Đề thi CosPro Level 2

Câu 1

□ Mô tả bài toán

Khi cho điểm thi của các học sinh, ta muốn tìm xem **học sinh số n xếp hạng mấy**.
Số báo danh của học sinh bắt đầu từ **0**, và **điểm thi được cho theo thứ tự số báo danh**.

Ví dụ, nếu điểm thi của 4 học sinh như sau:

Số học sinh	0	1	2	3
Điểm toán	20	60	98	59

Thì **học sinh số 3 xếp hạng 3**.

Để thực hiện việc này, cấu trúc chương trình được xây dựng như sau:

1. Lưu điểm của học sinh số n vào một biến.
2. Sắp xếp điểm theo thứ tự **giảm dần**.
3. Duyệt mảng từ phần tử đầu tiên đến phần tử cuối cùng để tìm điểm của học sinh số n.
4. Khi tìm được điểm bằng với điểm đã lưu ở bước 1 thì **return thứ hạng**.

Mảng `scores` chứa điểm thi của các học sinh theo thứ tự số báo danh, độ dài của mảng là `scores_len`, và số báo danh `n` được truyền làm tham số cho hàm `solution`.
Hãy viết hàm `solution` để trả về **thứ hạng của học sinh số n**.
Dựa trên cấu trúc trên, hãy điền đúng các hàm `func_a`, `func_b`, `func_c` và các tham số vào chỗ trống để chương trình hoạt động chính xác.

□ Giải thích tham số

Mảng `scores` chứa điểm thi của các học sinh theo thứ tự số báo danh, độ dài mảng là `scores_len`, và số báo danh `n` được truyền làm tham số cho hàm `solution`.

- `scores_len` là số nguyên từ **1 đến 100**.
- Điểm thi nằm trong khoảng từ **0 đến 100**, **không có học sinh nào bị trùng điểm**.
- `n` là số nguyên từ **0 đến (độ dài của scores − 1)**.

□ Giải thích giá trị trả về

Hãy **return thứ hạng của học sinh số n**.

□ Ví dụ

scores	scores_len	n	result
[20, 60, 98, 59]	4	3	3

□ Giải thích ví dụ

Điểm của học sinh số 3 là **59 điểm**.
Sắp xếp điểm theo thứ tự giảm dần ta được **[98, 60, 59, 20]**.
Trong mảng đã sắp xếp, **59 đứng ở vị trí thứ 3**.
Vì vậy, học sinh số 3 xếp **hạng 3**.

```python
def func_a(scores, score):
    rank = 1
    for s in scores:
        if s == score:
            return rank
        rank += 1
    return 0

def func_b(arr):
    arr.sort(reverse=True)

def func_c(arr, n):
    return arr[n]

def solution(scores, n):
    score = func_@@@(@@@)
    func_@@@(@@@)
    answer = func_@@@(@@@)
    return answer

scores = [20, 60, 98, 59]
n = 3
ret = solution(scores, n)

print("solution 함수의 반환 값은", ret, "입니다.")
```

Câu 2

□ Mô tả bài toán

Một trường học trao học bổng vào mỗi cuối học kỳ. Lúc này, cần tìm xem có **bao nhiêu học sinh nhận học bổng**. Điều kiện trao học bổng như sau:

1. Học sinh có **điểm học kỳ này từ 80 điểm trở lên** (thang 100 điểm) **và** có **xếp hạng trong top 10%**.
2. Học sinh có **điểm học kỳ này từ 80 điểm trở lên và đứng hạng 1**.

3. Học sinh có **mức tăng điểm lớn nhất so với học kỳ trước** (nếu có nhiều học sinh thì **tất cả** các học sinh đó).

Lưu ý:

- Các học sinh **đồng điểm** thì **có cùng thứ hạng**.
- **Không được nhận trùng lặp** (một học sinh không thể nhận học bổng nhiều lần).

