# 23일 (수)

## 1. 소수 찾기

### 에리토스테네스의 채  코드 구현

#### 에리토스테네스의 체로 구현한 코드

```
n=1000
a = [False,False] + [True]*(n-1)
primes=[]

for i in range(2,n+1):
  if a[i]:
    primes.append(i)
    for j in range(2*i, n+1, i):
        a[j] = False
print(primes)
```

#### 코드 결과

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89,
0.000580초 걸렸습니다.
```

위의 알고리즘의 표본 개수가 1000개 밖에 되지 않는데도 불구하고 속도의 차이가 10배 이상이 나는걸로 보인다.

표본 개수를 10000으로 늘려본다면 얼마나 차이가 나는걸까?

6.021144초 걸렸습니다.

0.004945초 걸렸습니다.

둘의 시간차이가 표본의 수가 늘어날수록 많은 차이가 벌어집니다.

둘의 시간복잡도는

```
O(√n)
O(n (log n) (log log n))
```

의 차이가 납니다.

이거 외에도 소스 코드에서 함수처리 호출 처리 등에 따라서 1000개의 표본에서 위의 알고리즘이 0.06초가 걸리는 것을 0.6초가 되게 구현할수 도 있습니다.

서버를 구축 하실 때에는 사소한 알고리즘의 차이가 큰 수준을 바꾸게 되니 항상 코드를 짜실때 시간 복잡도를 염두를 하시고 개발하시면 좋을것 같습니다.

자료가 유용하셨다면 구독과 아래의 하트 눌러주세요.

## Other Sol.

```
def solution(n):
        num= set(range(2, n+1))

    for i in range(2, n+1):
        if i in num:
                num -= set(range(2*i, n+1, i)

    return len(num)
```

에라토스테네스의 체 간단히 구현.

- 2~n까지의 수를 함수 num에 담기
- 2~n까지의 수 i 에 대해, i 가 num 안에 있으면, i의 배수set(i의 2배수 부터 n까지수 중 i 만큼 간격있는 수들)를 num에서 빼주기
- 나머지 num의 길이 리턴

### 2. 인덱스 슬라이싱 - 심화

### 11.4.2 인덱스 증가폭 사용하기

지금까지 지정된 범위의 요소를 모두 가져왔죠? 슬라이스는 인덱스의 증가폭을 지정하여 범위 내에서 인덱스를 건너뛰며 요소를 가져올 수 있습니다.

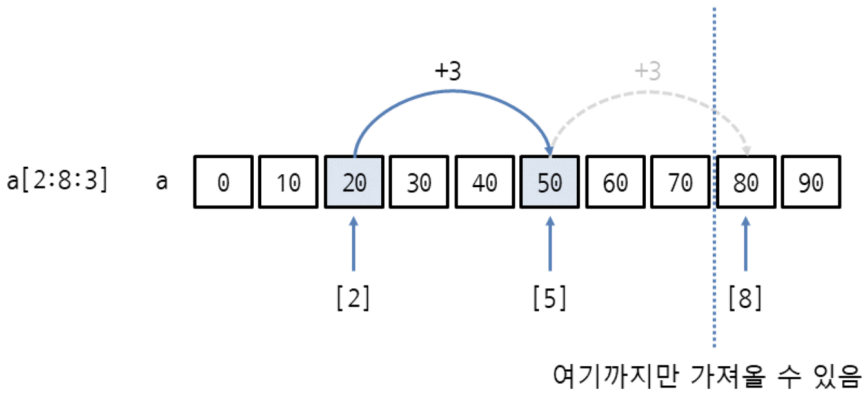
다음은 인덱스를 3씩 증가시키면서 요소를 가져옵니다. 여기서 주의할 점은 인덱스의 증가폭이지 요소의 값 증가폭이 아니라는 점입니다.

- 시퀀스객체[시작인덱스:끝인덱스:인덱스증가폭]

