3 b) WAP to simulate the working of a circular queue of integers using an array. Provide the following operations: Insert, Delete & Display The program should print appropriate messages for queue empty and queue overflow conditions

```
#include <stdio.h>

#include <stdlib.h>


#define MAX 50


int queue[MAX];

int front = -1, rear = -1;

int size;


void insert(int element) {

   if ((front == 0 && rear == size - 1) || (rear + 1) % size == front) {

      printf("Queue Overflow! Cannot insert %d\n", element);

      return;

   }


   if (front == -1) { // Queue empty

      front = 0;

      rear = 0;

   } else {

      rear = (rear + 1) % size;

   }


   queue[rear] = element;

   printf("%d inserted into queue.\n", element);

}
```

```
void delete() {
    if (front == -1) {
        printf("Queue Underflow! Queue is empty.\n");
        return;
    }

    int deleted = queue[front];
    if (front == rear) { // Only one element
        front = -1;
        rear = -1;
    } else {
        front = (front + 1) % size;
    }

    printf("%d deleted from queue.\n", deleted);
}

void display() {
    if (front == -1) {
        printf("Queue is empty.\n");
        return;
    }

    printf("Queue elements: ");
    int i = front;
    while (1) {
        printf("%d ", queue[i]);
```

```
        if (i == rear) break;
        i = (i + 1) % size;
    }
    printf("\n");
}

int main() {
    int choice, element;

    printf("Enter capacity of circular queue: ");
    scanf("%d", &size);

    while (1) {
        printf("\nCircular Queue Operations:\n");
        printf("1. Insert\n2. Delete\n3. Display\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter element to insert: ");
                scanf("%d", &element);
                insert(element);
                break;
            case 2:
                delete();
                break;
            case 3:
```

```
            display();
            break;
        case 4:
            printf("Exiting program.\n");
            exit(0);
        default:
            printf("Invalid choice! Try again.\n");
    }
}

return 0;
```

```
Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
Queue elements: 10 20 40

Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 32
Invalid choice! Try again.

Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2
10 deleted from queue.

Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
Queue elements: 20 40
```

```
Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 4
Exiting program.

Process returned 0 (0x0)   execution time : 116.308 s
Press any key to continue.




Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 4
Exiting program.

Process returned 0 (0x0)   execution time : 116.308 s
Press any key to continue.
```

```
Enter capacity of circular queue: 3

Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter element to insert: 10
10 inserted into queue.

Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter element to insert: 20
20 inserted into queue.

Circular Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter element to insert: 40
40 inserted into queue.
```