Để tính số học sinh nhận học bổng, chương trình được thiết kế theo cấu trúc sau:

1. Dựa trên điểm học kỳ này, tính **thứ hạng** của từng học sinh.
2. Tính giá trị lớn nhất của **(điểm học kỳ này − điểm học kỳ trước)** trong số các học sinh.
3. Khi phát hiện học sinh thỏa một trong các điều kiện dưới đây, **tăng số lượng học sinh nhận học bổng lên 1**:
   - 3-1. Điểm học kỳ này ≥ 80 và thứ hạng thuộc **top 10%**.
   - 3-2. Hoặc điểm học kỳ này ≥ 80 và **xếp hạng 1**.
   - 3-3. Hoặc (điểm học kỳ này − điểm học kỳ trước) **bằng** giá trị tìm được ở bước 2, và giá trị đó **là số dương**.
4. Trả về số học sinh nhận học bổng.

Mảng `current_grade` chứa điểm học kỳ này của học sinh và độ dài `current_grade_len` , mảng `last_grade` chứa điểm học kỳ trước và độ dài `last_grade_len` được truyền làm tham số.
Hãy viết hàm `solution` để trả về **số học sinh nhận học bổng**. Dựa theo cấu trúc trên, hãy điền đúng các hàm `func_a` , `func_b` , `func_c` và các tham số vào chỗ trống để mã hoạt động chính xác.

□ Giải thích tham số

Mảng `current_grade` chứa điểm học kỳ này và độ dài `current_grade_len` , mảng `last_grade` chứa điểm học kỳ trước và độ dài `last_grade_len` được truyền làm tham số.

- `current_grade_len` và `last_grade_len` bằng nhau, là số tự nhiên từ **1 đến 200**.
- Các phần tử của `current_grade` và `last_grade` là số nguyên từ **0 đến 100**.

□ Giải thích giá trị trả về

Hãy **return số học sinh nhận học bổng**.

□ Ví dụ

- `current_grade` : [70, 100, 70, 80, 50, 95]
- `current_grade_len` : 6
- `last_grade` : [35, 65, 80, 50, 20, 60]
- `last_grade_len` : 6
- `return` : 3

□ Giải thích ví dụ

Vì số học sinh ít hơn 10, nên **học sinh hạng 1** sẽ nhận học bổng.
Những học sinh có mức tăng điểm lớn nhất so với học kỳ trước gồm **3 người**, như sau:

- 35 → 70 điểm
- 65 → 100 điểm
- 60 → 95 điểm

Trong đó, học sinh thứ 2 đã nhận học bổng **hạng 1** rồi nên **không được nhận trùng lặp**; hai học sinh còn lại có thể nhận học bổng.
Vì vậy, tổng số học sinh nhận học bổng là **3**.

```python
def func_a(current_grade, last_grade, rank, max_diff_grade):
    arr_length = len(current_grade)
    count = 0
    for i in range(arr_length):
        if current_grade[i] >= 80 and rank[i] <= arr_length // 10:
            count += 1
        elif current_grade[i] >= 80 and rank[i] == 1:
            count += 1
        elif max_diff_grade > 0 and max_diff_grade == current_grade[i] - last_grade[i]:
            count += 1
    return count

def func_b(current_grade):
    arr_length = len(current_grade)
    rank = [1] * arr_length
    for i_1 in range(arr_length):
        for i_2 in range(arr_length):
            if current_grade[i_1] < current_grade[i_2]:
                rank[i_1] += 1
    return rank

def func_c(current_grade, last_grade):
    max_diff_grade = 0
    for i in range(len(current_grade)):
        max_diff_grade = max(max_diff_grade, current_grade[i] - last_grade[i])
    return max_diff_grade

def solution(current_grade, last_grade):
    rank = func_@@@(@@@)
    max_diff_grade = func_@@@(@@@)
    answer = func_@@@(@@@)
    return answer

current_grade = [70, 100, 70, 80, 50, 95]
last_grade = [35, 65, 80, 50, 20, 60]
ret = solution(current_grade, last_grade)

print("solution 함수의 반환 값은", ret, "입니다.")
```

Câu 3

□ Mô tả bài toán

Một vận động viên thể dục dụng cụ sẽ nhận **điểm cuối cùng** bằng cách **loại bỏ 1 điểm cao nhất và 1 điểm thấp nhất** trong số các điểm do nhiều giám khảo chấm, sau đó tính **trung bình cộng** của các điểm còn lại.
Lưu ý: **phần thập phân sẽ bị loại bỏ**.

