1. 주식가격

문제 설명

초 단위로 기록된 주식가격이 담긴 배열 prices 가 매개변수로 주어질 때, 가격이 떨어지지 않은 기간은 몇 초인지를 return 하도록 solution 함수를 완성하세요.

제한사항

- prices 의 각 가격은 1 이상 10,000 이하인 자연수입니다.
- prices 의 길이는 2 이상 100,000 이하입니다.

입출력 예

prices	return
[1, 2, 3, 2, 3]	[4, 3, 1, 1, 0]

입출력 예 설명

- 1 초 시점의 ₩1 은 끝까지 가격이 떨어지지 않았습니다.
- 2 초 시점의 ₩2 은 끝까지 가격이 떨어지지 않았습니다.
- 3 초 시점의 ₩3 은 1 초뒤에 가격이 떨어집니다. 따라서 1 초간 가격이 떨어지지 않은 것으로 봅니다.
- 4 초 시점의 ₩2 은 1 초간 가격이 떨어지지 않았습니다.
- 5 초 시점의 ₩3 은 0 초간 가격이 떨어지지 않았습니다.

※ 공지 - 2019 년 2 월 28 일 지문이 리뉴얼되었습니다.

```
class Solution {
    public int[] solution(int[] prices) {
        int[] answer = new int[prices.length];
        int count = 0;

        for(int i=0; i<prices.length-1; i++) {
            count = 0;
            for(int j=i+1; j<prices.length; j++) {
                if(prices[i]<=prices[j]) {
                    count++;
                } else {
                    count++;
                    break;
                }
            }
            answer[i] = count;
        }
```

```
        answer[prices.length-1] = 0;


        return answer;
    }
}
```

https://ju-nam2.tistory.com/35

2. 기능개발

문제 설명

프로그래머스 팀에서는 기능 개선 작업을 수행 중입니다. 각 기능은 진도가 100%일 때 서비스에 반영할 수 있습니다.

또, 각 기능의 개발속도는 모두 다르기 때문에 뒤에 있는 기능이 앞에 있는 기능보다 먼저 개발될 수 있고, 이때 뒤에 있는 기능은 앞에 있는 기능이 배포될 때 함께 배포됩니다.

먼저 배포되어야 하는 순서대로 작업의 진도가 적힌 정수 배열 progresses 와 각 작업의 개발 속도가 적힌 정수 배열 speeds 가 주어질 때 각 배포마다 몇 개의 기능이 배포되는지를 return 하도록 solution 함수를 완성하세요.

제한 사항

- 작업의 개수(progresses, speeds 배열의 길이)는 100 개 이하입니다.
- 작업 진도는 100 미만의 자연수입니다.
- 작업 속도는 100 이하의 자연수입니다.
- 배포는 하루에 한 번만 할 수 있으며, 하루의 끝에 이루어진다고 가정합니다. 예를 들어 진도율이 95%인 작업의 개발 속도가 하루에 4%라면 배포는 2 일 뒤에 이루어집니다.

입출력 예

progresses	speeds	return
[93, 30, 55]	[1, 30, 5]	[2, 1]
[95, 90, 99, 99, 80, 99]	[1, 1, 1, 1, 1, 1]	[1, 3, 2]

입출력 예 설명

입출력 예 #1
첫 번째 기능은 93% 완료되어 있고 하루에 1%씩 작업이 가능하므로 7 일간 작업 후 배포가 가능합니다.
두 번째 기능은 30%가 완료되어 있고 하루에 30%씩 작업이 가능하므로 3 일간 작업 후 배포가 가능합니다. 하지만 이전 첫 번째 기능이 아직 완성된 상태가 아니기 때문에 첫 번째 기능이 배포되는 7 일째 배포됩니다.
세 번째 기능은 55%가 완료되어 있고 하루에 5%씩 작업이 가능하므로 9 일간 작업 후 배포가 가능합니다.

따라서 7 일째에 2 개의 기능, 9 일째에 1 개의 기능이 배포됩니다.

입출력 예 #2
모든 기능이 하루에 1%씩 작업이 가능하므로, 작업이 끝나기까지 남은 일수는 각각 5 일, 10 일, 1 일, 1 일, 20 일, 1 일입니다. 어떤 기능이 먼저 완성되었더라도 앞에 있는 모든 기능이 완성되지 않으면 배포가 불가능합니다.

