뉴·진·스

스택, 큐

김준성 2023-11-01(화)

목차

1.스택

2.큐

3.Live Coding

스태

스택이란?

- 선입후출(or 후입선출)의 특성을 가지는 자료구조
- 스택은 추상(개념으로만 존재하는) 자료구조이기 때문에 다양한 방법으로 구현할 수 있음.(ex. 배열을 활용하는 방법, 연결리스트를 활용하는 방법)
- 스택이기 위한 최소 구현 조건
  push: 배열에 값을 넣는 행위
  pop: 배열에서 가장 최근에 삽입한 값을 뽑아내는 행위
  top: 배열에서 가장 최근에 삽입한 값을 그저 보여주기만 하는 행위
  empty: 스택이 비어있는지 확인하는 행위

7

5

6

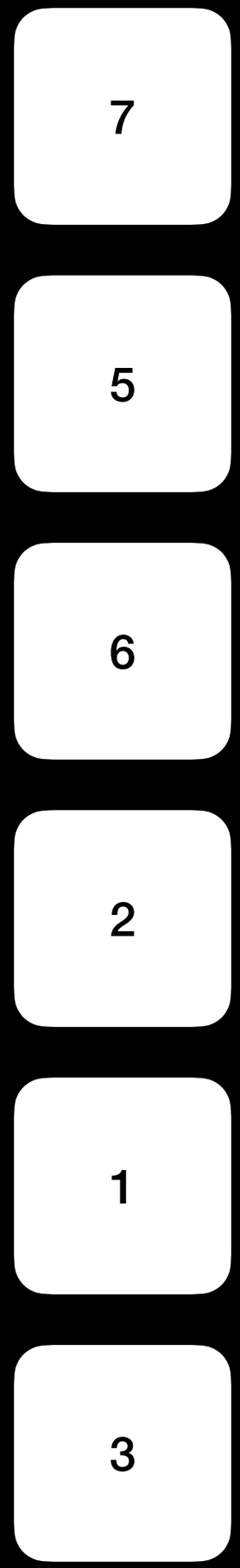
2

1

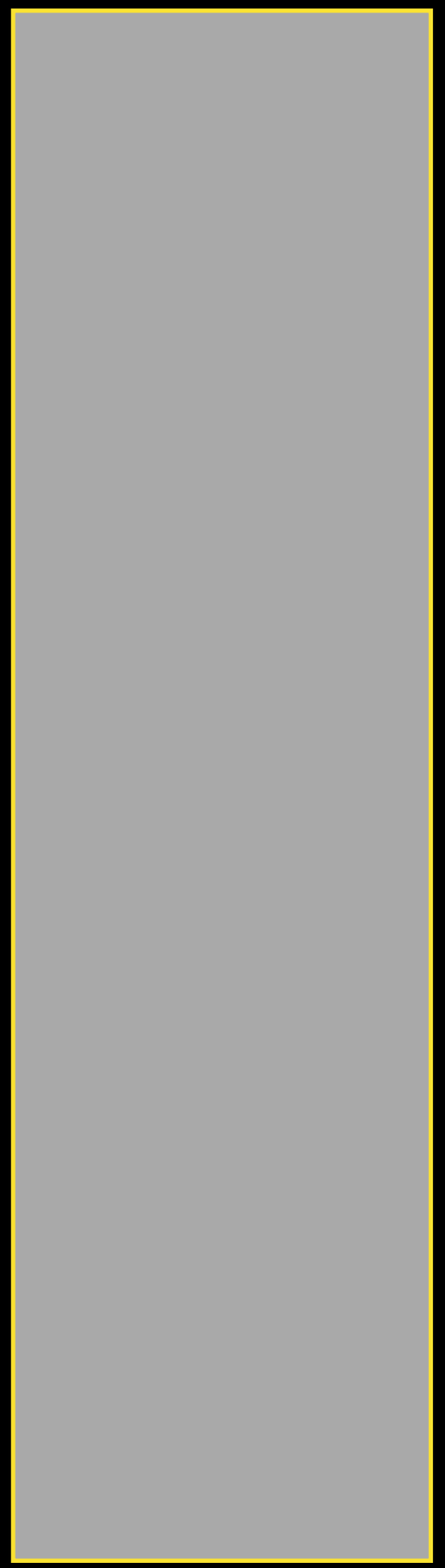
3

Stack ->

1. Push
2. Push
3. Pop
4. Push
5. Pop
6. Push
7. Push
8. Push
9. Pop
10. Pop



Stack ->



스택과 재귀함수

Stack ->

<- Array

index(0)	index(1)	index(2)	index(3)
0	0	1	2

```swift
// recursive.swift

func find(parentOf child: Int) -> Int {
    if child == array[child] {
        return child
    }

    return find(array[child])
}

print(find(parentOf: 3))
```

Swift는 Stack이
구현돼있을까?

