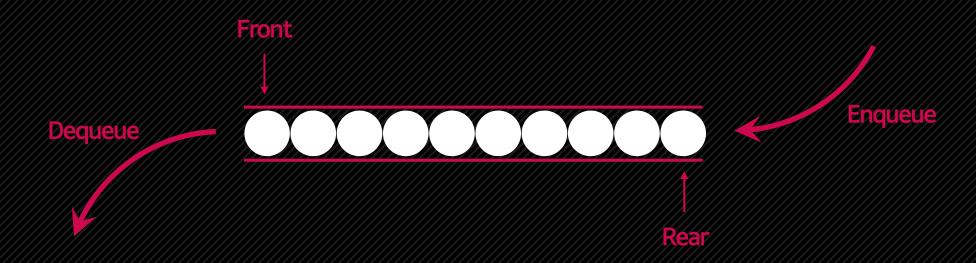


# 알고리즘 특강 큐/스택

기본적인 자료구조인 큐와 스택에 대해 알아봅시다. 이후 다양한 알고리즘을 공부할 때 기본이 되니 잘 숙지하셔야 합니다.







## Queue

- FIFO (First In, First Out) 구조를 띄고 있는 자료구조로, 삽입과 삭제 연산이 서로 다른 한군데에서 발생함.
- 삽입/삭제 연산에 있어 시간복잡도가 ○(1)임.
- 맨 앞을 front, 맨 뒤를 rear 라고 하고, 삽입 연산을 enqueue, 삭제 연산을 dequeue라고 함.
- 주로 순차적으로 진행되어야 하는 일을 스케줄링 할 때 사용 됨.



## 큐의 구현체

```
큐는 <queue> 헤더에 존재함.
#include <queue>
#include <iostream>
using namespace std;
int main() {
                                                                                         큐 선언 시 자료형 명시는 필수.
    queue<int> q; // 큐 생성 (템플릿 기반이므로 자료형 명시)
    for(int i = 0; i < 5; i++) {
                                                                                         void push(<T> element)
       q.push(i); -
                                                                                         해당하는 값을 큐의 맨 뒷부분에 삽입.
    cout << "front element : " << q.front() << endl;</pre>
                                                                                         <T> front/back()
    cout << "back element : " << q.back() << endl; -</pre>
                                                                                         큐의 맨 앞/맨뒤에 위치한 값을 리턴 함.
    cout << "queue size : " << q.size() << endl;</pre>
    cout << "Is it empty? : " << (q.empty() ? "Yes" : "No") << endl;</pre>
    for(int i = 0; i < 2; i++) {
       q.pop(); -
                                                                                         큐의 맨 앞에 위치하고 있는 값을 삭제함.
    cout << "front element : " << q.front() << endl;</pre>
    cout << "back element : " << q.back() << endl;</pre>
    cout << "queue size : " << q.size() << endl;_</pre>
    cout << "Is it empty? : " << (q.empty() ? "Yes" : "No") << endl;</pre>
                                                                                         큐에 담겨진 원소의 수를 리턴 함.
                                                                                         큐가비어 있는지 여부를 리턴 함.
```





### /◇ Silver 4 - 카드 2

## 요약

- N장의 카드가 있다. 각각의 카드는 차례로 1부터 N까지의 번호가 붙어 있으며, 1번 카드가 제일 위에, N번 카드가 제일 아래인 상태로 순서대로 카드가 놓여 있다.
- 카드가 한 장 남을 때 까지, 맨 위에 있는 카드를 바닥에 버린다. 그 다음, 제일 위에 있는 카드를 제일 아래에 있는 카드 밑으로 옮기는 행위를 반복합니다.
- N이 주어졌을 때, 제일 마지막에 남는 카드를 구하는 프로그램을 작성하시오.

# 제약조건

• N의 범위는 1 <= N <= 500,000 이다.





## Silver 4 - 카드 2



- N장의 카드가 있다. 각각의 카드는 차례로 1부터 N까지의 번호가 붙어 있으며, 1번 카드가 제일 위에, N번 카드가 제일 아래인 상태로 <del>순세대로 카드가 놓여 있다</del>.
- 카드가 한 장 남을 때 까지, <mark>맨 위에 있는 카드</mark>를 바닥에 버린다. 그 다음, <mark>제일 위에 있는 카드를 제일 아래에 있는 카드</mark> 밑으로 옮기는 행위를 반복합니다.
- N이 주어졌을 때, 제일 마지막에 남는 카드를 구하는 프로그램을 작성하시오.