```
>>> a = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> a[2:8:3]    # 인덱스 2부터 3씩 증가시키면서 인덱스 7까지 가져옴
[20, 50]
```

a[2:8:3]을 실행하니 [20, 50]이 나왔죠? 왜 이런 결과가 나왔을까요? 먼저 시작 인덱스가 2이므로 20부터 가져옵니다. 그리고 인덱스 증가폭을 3으로 지정했으므로 인덱스 5의 50, 인덱스 8의 80을 가져올 수 있습니다. 하지만, 끝 인덱스를 8로 지정했으므로 인덱스 7까지만 가져옵니다. 따라서 20과 50만 가져와서 [20, 50]이 나옵니다.

▼ **그림 11-23** 인덱스 증가폭 사용하기




공부는 링크 참조

https://dojang.io/mod/page/view.php?id=2208

## 3. 프로그래머스 문제 풀이

### 1) 완주하지 못한 선수 (못 푼 문제) - 해시 대표 문제

## 완주하지 못한 선수

### 문제 설명

수많은 마라톤 선수들이 마라톤에 참여하였습니다. 단 한 명의 선수를 제외하고는 모든 선수가 마라톤을 완주하였습니다.

마라톤에 참여한 선수들의 이름이 담긴 배열 participant와 완주한 선수들의 이름이 담긴 배열 completion이 주어질 때, 완주하지 못한 선수의 이름을 return 하도록 solution 함수를 작성해주세요.

### 제한사항

- 마라톤 경기에 참여한 선수의 수는 1명 이상 100,000명 이하입니다.
- completion의 길이는 participant의 길이보다 1 작습니다.
- 참가자의 이름은 1개 이상 20개 이하의 알파벳 소문자로 이루어져 있습니다.
- 참가자 중에는 동명이인이 있을 수 있습니다.

### 입출력 예

| participant | completion | return |
|---|---|---|
| ["leo", "kiki", "eden"] | ["eden", "kiki"] | "leo" |
| ["marina", "josipa", "nikola", "vinko", "filipa"] | ["josipa", "filipa", "marina", "nikola"] | "vinko" |
| ["mislav", "stanko", "mislav", "ana"] | ["stanko", "ana", "mislav"] | "mislav" |

### 입출력 예 설명

예제 #1
"leo"는 참여자 명단에는 있지만, 완주자 명단에는 없기 때문에 완주하지 못했습니다.

예제 #2
"vinko"는 참여자 명단에는 있지만, 완주자 명단에는 없기 때문에 완주하지 못했습니다.

예제 #3
"mislav"는 참여자 명단에는 두 명이 있지만, 완주자 명단에는 한 명밖에 없기 때문에 한명은 완주하지

```
#구글 검색  #zip 이용

def solution(participant, completion):
    participant.sort()
    completion.sort()

    for p,c in zip(participant, completion):
        if p != c:
            return p
    return participant.pop()
```

```
#해쉬 함수 이용

def solution(participant, completion):
    answer = ''
    temp = 0
    dic = {}
    for part in participant:
        dic[hash(part)] = part
        temp += int(hash(part))
    for com in completion:
        temp -= hash(com)
    answer = dic[temp]
```

