

바르게 살자

코딩테스트 week1

[1차] 비밀지도

1. 지도는 **한 변의 길이가 n**인 정사각형 배열 형태로, 각 칸은 "공백"(" ") 또는 "벽"("#") 두 종류로 이루어져 있다.
2. 전체 지도는 **두 장의 지도를 겹쳐서** 얻을 수 있다. 각각 "지도 1"과 "지도 2"라고 하자. 지도 1 또는 지도 2 중 **어느 하나라도 벽인 부분은 전체 지도에서도 벽**이다. 지도 1과 지도 2에서 모두 공백인 부분은 전체 지도에서도 공백이다.
3. "지도 1"과 "지도 2"는 **각각 정수 배열로 암호화**되어 있다.
4. 암호화된 배열은 지도의 각 가로줄에서 **벽 부분을 1, 공백 부분을 0으로 부호화했을 때 얻어지는 이진수**에 해당하는 값의 배열이다.

		$01001_{(2)} = 9$
		$10100_{(2)} = 20$
		$11100_{(2)} = 28$
		$10010_{(2)} = 18$
		$01011_{(2)} = 11$
		$11110_{(2)} = 30$
		$00001_{(2)} = 1$
		$10101_{(2)} = 21$
		$10001_{(2)} = 17$
		$11100_{(2)} = 28$

▼

[1차] 비밀지도

CODE

```
n = 5
arr1 = [9, 20, 28, 18, 11]
arr2 = [30, 1, 21, 17, 28]

def solution(n, arr1, arr2):
    answer = []
    # arr1, arr2 n번씩 나눈 나머지를 arr 배열에 거꾸로 저장
    i=0

    while i<n:
        j=n-1
        num1 = arr1[i]
        num2 = arr2[j]
        string = ""

        #print(arr[i])
        while j>=0:
            if num1 % 2 == num2 % 2 == 0:
                string += ' '
            else:
                string += '#'

            num1 = num1 // 2
            num2 = num2 // 2

            j-=1
        #print(string)
        answer.append(string[::-1])
        i+=1

    return answer

solution(n, arr1, arr2)
```

문제 해결 과정

1. n번 반복하면서 i번째 행의 값을 저장
2. 해당하는 값을 n번 나눈 나머지에 따라 문자 저장
나머지 0 -> ' '
나머지 1 -> '#'
3. 2번 과정에서 나온 결과를 뒤집어야 2진수 결과가 나오기 때문에 거꾸로 answer list에 저장

[1차] 비밀지도

BEST CODE

```
def solution(n, arr1, arr2):
    answer = []
    for i, j in zip(arr1, arr2):
        a12 = str(bin(i|j)[2:])
        a12 = a12.rjust(n, '0')
        a12 = a12.replace('1', '#')
        a12 = a12.replace('0', ' ')
        answer.append(a12)
    return answer
```

문제 해결 과정 개선

1. zip 함수로 묶어서 값 불러오기
2. bin 함수와 '|' 기능으로 진수 변환을 간단히 정리
3. rjust 함수로 우측부터 정렬 후 부족한 부분에 0을 채우면서 뒤집는 과정 삭제

[1차] 다트게임

1. 다트 게임은 **총 3번의 기회**로 구성된다.
2. 각 기회마다 얻을 수 있는 점수는 **0점에서 10점**까지이다.
3. 점수와 함께 **Single(S), Double(D), Triple(T)** 영역이 존재하고 각 영역 당첨 시 점수에서 **1제곱, 2제곱, 3제곱** (점수1, 점수2, 점수3)으로 계산된다.
4. 옵션으로 **스타상(*), 아차상(#)**이 존재하며 스타상(*) 당첨 시 **해당 점수와 바로 전에 얻은 점수를 각 2배**로 만든다. 아차상(#) 당첨 시 **해당 점수는 마이너스**된다.
5. **스타상(*)은 첫 번째 기회에서도 나올 수 있다.** 이 경우 **첫 번째 스타상(*)의 점수만 2배**가 된다. (예제 4번 참고)
6. **스타상(*)의 효과는 다른 스타상(*)의 효과와 중첩**될 수 있다. 이 경우 중첩된 스타상(*) 점수는 **4배**가 된다. (예제 4번 참고)
7. **스타상(*)의 효과는 아차상(#)의 효과와 중첩**될 수 있다. 이 경우 중첩된 아차상(#)의 점수는 **-2배**가 된다. (예제 5번 참고)
8. **Single(S), Double(D), Triple(T)**은 점수마다 **하나씩 존재**한다.
9. 스타상(*), 아차상(#)은 점수마다 **둘 중 하나만 존재**할 수 있으며, **존재하지 않을 수도** 있다.

0~10의 정수와 문자 s, d, t, *, #로 구성된 문자열이 입력될 시 총점수를 반환하는 함수를 작성하라.

