1. **Node Class:**
   * Represents a node with an **int data** attribute for seat number and pointers **next** and **prev** for the next and previous nodes.
2. **createNode Function:**
   * Dynamically allocates memory for a new node and initializes it with the given seat number.
3. **deque Function:**
   * Removes a node from the front of the queue (dequeues).
4. **dequeR Function:**
   * Removes a node from the rear (end) of the queue.
5. **enqueue Function:**
   * Adds a node to the end of the queue (enqueues).
6. **enqueueF Function:**
   * Adds a node to the front of the queue.
7. **printQ Function:**
   * Prints the content of the queue in either forward or reverse direction.
8. **main Function:**
   * Initializes the **front** and **rear** pointers to nodes.
   * Creates a queue with initial seat numbers from the array **arr**.
   * Enqueues a new node with seat number **1000** at the front of the queue.
   * Enqueues a new node with seat number **100** at the rear of the queue.
   * Displays the content of the queue using **printQ**.
   * Performs dequeuing operations on the front of the queue.
   * The output demonstrates the operations of enqueuing at both ends and dequeuing from the front.
9. **Note:**
   * The code is demonstrating basic queue operations using a doubly linked list.

Algorithm:

1. \*\*Include necessary libraries and define the Node class:\*\*

2. \*\*Implement the deque function to dequeue from the front (head) of the queue:\*\*

3. \*\*Implement the dequeR function to dequeue from the rear (end) of the queue:\*\*

4. \*\*Implement the enqueue function to enqueue at the rear of the queue:\*\*

5. \*\*Implement the enqueueF function to enqueue at the front of the queue:\*\*

6. \*\*Implement the printQ function to print the queue in both forward and reverse directions:\*\*

7. \*\*Implement the main function to create a circular doubly linked list queue and perform enqueue and dequeue operations:\*\*