```
    return answer
```

```
#collections 라이브러리 Counter 메소드 사용

import collections


def solution(participant, completion):
    answer = collections.Counter(participant) - collections.Counter(completion)
    return list(answer.keys())[0]
```

### 2) 로또 최고 순위와 최저 순위

**로또의 최고 순위와 최저 순위**

**문제 설명**

`로또 6/45` (이하 '로또'로 표기)는 1부터 45까지의 숫자 중 6개를 찍어서 맞히는 대표적인 복권입니다. 아래는 로또의 순위를 정하는 방식입니다. [1]

| 순위 | 당첨 내용 |
|---|---|
| 1 | 6개 번호가 모두 일치 |
| 2 | 5개 번호가 일치 |
| 3 | 4개 번호가 일치 |
| 4 | 3개 번호가 일치 |
| 5 | 2개 번호가 일치 |
| 6(낙첨) | 그 외 |

로또를 구매한 민우는 당첨 번호 발표일을 학수고대하고 있었습니다. 하지만, 민우의 동생이 로또에 낙서를 하여, 일부 번호를 알아볼 수 없게 되었습니다. 당첨 번호 발표 후, 민우는 자신이 구매했던 로또로 당첨이 가능했던 최고 순위와 최저 순위를 알아보고 싶어 졌습니다.
알아볼 수 없는 번호를 `0` 으로 표기하기로 하고, 민우가 구매한 로또 번호 6개가 `44, 1, 0, 0, 31 25` 라고 가정해보겠습니다. 당첨 번호 6개가 `31, 10, 45, 1, 6, 19` 라면, 당첨 가능한 최고 순위와 최저 순위의 한 예는 아래와 같습니다.

| 당첨 번호 | 31 | 10 | 45 | 1 | 6 | 19 | 결과 |
|---|---|---|---|---|---|---|---|
| 최고 순위 번호 | **31** | 0→**10** | 44 | **1** | 0→**6** | 25 | 4개 번호 일치, 3등 |
| 최저 순위 번호 | **31** | 0→11 | 44 | **1** | 0→7 | 25 | 2개 번호 일치, 5등 |

- 순서와 상관없이, 구매한 로또에 당첨 번호와 일치하는 번호가 있으면 맞힌 걸로 인정됩니다.
- 알아볼 수 없는 두 개의 번호를 각각 10, 6이라고 가정하면 3등에 당첨될 수 있습니다.
  - 3등을 만드는 다른 방법들도 존재합니다. 하지만, 2등 이상으로 만드는 것은 불가능합니다.
- 알아볼 수 없는 두 개의 번호를 각각 11, 7이라고 가정하면 5등에 당첨될 수 있습니다.
  - 5등을 만드는 다른 방법들도 존재합니다. 하지만, 6등(낙첨)으로 만드는 것은 불가능합니다.

## 로또의 최고 순위와 최저 순위

민우가 구매한 로또 번호를 담은 배열 lottos, 당첨 번호를 담은 배열 win_nums가 매개변수로 주어집니다. 이때, 당첨 가능한 최고 순위와 최저 순위를 차례대로 배열에 담아서 return 하도록 solution 함수를 완성해주세요.

### 제한사항

- lottos는 길이 6인 정수 배열입니다.
- lottos의 모든 원소는 0 이상 45 이하인 정수입니다.
  - 0은 알아볼 수 없는 숫자를 의미합니다.
  - 0을 제외한 다른 숫자들은 lottos에 2개 이상 담겨있지 않습니다.
  - lottos의 원소들은 정렬되어 있지 않을 수도 있습니다.
- win_nums은 길이 6인 정수 배열입니다.
- win_nums의 모든 원소는 1 이상 45 이하인 정수입니다.
  - win_nums에는 같은 숫자가 2개 이상 담겨있지 않습니다.
  - win_nums의 원소들은 정렬되어 있지 않을 수도 있습니다.

---

### 입출력 예

| lottos | win_nums | result |
|---|---|---|
| [44, 1, 0, 0, 31, 25] | [31, 10, 45, 1, 6, 19] | [3, 5] |
| [0, 0, 0, 0, 0, 0] | [38, 19, 20, 40, 15, 25] | [1, 6] |
| [45, 4, 35, 20, 3, 9] | [20, 9, 3, 45, 4, 35] | [1, 1] |

### 입출력 예 설명

입출력 예 #1
문제 예시와 같습니다.

입출력 예 #2
알아볼 수 없는 번호들이 아래와 같았다면, 1등과 6등에 당첨될 수 있습니다.

```
def solution(lottos, win_nums):

    prize = {
        6 : 1, 5 : 2, 4 : 3, 3 : 4, 2 : 5, 1 : 6 , 0 : 6
    }


    answer = []
    zero_num = lottos.count(0)
    suc_num = 0

    lottos.sort()
    win_nums.sort()

    for i in lottos:
        if i in win_nums:
            suc_num += 1


    answer.append(prize[suc_num + zero_num])
    answer.append(prize[suc_num])

    return answer
```

```
def solution(lottos, win_nums):

    rank=[6,6,5,4,3,2,1]