## Queue

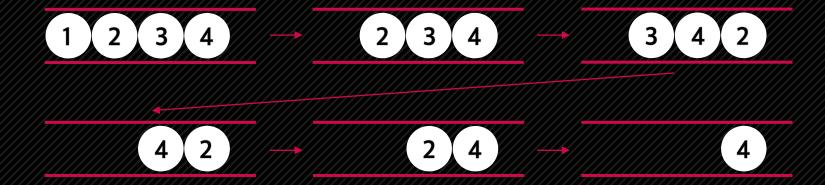
- FIFO (First In, First Out) 구조를 띄고 있는 자료구조로, 삽입과 삭제 연산이 서로 다른 한군데에서 발생함.
- 주로 순차적으로 진행되어야 하는 일을 스케줄링 할 때 사용 됨.





Silver 4 - 카드 2

N = 4







## Silver 4 - 카드 2

```
function solution(number) do
    set queue = []

foreach(i <- 1 ... number) do
    enqueue i to queue
  end

while loop (size of queue is 1) {
    dequeue of queue
    set anotherNumber = front of queue
    dequeue of queue
    enqueue anotherNumber to queue
}

return front of queue
end</pre>
```

Dequeue를 두 번 할 때 마다 큐의 길이가 1씩 감소 → O(2\*N) = O(N)





## ✓ Silver 4 - 요세푸스 문제 0

# 요약

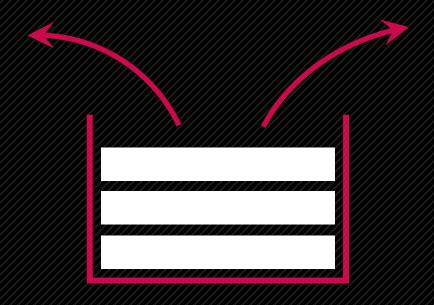
- 1번부터 N번까지 N명의 사람이 원을 이루면서 앉아있고, 양의 정수 K(≤ N)가 주어진다.
- 순서대로 K번째 사람을 제거한다. 한 사람이 제거되면 남은 사람들로 이루어진 원을 따라 이 과정을 계속해 나간다. 이 과정은 N명의 사람이 모두 제거될 때까지 계속된다.
- 사람들이 제거되는 순서를 (N, K)-요세푸스 순열이라고 한다. 예로 (7, 3)-요세푸스 순열은 <3, 6, 2, 7, 5, 1, 4>이다.
- 이 때, N, K가 주어지면 (N, K)-요세푸스 순열을 구해라.



• N과 K의 범위는 1 <= K <= N <= 1,000 이다.







### Stack

- FILO (First In, Last Out) 구조를 띄고 있는 자료구조로, 삽입과 삭제 연산이 동일한 한군데에서 발생함.
- 삽입/삭제 연산에 있어 시간복잡도가 O(1)임.
- 이전에 활용한 데이터를 역으로 추적하거나, 처음 들어온 데이터보다 나중에 들어온 데이터가 빨리 나가야 할 때 사용.



### 스택의 구현체

```
스택은 <stack> 헤더에 존재함.
#include <stack>
#include <iostream>
using namespace std;
int main() {
                                                                                        큐와 동일하게 자료형 명시 필요함.
    stack<int> s;
    for(int i = 0; i < 5; i++) {
                                                                                         void push(<T> element)
       s.push(i);
                                                                                        해당하는 값을 스택의 위쪽에 삽입.
    cout << "top element : " << s.top() << endl;</pre>
    cout << "stack size : " << s.size() << endl;</pre>
                                                                                        스택의 첫번째 값을 리턴. (큐와 함수가 다름!!)
    cout << "Is it empty? : " << (s.empty() ? "Yes" : "No") << endl;</pre>
    for(int i = 0; i < 2; i++) {
       s.pop();
                                                                                        스택의 첫번째 값을 삭제함.
   cout << "top element : " << s.top() << endl;</pre>
    cout << "stack size : " << s.size() << endl;</pre>
    cout << "Is it empty? : " << (s.empty() ? "Yes" : "No") << endl;</pre>
                                                                                        스택에 담겨진 원소의 수를 리턴 함.
                                                                                        스택이 비어 있는지 여부를 리턴 함.
```





## Silver 4 - 제로

## 요약

- 데이터를 입력 받다가, 중간에 잘못된 데이터가 들어오면 0이 들어온다.
- 0이 들어오면 잘못된 데이터가 들어온 것 이므로, 값을 삭제한다.
- 이후 남아있는 값들의 합을 구하라.