따라서 5 일째에 1 개의 기능, 10 일째에 3 개의 기능, 20 일째에 2 개의 기능이 배포됩니다.

※ 공지 - 2020 년 7 월 14 일 테스트케이스가 추가되었습니다.

```
import java.util.LinkedList;

class Solution {

	public int[] solution(int[] progresses, int[] speeds) {

		double[] days = new double[progresses.length];

		double max = 0;
		LinkedList<int[]> result = new LinkedList<>();
		for (int i = 0; i < progresses.length; i++) {

			// 끝낼 수 있는 일자
			days[i] = Math.ceil((100 - progresses[i]) /
speeds[i]);

			// max 보다 큰 값이 나올 경우 새로운 세트로 시작
			if (days[i] > max) {
				max = days[i];
				result.add(new int[] { 1 });

			// max 보다 작은 값일 경우 max 값이 시작된 count 와
한 세트
			} else {
				result.getLast()[0]++;
			}
		}

		// 일반 배열로 복사 후 리턴
		int size = result.size();
		int[] answer = new int[size];
		for (int i = 0; i < size; i++) {
			answer[i] = result.pollFirst()[0];
		}

		return answer;
	}
}
```

https://codevang.tistory.com/309

3. 다리를 지나는 트럭

문제 설명

트럭 여러 대가 강을 가로지르는 일 차선 다리를 정해진 순으로 건너려 합니다. 모든 트럭이 다리를 건너려면 최소 몇 초가 걸리는지 알아내야 합니다. 트럭은 1 초에 1 만큼 움직이며, 다리 길이는 bridge_length 이고 다리는 무게 weight 까지 견딥니다.
※ 트럭이 다리에 완전히 오르지 않은 경우, 이 트럭의 무게는 고려하지 않습니다.

예를 들어, 길이가 2 이고 10kg 무게를 견디는 다리가 있습니다. 무게가 [7, 4, 5, 6]kg 인 트럭이 순서대로 최단 시간 안에 다리를 건너려면 다음과 같이 건너야 합니다.

경과 시간	다리를 지난 트럭	다리를 건너는 트럭	대기 트럭
0	[]	[]	[7,4,5,6]
1~2	[]	[7]	[4,5,6]
3	[7]	[4]	[5,6]
4	[7]	[4,5]	[6]
5	[7,4]	[5]	[6]
6~7	[7,4,5]	[6]	[]
8	[7,4,5,6]	[]	[]

따라서, 모든 트럭이 다리를 지나려면 최소 8 초가 걸립니다.

solution 함수의 매개변수로 다리 길이 bridge_length, 다리가 견딜 수 있는 무게 weight, 트럭별 무게 truck_weights 가 주어집니다. 이때 모든 트럭이 다리를 건너려면 최소 몇 초가 걸리는지 return 하도록 solution 함수를 완성하세요.

제한 조건

- bridge_length 는 1 이상 10,000 이하입니다.
- weight 는 1 이상 10,000 이하입니다.
- truck_weights 의 길이는 1 이상 10,000 이하입니다.
- 모든 트럭의 무게는 1 이상 weight 이하입니다.

입출력 예

bridge_length	weight	truck_weights	return
2	10	[7,4,5,6]	8
100	100	[10]	101
100	100	[10,10,10,10,10,10,10,10,10,10]	110

출처

※ 공지 - 2020 년 4 월 06 일 테스트케이스가 추가되었습니다.

```
import java.util.*;

class Solution {
    class Truck {
        int weight;
        int entry;

        Truck(int weight, int entry){
            this.weight = weight;
            this.entry = entry;
        }
    }

    public int solution(int bridge_length, int weight, int[]
truck_weights) {
        Queue<Truck> waiting = new LinkedList<>();
        Queue<Truck> bridge = new LinkedList<>();

        for(int i = 0 ; i < truck_weights.length ; ++i){
            waiting.offer(new Truck(truck_weights[i], 0));
        }

        int time = 0;
        int totalWeight = 0;
        while(!waiting.isEmpty() || !bridge.isEmpty()){
        time++;
        if(!bridge.isEmpty()) {
                Truck t = bridge.peek();
                if(time - t.entry >= bridge_length) {
                        totalWeight -= t.weight;
                        bridge.poll();
                }
        }

        if(!waiting.isEmpty()) {
                if(totalWeight + waiting.peek().weight <= weight) {
                        Truck t = waiting.poll();
                        totalWeight += t.weight;

                        bridge.offer(new Truck(t.weight, time));
                }
        }
        }
        return time;
    }
```

```
}
https://velog.io/@hyeon930/%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%A8%B8%EC%8A%A4-%EB%8B%A4%EB%A6%AC%EB%A5%BC-%EC%A7%80%EB%82%98%EB%8A%94-%ED%8A%B8%EB%9F%AD-Java
```