Array의 내장함수에 push, pop, top, empty가 구현되어 있음!

Array<Element>의 내장함수

- append(_ newElement: Element)
- popLast() & removeLast() 둘의 차이점은 무엇일까?
  반환 타입에 차이가 있음! pop은 옵셔널 타입을 리턴하고, remove는 그냥 타입을 리턴함!
- isEmpty: Bool
- last: Element?

Array<Element>의 내장함수

- insert(_ newElement: Element, at: Int)
- removeFirst() -> T
- removeAll()
- remove(at: Int) -> T
- reverse()

큐

큐

- 선입선출(or 후입후출)
- 큐 또한 추상 자료형이기 때문에 다양한 방식으로 구현할 수 있다고 함!
- 큐이기 위한 최소 구현 조건
  enqueue: 배열에 값을 넣는 행위
  dequeue: 배열에서 가장 먼저 삽입한 값을 뽑아내는 행위
  front: 배열에서 가장 먼저 나와야 할 값을 그저 보여주기만 하는 행위
  back: 배열에서 가장 최근에 삽입한 값을 그저 보여주기만 하는 행위
  empty: 스택이 비어있는지 확인하는 행위

7

5

6

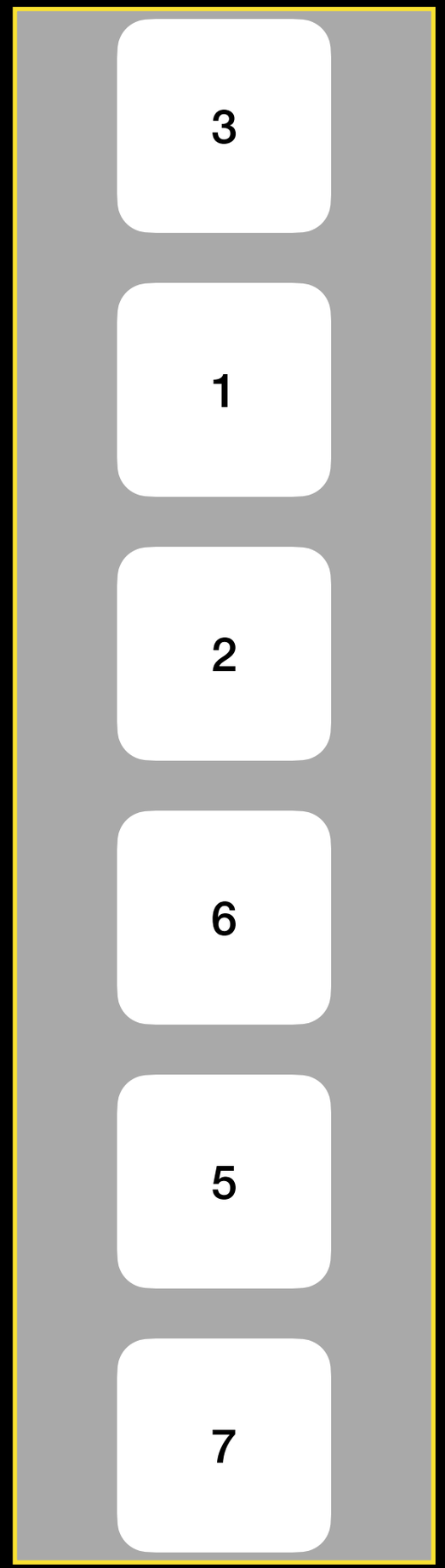
2

1

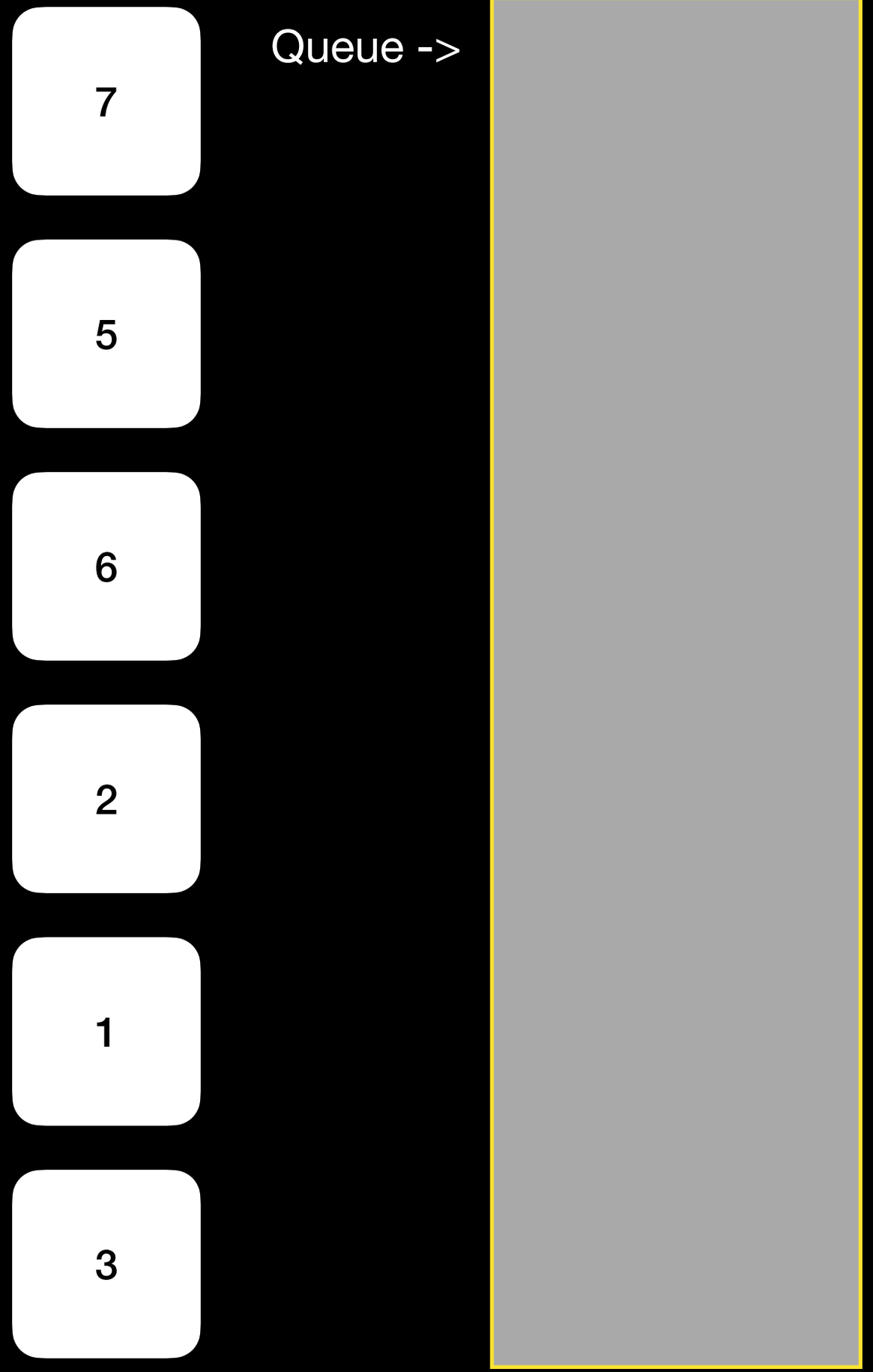
3

Queue ->

Queue ->



1. Enqueue
2. Dequeue
3. Enqueue
4. Enqueue
5. Enqueue
6. Dequeue
7. Enqueue
8. Dequeue
9. Deuqueue
10. Enqueue



Queue ->

큐의 구현방법

```
struct Queue {
    private var input = [String]()
    private var output = [String]()

    var size: Int { input.count + output.count }
    var isEmpty: Bool { input.isEmpty && output.isEmpty }
    var front: String? {
        if output.isEmpty {
            return input.first
        }
        return output.last
    }
    var back: String? {
        if input.isEmpty {
            return output.first
        }
        return input.last
    }

    mutating func enqueue(_ newElement: String) {
        input.append(newElement)
    }

    mutating func dequeue() -> String? {
        if output.isEmpty {
            output = input.reversed()
            input = [String]()
        }
        return output.popLast()
    }
}
```

큐의 또 다른 구현방법

```
var array = [String]()
var index = 0
for _ in 0..<Int(readLine()!)! {
    let order = readLine()!.split(separator: " ").map { String($0) }
    switch order[0] {
    case "push":
        array.append(order[1])
    case "pop":
        if array.count <= index {
            print(-1)
        } else {
            print(array[index])
            index += 1
        }
    case "size":
        let count = array.count - index
        print(count)
    case "empty":
        print(array.count <= index ? 1 : 0)
    case "front":
        print(array.count <= index ? -1 : array[index])
    case "back":
        print(array.count <= index ? -1 : array.last!)
    default:
        continue
    }
}
```

더 알아보기

실눈 상태

1874 스택 수열

2164 카드2

끝