

## 1. String 개념 (사전 지식)

고유한 이메일 (Unique Email Addresses)

보석과 돌 (JewelsAndStones)

포맷팅 (licenseKeyFormatting)

플러스원 (PlusOne)

1) `charAt(i)` => 문자열의 위치

2) `toCharArray()` => `char[]`

3) `IndexOf(), substring(begin, end)` `String str = "abcd"`

ex) `str.indexOf ( "a" )// 0`

`indexOf( "찾을 특정 문자" , "시작할 위치" )`

ex) `str.substring(0,3)= abc`

4) `startsWith(), endsWith(), split()`

ex) `String str = "abc" str.startsWith( "a" ); => true`

`str.endsWith( "c" );=true`

ex) `String str = "010 888 7777" String[] spStr = str.split( " " )`

`String s1 = spStr[0]; String s2 = spStr[1]; String s3 = spStr[0];`

5) `toLowerCase()`

6) `replace()`

7) `Character.isDigit(c), Character.isLetter(c)`

8) `StringBuilder sb = new StringBuilder(); sb.append( "aa" )`

# 고유한 이메일 (Unique Email Addresses)

## 설명

모든 유효한 이메일은 @을 기준으로 로컬이름과 도메인이름으로 구성됩니다.  
또한 소문자 외에 하나 이상의 '.' 또는 '+'을 포함합니다.

예) test@coding.com  
test는 로컬이름이고 codingtest.com은 도메인 이름입니다.  
이메일에는 소문자 외에 '.' 또는 '+'가 포함될 수 있습니다.

로컬이름에 일부 문자 사이에 마침표 ('.')를 추가하여  
전송된 메일은 로컬 이름에 점이없는 동일한 주소로 전달됩니다.  
예)

" test.email@codingtest.com " 및 " testemail@codingtest.com "은 동일한 이메일 주소로 전달됩니다.  
(이 규칙은 도메인 이름에는 적용되지 않습니다.)

로컬 이름에 더하기 ('+')를 추가하면 첫 번째 더하기 기호 뒤의 모든 항목이 무시됩니다.  
이를 통해 특정 이메일을 필터링 할 수 있습니다.

예) test.email+james@codingtest.com은  
test.email@codingtest.com으로 전달됩니다.  
(이 규칙은 도메인 이름에는 적용되지 않습니다.)  
이 두 규칙을 동시에 사용할 수 있습니다.

이메일 목록이 주어지면 목록의 각 주소로 하나의 이메일을 보냅니다.  
실제로 메일을받는 주소는 몇 개입니까?

# 고유한 이메일 (Unique Email Addresses)

## 입출력

**Input:** emails = [  
"test.email+james@coding.com",  
"test.e.mail+toto.jane@coding.com",  
"testemail+tom@cod.ing.com"]  
]

**Output:** 2

**Explanation:**

"testemail@coding.com"  
"testemail@cod.ing.com"

**Input:** emails = [  
"a@coding.com",  
"b@coding.com",  
"c@coding.com" ]

**Output:** 3

로컬네임이 다름

## 문제 Format

```
class Solution {  
    public int solve(String[] emails) {  
    }  
}
```

## 제한사항

1 ≤ emails.length ≤ 100  
1 ≤ emails[i].length ≤ 100  
email[i] consist of lowercase English letters, '+', '.' and '@'.  
Each emails[i] contains exactly one '@' character.  
All local and domain names are non-empty.  
Local names do not start with a '+' character.

# 보석과 돌 (Jewels And Stones)

## 설명

보석(jewels)과 돌(stones) 이 주어집니다.

보석을 이용해서 돌에 얼마나 많은 보석이 포함되어 있는지 알고 싶습니다.

문자는 대소 문자를 구분하므로 "a"와 "A " 은 다른 유형의 스톤으로 간주됩니다 .

## 입출력

**Input:** jewels = "aA",  
stones = "aAAbbbb"

**Output:** 3

**Input:** jewels = "z", stones = "ZZ"

**Output:** 0

## 문제 Format

```
class Solution {  
    public int solve(String jew, String stones){  
    }  
}
```

## 제한사항

1 <= jewels.length, stones.length <= 50  
jewels and stones consist of only English letters.  
All the characters of jewels are **unique**.

# 라이선스 키 포맷(License Key Formatting)

## 설명

숫자, 문자와 대시로만 구성된 문자열로 표시된 라이선스 키가 제공 됩니다.  
문자열은 대시(-)로  $n$  과  $n + 1$  그룹으로 구분됩니다 . 정수값  $k$ 도 주어집니다.  
첫 번째 그룹을 제외하고 각 그룹이 정확히  $k$ 로 문자열을 형식화(Formatting)해야 합니다.  
첫 번째 그룹은 더 짧을 수 있지만 여전히 적어도 하나의 문자를 포함해야 합니다.  
또한 두 그룹 사이에 대시가 삽입되어야 하며 모든 소문자를 대문자로 변환해야 합니다.  
재 포맷 된 라이선스 키를 리턴하세요

## 입출력

**Input:**  $s = "8F3Z-2e-9-w"$ ,  $k = 4$   
**Output:**  $"8F3Z-2E9W"$

**Input:**  $s = " 8-5g-3-J "$ ,  $k = 2$   
**Output:**  $"8-5G-3J"$

## 문제 Format

```
class Solution {  
    public int solve(String s, int k) {  
    }  
}
```

## 제한사항

$1 \leq s.length \leq 10^5$   
 $s$  는 영어문자, 숫자, 대쉬(-)로 이루어져 있습니다.  
 $1 \leq k \leq 10^4$

# 플러스 원 (Plus One)

## 설명

음이 아닌 정수를 나타내는 **비어 있지 않은** 십진수 배열이 주어지면 정수 1을 증가시킵니다.  
배열의 마지막에 숫자에 +1을 합니다.  
숫자 0 자체를 제외하고 정수에 앞에 0이 포함되지 않습니다.

## 입출력

**Input:** digits = [1,2,3]

**Output:** [1,2,4]

**Input:** [9,9,9]

**Output:** [1,0,0,0]

## 문제 Format

```
class Solution {  
    public int[] solve( int[] k) {  
    }  
}
```

## 제한사항

1 <= digits.length <= 100  
0 <= digits[i] <= 9