Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the functions of a normal stack (`push`, `top`, `pop`, and `empty`).

Implement the `MyStack` class:

- `void push(int x)` Pushes element x to the top of the stack.
- `int pop()` Removes the element on the top of the stack and returns it.
- `int top()` Returns the element on the top of the stack.
- `boolean empty()` Returns `true` if the stack is empty, `false` otherwise.

Notes:

- You must use **only** standard operations of a queue, which means that only `push to back`, `peek/pop from front`, `size` and `is empty` operations are valid.
- Depending on your language, the queue may not be supported natively. You may simulate a queue using a list or deque (double-ended queue) as long as you use only a queue's standard operations.

Example 1:

> **Input**
> `["MyStack", "push", "push", "top", "pop", "empty"]`
> `[[], [1], [2], [], [], []]`
> **Output**
> `[null, null, null, 2, 2, false]`

</> Code

C ⌄ Auto

```c
#include <stdlib.h>
#include <stdbool.h>

#define MAX 1000

typedef struct {
    int data[MAX];
    int front;
    int rear;
} Queue;

typedef struct {
    Queue q1;
    Queue q2;
} MyStack;

void initQueue(Queue* q) {
    q->front = 0;
    q->rear = -1;
}

bool isEmptyQueue(Queue* q) {
    return q->front > q->rear;
}

void enqueue(Queue* q, int x) {
    q->data[++q->rear] = x;
}

int dequeue(Queue* q) {
    return q->data[q->front++];
}

int peek(Queue* q) {
    return q->data[q->front];
}

MyStack* myStackCreate() {
    MyStack* obj = (MyStack*)malloc(sizeof(MyStack));
    initQueue(&obj->q1);
    initQueue(&obj->q2);
    return obj;
}

void myStackPush(MyStack* obj, int x) {
    enqueue(&obj->q2, x);

    while (!isEmptyQueue(&obj->q1)) {
        enqueue(&obj->q2, dequeue(&obj->q1));
    }

    Queue temp = obj->q1;
    obj->q1 = obj->q2;
    obj->q2 = temp;
}
```

```
56
57  int myStackPop(MyStack* obj) {
58      return dequeue(&obj->q1);
59  }
60
61  int myStackTop(MyStack* obj) {
62      return peek(&obj->q1);
63  }
64
65  bool myStackEmpty(MyStack* obj) {
66      return isEmptyQueue(&obj->q1);
67  }
68
69  void myStackFree(MyStack* obj) {
70      free(obj);
71  }
72
```

Testcase | >_ Test Result

Accepted Runtime: 0 ms

Case 1

Input

```
["MyStack","push","push","top","pop","empty"]
```

```
[[],[1],[2],[],[],[]]
```

Output

```
[null,null,null,2,2,false]
```

Expected

```
[null,null,null,2,2,false]
```

Contribute a testcase