

문제 1, 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

- $1 \leq \text{left} \leq \text{right} \leq 1,000$

입출력 예

left	right	result
13	17	43
24	27	52

문제 1, 약수의 개수와 덧셈

```
def solution(left, right):
    answer = 0
    for i in range(left, right+1): #left 부터 right까지 1씩 증가하는 for문
        now_count = 0;           #약수의 개수를 담기위한 변수
        for j in range(1, i+1):   #1부터 i까지 증가하며 약수를 찾아냅니다.
            if i % j == 0:        #나누어 떨어지는 수는 약수!
                now_count +=1;    #약수라면 개수를 증가시켜줍니다.

        if now_count % 2 == 0:    #이제 개수가 홀수인지 짝수인지 판별하여
            answer += i          #짝수라면 더해주고
        else:
            answer -= i          #홀수라면 빼줍니다.

    return answer                #끝!
```

```
#include <string>
#include <vector>

using namespace std;

int solution(int left, int right) {
    int answer = 0;
    for(int i = left; i <= right; ++i){
        int count = 0;
        for(int j = 1; j <= i; ++j)
            if (i % j == 0)
                ++count;
        if (count % 2 == 1)
            answer -= i;
        else
            answer += i;
    }
    return answer;
}
```

```
class Solution {
public int solution(int left, int right) {
    int answer = 0;

    for(int i = left; i <= right; i++) {
        int cnt = 0;

        for(int j = 1; j <= i; j++) {
            if(i % j == 0) cnt++;
        }

        if(cnt % 2 == 0) answer += i;
        else answer -= i;
    }

    return answer;
}
```

문제 2, 직사각형 별찍기

직사각형 별찍기

문제 설명

이 문제에는 표준 입력으로 두 개의 정수 n 과 m 이 주어집니다.
별(*) 문자를 이용해 가로의 길이가 n , 세로의 길이가 m 인 직사각형 형태를 출력해보세요.

제한 조건

- n 과 m 은 각각 1000 이하인 자연수입니다.

예시

입력

5 3

출력

```
*****  
*****  
*****
```

문제 2, 직사각형 별찍기

```
a, b = map(int, input().strip().split(' '))
for i in range(b):
    for j in range(a):
        print('*', end='')
    print()
```

```
#include <iostream>

using namespace std;

int main(void) {
    int a;
    int b;
    int i, j;

    cin >> a >> b;

    for(i = 0; i < b; i++) {
        for(j = 0; j < a; j++) {
            cout << "*";
        }
        cout << endl;
    }

    return 0;
}
```

```
import java.util.Scanner;
public class Solution {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();

        for(int i = 0; i < b ; i ++ ){
            for(int j = 0; j < a; j++){
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

문제 3, 복리

`pow(a,b)` //a의 b 승 (double 형태로 반환된다.)

`#include<iomanip>`

`setw(자릿수)` // (자릿수 만큼) 공백 생성

```
year = 0

money = 1000

while money<2000:

    money += money* 0.07

    year+=1

print(year, "년")
```

```
#include<iostream>
#include<cmath>
#include<iomanip>

using namespace std;

int main() {

    double amount = 0.0;
    double deposit = 1000.0;
    double rate = 0.05;

    cout << setw(10) << "Year" << setw(21) << "Amount on deposit" << endl; //10, 21
    cout << setprecision(2) << fixed; //소수점
    for (int i = 1; i <= 10; i++) {

        amount = deposit*pow(1 + rate, i); // 1+

        cout << setw(10) << i << setw(21) << amount << endl;

    }

}
```

```
package com.naver;

public class Test {
    public static void main(String[] args) {
        // 잔액=원금+(원금*이자율)
        double money = 10000; // 원금
        double rate = 0.1; // 10%의 금리
        int year = 10; // 예치 기간

        money = money + (money * rate);

        for (int i = 1; i <= year; i++) {
            money = money + (money * rate);
            System.out.println(i + "년 후의 금액" + money);
        }
    }
}
```

문제 4, 로또 번호 자동생성기

```
import random

print('***** 로또 번호 자동 생성기*****')
print('-----')

num = input('게임 수 : ')

for i in range(0,int(num)):
    lotto = random.sample(range(1,46),6)
    lotto.sort()
    print(lotto)

print('***** 로또 번호 자동 완료 *****')
```

```
import random

while(True):
    lottoQuantity = int(input("로또를 몇장 구매하시겠습니까? "))
    if lottoQuantity <= 0:
        print("종료합니다.")
        break
    while lottoQuantity>0:
        print("랜덤하게 생성된 로또 번호입니다.")
        for i in range(1,lottoQuantity+1):
            print("[%d]: " % i, end=" ")
            for j in range(6):
                #소수점 3번째 자리까지 보여주세요. 0:3d / 하지만 randint로 받기 때문에 의미 없음.
                print("{0:3d}".format(random.randint(1,45)), end=" ")
            print()
        break
```

문제 4, 로또 번호 자동생성기

```
1 #include <iostream> // 표준입출력 함수 사용을 위한 헤더파일
2 // C++ 에서는 iostream 을 포함하면 rand 함수를 사용할 수 있습니다.
3 #include <ctime> // 현재 시간 함수 사용을 위한 헤더파일
4
5 using namespace std;
6
7 int main()
8 {
9     int lotto[6]; // 생성된 번호를 저장할 배열
10    unsigned int seed; // 사용자에게 행운의 번호를 입력받을 변수
11    int i, j; // 반복을 위한 변수
12    int temp; // 값 교환 및 중복체크를 위한 임시변수
13    int cnt = 0; // 중복 횟수를 세기 위한 변수
14
15    // 현재 시간으로 랜덤함수의 시드값을 지정
16    srand(static_cast<unsigned int>(time(NULL)));
17
18    while (cnt != 7) // 숫자 중복의 횟수가 7이 아닐 동안 반복합니다.
19    {
20        cnt = 0; // 중복 횟수 초기화
21        for (i = 0; i < 6; i++)
22        { // 번호 6개 생성
23            temp = rand() % 45 + 1; // 범위는 1부터 45까지입니다.
24            lotto[i] = temp; // 현재 생성된 번호가
25            for (j = 0; j < i; j++) // 이전에 생성된 번호와 같은지 비교합니다.
26            {
27                if (lotto[j] == temp) // 중복된 게 있으면
28                { // 중복되지 않도록 다시 생성합니다.
29                    i--;
30                    cnt++; // 중복의 횟수도 증가시킵니다.
31                }
32            }
33            // 번호 6개가 생성되었을 때 중복의 횟수가 7회이면 종료됩니다.
34            // 이리하면 뭔가 조금 더 유니크한 숫자들이 나오지 않을까 싶어서 추가한 겁니다...
35        }
36
37        // 오름차순 정렬
38        for (i = 0; i < 5; i++)
39        {
40            for (j = i + 1; j < 6; j++)
41            {
42                if (lotto[i] > lotto[j])
43                {
44                    temp = lotto[i];
45                    lotto[i] = lotto[j];
46                    lotto[j] = temp;
47                }
48            }
49        }
50
51        //정렬 후 최종 출력
52        cout << "***** 로또 복권번호 생성기 *****\n\n";
53        cout << "생성된 행운의 번호 : ";
54        for (i = 0; i < 6; i++)
55        {
56            cout << lotto[i] << " ";
57        }
58        cout << endl << endl << "프로그램을 종료합니다." << endl;
59        system("pause"); // 프로그램이 자동으로 종료되지 않도록 일시정지합니다.
60        return 0;
61    }
```

