1. 모의고사

문제 설명
수포자는 수학을 포기한 사람의 준말입니다. 수포자 삼인방은 모의고사에 수학 문제를 전부 찍으려 합니다. 수포자는 1 번 문제부터 마지막 문제까지 다음과 같이 찍습니다.
1 번 수포자가 찍는 방식: 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, ...
2 번 수포자가 찍는 방식: 2, 1, 2, 3, 2, 4, 2, 5, 2, 1, 2, 3, 2, 4, 2, 5, ...
3 번 수포자가 찍는 방식: 3, 3, 1, 1, 2, 2, 4, 4, 5, 5, 3, 3, 1, 1, 2, 2, 4, 4, 5, 5, ...
1 번 문제부터 마지막 문제까지의 정답이 순서대로 들은 배열 answers 가 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, 5 중 하나입니다.
가장 높은 점수를 받은 사람이 여럿일 경우, return 하는 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 1 은 모든 문제를 맞혔습니다.
수포자 2 는 모든 문제를 틀렸습니다.
수포자 3 은 모든 문제를 틀렸습니다.
따라서 가장 문제를 많이 맞힌 사람은 수포자 1 입니다.
입출력 예 #2
모든 사람이 2 문제씩을 맞췄습니다.

```
import java.util.*;

class Solution {
    public int[] solution(int[] answers) {
        int[] answer = {};

        int first[] = {1,2,3,4,5}; //규칙 반복(이하 동일)
        int second[] = {2,1,2,3,2,4,2,5};
        int third[] = {3,3,1,1,2,2,4,4,5,5};

        int all[] = new int[3]; //3 명 비교

        for(int i = 0; i < answers.length; i++){ //정답과 비교하여
맞은 것이 있다면 ++
```

```
            if(answers[i] == first[i%first.length]){all[0]++;}
            if(answers[i] == second[i%second.length]){all[1]++;}
            if(answers[i] == third[i%third.length]){all[2]++;}
        }

        List<Integer> clearPerson = new ArrayList<Integer>(); //
List 만들어주고 최대값 비교
        int max = Math.max(Math.max(all[0], all[1]),all[2]);

        if(max == all[0]){clearPerson.add(1);}
        if(max == all[1]){clearPerson.add(2);}
        if(max == all[2]){clearPerson.add(3);}

        Collections.sort(clearPerson); //가장 높은 점수를 받은 사람이
 여러명일 경우 오름차순

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

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


        return answer;
    }
}
```

출처: https://wooaoe.tistory.com/42 [개발개발 울었다]

2. 소수 찾기

문제 설명
한자리 숫자가 적힌 종이 조각이 흩어져있습니다. 흩어진 종이 조각을 붙여 소수를 몇 개 만들 수 있는지 알아내려 합니다.
각 종이 조각에 적힌 숫자가 적힌 문자열 numbers 가 주어졌을 때, 종이 조각으로 만들 수 있는 소수가 몇 개인지 return 하도록 solution 함수를 완성해주세요.
제한사항
numbers 는 길이 1 이상 7 이하인 문자열입니다.
numbers 는 0~9 까지 숫자만으로 이루어져 있습니다.
013 은 0, 1, 3 숫자가 적힌 종이 조각이 흩어져있다는 의미입니다.
입출력 예