4. 프린터

문제 설명

일반적인 프린터는 인쇄 요청이 들어온 순서대로 인쇄합니다. 그렇기 때문에 중요한 문서가 나중에 인쇄될 수 있습니다. 이런 문제를 보완하기 위해 중요도가 높은 문서를 먼저 인쇄하는 프린터를 개발했습니다. 이 새롭게 개발한 프린터는 아래와 같은 방식으로 인쇄 작업을 수행합니다.

1. 인쇄 대기목록의 가장 앞에 있는 문서(J)를 대기목록에서 꺼냅니다.
2. 나머지 인쇄 대기목록에서 J 보다 중요도가 높은 문서가 한 개라도 존재하면 J 를 대기목록의 가장 마지막에 넣습니다.
3. 그렇지 않으면 J 를 인쇄합니다.

예를 들어, 4 개의 문서(A, B, C, D)가 순서대로 인쇄 대기목록에 있고 중요도가 2 1 3 2 라면 C D A B 순으로 인쇄하게 됩니다.

내가 인쇄를 요청한 문서가 몇 번째로 인쇄되는지 알고 싶습니다. 위의 예에서 C 는 1 번째로, A 는 3 번째로 인쇄됩니다.

현재 대기목록에 있는 문서의 중요도가 순서대로 담긴 배열 priorities 와 내가 인쇄를 요청한 문서가 현재 대기목록의 어떤 위치에 있는지를 알려주는 location 이 매개변수로 주어질 때, 내가 인쇄를 요청한 문서가 몇 번째로 인쇄되는지 return 하도록 solution 함수를 작성해주세요.

제한사항

- 현재 대기목록에는 1 개 이상 100 개 이하의 문서가 있습니다.
- 인쇄 작업의 중요도는 1~9 로 표현하며 숫자가 클수록 중요하다는 뜻입니다.
- location 은 0 이상 (현재 대기목록에 있는 작업 수 - 1) 이하의 값을 가지며 대기목록의 가장 앞에 있으면 0, 두 번째에 있으면 1 로 표현합니다.

입출력 예

priorities	location	return
[2, 1, 3, 2]	2	1
[1, 1, 9, 1, 1, 1]	0	5

입출력 예 설명

예제 #1

문제에 나온 예와 같습니다.

예제 #2

6 개의 문서(A, B, C, D, E, F)가 인쇄 대기목록에 있고 중요도가 1 1 9 1 1 1 이므로 C D E F A B 순으로 인쇄합니다.

출처

```
import java.util.*;
class Solution {
    public int solution(int[] priorities, int location) {
        int answer = 0;
        Queue<Printer> q = new LinkedList<>();

                for (int i = 0; i < priorities.length; i++) { // print 큐에 인덱스번호, 우선순위 삽입
                        q.offer(new Printer(i, priorities[i]));
                }

                while (!q.isEmpty()) {

                        boolean flag = false;
                        int com = q.peek().prior;
                        for (Printer p : q) {
                                if (com < p.prior) { // 맨앞의 수보다 큰 숫자가 존재하면

                                        flag = true;
                                }
                        }

                        if (flag) {
                                q.offer(q.poll());
                        } else {// 현재 맨앞의 숫자가 가장 클 때
                                if (q.poll().location == location) {
                                        answer = priorities.length - q.size();

                                }
                        }
                }
        return answer;
    }
    class Printer {
                int location;
                int prior;

                Printer(int location, int prior) {
                        this.location = location;
                        this.prior = prior;
```

```
                }
        }
}
```

https://velog.io/@qweadzs/%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%A8%B8%EC%8A%A4-%ED%94%84%EB%A6%B0%ED%84%B0-java