641. Design Circular Deque

Design your implementation of the circular double-ended queue (deque).

Implement the `MyCircularDeque` class:

- `MyCircularDeque(int k)` Initializes the deque with a maximum size of `k`.
- `boolean insertFront()` Adds an item at the front of Deque. Returns `true` if the operation is successful, or `false` otherwise.
- `boolean insertLast()` Adds an item at the rear of Deque. Returns `true` if the operation is successful, or `false` otherwise.
- `boolean deleteFront()` Deletes an item from the front of Deque. Returns `true` if the operation is successful, or `false` otherwise.
- `boolean deleteLast()` Deletes an item from the rear of Deque. Returns `true` if the operation is successful, or `false` otherwise.
- `int getFront()` Returns the front item from the Deque. Returns `-1` if the deque is empty.
- `int getRear()` Returns the last item from Deque. Returns `-1` if the deque is empty.
- `boolean isEmpty()` Returns `true` if the deque is empty, or `false` otherwise.
- `boolean isFull()` Returns `true` if the deque is full, or `false` otherwise.

Example 1:

```
Input
["MyCircularDeque", "insertLast", "insertLast", "insertFront",
"insertFront", "getRear", "isFull", "deleteLast", "insertFront", "getFront"]
[[3], [1], [2], [3], [4], [], [], [], [4], []]
Output
[null, true, true, true, false, 2, true, true, true, 4]

Explanation
MyCircularDeque myCircularDeque = new MyCircularDeque(3);
myCircularDeque.insertLast(1);  // return True
myCircularDeque.insertLast(2);  // return True
myCircularDeque.insertFront(3); // return True
myCircularDeque.insertFront(4); // return False, the queue is full.
myCircularDeque.getRear();      // return 2
myCircularDeque.isFull();       // return True
myCircularDeque.deleteLast();   // return True
```

```
myCircularDeque.insertFront(4); // return True
myCircularDeque.getFront();     // return 4
```

Constraints:

- `1 <= k <= 1000`
- `0 <= value <= 1000`
- At most `2000` calls will be made to `insertFront`, `insertLast`, `deleteFront`, `deleteLast`, `getFront`, `getRear`, `isEmpty`, `isFull`.

```
class Node:
    def __init__(self,val):
        self.val = val
        self.prev = None
        self.next = None

class MyCircularDeque:

    def __init__(self, k: int):
        self.size = 0
        self.maxSize = k
        self.head = None
        self.tail = None

    def insertFront(self, value: int) -> bool:
        if self.size==self.maxSize:
            return False
        else:
            if self.size==0:
                node = Node(value)
                self.head = node
                self.tail = node
            else:
                node = Node(value)
                node.next = self.head
                self.head.prev = node
                self.head = node
            self.size+=1
            return True

    def insertLast(self, value: int) -> bool:
        if self.size==self.maxSize:
            return False
```

```
        else:
            if self.size==0:
                node = Node(value)
                self.head = node
                self.tail = node
            else:
                node = Node(value)
                self.tail.next = node
                node.prev = self.tail
                self.tail = node
            self.size+=1
            return True


    def deleteFront(self) -> bool:
        if self.size==0:
            return False
        else:
            if self.size==1:
                self.head = None
                self.tail = None
            else:
                temp = self.head
                self.head = temp.next
                self.head.prev = None
                temp.next = None
            self.size-=1
            return True



    def deleteLast(self) -> bool:
        if self.size==0:
            return False
        else:
            if self.size==1:
                self.head = None
                self.tail = None
            else:
                temp = self.tail.prev
                temp.next = None
                self.tail.prev = None
                self.tail = temp
```

```
            self.size-=1
            return True


    def getFront(self) -> int:
        if self.size==0:
            return -1
        else:
            return self.head.val


    def getRear(self) -> int:
        if self.size==0:
            return -1
        else:
            return self.tail.val


    def isEmpty(self) -> bool:
        return self.size==0


    def isFull(self) -> bool:
        return self.size==self.maxSize



# Your MyCircularDeque object will be instantiated and called as such:
# obj = MyCircularDeque(k)
# param_1 = obj.insertFront(value)
# param_2 = obj.insertLast(value)
# param_3 = obj.deleteFront()
# param_4 = obj.deleteLast()
# param_5 = obj.getFront()
# param_6 = obj.getRear()
# param_7 = obj.isEmpty()
# param_8 = obj.isFull()
```