Ví dụ, vận động viên A nhận được điểm từ 10 giám khảo như sau:
[35, 28, 98, 34, 20, 50, 85, 74, 71, 7]

Loại bỏ điểm cao nhất là **98** và điểm thấp nhất là **7**, trung bình cộng của 8 điểm còn lại là **49.625**.
Sau khi bỏ phần thập phân, kết quả là **49 điểm**.

Khi mảng `scores` chứa điểm của các giám khảo và độ dài `scores_len` được truyền làm tham số, hãy viết hàm `solution` để **trả về điểm mà vận động viên này nhận được**.

□ Giải thích tham số

Mảng `scores` chứa điểm do các giám khảo chấm và độ dài `scores_len` được truyền làm tham số cho hàm `solution` .

- `scores_len` là số tự nhiên từ **3 đến 100**.
- Điểm do giám khảo chấm là số nguyên trong khoảng **0 đến 100**.

□ Giải thích giá trị trả về

Hãy **return giá trị trung bình (đã bỏ phần thập phân)** của các điểm sau khi **loại bỏ điểm cao nhất và điểm thấp nhất**.

□ Ví dụ

scores	scores_len	return
[35, 28, 98, 34, 20, 50, 85, 74, 71, 7]	10	49
[1, 1, 1, 1, 1]	5	1

□ Giải thích ví dụ

Ví dụ #1

Giống với ví dụ đã nêu trong đề bài.

Ví dụ #2

Điểm cao nhất là **1**, điểm thấp nhất cũng là **1**.
Sau khi loại bỏ hai điểm **1**, tổng các điểm còn lại là **3**, trung bình là **1**.

```
def solution(scores):
    answe = 0

    return answer

scores1 = [35, 28, 98, 34, 20, 50, 85, 74, 71, 7]
ret1 = solution(scores1)

print("solution 함수의 반환 값은", ret1, "입니다.")

scores2 = [1, 1, 1, 1, 1]
ret2 = solution(scores2)

print("solution 함수의 반환 값은", ret2, "입니다.")
```

Câu 4

□ Mô tả bài toán

Để học thuộc từ tiếng Anh, người ta gõ lặp lại một từ nhiều lần. Tuy nhiên, sau khi gõ xong thì phát hiện có **nhiều lỗi chính tả**. Để sửa các lỗi này, cần xác định **phải thay đổi bao nhiêu ký tự**.

Ví dụ, khi gõ từ **"CODE"** 3 lần thì thu được mảng:
["CODE", "COED", "CDEO"]

1. `"CODE"` được gõ đúng.
2. `"COED"` cần thay đổi **E thành D** và **D thành E**.
3. `"CDEO"` cần thay đổi **D, E, O** lần lượt thành **O, D, E**.

Vì vậy, **tổng số ký tự cần thay đổi là 5**.

Khi mảng `words` chứa các từ đã gõ, độ dài của mảng là `words_len` , và từ gốc muốn gõ là `word` , hãy hoàn thành hàm `solution` để **return số ký tự cần phải thay đổi**.

□ Giải thích tham số

Mảng `words` chứa các từ đã gõ, độ dài `words_len` , và từ gốc `word` được truyền làm tham số cho hàm `solution` .

- `word` là từ gồm **không quá 10 chữ cái in hoa**.
- Độ dài của các chuỗi trong `words` **bằng với độ dài của** `word` , và tất cả đều là **chữ cái in hoa**.
- `words_len` là số tự nhiên **không quá 15**.

□ Giải thích giá trị trả về

Hãy **return số lượng ký tự cần phải thay đổi**.

