Input code:

```
#include <stdio.h>

#include <stdlib.h>

#define MAX 5  // Define the size of the queue

// Circular queue structure

typedef struct {

    int arr[MAX];

    int front;

    int rear;

} CircularQueue;

// Initialize the queue

void initQueue(CircularQueue* queue) {

    queue->front = -1;

    queue->rear = -1;

}

// Check if the queue is full

int isFull(CircularQueue* queue) {

    return (queue->rear + 1) % MAX == queue->front;

}

// Check if the queue is empty

int isEmpty(CircularQueue* queue) {

    return queue->front == -1;

}

// Enqueue operation (Insert an element)

void enqueue(CircularQueue* queue, int value) {

    if (isFull(queue)) {

        printf("Queue Overflow! Cannot enqueue %d\n", value);

        return;

    }

    // If queue is empty, initialize front

    if (queue->front == -1) {

        queue->front = 0;
    }
    // Insert element at the rear
    queue->rear = (queue->rear + 1) % MAX;
    queue->arr[queue->rear] = value;
    printf("Enqueued: %d\n", value);
}
// Dequeue operation (Remove an element)
int dequeue(CircularQueue* queue) {
    if (isEmpty(queue)) {
        printf("Queue Underflow! No elements to dequeue\n");
        return -1;
    }
    int value = queue->arr[queue->front];
    // If only one element is left, reset the queue
    if (queue->front == queue->rear) {
        queue->front = -1;
        queue->rear = -1;
    } else {
        queue->front = (queue->front + 1) % MAX;
    }
    return value;
}
// Display the queue contents
void display(CircularQueue* queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty\n");
        return;
    }
    printf("Queue elements: ");
    int i = queue->front;
    while (i != queue->rear) {
        printf("%d ", queue->arr[i]);
        i = (i + 1) % MAX;
    }

    printf("%d\n", queue->arr[queue->rear]);

}

int main() {

    CircularQueue queue;

    initQueue(&queue);

    // Test the Circular Queue

    enqueue(&queue, 10);

    enqueue(&queue, 20);

    enqueue(&queue, 30);

    enqueue(&queue, 40);

    enqueue(&queue, 50);

    display(&queue);

    printf("Dequeued: %d\n", dequeue(&queue));

    printf("Dequeued: %d\n", dequeue(&queue));

    enqueue(&queue, 60);

    display(&queue);

    return 0;

}
```

<u>Output code:</u>

```
Enqueued: 10

Enqueued: 20

Enqueued: 30

Enqueued: 40

Enqueued: 50

Queue elements: 10 20 30 40 50

Dequeued: 10

Dequeued: 20

Enqueued: 60

Queue elements: 30 40 50 60
```