    cnt_0 = lottos.count(0)
    ans = 0
    for x in win_nums:
        if x in lottos:
            ans += 1
    return rank[cnt_0 + ans],rank[ans]
```

### 3) 폰켓몬

**폰켓몬**

**문제 설명**

당신은 폰켓몬을 잡기 위한 오랜 여행 끝에, 홍 박사님의 연구실에 도착했습니다. 홍 박사님은 당신에게 자신의 연구실에 있는 총 N 마리의 폰켓몬 중에서 N/2마리를 가져가도 좋다고 했습니다.
홍 박사님 연구실의 폰켓몬은 종류에 따라 번호를 붙여 구분합니다. 따라서 같은 종류의 폰켓몬은 같은 번호를 가지고 있습니다. 예를 들어 연구실에 총 4마리의 폰켓몬이 있고, 각 폰켓몬의 종류 번호가 [3번, 1번, 2번, 3번]이라면 이는 3번 폰켓몬 두 마리, 1번 폰켓몬 한 마리, 2번 폰켓몬 한 마리가 있음을 나타냅니다. 이때, 4마리의 폰켓몬 중 2마리를 고르는 방법은 다음과 같이 6가지가 있습니다.

1. 첫 번째(3번), 두 번째(1번) 폰켓몬을 선택
2. 첫 번째(3번), 세 번째(2번) 폰켓몬을 선택
3. 첫 번째(3번), 네 번째(3번) 폰켓몬을 선택
4. 두 번째(1번), 세 번째(2번) 폰켓몬을 선택
5. 두 번째(1번), 네 번째(3번) 폰켓몬을 선택
6. 세 번째(2번), 네 번째(3번) 폰켓몬을 선택

이때, 첫 번째(3번) 폰켓몬과 네 번째(3번) 폰켓몬을 선택하는 방법은 한 종류(3번 폰켓몬 두 마리)의 폰켓몬만 가질 수 있지만, 다른 방법들은 모두 두 종류의 폰켓몬을 가질 수 있습니다. 따라서 위 예시에서 가질 수 있는 폰켓몬 종류 수의 최댓값은 2가 됩니다.
당신은 최대한 다양한 종류의 폰켓몬을 가지길 원하기 때문에, 최대한 많은 종류의 폰켓몬을 포함해서 N/2마리를 선택하려 합니다. N마리 폰켓몬의 종류 번호가 담긴 배열 nums가 매개변수로 주어질 때, N/2마리의 폰켓몬을 선택하는 방법 중, 가장 많은 종류의 폰켓몬을 선택하는 방법을 찾아, 그때의 폰켓몬 종류 번호의 개수를 return 하도록 solution 함수를 완성해주세요.

**제한사항**

- nums는 폰켓몬의 종류 번호가 담긴 1차원 배열입니다.
- nums의 길이(N)는 1 이상 10,000 이하의 자연수이며, 항상 짝수로 주어집니다.
- 폰켓몬의 종류 번호는 1 이상 200,000 이하의 자연수로 나타냅니다.
- 가장 많은 종류의 폰켓몬을 선택하는 방법이 여러 가지인 경우에도, 선택할 수 있는 폰켓몬 종류 개수의 최댓값 하나만 return 하면 됩니다.

입출력 예

| nums | result |
| --- | --- |
| [3,1,2,3] | 2 |
| [3,3,3,2,2,4] | 3 |
| [3,3,3,2,2,2] | 2 |

입출력 예 설명

입출력 예 #1
문제의 예시와 같습니다.

입출력 예 #2
6마리의 폰켓몬이 있으므로, 3마리의 폰켓몬을 골라야 합니다.
가장 많은 종류의 폰켓몬을 고르기 위해서는 3번 폰켓몬 한 마리, 2번 폰켓몬 한 마리, 4번 폰켓몬 한 마리를 고르면 되며, 따라서 3을 return 합니다.

입출력 예 #3
6마리의 폰켓몬이 있으므로, 3마리의 폰켓몬을 골라야 합니다.
가장 많은 종류의 폰켓몬을 고르기 위해서는 3번 폰켓몬 한 마리와 2번 폰켓몬 두 마리를 고르거나, 혹은 3번 폰켓몬 두 마리와 2번 폰켓몬 한 마리를 고르면 됩니다. 따라서 최대 고를 수 있는 폰켓몬 종류의 수는 2입니다.

