```
//Name: 3. Circular Queue
#include <iostream>
 using namespace std;
 class CircularQueue {
 private:
   int front, rear, size;
   int* queue;
   const int maxSize;
 public:

   CircularQueue(int size) : maxSize(size), front(-1), rear(-1) {
      queue = new int[maxSize];
   }
   bool isFull() {
      return (rear + 1) % maxSize == front;
   }

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

   void enqueue(int value) {
      if (isFull()) {
         cout << "Queue is full! Cannot enqueue " << value << endl;
```

```
        return;
    }
    if (isEmpty()) {
        front = 0;
    }
    rear = (rear + 1) % maxSize;
    queue[rear] = value;
    cout << "Enqueued: " << value << endl;
}

int dequeue() {
    if (isEmpty()) {
        cout << "Queue is empty! Cannot dequeue." << endl;
        return -1;
    }
    int value = queue[front];
    if (front == rear) {
        front = rear = -1;
    } else {
        front = (front + 1) % maxSize;
    }
    return value;
}
```

```
  void display() {
    if (isEmpty()) {
      cout << "Queue is empty." << endl;
      return;
    }
    cout << "Circular Queue: ";
    int i = front;
    while (true) {
      cout << queue[i] << " ";
      if (i == rear) break;
      i = (i + 1) % maxSize;
    }
    cout << endl;
  }

  ~CircularQueue() {
    delete[] queue;
  }
};
int main() {
  CircularQueue cq(5);
  cq.enqueue(10);
  cq.enqueue(20);
  cq.enqueue(30);
```

```
    cq.enqueue(40);
    cout << "Dequeued: " << cq.dequeue() << endl;
    cout << "Dequeued: " << cq.dequeue() << endl;
    cq.display();
    return 0;
}
```