문제20,21,22

23.11.6 송이

문제20 - 모의고사

문제 이해하기

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]

문제20 - 모의고사

접근 방식 - 나머지 연산을 통해서 순차적으로 접근하기

answers [1,3,2,4,2]

student2 = [2,1,2,3,2,4,2,5]

for idx,answer in enumerate(answers):

if answer == student1[idx % len(student1)]:
score[0] +=1

0 1
1 3
2 2
3 4
4 2

idx, answer
0 1
1 3
2 2
3 4
4 2

student1[0] -> 2
student1[1] -> 1
student1[2] -> 2
student1[3] -> 3
student1[4] -> 2

문제20 - 모의고사

접근 방식 - 나머지 연산을 통해서 순차적으로 접근하기

answers [1,3,2,4,2,4,5,2,1]

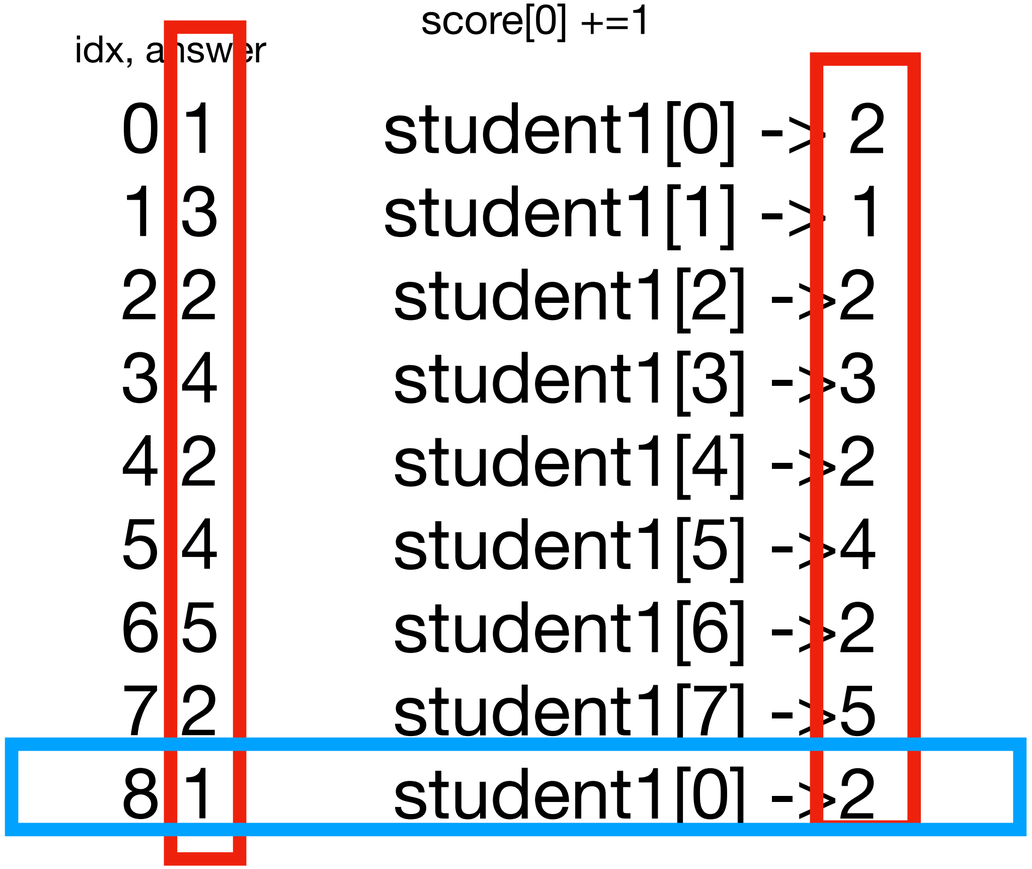
for idx,answer in enumerate(answers):

idx, answer

0 1
1 3
2 2
3 4
4 2
5 4
6 5
7 2
8 1

student2 = [2,1,2,3,2,4,2,5]

if answer == student1[idx % len(student1)]:
score[0] +=1



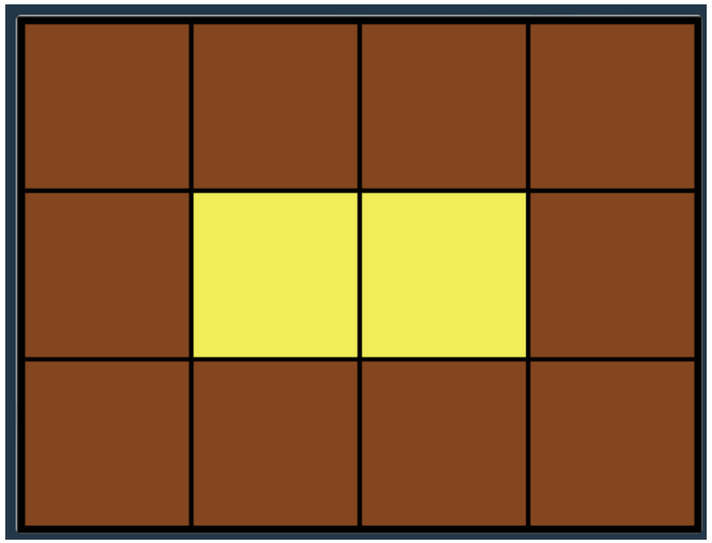
문제20 - 모의고사

전체 코드

```
def solution(answers):
    student1 = [1,2,3,4,5] #5
    student2 = [2,1,2,3,2,4,2,5] #8
    student3 = [3,3,1,1,2,2,4,4,5,5] #10
    score = [0,0,0]
    result = []
for idx,answer in enumerate(answers):
    if answer == student1[idx % len(student1)]:
        score[0] +=1
    if answer == student2[idx%len(student2)]:
        score[1] +=1
    if answer == student3[idx%len(student3)]:
        score[2] +=1
for idx,s in enumerate(score):
    if s ==max(score):
        result.append(idx+1)
return result
```

문제21 - 카펫

문제 이해하기



Leo는 집으로 돌아와서 아까 본 카펫의 노란색과 갈색으로 색칠된 격자의 개수는 기억했지만, 전체 카펫의 크기는 기억하지 못했습니다.