```
package com.kh.homework_method;
import java.util.*;

public class BinaidaBinaida {
    public void start() { //번호 6+1(보너스) 각 45개
        System.out.println("+++++");
        Scanner sc=new Scanner(System.in);

        System.out.print("1.자동번호 생성 횟수: ");
        int cycle=sc.nextInt();
        int[] arr=new int[7];
        int m=0;
        while(cycle>0) {
            for(int i=0; i<arr.length;i++) {
                arr[i] = (int)(Math.random()*45)+1;
                for(int j=0; j<i;j++) {
                    if(arr[i]==arr[j]) {
                        i--;
                    }
                }
            }
            for(int i=0; i<arr.length-1;i++) {
                System.out.printf("%5d", arr[i]);
            }
            System.out.printf("   +%5d", arr[6]);
            System.out.println();
            cycle--;
        }
        System.out.println("+++++");
    }
}
```

문제 5, 휴대폰 가리기

핸드폰 번호 가리기

문제 설명

프로그래머스 모바일은 개인정보 보호를 위해 고지서를 보낼 때 고객들의 전화번호의 일부를 가립니다.

전화번호가 문자열 `phone_number`로 주어졌을 때, 전화번호의 뒷 4자리를 제외한 나머지 숫자를 전부 *으로 가린 문자열을 리턴하는 함수, `solution`을 완성해주세요.

제한 조건

- `phone_number`는 길이 4 이상, 20이하인 문자열입니다.

입출력 예

phone_number	return
"01033334444"	"*****4444"
"027778888"	"*****8888"

문제 6, 휴대폰 가리기

```
def solution(phone_number):
    answer = ''

    phone_number_len = len(phone_number)

    answer = '*' * (phone_number_len - 4)

    answer += phone_number[-4:]

    return answer
```

```
#include <string>
#include <vector>
#include <iostream>

using namespace std;

string solution(string phone_number) {
    size_t s = phone_number.size();
    int stars = s - 4;
    cout << stars;

    for (int i = 0; i < stars; i++)
        phone_number[i] = '*';

    return phone_number;
}
```

```
class Solution {
public String solution(String phone_number) {
    String answer = "";
    for(int i = 0; i < phone_number.length() - 4){
        if(i < phone_number.length() - 4){
            answer += "*";
        }
        else{
            answer += phone_number.charAt(i);
        }
    }
    return answer;
}
```

문제 7, 최대공약수와 최소공배수

문제 설명

두 수를 입력받아 두 수의 최대공약수와 최소공배수를 반환하는 함수, solution을 완성해 보세요. 배열의 맨 앞에 최대공약수, 그다음 최소공배수를 넣어 반환하면 됩니다. 예를 들어 두 수 3, 12의 최대공약수는 3, 최소공배수는 12이므로 solution(3, 12)는 [3, 12]를 반환해야 합니다.

제한 사항

- 두 수는 1이상 1000000이하의 자연수입니다.

입출력 예

n	m	return
3	12	[3, 12]
2	5	[1, 10]

입출력 예 설명

입출력 예 #1

위의 설명과 같습니다.

입출력 예 #2

자연수 2와 5의 최대공약수는 1, 최소공배수는 10이므로 [1, 10]을 리턴해야 합니다.

문제 7, 최대공약수와 최소공배수

```
import math
def solution(n, m):
    answer = []
    # 최대공약수
    for i in range(min(n,m),0,-1):
        if n%i ==0 and m%i==0:
            answer.append(i)
            break

    # 최소공배수
    for i in range(max(n,m),n*m+1):
        if i%n == 0 and i%m == 0:
            answer.append(i)
            break
    return answer
```

유클리드 호제법

: 두 양의 정수, 혹은 두 다항식의 최대공약수를 구하는 방법

유클리드 호제법

두 양의 정수 a, b ($a > b$)에 대하여 $a = bq + r$ ($0 \leq r < b$)라 하면, a, b 의 최대공약수는 b, r 의 최대공약수와 같다. 즉,

$$\gcd(a, b) = \gcd(b, r)$$

$r = 0$ 이라면, a, b 의 최대공약수는 b 가 된다.

3. 활용

알고리즘이라는 이름에 걸맞게, 위 성질을 한 번만 사용해서는 제대로 된 활용이 힘들다. 보통은 나머지가 0이 될 때 까지 연속해서 사용한다. 예를 들면 아래와 같다.

$$\begin{aligned} b &= aq_1 + r_1 \quad (0 < r_1 < a) \\ a &= r_1q_2 + r_2 \quad (0 < r_2 < r_1) \\ r_1 &= r_2q_3 + r_3 \quad (0 < r_3 < r_2) \\ &\vdots \\ r_{n-2} &= r_{n-1}q_n + r_n \quad (0 < r_n < r_{n-1}) \\ r_{n-1} &= r_nq_{n+1} \\ \therefore \gcd(a, b) &= r_n \end{aligned}$$

```
class Solution {
    public int gcd(int n, int m) {
        int[] answer = new int[2];
        answer[0] = gcd(n, m);
        answer[1] = lcm(n, m);
        return answer[0];
    }

    private int gcd(int a, int b) {
        if (b == 0) return a;
        return gcd(b, a % b);
    }

    private int lcm(int a, int b) {
        return a * b / gcd(a, b);
    }
}
```

문제 7, 최대공약수와 최소공배수

```
import math
def solution(n, m):
    answer = []
    # 최대공약수
    for i in range(min(n,m),0,-1):
        if n%i ==0 and m%i==0:
            answer.append(i)
            break

    # 최소공배수
    for i in range(max(n,m),n*m+1):
        if i%n == 0 and i%m == 0:
            answer.append(i)
            break
    return answer
```

