Queue USING deque

Note Queue has a known size n

```
class QueueUsingDeque():
    def __init__(self,k):
        #k is useless for our Deque.
        # ONLY DATA STRUCTURE YOU CAN USE is Deque that you wrote
        self._maxSize = k
        self._s = Deque()
```

```
#Write all code as shown below
#Tests will fail if you don't write all routines
```

ALL operation must be CONSTANT

```
def enQueue(self, T)->'bool':
   # False means Queue is full
   # True means enQueued
def deQueue(self)->'bool':
   # does not return T
   # True: dequeued
   # False: Queue was empty
def Front(self)->'T':
   return -1 if Queue is empty
   else return T

def Rear(self)->'T':
   return -1 if Queue is empty
   else return T
def isEmpty(self)->'Bool':

def isFull(self)->'Bool':
def __len__(self)->'int':
```

```
k = 3
q = QueueUsingDeque(k)
x = q.enQueue(1);
print(q)
myassert(x == True)
x = q.Front()
myassert(x == 1)
x = q.Rear()
myassert(x == 1)
x = q.isEmpty()
myassert(x == False)
x = q.isFull()
myassert(x == False)
x = len(q)
myassert(x == 1)

x = q.enQueue(2);
print(q);
myassert(x == True)
x = q.Front()
myassert(x == 1)
x = q.Rear()
myassert(x == 2)
x = q.isEmpty()
myassert(x == False)
x = q.isFull()
myassert(x == False)
x = len(q)
myassert(x == 2)
```

Jeque([1])

Jeque([1, 2])

```
x = q.enQueue(3);
print(q);
x = q.Front()
myassert(x == 1)
x = q.Rear()
myassert(x == 3)
x = q.isEmpty()
myassert(x == False)
x = q.isFull()
myassert(x == True)
x = len(q)
myassert(x == 3)

x = q.enQueue(4);
myassert(x == False)
print(q);
x = q.deQueue();
print(q);
myassert(x == True)
x = q.Front()
myassert(x == 2)
x = q.Rear()
myassert(x == 3)
```

Jeque([1, 2, 3])

Jeque([2, 3])