입출력 예제

예제	dartResult	answer	설명
1	1S2D*3T	37	$1^1 * 2 + 2^2 * 2 + 3^3$
2	1D2S#10S	9	$1^2 + 2^1 * (-1) + 10^1$
3	1D2S0T	3	$1^2 + 2^1 + 0^3$
4	1S*2T*3S	23	$1^1 * 2 * 2 + 2^3 * 2 + 3^1$
5	1D#2S*3S	5	$1^2 * (-1) * 2 + 2^1 * 2 + 3^1$
6	1T2D3D#	-4	$1^3 + 2^2 + 3^2 * (-1)$
7	1D2S3T*	59	$1^2 + 2^1 * 2 + 3^3 * 2$

[1차] 다트게임

CODE

```
def solution(dartlist):
```

```
    score = 0
```

```
    idx = 0
```

```
    tmplist = [0] * 3
```

```
    j = 0
```

```
    while idx < len(dartlist):
```

```
        if dartlist[idx] >= '0' and dartlist[idx] <= '9':
```

```
            if dartlist[idx+1] == '0':
```

```
                tmp = 10
```

```
                idx+=1
```

```
            else:
```

```
                tmp = int(dartlist[idx])
```

```
            idx+=1
```

```
        if dartlist[idx] == 'S':
```

```
            tmp = tmp **1
```

```
        elif dartlist[idx] == 'D':
```

```
            tmp = tmp **2
```

```
        elif dartlist[idx] == 'T':
```

```
            tmp = tmp **3
```

```
        idx+=1
```

```
        tmplist[j] += tmp
```

```
        j+=1
```

```
        #print(tmplist)
```

```
        elif dartlist[idx] == '**':
```

```
            tmplist[j-2] *= 2
```

```
            tmplist[j-1] *= 2
```

```
            idx+=1
```

```
        else:
```

```
            if dartlist[idx] == '#':
```

```
                tmplist[j-1] *= -1
```

```
                idx+=1
```

```
        answer =  
        tmplist[0]+tmplist[1]+tmplist[2]
```

```
        return answer
```

문제 해결 과정

1. tmplist에 3번의 결과를 각각 저장하기 위해 0이 들어 있는 list 생성

2-1. 문자열의 길이만큼 반복하면서 0~10이면 해당하는 값을 저장하고 뒤에 S or D or T에 따라, 각각 1,2,3 제곱한 결과를 저장

2-2. S or D or T 이후, '*'이 나오면 j-1, j-2번째 idx * 2

2-3. S or D or T 이후, '#'이 나오면 j-1번째 idx * -1

3. tmplist에 저장된 값을 모두 더해서 return

[1차] 다트게임

BEST CODE

```
def solution(dartResult):
    dart = {'S':1, 'D':2, 'T':3}
    scores = []
    n = 0

    for i, d in enumerate(dartResult):
        if d in dart:
            scores.append(int(dartResult[n:i])**dart[d])
        if d == "*":
            scores[-2:] = [x*2 for x in scores[-2:]]
        if d == "#":
            scores[-1] = (-1)*scores[-1]
        if not (d.isnumeric()):
            n = i+1

    return sum(scores)
```

문제 해결 과정 개선

1. enumerate 함수로 idx, value를 한 번에 호출
2. list -idx를 이용하여 '*'이 나오면
- 1개가 있으면 해당하는 값만 2배
- 2개가 있으면 둘 다 2배
- 3개 이상이면 뒤에서 2개만 2배
3. list -idx를 이용하여 '#'이 나오면 마지막 idx * -1
4. 숫자가 아니면 n 증가 (0~9 위치까지 이동)

>>> s= "1S2D*3T"	0 1
>>> for idx, value in s:	1 S
print(idx, value)	2 2
	3 D
	4 *
	5 3
	6 T

키패트 누르기

1. 엄지손가락은 **상하좌우 4가지 방향으로만 이동**할 수 있으며 **키패드 이동 한 칸은 거리로 1**에 해당합니다.
2. 왼쪽 열의 3개의 숫자 **1, 4, 7**을 입력할 때는 **왼손 엄지손가락**을 사용합니다.
3. 오른쪽 열의 3개의 숫자 **3, 6, 9**를 입력할 때는 **오른손 엄지손가락**을 사용합니다.
4. 가운데 열의 4개의 숫자 **2, 5, 8, 0**을 입력할 때는 두 엄지손가락의 **현재 키패드의 위치에서 더 가까운 엄지손가락**을 사용합니다.
 - 4-1. 만약 두 엄지손가락의 **거리가 같다면, 오른손잡이는 오른손 엄지손가락, 왼손잡이는 왼손 엄지손가락**을 사용합니다.



키패트 누르기

CODE

```
def solution(numbers, hand):
    answer = ''
    key = [[1, 2, 3], [4, 5, 6], [7, 8, 9], ['*', 0, '#']]


    left = [3, 0]
    right = [3, 2]

    for num in numbers:
        if num % 3 == 1: # 1 4 7
            answer += 'L'
            left = [num//3, 0]
        elif num % 3 == 0 and num != 0: # 3 6 9
            answer += 'R'
            right = [num//3-1, 2]
        else: # 2 5 8 0
            middle = num//3
            if num == 0: middle = 3

            left_diff =
                abs(left[0]-middle) + abs(left[1]-1)
            right_diff =
                abs(right[0]-middle) + abs(right[1]-1)

            if left_diff < right_diff:
                # 왼쪽이 가까운 경우
                answer += 'L'
                left = [middle, 1]
            elif left_diff > right_diff:
                # 오른쪽이 가까운 경우
                answer += 'R'
                right = [middle, 1]
            else:
                if hand == "left":
                    answer += 'L'
                    left = [middle, 1]
                else:
                    answer += 'R'
                    right = [middle, 1]

    return answer
```



문제 해결 과정

1. 키패드 3 by 4 리스트 생성

2. left, right 위치를 저장하는 변수 선언

3. 3가지 경우로 분리하여

1 4 7은 왼손

3 6 9는 오른손

2 5 8 0은 해당 위치와 왼손 / 오른손의 거리를 파악하여
사용할 손가락 지정 후 좌표 변경

키패트 누르기

BEST CODE

문제 해결 과정 개선