Implementation of Circular Queue & Operations

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

struct circularQueue
{
    int size;
    int f;
    int r;
    int* arr;
};



int isEmpty(struct circularQueue *q){
    if(q->r==q->f){
        return 1;
    }
    return 0;
}

int isFull(struct circularQueue *q){
    if((q->r+1)%q->size == q->f){
        return 1;
    }
    return 0;
}

void enqueue(struct circularQueue *q, int val){
    if(isFull(q)){
        printf("This Queue is full");
    }
    else{
        q->r = (q->r +1)%q->size;
        q->arr[q->r] = val;
        printf("Enqueued element: %d\n", val);
    }
}

int dequeue(struct circularQueue *q){
    int a = -1;
    if(isEmpty(q)){
        printf("This Queue is empty");
    }
    else{
        q->f = (q->f +1)%q->size;
        a = q->arr[q->f];
    }
    return a;
}


int main(){
    struct circularQueue q;
    q.size = 4;
    q.f = q.r = 0;
    q.arr = (int*) malloc(q.size*sizeof(int));

    // Enqueue few elements
    enqueue(&q, 12);
    enqueue(&q, 15);
    enqueue(&q, 1);
    printf("Dequeuing element %d\n", dequeue(&q));
    printf("Dequeuing element %d\n", dequeue(&q));
    printf("Dequeuing element %d\n", dequeue(&q));
    enqueue(&q, 45);
    enqueue(&q, 45);
    enqueue(&q, 45);

    if(isEmpty(&q)){
        printf("Queue is empty\n");
    }
    if(isFull(&q)){
        printf("Queue is full\n");
    }

    return 0;
}
```