1주차 스터디 활동보고서(2기 2반 1조)

참석자	강희원(조장)	박상욱	조윤식	한경훈

주제: 프로그래머스에 lv1 이상의 정답률 70%이하의 문제 코드 리뷰

2016년(출처: https://school.programmers.co.kr/learn/courses/30/lessons/12901)
문제 설명:2016년이 윤달이고 1월 1일이 금요일이라고 할 때 a월 b일은 무슨요일인지 맞추는 문제 입니다

```
def solution(a, b):
    # 각 달의 일수를 리스트로 저장 (윤년이므로 2 월은 29 일)
    days_in_month = [31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

    # 1 월 1 일이 금요일이므로 'FRI'부터 시작하는 요일 리스트
    days_of_week = ['FRI', 'SAT', 'SUN', 'MON', 'TUE', 'WED', 'THU']

# 1 월 1 일에서 (a 월 b 일까지의) 총 일수를 계산
    total_days = sum(days_in_month[:a-1]) + (b - 1)

    # 총 일수를 7 로 나눈 나머지를 이용해 요일을 계산
    answer = days_of_week[total_days % 7]

    return answer
print(solution(5, 24))
```

순서: 달에 따른 일수 리스트 작성->요일 리스트 작성->sum 함수로 요일 수 계산 후 7로 나누기

붕대 감기(출처: https://school.programmers.co.kr/learn/courses/30/lessons/250137)
문제 설명: bandage 상태에서는 체력을 회복하는 데 일정시간 동안 공격을 받지 않고 회복하면 추가 회복이 일어납니다. Attacks 상태는 주어진 시간(초)에 데미지를 받습니다
Health는 건강,즉 체력을 말하면 체력이 0이거나 -면 -1을 반환합니다

```
# bandage [시전시간, 초당회복량, 추가회복량]

def solution(bandage, health, attacks):
    time = 1
    attacks = sorted(attacks, key = lambda x:x[0])
    max_hp = health
    banding_time = 1

    while attacks:
        if time == attacks[0][0]:
            banding_time = 1
            health -= attacks[0][1]
            if health <= 0:
```

```
                return -1
            attacks = attacks[1:]
            time += 1
            continue


        if banding_time == bandage[0]:
            health += bandage[2]
            banding_time = 0
        health += bandage[1]
        banding_time += 1

        if health > max_hp:
            health = max_hp # 최대 체력이 기존의 체력 이상으로 못 가는 코드

        time += 1


    return health


print(solution(
    [5, 1, 5], 30, [[2, 10], [9, 15], [10, 5], [11, 5]]))
```

순서도:bandage,time,attack,health 인자를 설정합니다-> 공격을 받을 경우 체력 회복이 정지 되며 추가회복을 하기 위한 시간이 초기화 됩니다 ->공격이 끝난 이후 다시 체력을 회복하고 일정시간이 지나면 추가회복을 합니다. 이때 주의사항으로 최대체력은 기존에 입력받았던 체력을 넘어갈 수 없다는 조건을 추가해야 합니다.-> 공격과 체력 회복이 끝난 이후의 체력을 가지고 0보다 작다면 -1 0보다 크다면 남은 체력을 반환합니다.

끝말잇기

문제 설명: 기존의 끝말잇기처럼 단어의 마지막 알파벳이 다음 시작하는 단어의 첫번째 알파벳과 일치하는 지를 확인하여 결과를 출력하는 문제입니다

```
# 끝말 잇기
# https://school.programmers.co.kr/learn/courses/30/lessons/12981

def solution(n, words):
    turn = 1
    answer = [words[0]]

    for i in range(1, len(words)):
        if words[i] in answer:
            return [i % n + 1, turn]

        if words[i-1][-1] != words[i][0]:
            return [i % n + 1, turn]
```

```
        answer.append(words[i])
        print(answer)

        if (i + 1) % n == 0:
            turn += 1

    return [0, 0]


# print(solution(5,
#                ["hello", "observe", "effect", "take",
#                 "either", "recognize", "encourage", "ensure",
#                 "establish", "hang", "gather", "refer",
#                 "reference", "estimate", "executive"]
#                ))
```