□ Ví dụ

words	words_len	word	return
["CODE", "COED", "CDEO"]	3	"CODE"	5

□ Giải thích ví dụ

Giống với ví dụ đã nêu trong đề bài.

```
def solution(words, word):
    count = 0

    return count

words = ["CODE", "COED", "CDEO"]
word = "CODE"
ret = solution(words, word)

print("solution 함수의 반환 값은", ret, "입니다.")
```

Câu 5

□ Mô tả bài toán

Cần tính **tổng chi phí đi lại** của các du khách. Có tổng cộng **3 loại phương tiện giao thông**: `"Bus"`, `"Ship"`, `"Airplane"`.
Nếu **tuổi từ 20 trở lên** thì áp dụng **giá người lớn**, ngược lại áp dụng **giá trẻ em**.
Giá của từng phương tiện như sau:

Phương tiện	Người lớn	Trẻ em
Bus	40.000 won	15.000 won

Phương tiện	Người lớn	Trẻ em
Ship	30.000 won	13.000 won
Airplane	70.000 won	45.000 won

Nếu **số lượng du khách từ 10 người trở lên**, sẽ được **giảm giá theo độ tuổi** như sau:

Đối tượng	Tỷ lệ giảm
Người lớn	10%
Trẻ em	20%

Mảng `member_age` chứa độ tuổi của các du khách, độ dài mảng là `member_age_len` , và phương tiện giao thông `transportation` được truyền làm tham số.
Hãy viết hàm `solution` để **return tổng chi phí đi lại**.
Hãy điền vào chỗ trống để hoàn thành toàn bộ đoạn mã.

□ Giải thích tham số

Mảng `member_age` chứa độ tuổi của các du khách, độ dài mảng là `member_age_len` , và phương tiện giao thông `transportation` được truyền làm tham số cho hàm `solution` .

- `member_age_len` là số tự nhiên từ **1 đến 1.000**.
- Các phần tử của `member_age` là số nguyên từ **1 đến 100**.
- `transportation` là một trong các giá trị: `"Bus"` , `"Ship"` hoặc `"Airplane"` .

□ Giải thích giá trị trả về

Hãy **return tổng số tiền chi phí đi lại**.

□ Ví dụ

member_age	member_age_len	transportation	return
[13, 33, 45, 11, 20]	5	"Bus"	150000
[25, 11, 27, 56, 7, 19, 52, 31, 77, 8]	10	"Ship"	203600

□ Giải thích ví dụ

Ví dụ #1

Độ tuổi của du khách là `[13, 33, 45, 11, 20]` và sử dụng phương tiện `"Bus"`.
Có **2 người dưới 20 tuổi**, chi phí trẻ em là **30.000 won**, và **3 người từ 20 tuổi trở lên**, chi phí người lớn là **120.000 won**.
Vì vậy, **tổng chi phí đi lại là 150.000 won**.

Ví dụ #2

Độ tuổi của du khách là `[25, 11, 27, 56, 7, 19, 52, 31, 77, 8]` và sử dụng phương tiện `"Ship"`.
Do số lượng du khách **từ 10 người trở lên**, nên được **giảm giá**.
Có **4 người dưới 20 tuổi**, chi phí trẻ em là **41.600 won**, và **6 người từ 20 tuổi trở lên**, chi phí người lớn là **162.000 won**.
Vì vậy, **tổng chi phí đi lại là 203.600 won**.

```python
def solution(member_age, transportation):
    if transportation == 'Bus':
        adult_expense = 40000
        child_expense = 15000
    elif transportation == 'Ship':
        adult_expense = 30000
        child_expense = 13000
    elif transportation == 'Airplane':
        adult_expense = 70000
        child_expense = 45000

    if len(member_age) >= 10:
        adult_expense = @@@
        child_expense = @@@

    total_expenses = 0
    for age in member_age:
        if @@@:
            total_expenses += adult_expense
        else:
            total_expenses += child_expense

    return total_expenses

member_age1 = [13, 33, 45, 11, 20]
transportation1 = "Bus"
ret1 = solution(member_age1, transportation1)

print("solution 함수의 반환 값은", int(ret1), "입니다.")

member_age2 = [25, 11, 27, 56, 7, 19, 52, 31, 77, 8]
transportation2 = "Ship"
ret2 = solution(member_age2, transportation2)

print("solution 함수의 반환 값은", int(ret2), "입니다.")
```

Câu 6

□ Mô tả bài toán

Ví dụ, nếu **độ dài của gạch (tile)** là **11**, thì có thể sơn theo thứ tự màu **"RRRGGBRRRGG"**.

Khi độ dài của gạch `tile_length` được truyền làm tham số, hãy viết hàm `solution` để **return chuỗi biểu diễn thứ tự sơn gạch**.
Nếu **không thể sơn theo đúng thứ tự**, hãy **return -1**.