numbers	return
17	3
011	2

입출력 예 설명
예제 #1
[1, 7]으로는 소수 [7, 17, 71]를 만들 수 있습니다.
예제 #2
[0, 1, 1]으로는 소수 [11, 101]를 만들 수 있습니다.
11 과 011 은 같은 숫자로 취급합니다.

```
import java.util.*;

class Solution {
    static int answer = 0;
    static Set<Integer> set = new HashSet<>();

    public int solution(String numbers) {
        char[] arr = numbers.toCharArray();
        boolean[] select = new boolean[numbers.length()];
        char[] output = new char[numbers.length()];

        for (int i = 1; i <= numbers.length(); i++) {
            perm(arr, select, output, 0, numbers.length(), i);
        }

        return answer;
    }

    // 순열
    public static void perm(char[] arr, boolean[] select, char[]
 output, int num, int size, int r) {
        if (num == r) {
            if (isPrime(output, r)) {
                answer++;
            }
```

```
            return;
        }

        for (int i = 0; i < size; i++) {
            if (select[i]) {
                continue;
            }
            select[i] = true;
            output[num] = arr[i];
            perm(arr, select, output, num + 1, size, r);
            select[i] = false;
        }
    }

    public static boolean isPrime(char[] arr, int size) {
        if (arr[0] == '0') {
            return false;
        }

        String str = "";
        for (int i = 0; i < size; i++) {
            str += arr[i];
        }

        int temp = Integer.parseInt(str);

        if (temp <= 1 || set.contains(temp)) {
            return false;
        }

        for (int i = 2; i * i <= temp; i++) {
            if (temp % i == 0) {
                return false;
            }
        }

        set.add(temp);
        return true;
    }
}
```

https://steady-coding.tistory.com/70

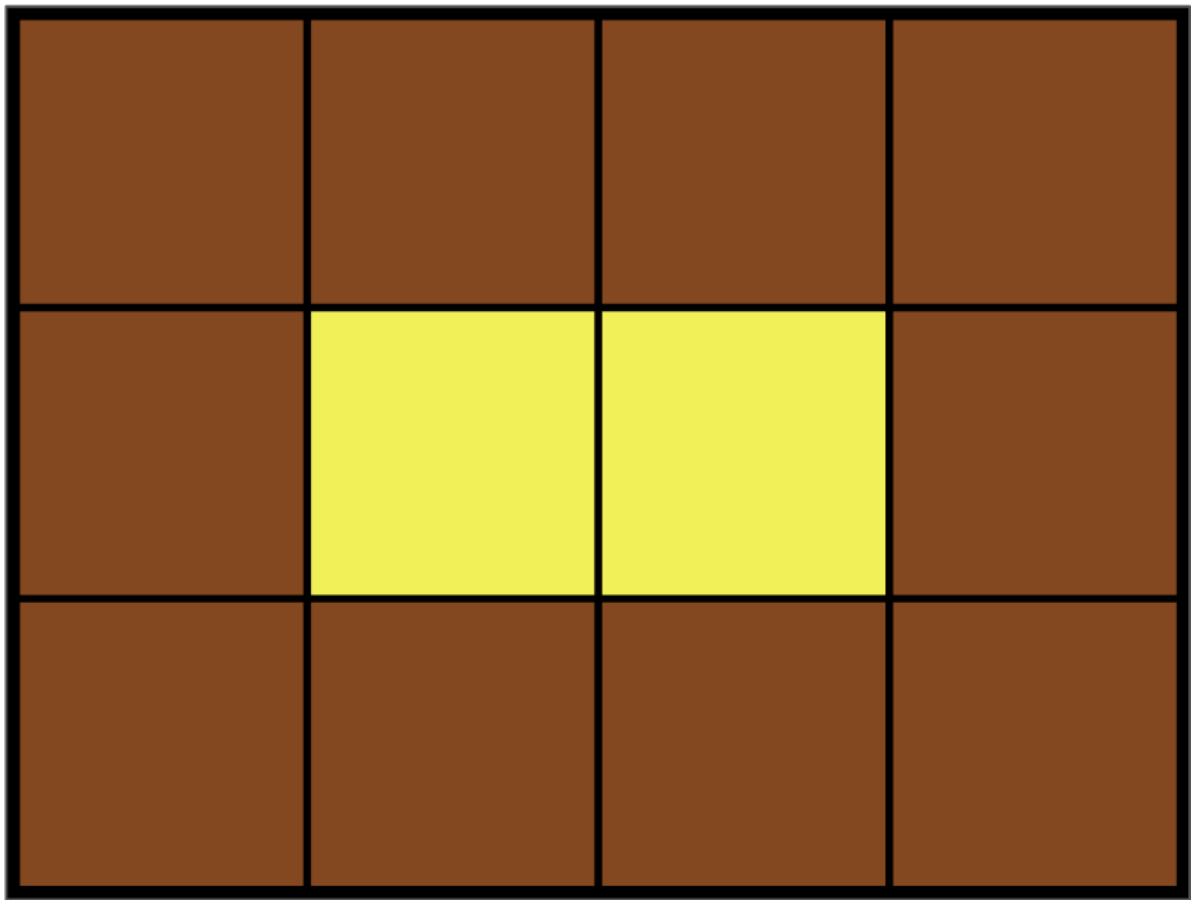
```
class Solution {
  public int solution(int n) {
              int answer = 0;
              boolean[] checked = new boolean[n + 1];

              for (int i = 2; i <= n; i++) {
                    if (!checked[i])
                          answer++;
```

```
                    for (int j = i; j <= n; j += i) {
                        if (!checked[j])
                            checked[j] = true;
                    }
                }
        return answer;
    }
}
```
https://javacoding.tistory.com/133