순서도: turn은 참가자들이 몇 바퀴를 돌았는지를 나타내며 answer는 단어들의 리스트를 나타냅니다-> for문, "in"으로 다음의 if문의 조건이 맞추어지는지를 반복합니다-> 첫 번째 if문으로 단어의 중복 여부를 확인합니다. -> 두 번째 if문으로 끝난 단어의 알파벳과 시작하는 단어의 알파벳 일치성을 확인합니다. -> 이상이 없을 경우 단어를 리스트에 추가하고 바퀴 수를 1증가합니다->최종적으로 이상이 없을 경우 return값을 [0,0]으로 반환합니다

공원(출처: https://school.programmers.co.kr/learn/courses/30/lessons/340198)

문제 설명:사람들이 앉아 있는 공원에서 정사각형 형태의 돗자리를 펴려고 할 때 최대 NxN형태의 돗자리를 펼 수 있는지 확인하는 문제입니다

```
def squre(n,x,y,park):
    for i in range(n):
        for j in range(n):
            if park[y+i][x+j]!='-1':
                return False
    return n

def solution(mats, park):
    col = len(park)
    row = len(park[0])
    mats = sorted(mats)
    answer = -1

    for n in mats:
        tmp=False
        for y in range(col):
            for x in range(row):
                if (x+n-1<row) and (y+n-1<col):
                    tmp=squre(n,x,y,park)
                if tmp:
                    answer=tmp
                    break
            if tmp:
```

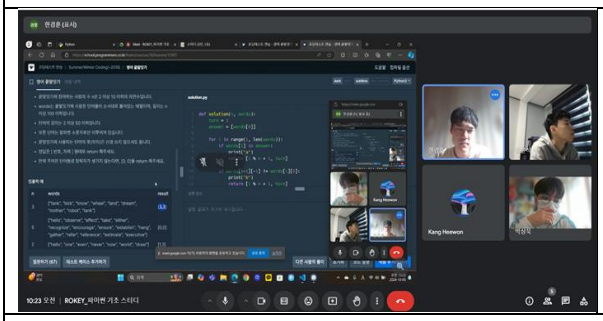
```
                break
    return answer
```

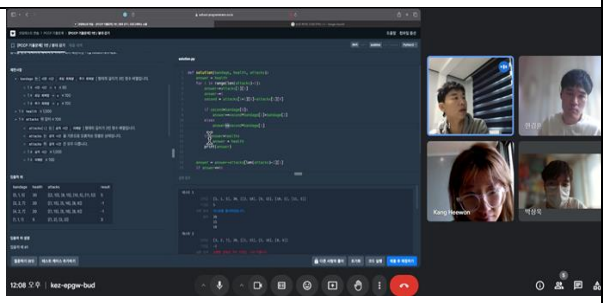
순서도: 첫번째 함수에서 n=매트의 크기,(x, y)는 매트를 놓을 시작 좌표,.이중 for 루프는 n x n 크기만큼 공원의 해당 영역을 탐색하는 코드를 설정(이때 만약 탐색하는 영역 중 하나라도 '-1'이 아닌 값이 있으면 매트를 놓을 수 없다고 판단하고 False를 반환합니다. 예를 들어 2x2의 경우 3개는 되는데 왼쪽 아래가 설치할 수 없는 경우 false를 반환)

-> 두번째 함수의 경우 col과 row는 공원의 행과 열의 크기를 설정,mats 리스트는 돗자리의 크기를 저장한 것으로, 이를 sorted를 통해 오름차순으로 정렬합니다. 작은 매트부터 순차적으로 확인할 수 있게 됩니다.(이 코드가 공원에서 설치할 수 있는 최대 돗자리의 구할 수 있게 도와줌)answer는 가장 큰 매트 크기를 저장하는 변수로, 처음엔 -1로 초기화되어 있습니다.

-> 마지막인 이중 for 루프문을 통하여 돗자리가 공원의 크기를 넘어서지 않는지 확인 후 넘어서지 않는다면 돗자리가 해당 좌표에서 놓을 수 있는 지 확인합니다. 이제 answer를 통하여 돗자리의 최대 크기를 구하고 그 값을 반환합니다.(이때 break문을 통해 조건을 만족하면 반복을 멈추도록 설정합니다)

스터디 사진;





미참석자 및 사유:이번주에는 없음