□ Giải thích tham số

Độ dài của gạch `tile_length` được truyền làm tham số cho hàm `solution` .

- `tile_length` là số tự nhiên **không quá 1.000**.

□ Giải thích giá trị trả về

Hãy **return chuỗi biểu diễn thứ tự sơn gạch**.
Nếu không thể sơn gạch theo đúng thứ tự, hãy **return -1**.

□ Ví dụ

tile_length	answer
11	"RRRGGBRRRGG"
16	"-1"

□ Giải thích ví dụ

Ví dụ #1

Có thể sơn gạch theo thứ tự sau:

- `'R'` – 3 viên
- `'G'` – 2 viên
- `'B'` – 1 viên
- `'R'` – 3 viên
- `'G'` – 2 viên

Do đó, **return "RRRGGBRRRGG"**.

Ví dụ #2

Độ dài gạch là **16**.
Nếu sơn theo thứ tự `'R'`, `'G'`, `'B'` thì kết quả như sau:

Vị trí	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Màu	R	R	R	G	G	B	R	R	R	G	G	B	R	R	R	-

Sau khi sơn đến viên thứ **15**, còn **1 viên gạch trống**.
Do màu `'G'` cần sơn **2 viên**, nên không thể tiếp tục sơn được.
Vì vậy, **return "-1"**.

```
def solution(tile_length):
    answer = ''
    com = 'RRRGGB'
    if tile_length%6 == 1 or tile_length%6 == 2 or @@@:
        answer = '-1'
    else:
        for i in range(tile_length):
            answer += com[i % 6]
    return answer

tile_length1 = 11
ret1 = solution(tile_length1)

print("solution 함수의 반환 값은", ret1, "입니다.")

tile_length2 = 16
ret2 = solution(tile_length2)

print("solution 함수의 반환 값은", ret2, "입니다.")
```

Câu 7

□ Mô tả bài toán

Để làm **1 ly nước ép**, cần **3 quả táo** và **1 củ cà rốt**.
Tuy nhiên, để cho thỏ ăn, người ta muốn **dành ra k đơn vị thức ăn**, **không phân biệt là táo hay cà rốt**.
Mục tiêu là **làm được càng nhiều ly nước ép càng tốt**.

Khi số lượng táo `num_apple` , số lượng cà rốt `num_carrot` , và số lượng thức ăn dành cho thỏ `k` được cho, hãy viết hàm `solution` để **return số ly nước ép tối đa có thể làm được**.
Tuy nhiên, do **một phần của đoạn mã bị sai**, nên với một số dữ liệu đầu vào, chương trình **không hoạt động đúng**.
Hãy **chỉ thay đổi đúng một dòng trong đoạn mã đã cho** để chương trình hoạt động chính xác với **mọi dữ liệu đầu vào**.

□ Giải thích tham số

Số lượng táo `num_apple` , số lượng cà rốt `num_carrot` , và số lượng thức ăn dành cho thỏ `k` được truyền làm tham số cho hàm `solution` .

- `num_apple` và `num_carrot` là số nguyên từ **0 đến 200**.
- `k` là số nguyên từ **0 đến (`num_apple + num_carrot`)**.

□ Giải thích giá trị trả về

Hãy **return số ly nước ép tối đa có thể làm được**.

□ Ví dụ

num_apple	num_carrot	k	return
5	1	2	1
10	5	4	2

□ Giải thích ví dụ

Ví dụ #1

Nếu dùng **2 quả táo** làm thức ăn cho thỏ, thì còn lại **3 quả táo** và **1 củ cà rốt**.
Với số nguyên liệu còn lại này, có thể làm được **1 ly nước ép**.