```
#include <string>
#include <vector>

using namespace std;

vector<int> solution(int n, int m) {
    vector<int> answer;
    int a, b, r;

    a = n;
    b = m;

    while(b != 0) {
        r = a % b;
        a = b;
        b = r;
    }

    answer.push_back(a);
    answer.push_back(n * m / a);

    return answer;
}
```

```
class Solution {
public int[] solution(int n, int m) {
    int[] answer = new int[2];
    int big = Math.max(n, m);
    int small = Math.min(n, m);

    answer[0] = gcd(big, small);
    answer[1] = big*small/answer[0];

    return answer;
}

static int gcd(int a, int b) {
    if(a % b == 0) {
        return b;
    }
    return gcd(b, a%b);
}
```

문제 7, 최대공약수와 최소공배수

유클리드 호제법

: 두 양의 정수, 혹은 두 다항식의 최대공약수를 구하는 방법

유클리드 호제법

두 양의 정수 a, b ($a > b$)에 대하여 $a = bq + r$ ($0 \leq r < b$)라 하면, a, b 의 최대공약수는 b, r 의 최대공약수와 같다. 즉,

$$\gcd(a, b) = \gcd(b, r)$$

$r = 0$ 이라면, a, b 의 최대공약수는 b 가 된다.

3. 활용

[편집]

알고리즘이라는 이름에 걸맞게, 위 성질을 한 번만 사용해서는 제대로 된 활용이 힘들다. 보통은 나머지가 0이 될 때 까지 연속해서 사용한다. 예를 들면 아래와 같다.

$$\begin{aligned} b &= aq_1 + r_1 \quad (0 < r_1 < a) \\ a &= r_1q_2 + r_2 \quad (0 < r_2 < r_1) \\ r_1 &= r_2q_3 + r_3 \quad (0 < r_3 < r_2) \\ &\vdots \\ r_{n-2} &= r_{n-1}q_n + r_n \quad (0 < r_n < r_{n-1}) \\ r_{n-1} &= r_nq_{n+1} \\ &\therefore \gcd(a, b) = r_n \end{aligned}$$

문제 8, 수박수박수박수박수?

길이가 n 이고, "수박수박수박수...."와 같은 패턴을 유지하는 문자열을 리턴하는 함수, `solution`을 완성하세요. 예를들어 n 이 4이면 "수박수박"을 리턴하고 3이라면 "수박수"를 리턴하면 됩니다.

제한 조건

- n 은 길이 10,000이하인 자연수입니다.

입출력 예

n	return
3	"수박수"
4	"수박수박"

문제 8, 수박수박수박수박수박수?

```
def solution(n):
    answer = ''
    for i in range(1, n+1):
        if i % 2 != 0:
            answer += '수'
        else:
            answer += '박'
    return answer
```

```
def solution(n):
    s = '수박' * n
    return s[:n]
```

```
def solution(s):
    return '수박' * (n//2) + '수' * (n%2)
```

```
#include <string>
#include <vector>

using namespace std;

string solution(int n) {
    string answer = "";

    for(int i=0; i<n; i++){
        if(i%2 == 0)
            answer += "수";
        else
            answer += "박";
    }

    return answer;
}
```

```
class Solution {
public:
    String solution(int n) {
        String answer = "";

        for (int i = 0; i < n; i++) {
            if(i%2 == 0) {
                answer += "수";
            }

            if(i%2 == 1) {
                answer += "박";
            }
        }

        return answer;
    }
}
```

문제 9, 소수 찾기

1부터 입력받은 숫자 n 사이에 있는 소수의 개수를 반환하는 함수, solution을 만들어 보세요.

소수는 1과 자기 자신으로만 나누어지는 수를 의미합니다.
(1은 소수가 아닙니다.)

제한 조건

- n은 2이상 1000000이하의 자연수입니다.

입출력 예

n	result
10	4
5	3

입출력 예 설명

입출력 예 #1

1부터 10 사이의 소수는 [2,3,5,7] 4개가 존재하므로 4를 반환

입출력 예 #2

1부터 5 사이의 소수는 [2,3,5] 3개가 존재하므로 3를 반환

문제 9, 소수 찾기

에라토스테네스의 체

```
def is_prime_num(n):
    for i in range(2, n):
        if n % i == 0:
            return False # i로 나누어 떨어지면 소수가 아니므로 False 리턴

    return True # False가 리턴되지 않고 for문을 빠져나왔다면 소수이므로 True 리턴
```

제곱근을 구하기 위해 math 라이브러리 импорт

```
import math
```

```
def is_prime_num(n):
    for i in range(2, int(math.sqrt(n))+1): # n의 제곱근을 정수화 시켜준 후 + 1
        if n % i == 0:
            return False

    return True
```

```
def is_prime_num(n):
    arr = [True] * (n + 1) # 특정 수가 지워졌는지 아닌지 확인하기 위한 배열
    arr[0] = False
    arr[1] = False

    for i in range(2, n + 1):
        if arr[i] == True: # 특정 수가 지워지지 않았다면 (소수여서)
            j = 2

            while (i * j) <= n:
                arr[i*j] = False # i의 배수의 값을 False로 지워준다.
                j += 1

    return arr
```

arr = is_prime_num(50) # 0 ~ 50중 소수를 구하기 위한 함수

```
for i in range(len(arr)):
    if arr[i] == True:
        print(i, end=' ')
```

```
import math
```

```
def is_prime_num(n):
    arr = [True] * (n + 1)
    arr[0] = False
    arr[1] = False

    for i in range(2, int(math.sqrt(n)+1)):
        if arr[i] == True:
            j = 2

            while (i * j) <= n:
                arr[i*j] = False
                j += 1

    return arr

arr = is_prime_num(50)

for i in range(len(arr)):
    if arr[i] == True:
        print(i, end=' ')
```

문제 9, 소수 찾기

```
#include <string>
#include <vector>

using namespace std;

int solution(int n) {
    int answer = 0;
    bool isPrime;
    for(int i = 2; i <= n; i++)
    {
        isPrime = true;
        for(int j = 2; j < i; j++)
        {
            if(i % j == 0)
            {
                isPrime = false;
                break;
            }
        }
        if(isPrime)
            answer++;
    }

    return answer;
}
```

$O(N^2)$ 의 시간복잡도 -> 시간초과

에라토스테네스의 체

```
#include <string>
#include <vector>

using namespace std;

int solution(int n) {
    int answer = 0;
    vector<bool> v(n+1, true);

    for(int i = 2; i <= n; i++)
    {
        if(v[i] == true)
        {
            for(int j = 2; j*i <= n; j++)
            {
                v[j*i] = false;
            }
            answer++;
        }
    }

    return answer;
}
```

문제 9, 소수 찾기

에라토스테네스의 체 이용.

```
class Solution {
    public int solution(int n) {
        int answer = 0;

        for(int i=2; i<=n; i++) {
            boolean flag = true;
            for(int j=2; j<i; j++) { //두 번째 방법에서는 j<i 부분을 j<Math.sqrt(i) 로 바꾼다.
                if(i%j==0) {
                    flag = false;
                    break;
                }
            }

            if(flag==true) answer++;
        }

        return answer;
    }
}
```

```
class Solution {
    public int solution(int n) {
        int answer = 0;

        int[] number = new int[n+1];

        //2부터 n까지의 수를 배열에 넣는다.
        for(int i=2; i<=n; i++) {
            number[i] = i;
        }

        //2부터 시작해서 그의 배수들을 0으로 만든다.
        //후에 0이면 넘어가고 아니면 그의 배수들을 다시 0으로 만든다.
        for(int i=2; i<=n; i++) {
            if(number[i]==0) continue;

            for(int j= 2*i; j<=n; j += i) {
                number[j] = 0;
            }
        }

        //배열에서 0이 아닌 것들의 개수를 세준다.
        for(int i=0; i<number.length; i++) {
            if(number[i]!=0) {
                answer++;
            }
        }

        return answer;
    }
}
```

<http://colorscrippter.com/info#e> target="_blank" style="color: #4f4f4f; text-decoration: none;">

문제 10, 모의고사

