Problem 641: Design Circular Deque

Problem Information

Difficulty: Medium
Acceptance Rate: 64.39%
Paid Only: No
Tags: Array, Linked List, Design, Queue

Problem Description

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`.

Code Snippets

C++:

```
class MyCircularDeque {
public:
MyCircularDeque(int k) {

}

bool insertFront(int value) {

}

bool insertLast(int value) {

}

bool deleteFront() {

}

bool deleteLast() {

}

int getFront() {

}

int getRear() {

}

bool isEmpty() {

}
bool isFull() {

}
};

/**
* Your MyCircularDeque object will be instantiated and called as such:
* MyCircularDeque* obj = new MyCircularDeque(k);
* bool param_1 = obj->insertFront(value);
* bool param_2 = obj->insertLast(value);
* bool param_3 = obj->deleteFront();
* bool param_4 = obj->deleteLast();
* int param_5 = obj->getFront();
* int param_6 = obj->getRear();
* bool param_7 = obj->isEmpty();
* bool param_8 = obj->isFull();
*/
```

Java:

```
class MyCircularDeque {

public MyCircularDeque(int k) {

}

public boolean insertFront(int value) {

}

public boolean insertLast(int value) {

}

public boolean deleteFront() {

}

public boolean deleteLast() {

}
public int getFront() {

}

public int getRear() {

}

public boolean isEmpty() {

}

public boolean isFull() {

}
}

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

Python3:

```
class MyCircularDeque:

def __init__(self, k: int):



def insertFront(self, value: int) -> bool:



def insertLast(self, value: int) -> bool:
```

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


def deleteLast(self) -> bool:


def getFront(self) -> int:


def getRear(self) -> int:


def isEmpty(self) -> bool:


def isFull(self) -> bool:



# 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()
```