```
def solution(nums):
    answer = 0
    kinds = len(set(nums))
    take_num = len(nums) / 2

    if kinds >= take_num:
        answer += take_num
    else:
        answer += kinds
    return answer
```

```
def solution(ls):
    return min(len(ls)/2, len(set(ls)))
```

#### 4) 신규 아이디 추천

## 신규 아이디 추천

**문제 설명**

카카오에 입사한 신입 개발자 `네오` 는 "카카오계정개발팀"에 배치되어, 카카오 서비스에 가입하는 유저들의 아이디를 생성하는 업무를 담당하게 되었습니다. "네오"에게 주어진 첫 업무는 새로 가입하는 유저들이 카카오 아이디 규칙에 맞지 않는 아이디를 입력했을 때, 입력된 아이디와 유사하면서 규칙에 맞는 아이디를 추천해주는 프로그램을 개발하는 것입니다.
다음은 카카오 아이디의 규칙입니다.

- 아이디의 길이는 3자 이상 15자 이하여야 합니다.
- 아이디는 알파벳 소문자, 숫자, 빼기( `-` ), 밑줄( `_` ), 마침표( `.` ) 문자만 사용할 수 있습니다.
- 단, 마침표( `.` )는 처음과 끝에 사용할 수 없으며 또한 연속으로 사용할 수 없습니다.

"네오"는 다음과 같이 7단계의 순차적인 처리 과정을 통해 신규 유저가 입력한 아이디가 카카오 아이디 규칙에 맞는 지 검사하고 규칙에 맞지 않은 경우 규칙에 맞는 새로운 아이디를 추천해 주려고 합니다.
신규 유저가 입력한 아이디가 `new_id` 라고 한다면,

```
1단계 new_id의 모든 대문자를 대응되는 소문자로 치환합니다.
2단계 new_id에서 알파벳 소문자, 숫자, 빼기(-), 밑줄(_), 마침표(.)를 제외한 모든 문자를 제거합니다
3단계 new_id에서 마침표(.)가 2번 이상 연속된 부분을 하나의 마침표(.)로 치환합니다.
4단계 new_id에서 마침표(.)가 처음이나 끝에 위치한다면 제거합니다.
5단계 new_id가 빈 문자열이라면, new_id에 "a"를 대입합니다.
6단계 new_id의 길이가 16자 이상이면, new_id의 첫 15개의 문자를 제외한 나머지 문자들을 모두 제거합
      만약 제거 후 마침표(.)가 new_id의 끝에 위치한다면 끝에 위치한 마침표(.) 문자를 제거합니다.
7단계 new_id의 길이가 2자 이하라면, new_id의 마지막 문자를 new_id의 길이가 3이 될 때까지 반복해
```

예를 들어, new_id 값이 "...!@BaT#*..y.abcdefghijklm" 라면, 위 7단계를 거치고 나면 new_id는 아래와 같이 변경됩니다.

1단계 대문자 'B'와 'T'가 소문자 'b'와 't'로 바뀌었습니다.
`"...!@BaT#*..y.abcdefghijklm"` → `"...!@bat#*..y.abcdefghijklm"`

2단계 '!', '@', '#', '*' 문자가 제거되었습니다.
`"...!@bat#*..y.abcdefghijklm"` → `"...bat..y.abcdefghijklm"`

3단계 '...'와 '..' 가 '.'로 바뀌었습니다.
`"...bat..y.abcdefghijklm"` → `".bat.y.abcdefghijklm"`

4단계 아이디의 처음에 위치한 '.'가 제거되었습니다.
`".bat.y.abcdefghijklm"` → `"bat.y.abcdefghijklm"`

5단계 아이디가 빈 문자열이 아니므로 변화가 없습니다.
`"bat.y.abcdefghijklm"` → `"bat.y.abcdefghijklm"`

6단계 아이디의 길이가 16자 이상이므로, 처음 15자를 제외한 나머지 문자들이 제거되었습니다.
`"bat.y.abcdefghijklm"` → `"bat.y.abcdefghi"`

7단계 아이디의 길이가 2자 이하가 아니므로 변화가 없습니다.
`"bat.y.abcdefghi"` → `"bat.y.abcdefghi"`

따라서 신규 유저가 입력한 new_id가 "...!@BaT#*..y.abcdefghijklm"일 때, 네오의 프로그램이 추천하는 새로운 아이디는 "bat.y.abcdefghi" 입니다.

---

**[문제]**

신규 유저가 입력한 아이디를 나타내는 new_id가 매개변수로 주어질 때, "네오"가 설계한 7단계의 처리 과정을 거친 후의 추천 아이디를 return 하도록 solution 함수를 완성해 주세요.

**[제한사항]**

new_id는 길이 1 이상 1,000 이하인 문자열입니다.
new_id는 알파벳 대문자, 알파벳 소문자, 숫자, 특수문자로 구성되어 있습니다.
new_id에 나타날 수 있는 특수문자는 `-_.~!@#$%^&*()=+[{]}:?,<>/` 로 한정됩니다.

---

**[입출력 예]**

| no | new_id | result |
|---|---|---|
| 예1 | `"...!@BaT#*..y.abcdefghijklm"` | `"bat.y.abcdefghi"` |
| 예2 | `"z-+.^."` | `"z--"` |
| 예3 | `"=.="` | `"aaa"` |
| 예4 | `"123_.def"` | `"123_.def"` |
| 예5 | `"abcdefghijklmn.p"` | `"abcdefghijklmn"` |

**입출력 예에 대한 설명**

---

**입출력 예 #1**
문제의 예시와 같습니다.

**입출력 예 #2**
7단계를 거치는 동안 new_id가 변화하는 과정은 아래와 같습니다.