```
def solution(answers):
    answer = [0 for i in range(3)]

    man1 = [1,2,3,4,5]
    man2 = [2,1,2,3,2,4,2,5]
    man3 = [3,3,1,1,2,2,4,4,5,5]

    for i in range(len(answers)):
        ans = answers[i]
        if(man1[i%len(man1)] == ans):
            answer[0] += 1
        if(man2[i%len(man2)] == ans):
            answer[1] += 1
        if(man3[i%len(man3)] == ans):
            answer[2] += 1

    result = []
    for i in range(len(answer)):
        if(answer[i] == max(answer)):
            result.append(i+1)

    return sorted(result)
```

```
def solution(answers):
    pattern1 = [1,2,3,4,5]
    pattern2 = [2,1,2,3,2,4,2,5]
    pattern3 = [3,3,1,1,2,2,4,4,5,5]
    score = [0,0,0]
    result = []

    for idx, answer in enumerate(answers):
        if answer == pattern1[idx%len(pattern1)]:
            score[0] += 1
        if answer == pattern2[idx%len(pattern2)]:
            score[1] += 1
        if answer == pattern3[idx%len(pattern3)]:
            score[2] += 1
    for idx, s in enumerate(score):
        if s == max(score):
            result.append(idx+1)
    return result
```

문제 10, 모의고사

```
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

int c1[5] = {1,2,3,4,5};
int c2[8] = {2,1,2,3,2,4,2,5};
int c3[10] = {3,3,1,1,2,2,4,4,5,5};

vector<int> solution(vector<int> answers) {
    vector<int> answer;
    vector<int> score;
    int cnt1 = 0;
    int cnt2 = 0;
    int cnt3 = 0;
    int max = 0;
    for(int i = 0; i < answers.size(); i++){
        int x1 = i%5;
        int x2 = i%8;
        int x3 = i%10;
        if(c1[x1] == answers[i]) cnt1++;
        if(c2[x2] == answers[i]) cnt2++;
        if(c3[x3] == answers[i]) cnt3++;
        if(max <= cnt1) max = cnt1;
        if(max <= cnt2) max = cnt2;
        if(max <= cnt3) max = cnt3;
    }

    score.push_back(cnt1);
    score.push_back(cnt2);
    score.push_back(cnt3);

    for(int i = 0; i < 3; i++){
        if(score[i] == max){
            answer.push_back(i+1);
        }
    }

    sort(answer.begin(), answer.end());
    return answer;
}
```

```
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

vector<int> one = {1,2,3,4,5};
vector<int> two = {2,1,2,3,2,4,2,5};
vector<int> thr = {3,3,1,1,2,2,4,4,5,5};

vector<int> solution(vector<int> answers) {
    vector<int> answer;
    vector<int> they(3);
    for(int i=0; i<answers.size(); i++) {
        if(answers[i] == one[i%one.size()]) they[0]++;
        if(answers[i] == two[i%two.size()]) they[1]++;
        if(answers[i] == thr[i%thr.size()]) they[2]++;
    }
    int they_max = *max_element(they.begin(), they.end());
    for(int i = 0; i < 3; i++) {
        if(they[i] == they_max) answer.push_back(i+1);
    }
    return answer;
}
```

문제 10, 모의고사

```
import java.util.ArrayList;
class Solution {
    public int[] solution(int[] answers) {
        int[] answer = {};
        int[] person1 = {1,2,3,4,5}; //0이만큼씩 반복
        int[] person2 = {2,1,2,3,2,4,2,5};
        int[] person3 = {3,3,1,1,2,2,4,4,5,5};
        int answer1=0, answer2 =0, answer3 =0;

        for(int i =0; i<answers.length; i++){
            if(person1[i%person1.length] == answers[i]) answer1++;
            if(person2[i%person2.length] == answers[i]) answer2++;
            if(person3[i%person3.length] == answers[i]) answer3++;
        }
        int max = Math.max(Math.max(answer1, answer2),answer3); // max값 구하기
        ArrayList<Integer> list = new ArrayList<Integer>();
        if(max==answer1) list.add(1); //max값이랑 같으면 넣는다.
        if(max==answer2) list.add(2);
        if(max==answer3) list.add(3);

        answer = new int[list.size()];

        for(int i =0; i<answer.length; i++) {
            answer[i] = list.get(i);
        }

        return answer;
    }
}
```

```
class Solution {
    public int[] solution(int[] answer) {
        int[] a = {1, 2, 3, 4, 5};
        int[] b = {2, 1, 2, 3, 2, 4, 2, 5};
        int[] c = {3, 3, 1, 1, 2, 2, 4, 4, 5, 5};
        int[] score = new int[3];
        for(int i=0; i<answer.length; i++) {
            if(answer[i] == a[i%a.length]) {score[0]++;}
            if(answer[i] == b[i%b.length]) {score[1]++;}
            if(answer[i] == c[i%c.length]) {score[2]++;}
        }
        int maxScore = Math.max(score[0], Math.max(score[1], score[2]));
        ArrayList<Integer> list = new ArrayList<>();
        if(maxScore == score[0]) {list.add(1);}
        if(maxScore == score[1]) {list.add(2);}
        if(maxScore == score[2]) {list.add(3);}
        return list.stream().mapToInt(i->i.intValue()).toArray();
    }
}
```

문제 11, 베스킨 라빈스 31 게임

```
import random

print("베스킨라빈스 31 게임 프로그램입니다!")

order = input('순서를 입력하세요. (선공 1, 후공 0 입력) : ')
order = int(order)

call = 0
count = 1

while call < 31:
    if count % 2 == order:
        # 사용자의 차례
        print('사용자의 차례')
        size_of_call = input("호출할 개수를 입력하세요 : ")
        size_of_call = int(size_of_call)

        for _ in range(size_of_call):
            call += 1
            print("사용자 : '{0}'!!!".format(call))

    else:
        # 컴퓨터의 차례
        print('컴퓨터의 차례')
        size_of_call = random.randint(1, 3)

        for _ in range(size_of_call):
            call += 1
            print("컴퓨터 : '{0}'!!!".format(call))

    count += 1

if count % 2 == order:
    print("사용자의 승리!!")
else:
    print("컴퓨터의 승리!!")
```

문제 11, 베스킨 라빈스 31 게임

- 1) 베스킨라빈스 31 게임 필승 전략 : $[2, 6, 10, 14, 18, 22, 26, 30]$ $4n+2$ 수열의 숫자들을 외치는 쪽이 이기게 된다. 따라서, 사실상 먼저 숫자 2를 외치는 쪽이 이기게 된다. 그리고 이것은 반대로 위의 필승 알고리즘으로 만든다해도 유일하게 컴퓨터가 패할 수 있는 경우의 수이다.

cf. 필승의 해당 수열이 나온 논리에 대해 설명해보자면 내가 30을 외치면 승리! 이므로 내가 마지막 숫자로 30을 외치기 위해선 상대방이 29, 28, 27 중 하나를 마지막 숫자로 외쳐주면 된다. 따라서, 상대방이 외쳐주길 원하는 숫자 3개를 빼면 26이 나오게 되고, 내가 26을 외치면 다음에 다시 내 차례가 될때 30을 외칠 수 있게된다. 이런식으로 생각해보면 필승 숫자 30부터 4씩 감소한다는 것을 알 수 있으며 따라서, 처음 외치면 무조건 이길 수 있게 되는 숫자는 2가 된다. 즉, 필승 숫자 30을 4로 나눈 나머지 값인 2가 필승 숫자를 외칠수 있게 하는 씨앗이된다.

