基于Python实现的循环队列，参考了LeetCode上的代码，使用了不需要少一个元素的储存方式。

主要思想是：将head和tail都设置为-1，进队时，让head和tail都为0。当head和tail相等且不为-1时，说明head和tail都只有一个数在队列中。这时如果要出队，那么head和tail都回到-1，表示队列为空。如果要入队，那么tail按照公式 (tail+1)%队列总长度 将tail增加。同样出队将head增加。

这样的好处是n个长度的队列可以储存n个元素，如果用head和tail等于0的方式那么tail最后需要放在head的后一个位置，否则无法判断队列的空和满。这种方式以head和tail为-1作为空，以tail在head的后一个位置作为满。

```
class MyCircularQueue:
    def __init__(self, k: int):
        """
        Initialize your data structure here.
        Set the size of the queue to be k.
        """
        self.__k = k
        self.__head = -1
        self.__tail = -1
        self.__data = [None] * self.__k

    def enQueue(self, value: int) -> bool:
        """
        Insert an element into the circular queue. Return true
        if the operation is successful.
        """
        if self.isFull():
            return False
        if self.isEmpty():
            self.__head = 0
        self.__tail = (self.__tail + 1) % self.__k
        self.__data[self.__tail] = value
        return True

    def deQueue(self) -> bool:
        """
        Delete an element from the circular queue. Return true
        if the operation is successful.
        """
        if self.isEmpty():
            return False
        if self.__head == self.__tail:
            self.__head, self.__tail = -1, -1
            return True
```

```
        self.__head = (self.__head + 1) % self.__k
        return True

    def Front(self):
        """
        Get the front item from the queue.
        """
        if self.isEmpty() is True:
            return None
        return self.__data[self.__head]

    def Rear(self):
        """
        Get the last item from the queue.
        """
        if self.isEmpty() is True:
            return None
        return self.__data[self.__tail]

    def isEmpty(self) -> bool:
        """
        Checks whether the circular queue is empty or not.
        """
        if self.__head == -1:
            return True
        return False

    def isFull(self) -> bool:
        """
        Checks whether the circular queue is full or not.
        """
        if (self.__tail + 1) % self.__k == self.__head:
            return True
        return False


if __name__ == '__main__':

    queue = MyCircularQueue(5)  # 定义一个大小为5的队列
    for i in range(3):  # 进队三次
        queue.enQueue(i)

    print(queue.Front())
    print(queue.Rear())

    for i in range(2):  # 出队两次
        queue.deQueue()
```

```
print(queue.Front())
print(queue.Rear())

for i in range(5):  # 进队五次（最后一次因为满了进不去）
    queue.enQueue(i)

print(queue.Front())
print(queue.Rear())
```