Leo가 본 카펫에서 갈색 격자의 수 brown, 노란색 격자의 수 yellow가 매개변수로 주어질 때 카펫의 가로, 세로 크기를 순서대로 배열에 담아 return 하도록 solution 함수를 작성해주세요.

제한사항

- 갈색 격자의 수 brown은 8 이상 5,000 이하인 자연수입니다.
- 노란색 격자의 수 yellow는 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]

문제21 - 카펫

접근 방식

- step1 - 전체 넓이를 구하기 - grid = brown + yellow
- step2 - 전체 넓이를 만들 수 있는 가로와 세로의 길이를 구하기

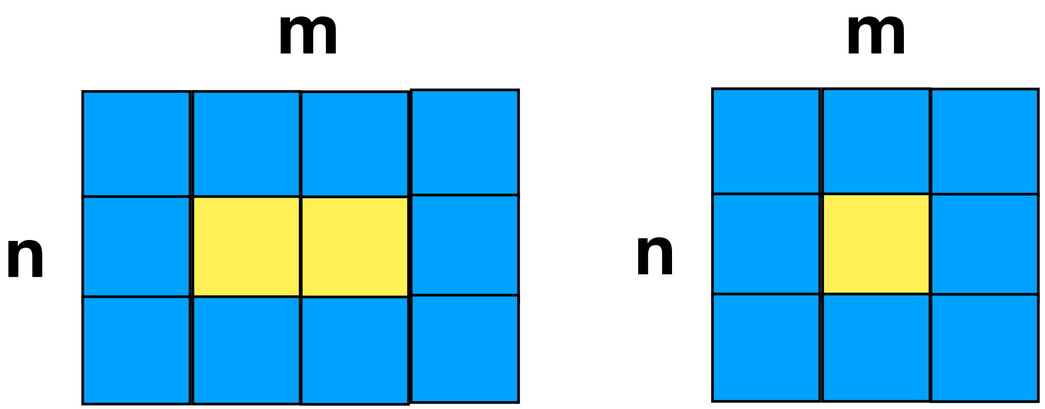
문제21 - 카펫

step2 - 전체 넓이를 구할 수 있는 가로와 세로의 길이 구하기

grid = brown + yellow ⟶ 직사각형의 넓이

넓이를 만들 수 있는 두 수의 곱을 찾아 조건을 만족하는지 찾아보기

직사각형의 넓이 48 12 x 4 6 x 8 1 x 48



m,n은 최소 3부터 시작한다.

(m-2) * (n-2)의 값과 yellow의 값이 같아야 조건 성립한다.

문제21 - 카펫

전체 코드

```
def solution(answers):
    grid = brown + yellow
    for n in range(3,int(grid**0.5)+1):
        if grid % n !=0: continue
        m = grid // n
        if (n-2) * (m-2) == yellow:
            return [m,n]
```

전체 사각형의 넓이

n이 grid의 약수가 아니라면 넘어가기

3 * 16 , 4 * 12와 같이 직사각형을 만들 수 있는 조합을 만들기

문제에서 원하는 조건

안에는 노란색 카펫이 있고

바깥쪽에 갈색 카펫이 있어야 한다.

문제22 - 소수 찾기

문제 이해하기

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 numbers가 주어졌을 때, 종이 조각으로 만들 수 있는 소수가 몇 개인지 return 하도록 solution 함수를 완성해주세요.

제한사항

- numbers는 길이 1 이상 7 이하인 문자열입니다.
- numbers는 0~9까지 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.

- 11과 011은 같은 숫자로 취급합니다.

문제22 - 소수 찾기

접근방식

- step1 - 소수인지 판별하는 함수 만들기
- step2 - 주어진 종이 조각으로부터 만들 수 있는 모든 숫자(순열)을 찾기
- step3 - 1번과 2번을 사용하여 나온 소수 배열의 길이를 계산하기

문제22 - 소수 찾기

step1 - 소수인지 판별하는 함수 만들기

```
def checkPrime(n):
    if n<2:
        return False
    for i in range(2,int(n**0.5)+1):
        if n %i==0:
            return False
    return True
```

0,1은 소수가 아님.

소수를 판별할 때
주어진 수 n의 제곱근 이하에서
나누어지는만 확인하면 ok

for문을 다 돌고나면 True 리턴

문제22 - 소수 찾기

step2 - 주어진 조각으로 만들 수 있는 모든 숫자를 찾기


```
from itertools import permutations

for i in range(1,len(numbers)+1):
    num.append(list(permutations(numbers,i)))
num = [int(‘’.join(y)) for x in num for y in x]
```

```
[[('1',), ('7',)], [('1', '7'), ('7', '1')]]
```

```
[1, 7, 17, 71]
```

문제22 - 소수 찾기

step3 - 소수 배열의 길이를 계산하기

```
for i in num:
    if checkPrime(i):
        answer.append(i)
```

```
[1, 7, 17, 71]
```

num에 있는 순열들을 하나씩 가져와서
checkPrime에서 소수인지 체크하고
만약 소수가 맞다면 answer 배열에 넣어주기

```
return len(set(answer))
```

정답은 answer 배열을 집합으로 만들어서 중복을 제거하기
그리고 len을 이용해 길이를 반환하기