3. 카펫

문제 설명
Leo 는 카펫을 사러 갔다가 아래 그림과 같이 중앙에는 노란색으로 칠해져 있고 테두리 1 줄은 갈색으로 칠해져 있는 격자 모양 카펫을 봤습니다.



Leo 는 집으로 돌아와서 아까 본 카펫의 노란색과 갈색으로 색칠된 격자의 개수는 기억했지만, 전체 카펫의 크기는 기억하지 못했습니다.
Leo 가 본 카펫에서 갈색 격자의 수 brown, 노란색 격자의 수 yellow 가 매개변수로 주어질 때 카펫의 가로, 세로 크기를 순서대로 배열에 담아 return 하도록 solution 함수를 작성해주세요.
제한사항
갈색 격자의 수 brown 은 8 이상 5,000 이하인 자연수입니다.
노란색 격자의 수 yellow 는 1 이상 2,000,000 이하인 자연수입니다.
카펫의 가로 길이는 세로 길이와 같거나, 세로 길이보다 깁니다.
입출력 예

brown	yellow	return
10	2	[4, 3]
8	1	[3, 3]
24	24	[8, 6]

출처

```
package pojoPrj;
```

※ 공지 - 2020 년 2 월 3 일 테스트케이스가 추가되었습니다.
※ 공지 - 2020 년 5 월 11 일 웹접근성을 고려하여 빨간색을 노란색으로 수정하였습니다.

```
import java.util.Arrays;

class Solution {

	public int[] solution(int brown, int red) {

		int height = 0;
		int width = 0;
		for (height = 3; height <= (int) (brown + 4) / 2;
height++) {

			width = ((brown + 4) / 2) - height;
			if (width < height) {
				break;
			}

			int redCount = (width - 2) * (height - 2);
			if (red == redCount) {
				break;
			}
		}

		int[] answer = new int[] { width, height };
		return answer;
	}
}
```

https://codevang.tistory.com/301

```
import java.util.*;

class Solution {
    public int[] solution(int brown, int yellow) {
        int[] answer = new int[2];
        int sum = (brown + 4) / 2; // 가로와 세로의 합.
        int m = 3; // 세로
        int n = sum - m; // 가로

        // 노란색 칸이 최소 1 개이기때문에 n 은 3 이상이어야 함.
        // 문제에서 가로 길이는 세로 길이보다 크거나 같다고 명시되어 있음.
        while(n >= 3 && n >= m) {
            // (가로 - 2) * (세로 - 2)는 노란색 칸의 개수와 같음.
            if((n - 2) * (m - 2) == yellow){
                answer[0] = n;
                answer[1] = m;
```

```
                break;
            }

            n--; m++;
        }

        return answer;
    }
}
```
https://steady-coding.tistory.com/66