```
# 컴퓨터가 몇을 호출할 것인지 호출할 숫자의 범위를 결정하는 것
# 컴퓨터가 랜덤하게 호출하는 기존의 함수에서 이 if~else 구문만 추가됨
#  $4n+2$  수열의 숫자를 불러나가는 쪽이 승리하게되며, 따라서 사실상 먼저 2를 외치는 쪽이 승리할 수 있음
#  $4n$   $n$ 을 4로 나누면 나머지는 0~3이 나올 수 있는데 나누었을 때 나머지 2가 나올 수 있는 숫자는 2를
# 전부 필승 수열의 숫자뿐이다. 따라서, 숫자 2를 먼저 외치게 되는 쪽이 승리할 수 있게 된다.
# 그러므로  $last\_num \% 4 == 2$ : 인  $last\_num$ 은 의미가 없다. 1~3 중 몇을 더해도 지기 때문.
if last_num % 4 == 0:
    call_num_size = 2 # 처음 시작시 2를 먼저 외침
elif last_num % 4 == 1:
    call_num_size = 1 # 상대가 1을 외치면 1을 추가하여 2를 먼저 외침
elif last_num % 4 == 3:
    call_num_size = 3 # 상대가 3을 외치면  $4n+2$  수열의  $n=1$ 을 넣은 값인 6을 외치도록 3을 추가함
else:
    call_num_size = random.randint(1, 3)

computer_call = [last_num+i for i in range(1, call_num_size+1)]
```


문제 11, 베스킨 라빈스 31 게임

- 2) **턴방식 진행** : 교차진행이랄까? 상대턴 -> 내턴 -> 상대턴 이런식으로 번갈아가면서 진행하는 방식을 코드로 구현해보았다.

```
if start_player == 'computer':
    turn = 0 # 역할에 가깝다( 2로 나눈 나머지 값이 0인 숫자들 아무거나 상관없다. 짝수! 단, 그 숫
    print('-- 각오해라 휴우먼 --')
else:
    turn = 1
    print('-- 기계제국으로 돌아가시지 --')

# 첫 시작시 last_num 값을 0으로 초기화 후 진입
last_num = 0

# 게임의 본체
while last_num < 31:
    if turn % 2 == 0: # 2로 나눈 나머지 값이 0인 즉, 0(첫번째턴), 2(세번째턴), 4(다섯번째턴) (
        # 두 함수 모두 return 값은 last_num이다.
        # 컴퓨터가 숫자를 호출하는 함수
        last_num = computer_call_number(last_num)
    else:
        # 플레이어가 숫자를 호출하는 함수
        last_num = player_call_number(last_num)

    turn += 1
```

문제 11, 베스킨 라빈스 31 게임

- 3) **리스트 컴프리헨션** : 조건 중 마지막에 꼭 31까지 외치도록 하는 조건이 있었다. 따라서, 32 혹은 33을 외치지 않도록 하기 위해 리스트 컴프리헨션을 사용하여 32, 33을 리스트에서 코드 한줄로 제거해준다. (숫자 호출도 리스트로 나란히 나오도록 하기에 이렇게 작성한 것
ex. [1,2] -> [3] -> [4,5,6]))

cf. 리스트에서 한번에 여러 요소를 제거해 주는 함수는 numpy 패키지의 함수밖에 없다.
따라서, numpy 패키지를 import 하지 않는 이상 이와 같은 방법이 리스트에서 한번에 여러 요소를 지우기에 적합하다고 생각한다.

```
# 상대로 부터 받은 마지막 숫자(last_num)에 더해줄 숫자값(i)의 범위를
# range(1, 호출할 숫자의 갯수(call_num_size) + 1)를 사용하여 표현한다.
# 선공인 경우 숫자 0부터 증가가 시작되고, 후공인 경우 상대의 마지막 호출 숫자로부터 증가가 시작된다.
# 리스트 컴프리헨션 파이썬 문법을 통해 코드 간소화
call_list = [last_num+i for i in range(1, call_num_size+1)]

# 호출 리스트의 마지막 숫자값
last_num = call_list[-1]

if last_num >= 31:
    # 리스트 컴프리헨션을 사용하여 마지막 숫자값이 32, 33인 경우는 호출 리스트에서 제외한다.
    # 이유는 31을 마지막에 꼭 외치도록 하기 위해.
    # call_list에서 숫자 하나씩 빼서 item에 넣는데, 그 값이 32, 33이 아닌값만 넣겠다.
    # 그리고 그 item들로 list를 만들겠다.
    call_list = [item for item in call_list if item != 32 and item != 33]
    print(f'call_list : {call_list}')
```

문제 11, 베스킨 라빈스 31 게임

```
int Answer = 0; //기존 숫자

while (true)
{
    Console.WriteLine("증가 시킬 숫자를 입력해주세요[1~3] : ");
    string Count = Console.ReadLine();
    int Plus;
    if (int.TryParse(Count, out Plus))
    {
        if (Plus <= 3 && Plus > 0)
        {
            Answer += Plus;
        }
        else
        {
            Console.WriteLine("숫자는 최대 3까지만 증가시킬수 있습니다.");
            continue;
        }
    }
    else
    {
        Console.WriteLine("숫자 형식을 입력해주세요.");
        continue;
    }

    Console.WriteLine("[Me] 현재 베스킨 라빈스 숫자 : " + Answer);

    //컴퓨터가 입력한 숫자

    for (int i = 1; i <= 8; i++)
    {
        int Temp = 4 * i - 2;
        if (Answer + 1 == Temp || Answer + 2 == Temp || Answer + 3 == Temp)
        {
            Answer = Temp;
        }
    }

    Console.WriteLine("[컴퓨터] 현재 베스킨 라빈스 숫자 : " + Answer);
```

```
if (Answer == 30)
{
    Console.WriteLine("컴퓨터가 이겼습니다.");
    break;
}

Console.ReadKey();
```

```
public class MyClass {
    public static void main(String []args){
        int currentNumber = 0;
        int userCount = 0;

        Scanner s = new Scanner(System.in);

        System.out.println("게임을 시작하겠습니다.");
        System.out.println();
        System.out.println("CPU >> 1 2");
        currentNumber = 2;

        while(currentNumber < 31) {
            System.out.print("몇 개의 수를 부르시겠습니까? >>");
            while(true) {
                userCount = s.nextInt();
                if (userCount <= 3 && userCount >= 1) {
                    System.out.println();
                    break;
                }
                else {
                    System.out.println();
                    System.out.print("다시 입력하십시오>>");
                }
            }

