Circular Queue Implementation

```
#include <stdio.h>

#define SIZE 5

int items[SIZE];
int front = -1, rear =-1;

int isFull()
{
   if( (front == rear + 1) || (front == 0 && rear == SIZE-1)) return 1;
   return 0;
}

int isEmpty()
{
   if(front == -1) return 1;
   return 0;
}

void enQueue(int element)
{
   if(isFull()) printf("\n Queue is full!! \n");
   else
   {
      if(front == -1) front = 0;
      rear = (rear + 1) % SIZE;
      items[rear] = element;
      printf("\n Inserted -> %d", element);
   }
}


int deQueue()
{
   int element;
   if(isEmpty()) {
      printf("\n Queue is empty !! \n");
      return(-1);
   } else {
      element = items[front];
      if (front == rear){
         front = -1;
         rear = -1;
```

```
        } /* Q has only one element, so we reset the queue after dequeing it. ? */
        else {
            front = (front + 1) % SIZE;

        }
        printf("\n Deleted element -> %d \n", element);
        return(element);
    }
}




void display()
{
    int i;
    if(isEmpty()) printf(" \n Empty Queue\n");
    else
    {
        printf("\n Front -> %d ",front);
        printf("\n Items -> ");
        for( i = front; i!=rear; i=(i+1)%SIZE) {
            printf("%d ",items[i]);
        }
        printf("%d ",items[i]);
        printf("\n Rear -> %d \n",rear);
    }
}

int main()
{
    // Fails because front = -1
    deQueue();

    enQueue(1);
    enQueue(2);
    enQueue(3);
    enQueue(4);
    enQueue(5);

    // Fails to enqueue because front == 0 && rear == SIZE - 1
    enQueue(6);

    display();
    deQueue();
```

```
    display();

    enQueue(7);
    display();

    // Fails to enqueue because front == rear + 1
    enQueue(8);

    return 0;
}
```