```
def solution(new_id):

    edit_id = ''

    special = ['~', '!', '@' , '#', '$', '%', '^', '&', '*', '(', ')', '=', '+', '[', '{', ']', '}', ':', '?', ',', '<', '>', '/']

    # 1단계
    for i in list(new_id):
        if i.isupper():
            x = i.lower()
            edit_id += x
        else:
            edit_id += i

    # 2단계
    for i in edit_id:
        if i in special:
             edit_id = edit_id.replace(i, '')
        else:
             continue
```

```
    # 3단계
    check = '..' in edit_id

    while check:
        edit_id = edit_id.replace('..', '.')
        check = '..' in edit_id


    # 4단계
    ls = list(edit_id)

    if ls[0] == '.':
        ls = ls[1:]

    if list(edit_id)[-1] == '.':
        ls = ls[:-1]
    edit_id = ''.join(ls)


    # 5단계
    if edit_id == '':
        edit_id += 'a'

    # 6단계
    if len(edit_id) >= 16:
        edit_id = edit_id[:15] #TIL 문자열도 슬라이싱이 된다.

        if edit_id[-1] == '.':
            edit_id = edit_id[:-1]

    # 7단계
    while len(edit_id) <= 2:
        edit_id += edit_id[-1]
    return edit_id
```

```
def solution(new_id):
    answer = ''
    # 1단계 - 대문자 소문자로 바꾸기
    new_id = new_id.lower()   # 기가 막힌 부분..
    # 2단계 - 특수문자 제거
    for c in new_id:
        if c.isalpha() or c.isdigit() or c in ['-', '_', '.']:
            answer += c
    # 3단계 - 치환
    while '..' in answer:   # 내가 했을 땐 왜 안됐을까? 변수지정을 안한 것 같다.
        answer = answer.replace('..', '.')
    # 4단계 - 양쪽 . 제거
    if answer[0] == '.':
        answer = answer[1:] if len(answer) > 1 else '.'
    if answer[-1] == '.':
        answer = answer[:-1]
    # 5단계 - 빈배열 채우기
    if answer == '': # 빈 문자열 표현
        answer = 'a'
    # 6단계 15자이상 짜르기
    if len(answer) > 15:  # 슬라이싱 인덱스 설정 주의하자. //
        answer = answer[:15]
        if answer[-1] == '.':
            answer = answer[:-1]
    # 7단계 3자이하 채우기
    while len(answer) < 3:
        answer += answer[-1]
    return answer
```