# 제약조건

- 전체 수의 수는 1 <= K <= 100,000 이다.
- 최종적으로 적어낸 수의 합은 231-1 보다 같거나 작다.





## ✓ Silver 4 - 제로

## 요약

- 데이터를 입력 받다가, 중간에 잘못된 데이터가 들어오면 0이 들어온다.
- 0이 들어오면 잘못된 데이터가 들어온 것 이므로, 값을 삭제한다.
- 이후 남아있는 값들의 합을 구하라.

### Stack

- FILO (First In, Last Out) 구조를 띄고 있는 자료구조로, 삽입과 삭제 연산이 동일한 한군데에서 발생함.
- 삽입/삭제 연산에 있어 시간복잡도가 O(1)임.
- 이전에 활용한 데이터를 역으로 추적하거나, 처음 들어온 데이터보다 나중에 들어온 데이터가 빨리 나가야 할 때 사용.





## Silver 4 - 제로

```
function solution(number) do
    set stack = []
    foreach (numberElement in number) do
        if numberElement != 0 do
           push numberElement to stack
       end else do
            pop numberElement to stack
        end
    end
    set answer = 0
    while loop(stack is not empty) {
        answer += top of stack
        pop element of stack
    return answer
end
```

숫자를 순차적으로 push하고, 순차적으로 pop함. → O(2 \* N) = O(N)





## ✓ Silver 4 - 괄호

## 요약

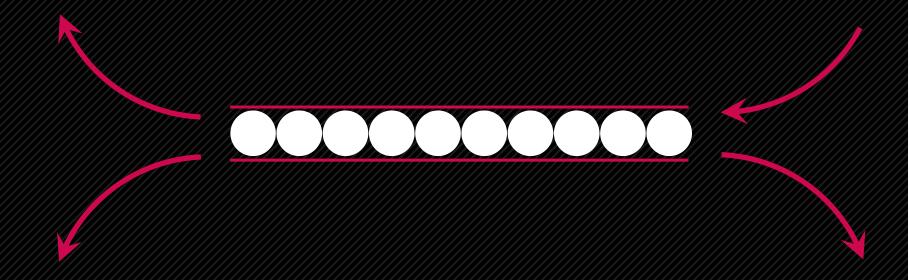
- 괄호 문자열(Parenthesis String, PS)은 두 개의 괄호 기호인 '(' 와 ')' 만으로 구성되어 있는 문자열이다. 그 중에서 괄호의 모양이 바르게 구성된 문자열을 <mark>올바른 괄호 문자열(Valid PS, VPS)</mark>이라고 부른다.
- 예를 들어 "(())()"와 "((()))" 는 VPS 이지만 "(()(", "(())()))" , 그리고 "(()" 는 모두 VPS 가 아닌 문자열이다.
- 주어진 괄호 문자열이 VPS인지 아닌지 판단하는 프로그램을 작성하시오.

# 제약조건

•하나의 괄호 문자열의 길이는 2이상 50 이하이다.







## Deque

- Queue/Stack의 구조를 합친 구조로, 양쪽에서 삽입/삭제를 모두 할 수 있음.
- 삽입/삭제 연산에 있어 시간복잡도가 O(1)임.
- C++의 Queue/Stack의 구현체는 Deque 자료구조에서 인터페이스를 제한 한 것.

## 추가 추천 문제



- ✓ Silver 3 과제는 끝나지 않아!
  - 문제를 보자 마자 어떤 자료구조를 사용해야 하는지 감이 와야 합니다.
- ⋉ Silver 1 세훈이의 선물가게
  - 구현에 집중된 문제지만, 결국 구현 과정에서 배운 내용을 사용해야 합니다.
- ▶ Level 2 기능개발
  - 굳이 자료구조를 사용하지 않아도 풀 수 있지만, 우선은 배운 내용을 연습해봅시다.
- Level 2 주식가격
  - 배운 자료구조들의 크기는 어떤 식으로 확인할까요?

</s>

"Any question?"