            System.out.print("User >> ");
            for(int n = 0; n < userCount; n++) {
                ++currentNumber;
                System.out.print(currentNumber + " ");
            }
            System.out.println();

            System.out.print("CPU >> ");
            for(int i = 0; i < (4-userCount); i++) {
                ++currentNumber;
                if(currentNumber < 31)
                    System.out.print(currentNumber + " ");
            }
            System.out.println();

            userCount = 0;

        }
        s.close();
    }
}
```

문제 12, 같은 숫자는 싫어

```
def solution(arr):
    answer = []

    for i in range(len(arr)):
        if i != len(arr)-1 and arr[i] is not arr[i+1]:
            answer.append(arr[i])
        elif i == len(arr)-2 and arr[i] is arr[i+1]:
            answer.append(arr[i])
        elif i == len(arr)-1 and arr[i] is not arr[i-1]:
            answer.append(arr[i])

    return answer
```

```
def solution(arr):
    answer = []

    for i in range(len(arr)):
        if i == 0:
            answer.append(int(arr[i]))
        elif arr[i] != arr[i-1]:
            answer.append(int(arr[i]))

    return answer
```

문제 12, 같은 숫자는 싫어

```
#include <vector>
#include <iostream>

using namespace std;

vector<int> solution(vector<int> arr)
{
    vector<int> answer;
    answer.push_back(arr[0]);
    for(int i=1;i<arr.size();i++){
        if(arr[i-1]!=arr[i])    answer.push_back(arr[i]);
    }
    return answer;
}
```

Colored by Color Scripter

```
import java.util.*;

public class Solution {
    public int[] solution(int []arr) {
        ArrayList<Integer> answerList = new ArrayList<Integer>();
        //
        int value = -1;
        for(int i=0; i<arr.length; i++) {
            if(arr[i] != value) {
                answerList.add(arr[i]);
                value = arr[i];
            }
        }
        return answerList.stream().mapToInt(i->i).toArray();
    }
}
```

문제 8, 소수 찾기 DFS와 BFS로 찾아보자.

1부터 입력받은 숫자 n 사이에 있는 소수의 개수를 반환하는 함수, solution을 만들어 보세요.

소수는 1과 자기 자신으로만 나누어지는 수를 의미합니다.
(1은 소수가 아닙니다.)

제한 조건

- n은 2이상 1000000이하의 자연수입니다.

입출력 예

n	result
10	4
5	3

입출력 예 설명

입출력 예 #1

1부터 10 사이의 소수는 [2,3,5,7] 4개가 존재하므로 4를 반환

입출력 예 #2

1부터 5 사이의 소수는 [2,3,5] 3개가 존재하므로 3를 반환

문제 8, 소수 찾기 DFS로 찾아보자.

DFS

```
from itertools import permutations
import math

def solution(numbers):
    length = len(numbers)
    answer = 0
    numberList = []
    for i in range(1, length+1):
        for temp in permutations(numbers, i):
            numberList.append(int("".join(temp)))

    numberList = sorted(set(numberList))
    for num in numberList:
        isValue = True
        for i in range(2, int(math.sqrt(num))+1):
            if num % i == 0:
                isValue = False
                break
        if isValue and (num != 0 and num != 1):
            answer += 1

    return answer
```

```
#include <string>
#include <vector>
#include <iostream>
#include <algorithm>

using namespace std;
bool prime[10000000] = {false,};

int solution(string numbers) {
    int answer = 0;
    int max = 0;
    int tmp = 0;
    sort(numbers.begin(), numbers.end(), greater<int>());
    max = stoi(numbers);

    for(int i = 2; i <= max; i++)
    {
        for(int j = i + i; j <= max; j += i)
            prime[j] = true;
    }
    sort(numbers.begin(), numbers.end());

    do
    {
        string str = "";
        tmp = 0;
        for(int j = 0; j < numbers.size(); j++)
        {
            str += numbers[j];
            tmp = stoi(str);
            if(tmp > 1 && prime[tmp] == false)
            {
                answer++;
                prime[tmp] = true;
            }
        }
    }while(next_permutation(numbers.begin(), numbers.end()));

    return answer;
}
```

```
import java.util.*;

class Solution {
    int answer;
    boolean[] check;
    ArrayList<Integer> arr = new ArrayList<Integer>();

    public int solution(String numbers) {
        String tmp = "";
        check = new boolean[numbers.length()];

        for (int i = 1; i <= numbers.length(); i++) {
            dfs(numbers, tmp, i);
        }

        for (int i = 0; i < arr.size(); i++) {
            isPrime(arr.get(i));
        }

        return answer;
    }

    void dfs(String str, String tmp, int dept) {
        for (int i = 0; i < str.length(); i++) {
            if (tmp.length() == dept) {
                if (!arr.contains(Integer.parseInt(tmp)))
                    arr.add(Integer.parseInt(tmp));
                return;
            } else {
                if (!check[i]) {
                    check[i] = true;
                    tmp += str.charAt(i);

                    dfs(str, tmp, dept);

                    check[i] = false;
                    tmp = tmp.substring(0, tmp.length() - 1);
                }
            }
        }
    }

    void isPrime(int x) {
        if (x == 0) return;
        if (x == 1) return;
        for (int i = 2; i < x; i++) {
            if (x % i == 0) return;
        }
        answer++;
    }
}
```

문제 13, DFS와 BFS.

문제

그래프를 DFS로 탐색한 결과와 BFS로 탐색한 결과를 출력하는 프로그램을 작성하시오. 단, 방문할 수 있는 정점이 여러 개인 경우에는 정점 번호가 작은 것을 먼저 방문하고, 더 이상 방문할 수 있는 점이 없는 경우 종료한다. 정점 번호는 1번부터 N번까지이다.

입력

첫째 줄에 정점의 개수 N ($1 \leq N \leq 1,000$), 간선의 개수 M ($1 \leq M \leq 10,000$), 탐색을 시작할 정점의 번호 V 가 주어진다. 다음 M 개의 줄에는 간선이 연결하는 두 정점의 번호가 주어진다. 어떤 두 정점 사이에 여러 개의 간선이 있을 수 있다. 입력으로 주어지는 간선은 양방향이다.

출력

첫째 줄에 DFS를 수행한 결과를, 그 다음 줄에는 BFS를 수행한 결과를 출력한다. V 부터 방문된 점을 순서대로 출력하면 된다.

예제 입력 1 [복사](#)

```
4 5 1
1 2
1 3
1 4
2 4
3 4
```

예제 출력 1 [복사](#)

```
1 2 4 3
1 2 3 4
```


문제 13, DFS와 BFS.

