Program 06 : Circular Queue

Develop a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)

a. Insert an Element on to Circular QUEUE
b. Delete an Element from Circular QUEUE
c. Demonstrate Overflow and Underflow situations on Circular QUEUE
d. Display the status of Circular QUEUE
e. Exit
Support the program with appropriate functions for each of the above operations

Program:

```
#include <stdio.h>
#include <stdbool.h>
#define MAX 10 // Maximum size of the Circular Queue
// Global Variables
char queue[MAX];
int front = -1;
int rear = -1;
// Function Prototypes
void enqueue(char element);
void dequeue();
void displayQueue();
bool isFull();
bool isEmpty();
// Main Function
int main() {
   int choice;
   char element;

   while (1) {
      printf("\n--- Circular Queue Operations ---\n");
      printf("1. Insert an Element\n");
      printf("2. Delete an Element\n");
      printf("3. Display Queue\n");
      printf("4. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);
      switch (choice) {
```

```
            case 1:
                printf("Enter a character to insert: ");
                scanf(" %c", &element);
                enqueue(element);
                break;
            case 2:
                dequeue();
                break;
            case 3:
                displayQueue();
                break;
            case 4:
                printf("Exiting...\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
}
// Function Definitions
void enqueue(char element) {
    if (isFull()) {
        printf("Queue Overflow\n");
        return;
    }
    if (front == -1) front = 0;
    rear = (rear + 1) % MAX;
    queue[rear] = element;
    printf("Inserted '%c'\n", element);
}
void dequeue() {
    if (isEmpty()) {
        printf("Queue Underflow\n");
        return;
    }
    printf("Deleted '%c'\n", queue[front]);
    if (front == rear) {
        front = -1;
        rear = -1;
    } else {
```

```
        front = (front + 1) % MAX;
    }
}

void displayQueue() {
    if (isEmpty()) {
        printf("Queue is empty\n");
        return;
    }
    printf("Queue elements: ");
    for (int i = front; i != rear; i = (i + 1) % MAX) {
        printf("%c ", queue[i]);
    }
    printf("%c\n", queue[rear]); // Display the last element
}

bool isFull() {
    return ((rear + 1) % MAX) == front;
}

bool isEmpty() {
    return front == -1;
}
```