```
#정규표현식으로 풀기

import re

def solution(new_id):
    st = new_id
    st = st.lower()
    st = re.sub('[^a-z0-9\-_.]', '', st)
    st = re.sub('\.+', '.', st)
    st = re.sub('^[.]|[.]$', '', st)
    st = 'a' if len(st) == 0 else st[:15]
    st = re.sub('^[.]|[.]$', '', st)
    st = st if len(st) > 2 else st + "".join([st[-1] for i in range(3-len(st))])
    return st
```

### 5) 모의고사 (못 푼 문제)

**모의고사**

**문제 설명**

수포자는 수학을 포기한 사람의 준말입니다. 수포자 삼인방은 모의고사에 수학 문제를 전부 찍으려 합니다. 수포자는 1번 문제부터 마지막 문제까지 다음과 같이 찍습니다.

1번 수포자가 찍는 방식: 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, ...
2번 수포자가 찍는 방식: 2, 1, 2, 3, 2, 4, 2, 5, 2, 1, 2, 3, 2, 4, 2, 5, ...
3번 수포자가 찍는 방식: 3, 3, 1, 1, 2, 2, 4, 4, 5, 5, 3, 3, 1, 1, 2, 2, 4, 4, 5, 5, ...

1번 문제부터 마지막 문제까지의 정답이 순서대로 들은 배열 answers가 주어졌을 때, 가장 많은 문제를 맞힌 사람이 누구인지 배열에 담아 return 하도록 solution 함수를 작성해주세요.

**제한 조건**

- 시험은 최대 10,000 문제로 구성되어있습니다.
- 문제의 정답은 1, 2, 3, 4, 5중 하나입니다.
- 가장 높은 점수를 받은 사람이 여럿일 경우, return하는 값을 오름차순 정렬해주세요.

**입출력 예**

| answers | return |
|---|---|
| [1,2,3,4,5] | [1] |
| [1,3,2,4,2] | [1,2,3] |

**입출력 예 설명**

입출력 예 #1

- 수포자 1은 모든 문제를 맞혔습니다.
- 수포자 2는 모든 문제를 틀렸습니다.
- 수포자 3은 모든 문제를 틀렸습니다.

따라서 가장 문제를 많이 맞힌 사람은 수포자 1입니다.

```
#내가 쓴 코드 - 50점

def solution(answers):
    answer = []

    # 정답 개수 변수 설정
    one_count = 0
    two_count = 0
    three_count = 0
    problem_num = len(answers)
```

```
    #1, 2, 3번 수포자 답안지 생성
    one_solve = [1, 2, 3, 4, 5] * ((problem_num // 5) + 1)
    one_solve_f = one_solve[:problem_num]

    two_solve = [2, 1, 2, 3, 2, 4, 2, 5] * ((problem_num // 8) + 1)
    two_solve_f = two_solve[:problem_num]

    three_solve = [3, 3, 1, 1, 2, 2, 4, 4, 5, 5] * ((problem_num // 10) + 1)
    three_solve_f = three_solve[:problem_num]

    # 정답 갯수 구하기
    for i, j in zip(one_solve_f, answers):
        if i == j:
            one_count += 1

    for i, j in zip(two_solve_f, answers):
        if i == j:
            two_count += 1

    for i, j in zip(three_solve_f, answers):
        if i == j:
            three_count += 1


    if one_count == max(one_count, two_count, three_count):
        answer.append(1)

    if two_count == max(one_count, two_count, three_count):
        answer.append(2)

    if two_count == max(one_count, two_count, three_count):
        answer.append(3)

    answer = sorted(answer)


    return answer
```

```
def solution(answers):
    counts = [0,0,0]
    winner = []
    s1 = [1, 2, 3, 4, 5]
    s2 = [2, 1, 2, 3, 2, 4, 2, 5]
    s3 = [3, 3, 1, 1, 2, 2, 4, 4, 5, 5,]

    for i in range(len(answers)):
        if answers[i] == s1[(i%5)]:
            counts[0] += 1
        if answers[i] == s2[(i%8)]:
            counts[1] += 1
        if answers[i] == s3[(i%10)]:
            counts[2] += 1

    for i in range(3):
        if counts[i] == max(counts):
            winner.append(i+1)

    return winner
```

```
from itertools import cycle

def solution(answers):
    giveups = [
        cycle([1,2,3,4,5]),
        cycle([2,1,2,3,2,4,2,5]),
        cycle([3,3,1,1,2,2,4,4,5,5]),
    ]
    scores = [0, 0, 0]
    for num in answers:
        for i in range(3):
            if next(giveups[i]) == num:
                scores[i] += 1
    highest = max(scores)