```
import sys
from collections import deque
input=sys.stdin.readline

n,m,start=map(int,input().split())
visited=[False]*(n+1)

graph=[[[] for _ in range(n+1)]

for _ in range(m):
    a,b=map(int,input().split())
    graph[a].append(b)
    graph[b].append(a)

for i in range(len(graph)):
    graph[i].sort()

def dfs(start):
    print(start,end=' ')
    visited[start]=True
    for i in graph[start]:
        if not visited[i]:
            dfs(i)
    visited[i]=True

def bfs(start):
    q=deque([start])
    visited[start]=True
    while q:
        now=q.popleft()
        print(now,end=' ')
        for i in graph[now]:
            if not visited[i]:
                q.append(i)
                visited[i]=True

dfs(start)
visited=[False]*(n+1)
print()
bfs(start)
```

```
from collections import deque
```

```
N, M, V = map(int, input().split())
```

```
graph = [[0] * (N + 1) for _ in range(N + 1)]
```

```
for _ in range(M):
```

```
    m1, m2 = map(int, input().split())
```

```
    # 노드 연결 하기
```

```
    graph[m1][m2] = graph[m2][m1] = 1
```

```
# 너비 우선 탐색
```

```
def bfs(start_v):
```

```
    discovered = [start_v]
```

```
    # 리스트를 써서 pop(0)하게 되면 시간복잡도가 O(N)이다
```

```
    # 그래서 시간복잡도가 O(1)인 deque를 사용한다.
```

```
    queue = deque()
```

```
    queue.append(start_v)
```

```
while queue:
```

```
    v = queue.popleft()
```

```
    print(v, end=' ')
```

```
    for w in range(len(graph[start_v])):
```

```
        if graph[v][w] == 1 and (w not in discovered):
```

```
            discovered.append(w)
```

```
            queue.append(w)
```

```
# 깊이 우선 탐색
```

```
def dfs(start_v, discovered=[]):
```

```
    discovered.append(start_v)
```

```
    print(start_v, end=' ')
```

```
    for w in range(len(graph[start_v])):
```

```
        if graph[start_v][w] == 1 and (w not in discovered):
```

```
            dfs(w, discovered)
```

```
dfs(V)
```

```
print()
```

```
bfs(V)
```

문제 13, DFS와 BFS.

```
#include <stdio.h>
#define MAX_VERTICES 1001
int DFS_V[MAX_VERTICES] = { 0, }; //DFS를 실행하면서 방문한 정점을 표시하기 위한 배열
int BFS_V[MAX_VERTICES] = { 0, }; //BFS를 실행하면서 방문한 정점을 표시하기 위한
int graph[MAX_VERTICES][MAX_VERTICES] = { 0, };
int queue[MAX_VERTICES];
void dfs(int v, int vertices);
void bfs(int v, int vertices);

int main() {
    int vertices, edges, vertex, i, j;
    scanf("%d %d %d", &vertices, &edges, &vertex);

    while (edges--) {
        scanf("%d %d", &i, &j);
        graph[i][j] = 1;
        graph[j][i] = 1;
    }

    dfs(vertex, vertices);
    printf("\n");
    bfs(vertex, vertices);

    return 0;
}
```

```
void dfs(int v, int vertices) {
    int w;
    DFS_V[v] = 1;
    printf("%d ", v);
    for (w = 1; w <= vertices; w++) {
        if (graph[v][w] == 1 && DFS_V[w] == 0) {
            dfs(w, vertices);
        }
    }
}

void bfs(int v, int vertices) {
    int w;
    int front, rear, pop;
    front = rear = 0;
    printf("%d ", v);
    BFS_V[v] = 1;
    queue[0] = v; rear++;
    while (front < rear) {
        pop = queue[front]; front++;
        for (w = 1; w <= vertices; w++) {
            if (graph[pop][w] == 1 && BFS_V[w] == 0) {
                printf("%d ", w);
                queue[rear] = w; rear++;
                BFS_V[w] = 1;
            }
        }
    }
}
```

문제 13, DFS와 BFS.

```
import java.util.*;

public class Main {
    static ArrayList<Integer>[] a;
    static boolean[] c;
    public static void dfs(int x) {
        if (c[x]) {
            return;
        }
        c[x] = true;
        System.out.print(x + " ");
        for (int y : a[x]) {
            if (c[y] == false) {
                dfs(y);
            }
        }
    }
    public static void bfs(int start) {
        Queue<Integer> q = new LinkedList<Integer>();
        q.add(start);
        c[start] = true;
        while (!q.isEmpty()) {
            int x = q.remove();
            System.out.print(x + " ");
            for (int y : a[x]) {
                if (c[y] == false) {
                    c[y] = true;
                    q.add(y);
                }
            }
        }
    }
}
```

```
public static void main(String args[]) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    int m = sc.nextInt();
    int start = sc.nextInt();
    a = (ArrayList<Integer>[]) new ArrayList[n+1];
    for (int i=1; i<=n; i++) {
        a[i] = new ArrayList<Integer>();
    }
    for (int i=0; i<m; i++) {
        int u = sc.nextInt();
        int v = sc.nextInt();
        a[u].add(v);
        a[v].add(u);
    }
    for (int i=1; i<=n; i++) {
        Collections.sort(a[i]);
    }
    c = new boolean[n+1];
    dfs(start);
    System.out.println();
    c = new boolean[n+1];
    bfs(start);
    System.out.println();
}
```

문제 14, 정렬 문제

단계	문제 번호	제목	정보	정답	제출	정답 비율
1	2750	수 정렬하기		74502	130735	58.337%
시간 복잡도가 $O(n^2)$ 인 정렬 알고리즘으로 풀 수 있습니다. 예를 들면 삽입 정렬, 거품 정렬 등이 있습니다.						
2	2751	수 정렬하기 2		56058	197840	30.353%
시간 복잡도가 $O(n \log n)$ 인 정렬 알고리즘으로 풀 수 있습니다. 예를 들면 병합 정렬, 힙 정렬 등이 있지만, 어려운 알고리즘이므로 지금은 언어에 내장된 정렬 함수를 쓰는 것을 추천드립니다.						
3	10989	수 정렬하기 3		40777	176535	23.426%
수의 범위가 작다면 카운팅 정렬을 사용하여 더욱 빠르게 정렬할 수 있습니다.						
4	2108	통계학		20601	98809	25.421%
정렬을 활용하는 문제						
5	1427	소트인사이드		36614	57639	63.884%
숫자를 정렬하는 문제						
6	11650	좌표 정렬하기		38354	81199	48.151%
좌표를 정렬하는 문제						
7	11651	좌표 정렬하기 2		28739	44337	67.521%
좌표를 다른 순서로 정렬하는 문제						
8	1181	단어 정렬		41784	100543	40.291%
단어의 순서를 정의하여 정렬하는 문제						
9	10814	나이순 정렬		35853	81976	42.618%
값이 같은 원소의 전후관계가 바뀌지 않는 정렬 알고리즘을 안정 정렬(stable sort)이라고 합니다.						
10	18870	좌표 압축		16449	38516	40.599%
만약 정확한 값이 필요 없고 값의 대소 관계만 필요하다면, 모든 수를 0 이상 N 미만의 수로 바꿀 수 있습니다.						

문제 15, K번째 수 / 정렬

배열 array의 i번째 숫자부터 j번째 숫자까지 자르고 정렬했을 때, k번째에 있는 수를 구하려 합니다.

예를 들어 array가 [1, 5, 2, 6, 3, 7, 4], i = 2, j = 5, k = 3이라면

- array의 2번째부터 5번째까지 자르면 [5, 2, 6, 3]입니다.
- 1에서 나온 배열을 정렬하면 [2, 3, 5, 6]입니다.
- 2에서 나온 배열의 3번째 숫자는 5입니다.

배열 array, [i, j, k]를 원소로 가진 2차원 배열 commands가 매개변수로 주어질 때, commands의 모든 원소에 대해 앞서 설명한 연산을 적용했을 때 나온 결과를 배열에 담아 return 하도록 solution 함수를 작성해주세요.

제한사항

- array의 길이는 1 이상 100 이하입니다.
- array의 각 원소는 1 이상 100 이하입니다.
- commands의 길이는 1 이상 50 이하입니다.
- commands의 각 원소는 길이가 3입니다.

입출력 예

array	commands	return
[1, 5, 2, 6, 3, 7, 4]	[[2, 5, 3], [4, 4, 1], [1, 7, 3]]	[5, 6, 3]

입출력 예 설명

[1, 5, 2, 6, 3, 7, 4]를 2번째부터 5번째까지 자른 후 정렬합니다. [2, 3, 5, 6]의 세 번째 숫자는 5입니다.

[1, 5, 2, 6, 3, 7, 4]를 4번째부터 4번째까지 자른 후 정렬합니다. [6]의 첫 번째 숫자는 6입니다.

[1, 5, 2, 6, 3, 7, 4]를 1번째부터 7번째까지 자릅니다. [1, 2, 3, 4, 5, 6, 7]의 세 번째 숫자는 3입니다.

문제 15, K번째 수 / 정렬

