```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
// Circular Queue structure
struct CircularQueue {
    char items[MAX];
    int front, rear;
};
// Function prototypes
void initializeQueue(struct CircularQueue *cq);
void insertElement(struct CircularQueue *cq, char element);
void deleteElement(struct CircularQueue *cq);
void displayQueue(struct CircularQueue *cq);
int isQueueFull(struct CircularQueue *cq);
int isQueueEmpty(struct CircularQueue *cq);
int main() {
    struct CircularQueue cq;
    initializeQueue(&cq);
    int choice;
    char element;
    do {
        printf("\nCircular Queue Menu:\n");
        printf("1. Insert Element\n");
        printf("2. Delete Element\n");
        printf("3. Display Queue\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter element to insert: ");
                scanf(" %c", &element);

                if (isQueueFull(&cq)) {
                    printf("Queue is full. Cannot insert 
element.\n");
                } else {
                    insertElement(&cq, element);
                    printf("%c inserted into the queue.\n", 
element);
                }
                break;
            case 2:
                if (isQueueEmpty(&cq)) {
                    printf("Queue is empty. Cannot delete
element.\n");
                } else {
                    deleteElement(&cq);
                    printf("Element deleted from the queue.
\n");
                }
                break;
            case 3:
                displayQueue(&cq);
                break;
            case 4:
                printf("Exiting program.\n");
                break;
            default:
                printf("Invalid choice. Please enter a valid
option.\n");
        }
    } while (choice != 4);
    return 0;
}
// Function to initialize the Circular Queue
void initializeQueue(struct CircularQueue *cq) {
    cq->front = cq->rear = -1;
}
// Function to insert an element into the Circular Queue
void insertElement(struct CircularQueue *cq, char element) {
    if (isQueueEmpty(cq)) {
        cq->front = cq->rear = 0;
    } else {
        cq->rear = (cq->rear + 1) % MAX;
    }
    cq->items[cq->rear] = element;
}
// Function to delete an element from the Circular Queue
void deleteElement(struct CircularQueue *cq) {
    char deletedElement = cq->items[cq->front];
    if (cq->front == cq->rear) {
        // Only one element in the queue
        initializeQueue(cq);
    } else {
        cq->front = (cq->front + 1) % MAX;
    }
    printf("Deleted element: %c\n", deletedElement);
}
// Function to display the Circular Queue
```

```c
void displayQueue(struct CircularQueue *cq) {
    if (isQueueEmpty(cq)) {
        printf("Queue is empty.\n");
    } else {
        int i = cq->front;
        printf("Circular Queue: ");
        do {
            printf("%c ", cq->items[i]);
            i = (i + 1) % MAX;
        } while (i != (cq->rear + 1) % MAX);
        printf("\n");
    }
}
// Function to check if the Circular Queue is full
int isQueueFull(struct CircularQueue *cq) {
    return (cq->front == (cq->rear + 1) % MAX);
}
// Function to check if the Circular Queue is empty
int isQueueEmpty(struct CircularQueue *cq) {
    return (cq->front == -1 && cq->rear == -1);
}
```