//Assignment 3

```
#include <iostream>
using namespace std;
#define SIZE 5  // Fixed size of the queue

class CircularQueue {
   int arr[SIZE];
   int front, rear;

public:
   CircularQueue() {
      front = -1;
      rear = -1;
   }

   // Check if queue is full
   bool isFull() {
      return ((front == 0 && rear == SIZE - 1) || (rear + 1 == front));
   }

   // Check if queue is empty
   bool isEmpty() {
      return (front == -1);
   }

   // Enqueue operation
   void enqueue(int value) {
      if (isFull()) {
         cout << "Queue is FULL! Cannot insert " << value << endl;
         return;
      }

      if (front == -1)  // First element
         front = 0;

      rear = (rear + 1) % SIZE; // Circular increment
      arr[rear] = value;
```

```
    cout << "Inserted: " << value << endl;
}

// Dequeue operation
void dequeue() {
    if (isEmpty()) {
        cout << "Queue is EMPTY! Cannot delete." << endl;
        return;
    }

    cout << "Deleted: " << arr[front] << endl;

    if (front == rear) {
        // Queue has only one element
        front = -1;
        rear = -1;
    } else {
        front = (front + 1) % SIZE; // Circular increment
    }
}

// Display queue
void display() {
    if (isEmpty()) {
        cout << "Queue is EMPTY!" << endl;
        return;
    }

    cout << "Queue elements: ";
    int i = front;
    while (true) {
        cout << arr[i] << " ";
        if (i == rear)
            break;
        i = (i + 1) % SIZE;
    }
    cout << endl;
}
```

```
};

int main() {
    CircularQueue q;
    int choice, value;

    do {
        cout << "\n--- Circular Queue Menu ---\n";
        cout << "1. Enqueue (Insert)\n";
        cout << "2. Dequeue (Delete)\n";
        cout << "3. Display\n";
        cout << "4. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter value to insert: ";
                cin >> value;
                q.enqueue(value);
                break;
            case 2:
                q.dequeue();
                break;
            case 3:
                q.display();
                break;
            case 4:
                cout << "Exiting program..." << endl;
                break;
            default:
                cout << "Invalid choice! Try again." << endl;
        }
    } while (choice != 4);

    return 0;
}
```

--- Circular Queue Menu ---
1. Enqueue (Insert)
2. Dequeue (Delete)
3. Display
4. Exit
Enter choice: 1
Enter value to insert: 99
Inserted: 99

--- Circular Queue Menu ---
1. Enqueue (Insert)
2. Dequeue (Delete)
3. Display
4. Exit
Enter choice: 1
Enter value to insert: 55
Inserted: 55

--- Circular Queue Menu ---
1. Enqueue (Insert)
2. Dequeue (Delete)
3. Display
4. Exit
Enter choice: 1
Enter value to insert: 102
Inserted: 102

--- Circular Queue Menu ---
1. Enqueue (Insert)
2. Dequeue (Delete)
3. Display
4. Exit
Enter choice: 1
Enter value to insert: 55
Inserted: 55

--- Circular Queue Menu ---

1. Enqueue (Insert)
2. Dequeue (Delete)
3. Display
4. Exit
Enter choice: 3
Queue elements: 99 55 102 55

--- Circular Queue Menu ---
1. Enqueue (Insert)
2. Dequeue (Delete)
3. Display
4. Exit
Enter choice: 2
Deleted: 99

--- Circular Queue Menu ---
1. Enqueue (Insert)
2. Dequeue (Delete)
3. Display
4. Exit
Enter choice: 3
Queue elements: 55 102 55

--- Circular Queue Menu ---
1. Enqueue (Insert)
2. Dequeue (Delete)
3. Display
4. Exit
Enter choice: 1
Enter value to insert: 111
Inserted: 111

--- Circular Queue Menu ---
1. Enqueue (Insert)
2. Dequeue (Delete)
3. Display
4. Exit
Enter choice: 3

Queue elements: 55 102 55 111
--- Circular Queue Menu ---
1. Enqueue (Insert)
2. Dequeue (Delete)
3. Display
4. Exit
Enter choice: 4