Circular Queue -

```
#include <stdio.h>
#include <conio.h>
#define qsize 5
int f=0, r=-1, ch;
int item, q[10];
int isfull(){
  return (r==qsize-1)?1:0;
}
int isempty(){
  return (f<r)?1:0;
}
void insert_rear(){
 if (isfull()){
   printf("queue overflow\n");
   return;
 }
 r=r+1;
 q[r]= item;
}
void delete_front(){
 if (isempty()){
   printf("queue empty \n");
   return;
 }
 printf("item 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[++r]=item;
  return;
}
else
  printf("insertion not possible");
}
void delete_rear() {
  if (isempty()) {
    printf("queue is empty\n");
    return;
  }
  printf("item deleted=%d", q[r--]);
  if (f>r) {
      f=0
      r=-1;
    }
}
void display() {
  int i;
  if (isempty()) {
    printf("queue empty\n");
    return;
  }
  for(i=f;i<=r;i++)
    printf("%d", q[i]);
}
```

```
int main() {
for (;;) {
 printf ("1. insert rear \n 2. insert front \n 3.
      delete rear \n 4. delete front \n 5. display
           \n 6. exit \n");
 scanf ("%d", &ch);
 switch (ch) {
  case 1: printf ("Enter item \n");
       scanf ("%d, & item);
       insert - rear ();
       printf ("------ \n");
       break;
  case 2: printf ("Enter item \n");
       scanf ("%d", & item);
       insert - front ();
       printf ("------ \n");
       break.
  case 3: delete - rear ();
       printf ("------ \n");
       break;
  case 4: delete - front ();
       printf ("------ \n");
       break;
  case 5: display ();
       printf ("------- \n");
       break;
  default : break;
    }
  }
  return 0;
}
```

```c
#include <stdio.h>
#include <conio.h>
#define qsize 5
int f=0,r=-1,ch;
int item,q[10];

int isfull()
  {
   return(r==qsize-1)?1:0;
  }
int isempty()
  {
   return(f>r)?1:0;
  }
void insert_rear()
  {
   if(isfull())
     {
      printf("queue overflow\n");
      return;
     }
   r=r+1;
   q[r]=item;
  }
void delete_front()
  {
   if(isempty())
     {
      printf("queue empty\n");
      return;
     }
   printf("item deleted is %d\n",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[++(r)]=item;
    return;
   }
  else
    printf("insertion not possible\n");
}
void delete_rear()
{
 if(isempty())
   {
    printf("queue is empty\n");
    return;
   }
 printf("item deleted is %d\n",q[(r)--]);
 if(f>r)
   {
    f=0;
    r=-1;
   }
}
void display()
{
 int i;
 if(isempty())
   {
    printf("queue empty\n");
    return;
```

```
      return;
    }
   for(i=f;i<=r;i++)
    printf("%d\n",q[i]);
  }
int main()
 {
  for(;;)
   {
    printf("1.insert_rear\n2.insert_front\n3.delete_rear\n4.delete_front
    printf("enter choice\n");
    scanf("%d",&ch);
    switch(ch)
      {
       case 1:printf("enter the item\n");
              scanf("%d",&item);
              insert_rear();
              printf("----------------------\n");
              break;
      case 2:printf("enter the item\n");
              scanf("%d",&item);
              insert_front();
              printf("----------------------\n");
              break;
      case 3:delete_rear();
              printf("----------------------\n");
              break;
      case 4:delete_front();
              printf("----------------------\n");
              break;
      case 5:display();
              printf("----------------------\n");
              break;
      default: break;
     }
  }
 return 0;
```

```
1.insert rear
2.insert front
3.delete rear
4.delete front
5.display
6.exit
1
enter item
10
---------
1.insert rear
2.insert front
3.delete rear
4.delete front
5.display
6.exit
1
enter item
20
---------
1.insert rear
2.insert front
3.delete rear
4.delete front
5.display
6.exit
2
Enter the item
30
Insertion not possible
---------
1.insert rear
2.insert front
3.delete
```