```
def solution(array, commands):
    answer = []

    for i in commands:
        ary = array[i[0]-1: i[1]]    # 문제에서 주어진 크기만큼 자르기
        ary.sort()                  # sort 함수로 정렬
        answer.append(ary[i[2]-1])   # k 번째 값 집어넣기

    return answer
```

```
def solution(array, commands):
    return list(map(lambda x: sorted(array[x[0]-1:x[1]])[x[2]-1], commands))
```

문제 15, K번째 수 / 정렬

```
#include <vector>
#include <algorithm>
using namespace std;

vector<int> solution(vector<int> array1, vector<vector<int>> commands) {
    vector<int> answer;
    //반환되어야하는 세트동안 반복
    for (int i = 0; i < commands.size(); i++){
        //임시 벡터 생성
        vector<int> temp;
        //필요한 부분만 임시벡터에 넣음
        for (int j = commands[i][0] - 1; j < commands[i][1]; j++)
            temp.push_back(array1[j]);
        //임시 벡터정렬
        sort(temp.begin(), temp.end());
        //임시벡터의 index번째를 answer에 넣음
        answer.push_back(temp.at(commands[i][2] - 1));
    }
    return answer;
}
```

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

class Solution {
    public int[] solution(int[] array, int[][] commands) {
        int[] answer = {};
        answer = new int[commands.length];
        for(int i = 0; i < commands.length; i++) {
            List<Integer> arrInt = new ArrayList<Integer>();
            for(int j = commands[i][0]-1; j < commands[i][1]; j
            ++){
                arrInt.add(array[j]);
            }
            Collections.sort(arrInt);
            answer[i] = arrInt.get(commands[i][2]-1);
        }
        return answer;
    }
}
```

```
import java.util.Arrays;
class Solution {
    public int[] solution(int[] array, int[][] commands) {
        int[] answer = new int[commands.length];

        for(int i=0; i<commands.length; i++){
            int[] temp = Arrays.copyOfRange(array, commands[i][0]-1, commands[i][1]);
            Arrays.sort(temp);
            answer[i] = temp[commands[i][2]-1];
        }

        return answer;
    }
}
```

문제 16, 리트코드 771. Jewels and Stones

771. Jewels and Stones

Easy  3699  496  Add to List  Share

You're given strings `jewels` representing the types of stones that are jewels, and `stones` representing the stones you have. Each character in `stones` is a type of stone you have. You want to know how many of the stones you have are also jewels.

Letters are case sensitive, so `"a"` is considered a different type of stone from `"A"`.

Example 1:

```
Input: jewels = "aA", stones = "aAAbbbb"
Output: 3
```

Example 2:

```
Input: jewels = "z", stones = "ZZ"
Output: 0
```

Constraints:

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

돌을 나타내는 S와 보석을 나타내는 J가 주어졌을 때, 돌 안에 보석이 몇 개 들어있는지 알려주기

문제 16, 리트코드 771. Jewels and Stones

```
class Solution:
    def numJewelsInStones(self, J: str, S: str) -> int:
        freq = {}
        count = 0

        for char in S:
            if not freq[char]:
                freq[char] = 1
            else:
                freq[char] += 1

        for char in J:
            if freq[char]:
                count += freq[char]

        return count
```

```
class Solution:
    def numJewelsInStones(self, J: str, S: str) -> int:
        freq = collections.defaultdict()
        count = 0

        for char in S:
            freq[char] += 1

        for char in J:
            if char in freq:
                count += freq[char]

        return count
```

```
class Solution:
    def numJewelsInStones(self, J: str, S: str) -> int:
        freq = collections.Counter(S)
        count = 0

        for char in J:
            count += freqs[char]

        return count
```

```
class Solution:
    def numJewelsInStones(self, J: str, S: str) -> int:
        return sum(s in J for s in S)
```

문제 17, 완주하지 못한 선수

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 명 밖에 없기 때문에 한명은 완주하지 못했습니다.

문제 17, 완주하지 못한 선수

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

```
import collections
def solution(participant, completion):
    # 1. participant의 Counter를 구한다
    # 2. completion의 Counter를 구한다
    # 3. 둘의 차를 구하면 정답만 남아있는 counter를 반환한다
    answer = collections.Counter(participant) - collections.Counter(completion)

    # 4. counter의 key값을 반환한다
    return list(answer.keys())[0]

print(solution(["marina", "josipa", "nikola", "vinko", "filipa"],
               ["josipa", "filipa", "marina", "nikola"]))
```

```
#include <iostream>
#include <string>
#include <vector>
#include <set>

using namespace std;

string solution(vector<string> participant, vector<string> completion)
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);

    string answer = "";

    multiset<string> s1;
    multiset<string> s2;

    for (int i = 0; i < participant.size(); i++)
        s1.insert(participant[i]);

    for (int i = 0; i < completion.size(); i++)
        s2.insert(completion[i]);

    for (int i = 0; i < participant.size(); i++)
    {
        if (s1.count(participant[i]) != s2.count(participant[i]))
            answer = participant[i];
    }

    return answer;
}
```

```
import java.util.HashMap;

class Solution {
    public String solution(String[] participant, String[] completion) {
        String answer = "";
        HashMap<String, Integer> hm = new HashMap<>();
        for(String player : participant) hm.put(player, hm.getOrDefault(player, 0) + 1);
        for(String player : completion) hm.put(player, hm.get(player) - 1);
        for(String key : hm.keySet()) {
            if(hm.get(key) != 0) {
                answer = key;
                System.out.println(answer);
                break;
            }
        }
        return answer;
    }
}
```