1. **Global Definitions:**
   * **SIZE** is defined as **3**.
   * **Queue** is an array used to implement the queue.
   * **front** and **rear** are initialized to **-1** to signify an empty queue.
2. **isEmpty Function:**
   * Checks if the queue is empty based on the positions of **front** and **rear**.
   * Returns **1** if the queue is empty; otherwise, returns **0**.
3. **Qfull Function:**
   * Checks if the queue is full based on the positions of **front** and **rear**.
   * Returns **1** if the queue is full; otherwise, returns **0**.
4. **enque Function:**
   * Enqueues (inserts) an element into the queue.
   * Checks if the queue is not full (**!Qfull()**).
   * If the queue is initially empty, sets **front** to **0**.
   * Increments **rear** and inserts the **data** into the **Queue**.
5. **deque Function:**
   * Dequeues (removes) an element from the front of the queue.
   * Checks if the queue is not empty (**!isEmpty()**).
   * Returns the element at the front of the queue (**Queue[front]**).
   * Increments **front** to remove the front element.
6. **main Function:**
   * Calls **enque** to insert elements **10**, **20**, **30**, and **40** into the queue.
   * Calls **deque** to dequeue and print elements from the front of the queue.
   * The output demonstrates the basic operations of enqueueing and dequeueing in a circular queue.
7. **Note:**
   * The code is implementing a circular queue using an array with predefined size **SIZE**.

Algorithm:

1. \*Include necessary libraries and define constants.\*

2. \*\*Define functions to check if the queue is empty (isEmpty) or full (Qfull).\*\*

3. \*\*Define functions for enqueuing (enque) and dequeuing (deque).\*\*

4. \*\*In the main function, create a circular queue, enqueue elements, and dequeue elements.\*\*