    return [i + 1 for i, v in enumerate(scores) if v == highest]
```

### 6) 약수의 개수와 덧셈

약수의 개수와 덧셈

**문제 설명**

두 정수 `left` 와 `right` 가 매개변수로 주어집니다. `left` 부터 `right` 까지의 모든 수들 중에서, 약수의 개수가 짝수인 수는 더하고, 약수의 개수가 홀수인 수는 뺀 수를 return 하도록 solution 함수를 완성해주세요.

---

**제한사항**

- 1 ≤ `left` ≤ `right` ≤ 1,000

---

**입출력 예**

| left | right | result |
|---|---|---|
| 13 | 17 | 43 |
| 24 | 27 | 52 |

---

**입출력 예 설명**

**입출력 예 #1**

- 다음 표는 13부터 17까지의 수들의 약수를 모두 나타낸 것입니다.

| 수 | 약수 | 약수의 개수 |
|---|---|---|
| 13 | 1, 13 | 2 |
| 14 | 1, 2, 7, 14 | 4 |
| 15 | 1, 3, 5, 15 | 4 |
| 16 | 1, 2, 4, 8, 16 | 5 |
| 17 | 1, 17 | 2 |

- 따라서, 13 + 14 + 15 - 16 + 17 = 43을 return 해야 합니다.

**입출력 예 #2**

- 다음 표는 24부터 27까지의 수들의 약수를 모두 나타낸 것입니다.

| 수 | 약수 | 약수의 개수 |
|---|---|---|
| 24 | 1, 2, 3, 4, 6, 8, 12, 24 | 8 |
| 25 | 1, 5, 25 | 3 |
| 26 | 1, 2, 13, 26 | 4 |
| 27 | 1, 3, 9, 27 | 4 |

- 따라서, 24 - 25 + 26 + 27 = 52를 return 해야 합니다.

```
def get_num(n):

    count = 0

    for i in range(1, n+1):
        if n % i == 0:
            count += 1

    return count


def solution(left, right):
    answer = 0

    for i in range(left, right+1):
        if get_num(i) % 2 == 0:
            answer += i

        else:
            answer -= i

    return answer
```

```
# 1위

def solution(left, right):
    answer = 0
    for i in range(left,right+1):
        if int(i**0.5)==i**0.5:
            answer -= i
        else:
            answer += i
    return answer
```

```
#2위

import math

def solution(left, right):
    answer = 0
    for i in range(left, right + 1, 1):
        sqrt = math.sqrt(i)
        if int(sqrt) == sqrt:
            answer -= i
        else:
            answer += i

    return answer
```

### 7) 직사각형 좌표 구하기 - 부스트캠프 데모 문제

**문제 설명**

직사각형을 만드는 데 필요한 4개의 점 중 3개의 좌표가 주어질 때, 나머지 한 점의 좌표를 구하려고 합니다. 점 3개의 좌표가 들어있는 배열 v가 매개변수로 주어질 때, 직사각형을 만드는 데 필요한 나머지 한 점의 좌표를 return 하도록 solution 함수를 완성해주세요. 단, 직사각형의 각 변은 x축, y축에 평행하며, 반드시 직사각형을 만들 수 있는 경우만 입력으로 주어집니다.

**제한사항**

- v는 세 점의 좌표가 들어있는 2차원 배열입니다.
- v의 각 원소는 점의 좌표를 나타내며, 좌표는 [x축 좌표, y축 좌표] 순으로 주어집니다.
- 좌표값은 1 이상 10억 이하의 자연수입니다.
- 직사각형을 만드는 데 필요한 나머지 한 점의 좌표를 [x축 좌표, y축 좌표] 순으로 담아 return 해주세요.

---

**입출력 예**

| v | result |
|---|---|
| [[1, 4], [3, 4], [3, 10]] | [1, 10] |
| [[1, 1], [2, 2], [1, 2]] | [2, 1] |

**입출력 예 설명**

입출력 예 #1
세 점이 [1, 4], [3, 4], [3, 10] 위치에 있을 때, [1, 10]에 점이 위치하면 직사각형이 됩니다.

입출력 예 #2
세 점이 [1, 1], [2, 2], [1, 2] 위치에 있을 때, [2, 1]에 점이 위치하면 직사각형이 됩니다.

```
def solution(v):


    answer = []
    # x,y 좌표 리스트 생성
    x_ls = []
    y_ls = []

    # x,y 좌표 리스트에 삽입
    for i in v:
        x_ls.append(i[0])
        y_ls.append(i[1])

    # x 좌표 리스트 내에 갯수가 1개인 원소를 answer 리스트에 추가
    for i in x_ls:
```

```
        if x_ls.count(i) == 1:
            answer.append(i)
    # y 좌표 리스트 내에 갯수가 1개인 원소를 answer 리스트에 추가
    for i in y_ls:
        if y_ls.count(i) == 1:
            answer.append(i)


    return answer
```