Mock Interviews @ ACM

Problem Card

Course Queue

Problem Statement

The OIT admins are finally updating TigerHub with a new front-end... as well as a new back-end! Princeton has acquired a new high-speed computing cluster that runs optimally when most operations are performed in small, localized sections of memory. As the head of technology at OIT, you are assigned the task of implementing a very specific data structure for the course queue—a circular queue.

Design your implementation of the circular queue. The circular queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle.

Implementation the `MyCircularQueue` class:

- `MyCircularQueue(k)` Initializes the object with the size of the queue to be `k`.
- `int Front()` Gets the front item from the queue. If the queue is empty, return `-1`.
- `int Rear()` Gets the last item from the queue. If the queue is empty, return `-1`.
- `boolean enQueue(int value)` Inserts an element into the circular queue. Return `true` if the operation is successful.
- `boolean deQueue()` Deletes an element from the circular queue. Return `true` if the operation is successful.
- `boolean isEmpty()` Checks whether the circular queue is empty or not.
- `boolean isFull()` Checks whether the circular queue is full or not.

Note: You must solve the problem without using the built-in queue data structure in your programming language.

Test Cases

```
// Input
["MyCircularQueue", "enQueue", "enQueue", "enQueue", "enQueue", "Rear",
```

```
"isFull", "deQueue", "enQueue", "Rear"]
[[3], [1], [2], [3], [4], [], [], [], [4], []]
Output
[null, true, true, true, false, 3, true, true, true, 4]

// Explanation
MyCircularQueue myCircularQueue = new MyCircularQueue(3);
myCircularQueue.enQueue(1); // return True
myCircularQueue.enQueue(2); // return True
myCircularQueue.enQueue(3); // return True
myCircularQueue.enQueue(4); // return False
myCircularQueue.Rear();     // return 3
myCircularQueue.isFull();   // return True
myCircularQueue.deQueue();  // return True
myCircularQueue.enQueue(4); // return True
myCircularQueue.Rear();     // return 4
```