Problem 225: Implement Stack using Queues

Problem Information

Difficulty: Easy
Acceptance Rate: 68.70%
Paid Only: No
Tags: Stack, Design, Queue

Problem Description

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] **Explanation** MyStack myStack = new MyStack(); myStack.push(1); myStack.push(2); myStack.top(); // return 2 myStack.pop(); // return 2 myStack.empty(); // return False

Constraints:

* `1 <= x <= 9` * At most `100` calls will be made to `push`, `pop`, `top`, and `empty`. * All the calls to `pop` and `top` are valid.

Follow-up: Can you implement the stack using only one queue?

Code Snippets

C++:

```
class MyStack {
public:
MyStack() {

}

void push(int x) {

}

int pop() {

}

int top() {

}

bool empty() {

}
};

/**
* Your MyStack object will be instantiated and called as such:
* MyStack* obj = new MyStack();
* obj->push(x);
* int param_2 = obj->pop();
* int param_3 = obj->top();
* bool param_4 = obj->empty();
*/
```

Java:

```
class MyStack {

public MyStack() {

}

public void push(int x) {

}

public int pop() {

}

public int top() {

}

public boolean empty() {

}
}

/**
* Your MyStack object will be instantiated and called as such:
* MyStack obj = new MyStack();
* obj.push(x);
* int param_2 = obj.pop();
* int param_3 = obj.top();
* boolean param_4 = obj.empty();
*/
```

Python3:

```
class MyStack:

def __init__(self):


def push(self, x: int) -> None:
```

```
def pop(self) -> int:


def top(self) -> int:


def empty(self) -> bool:



# Your MyStack object will be instantiated and called as such:
# obj = MyStack()
# obj.push(x)
# param_2 = obj.pop()
# param_3 = obj.top()
# param_4 = obj.empty()
```