\n");
  if (front == -1 && rear == -1)
      printf("empty \n");
  else {
      if (front < rear) {
        for (i = front; i < MAX; i++)
            printf("%d ", queue[i]) }
      else {
        for (i = front; i < MAX; i++)
            printf("%d", queue[i]);
        for (i = 0; i < rear; i++)
            printf("%d", queue[i]);
      } } }
```