Ví dụ #2

Nếu dùng **2 quả táo** và **2 củ cà rốt** làm thức ăn cho thỏ, thì còn lại **8 quả táo** và **3 củ cà rốt**.
Với số nguyên liệu còn lại này, có thể làm được **2 ly nước ép**.

```
def solution(num_apple, num_carrot, k):
        answer = 0

        if num_apple < (3 * num_carrot):
                answer = num_apple // 3
        else:
                answer = num_carrot

        num_apple -= answer * 3
        num_carrot -= answer

        i = 0
        k = k - (num_apple + num_carrot)

        while k > 0:
                if i % 4 == 0:
                        answer = answer + 1
                i = i + 1
                k = k - 1

        return answer

num_apple1 = 5
num_carrot1 = 1
k1 = 2
ret1 = solution(num_apple1, num_carrot1, k1)

print("solution 함수의 반환 값은", ret1, "입니다.")

num_apple2 = 10
num_carrot2 = 5
k2 = 4
ret2 = solution(num_apple2, num_carrot2, k2)

print("solution 함수의 반환 값은", ret2, "입니다.")
```

Câu 8

□ Mô tả bài toán

Anh A muốn xác định **khoảng thời gian trong một ngày mà anh ấy bật từ 2 chiếc TV trở lên**.
Mỗi ngày, anh A xem **3 chương trình truyền hình**.
Khi thời gian phát sóng của các chương trình **bị trùng nhau**, anh A sẽ bật **nhiều TV** để xem tất cả các chương trình.

- Nếu **2 chương trình** có thời gian phát sóng trùng nhau thì bật **2 TV**.
- Nếu **3 chương trình** có thời gian phát sóng trùng nhau thì bật **3 TV**.

Khi mảng hai chiều `programs` chứa thời gian **bắt đầu và kết thúc** của 3 chương trình, và số hàng của mảng là `programs_len` , hãy viết hàm `solution` để **return tổng thời gian trong ngày mà anh A bật từ 2 TV trở lên**.
Tuy nhiên, do **một phần của đoạn mã bị sai**, nên với một số dữ liệu đầu vào, chương trình **không hoạt động đúng**.
Hãy **chỉ thay đổi đúng một dòng trong đoạn mã đã cho** để chương trình hoạt động chính xác với **mọi dữ liệu đầu vào**.

□ Giải thích tham số

Mảng hai chiều `programs` chứa **thời gian bắt đầu và kết thúc** của 3 chương trình, và số hàng của mảng là `programs_len` , được truyền làm tham số cho hàm `solution` .

- `programs_len` **luôn bằng 3**.
- Mỗi phần tử trong mảng `programs` có dạng **[thời gian bắt đầu, thời gian kết thúc]**.
- Thời gian bắt đầu và kết thúc của chương trình là số nguyên từ **0 đến 24**.
- Thời gian bắt đầu **luôn sớm hơn** thời gian kết thúc.

□ Giải thích giá trị trả về

Hãy **return tổng thời gian mà anh A bật từ 2 TV trở lên**.

□ Ví dụ

programs	programs_len	return
1, 6], [3, 5], [2, 8	3	4

□ Giải thích ví dụ

Thời gian bật **2 TV** là từ **2 giờ đến 3 giờ** và từ **5 giờ đến 6 giờ**, tổng cộng **2 giờ**.
Thời gian bật **3 TV** là từ **3 giờ đến 5 giờ**, tổng cộng **2 giờ**.
Do đó, tổng thời gian bật **từ 2 TV trở lên** là **4 giờ**.

```
def solution(programs):
        answer = 0
        used_tv = [0] * 25

        for program in programs:
                for i in range(program[0], program[1]):
                        used_tv[i] = used_tv[i] + 1

        for i in used_tv:
                if i >= 1:
                        answer = answer + 1
        return answer

programs = [[1, 6], [3, 5], [2, 8]]
ret = solution(programs)

print("solution 함수의 반환 값은", ret, "입니다.")
```

Câu 9

□ Mô tả bài toán

Tại bãi đỗ xe của cơ quan nhà nước, áp dụng **chế độ chẵn – lẻ theo biển số xe (2부제)**.
Theo chế độ này:

- Xe có **chữ số cuối của biển số là số lẻ** chỉ được vào bãi vào **ngày lẻ**.
- Xe có **chữ số cuối của biển số là số chẵn** chỉ được vào bãi vào **ngày chẵn**.

