원형 큐를 디자인하라.

- 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.

1.

```
class MyCircularQueue:

    def __init__(self, k: int):
        self.q = [None] * k
        self.maxLen = k
        self.p1 = self.p2 = 0


    def enQueue(self, value: int) -> bool:
        if self.q[self.p2] is None:
            self.q[self.p2] = value
            self.p2 = (self.p2 + 1) % self.maxLen
            return True
        return False


    def deQueue(self) -> bool:
        if self.q[self.p1] is not None:
            self.q[self.p1] = None
            self.p1 = (self.p1 + 1) % self.maxLen
            return True
        return False


    def Front(self) -> int:
        return -1 if self.q[self.p1] is None else self.q[self.p1]


    def Rear(self) -> int:
        return -1 if self.q[self.p2 - 1] is None else self.q[self.p2 - 1]


    def isEmpty(self) -> bool:
        return self.p1 == self.p2 and self.q[self.p1] is None


    def isFull(self) -> bool:
```

```
        return self.p1 == self.p2 and self.q[self.p2] is not None



# Your MyCircularQueue object will be instantiated and called as such:
# obj = MyCircularQueue(k)
# param_1 = obj.enQueue(value)
# param_2 = obj.deQueue()
# param_3 = obj.Front()
# param_4 = obj.Rear()
# param_5 = obj.isEmpty()
# param_6 = obj.isFull()

class MyCircularQueue:

    def __init__(self, k: int):
        self.size = k + 1
        self.front = self.rear = -1
        self.q = [0] * self.size


    def enQueue(self, value: int) -> bool:
        if(self.isFull()):
            return False
        self.rear = (self.rear + 1) % self.size
        self.q[self.rear] = value
        return True


    def deQueue(self) -> bool:
        if(self.isEmpty()):
            return False
        self.front = (self.front + 1) % self.size
        return True


    def Front(self) -> int:
        if self.isEmpty():
            return -1
        self.front = (self.front + 1) % self.size
        temp = self.q[self.front]
        self.front = (self.front - 1) % self.size
        return temp


    def Rear(self) -> int:
        if self.isEmpty():
            return -1
        return self.q[self.rear]
```

```
    def isEmpty(self) -> bool:
        return self.front == self.rear


    def isFull(self) -> bool:
        return (self.front + 1) % self.size  == (self.rear + 2) % self.size



# Your MyCircularQueue object will be instantiated and called as such:
# obj = MyCircularQueue(k)
# param_1 = obj.enQueue(value)
# param_2 = obj.deQueue()
# param_3 = obj.Front()
# param_4 = obj.Rear()
# param_5 = obj.isEmpty()
# param_6 = obj.isFull()
```