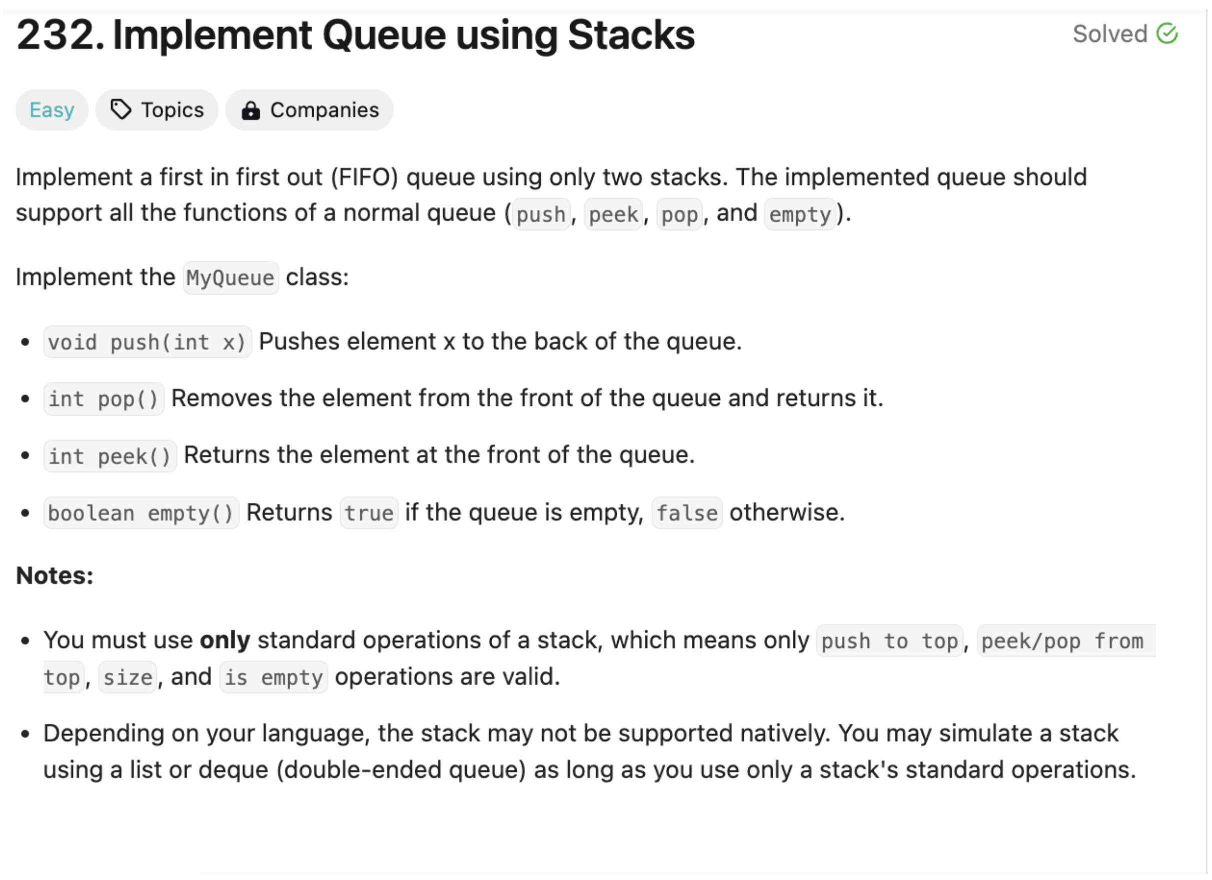

232. Implement Queue using Stacks

Solved ✓

Easy · Topics · Companies

Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the functions of a normal queue (`push`, `peek`, `pop`, and `empty`).

Implement the `MyQueue` class:

- `void push(int x)` Pushes element x to the back of the queue.
- `int pop()` Removes the element from the front of the queue and returns it.
- `int peek()` Returns the element at the front of the queue.
- `boolean empty()` Returns `true` if the queue is empty, `false` otherwise.

Notes:

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

```
class MyQueue {
   var stack1:[Int] = []
   var stack2:[Int] = []

   init() {
       stack1 = []
       stack2 = []
   }

   func push(_ x: Int) {
       stack1.append(x)
   }

   func pop() -> Int {
       stack2 = stack1
       let s = stack2.removeFirst()
       stack1 = stack2
```

```swift
        return s
    }

    func peek() -> Int {
        return stack1.first!
    }

    func empty() -> Bool {
        if(stack1.isEmpty) {
            return true
        }

        return false
    }
}

/**
* Your MyQueue object will be instantiated and called as such:
* let obj = MyQueue()
* obj.push(x)
* let ret_2: Int = obj.pop()
* let ret_3: Int = obj.peek()
* let ret_4: Bool = obj.empty()
*/
```