Khi số ngày `day` , mảng `numbers` chứa **biển số các xe muốn vào bãi trong ngày đó**, và độ dài mảng là `numbers_len` được cho làm tham số, hãy viết hàm `solution` để **return số lượng xe có thể vào bãi đỗ**.
Tuy nhiên, do **một phần của đoạn mã bị sai**, nên chương trình **không hoạt động đúng**.
Hãy **chỉ thay đổi đúng một dòng trong đoạn mã đã cho** để chương trình hoạt động chính xác với **mọi dữ liệu đầu vào**.

Lưu ý: **Không có biển số xe bị trùng lặp**.

□ Giải thích tham số

- `day` : ngày trong tháng, là số tự nhiên từ **1 đến 31**.
- `numbers` : mảng chứa biển số các xe muốn vào bãi trong ngày đó.
- `numbers_len` : độ dài của mảng `numbers` .

Điều kiện:

- `numbers_len` là số tự nhiên từ **1 đến 9.000**.
- Các phần tử của `numbers` là số tự nhiên từ **1.000 đến 9.999**.
- Trong mảng `numbers` **không có biển số trùng lặp**.

□ Giải thích giá trị trả về

Hãy **return số lượng xe có thể vào bãi đỗ xe**.

□ Ví dụ

day	numbers	numbers_len	return
17	[3285, 1724, 4438, 2988, 3131, 2998]	6	2

□ Giải thích ví dụ

Ngày **17** là **ngày lẻ**, vì vậy chỉ những xe có **biển số kết thúc bằng số lẻ** mới được vào bãi.
Các xe thỏa điều kiện là **3285** và **3131**, tổng cộng **2 xe**.

```
def solution(day, numbers):
          count = 0
          for number in numbers:
                    if number%2 != day%2:
                              count += 1
          return count

day = 17
numbers = [3285, 1724, 4438, 2988, 3131, 2998]
ret = solution(day, numbers)

print("solution 함수의 반환 값은", ret, "입니다.")
```

Câu 10

□ Mô tả bài toán

Tại bãi đỗ xe của cơ quan nhà nước, áp dụng **chế độ chẵn – lẻ theo biển số xe**.
Theo chế độ này:

- Xe có **chữ số cuối của biển số là số lẻ** chỉ được vào bãi vào **ngày lẻ**.
- Xe có **chữ số cuối của biển số là số chẵn** chỉ được vào bãi vào **ngày chẵn**.

Khi số ngày `day` , mảng `numbers` chứa **biển số các xe muốn vào bãi trong ngày đó**, và độ dài mảng là `numbers_len` được cho làm tham số, hãy viết hàm `solution` để **return số lượng xe có thể vào bãi đỗ**.
Tuy nhiên, do **một phần của đoạn mã bị sai**, nên chương trình **không hoạt động đúng**.
Hãy **chỉ thay đổi đúng một dòng trong đoạn mã đã cho** để chương trình hoạt động chính xác với **mọi dữ liệu đầu vào**.

Lưu ý: **Không có biển số xe bị trùng lặp**.

□ Giải thích tham số

- `day` : ngày trong tháng, là số tự nhiên từ **1 đến 31**.
- `numbers` : mảng chứa biển số các xe muốn vào bãi trong ngày đó.
- `numbers_len` : độ dài của mảng `numbers` .

Điều kiện:

- `numbers_len` là số tự nhiên từ **1 đến 9.000**.
- Các phần tử của `numbers` là số tự nhiên từ **1.000 đến 9.999**.
- Trong mảng `numbers` **không có biển số trùng lặp**.

□ Giải thích giá trị trả về

Hãy **return số lượng xe có thể vào bãi đỗ xe**.

□ Ví dụ

day	numbers	numbers_len	return
17	[3285, 1724, 4438, 2988, 3131, 2998]	6	2

□ Giải thích ví dụ

Ngày **17** là **ngày lẻ**, vì vậy chỉ những xe có **biển số kết thúc bằng số lẻ** mới được vào bãi.
Các xe thỏa điều kiện là **3285** và **3131**, tổng cộng **2 xe**.

```
def solution(arr):
          answer = 0
          for i in arr:
                    for i/2 in arr:
                              answer += 1
          return answer

arr = [4, 8, 3, 6, 7]
ret = solution(arr)

# [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
print("solution 함수의 반환 값은", ret, "입니다.")
```