Đề thi CosPro Level 2

Câu 1

□ **Mô tả bài toán**

Có 10 học sinh, mỗi ngày lần lượt từng người một đến phòng tư vấn để được tư vấn. Việc tư vấn chỉ có thể diễn ra khi thầy/cô tư vấn có mặt. Ngoài ra, mỗi học sinh chỉ đến phòng tư vấn **một lần duy nhất**, không quay lại lần thứ hai.

Khi có lịch làm việc của thầy/cô, ta muốn biết **những học sinh nào không được tư vấn**.
Trong lịch làm việc:

- Ngày có thầy/cô được ký hiệu là `"O"`
- Ngày không có thầy/cô được ký hiệu là `"X"`

Ví dụ, nếu lịch của thầy/cô là
`["O", "X", "X", "O", "O", "O", "X", "O", "X", "X"]`
thì các học sinh số **2, 3, 7, 9, 10** sẽ không được tư vấn.

Khi mảng `schedule` chứa lịch làm việc của thầy/cô và độ dài của mảng `schedule_len` được truyền làm tham số, hãy viết hàm `solution` sao cho **trả về danh sách số thứ tự của các học sinh không được tư vấn**, được **sắp xếp theo thứ tự tăng dần**.
Hãy điền vào chỗ trống để hoàn thành toàn bộ đoạn mã.

□ **Giải thích tham số**

Mảng `schedule` chứa lịch làm việc của thầy/cô và độ dài của mảng `schedule_len` được truyền vào hàm `solution` làm tham số.

- Phần tử của `schedule` là `"O"` hoặc `"X"`
- `schedule_len` luôn bằng **10**

□ Giải thích giá trị trả về

Trả về danh sách số thứ tự của các học sinh **không được tư vấn**, được sắp xếp theo **thứ tự tăng dần**.

□ Ví dụ

schedule	schedule_len	return
["O", "X", "X", "O", "O", "O", "X", "O", "X", "X"]	10	[2, 3, 7, 9, 10]

□ Giải thích ví dụ

Vì thầy/cô chỉ có mặt tại phòng tư vấn vào các ngày **1, 4, 5, 6, 8**, nên các học sinh số **2, 3, 7, 9, 10** không được tư vấn.

```
def solution(schedule):
    answer = []
    for idx, i in enumerate(schedule):
        if i == @@@:
            answer.append(@@@)
    return answer

schedule = ["O", "X", "X", "O", "O", "O", "X", "O", "X", "X"]
ret = solution(schedule)

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

Câu 2

□ Mô tả bài toán

Ta muốn biết số người **đậu** bài kiểm tra thể lực. Các nội dung kiểm tra thể lực gồm: **gập bụng**, **chống đẩy**, **chạy**. Tiêu chuẩn đậu theo từng nội dung như sau:

- **Gập bụng:** từ **80 điểm** trở lên
- **Chống đẩy:** từ **88 điểm** trở lên
- **Chạy:** từ **70 điểm** trở lên

Nếu số nội dung vượt qua **ít hơn hoặc bằng 1**, hoặc có nội dung **không vượt quá một nửa điểm chuẩn đậu**, thì **rớt**.
Ngoài ra thì **đậu**.

Để tìm số người đậu, cấu trúc chương trình được 작성 như sau:

1. Đếm xem đã vượt qua bao nhiêu nội dung.
2. Đếm xem có bao nhiêu nội dung không vượt quá một nửa điểm chuẩn đậu.
3. Nếu số nội dung vượt qua nhiều hơn 1 và không có nội dung nào không vượt quá một nửa điểm chuẩn đậu thì tính là người đậu.

Khi mảng `scores` chứa điểm của từng nội dung và độ dài theo hàng của `scores` là `scores_len` được đưa vào làm tham số, ta muốn viết hàm `solution` để **trả về số người đậu**. Dựa theo cấu trúc trên, hãy điền vào chỗ trống các hàm `func_a`, `func_b`, `func_c` và các tham số sao cho mã có thể hoạt động đúng.

□ Giải thích tham số

Mảng `scores` chứa điểm của từng nội dung và độ dài theo hàng của `scores` là `scores_len` được truyền vào hàm `solution` làm tham số.

- Trong `scores` lần lượt chứa điểm **gập bụng**, **chống đẩy**, **chạy**
- Phần tử của `scores` là số nguyên từ **0 đến 100**
- `scores_len` là số nguyên từ **0 đến 10,000**

□ Giải thích giá trị trả về

Trả về **số người đậu** bài kiểm tra.

□ Ví dụ

scores	scores_len	return
Ví dụ #1: 30, 40, 100], [97, 88, 95	2	1
Ví dụ #2: 90, 88, 70], [85, 90, 90], [100, 100, 70], [30, 90, 80], [40, 10, 20], [83, 88, 80	6	4

□ **Giải thích ví dụ**

Ví dụ #1

Người thứ 1 gập bụng 30 điểm, chống đẩy 40 điểm nên không vượt qua một nửa điểm chuẩn đậu, vì vậy rớt.

Người thứ 2 gập bụng 97 điểm, chống đẩy 88 điểm, chạy 95 điểm nên vượt qua tất cả, vì vậy đậu. Do đó tổng cộng có **1 người** đậu.

Ví dụ #2

Người thứ **1, 2, 3, 6** thỏa mãn tiêu chuẩn đậu nên tổng cộng có **4 người** đậu.

```python
def func_a(passed, non_passed):
    return (passed > 1 and non_passed == 0)


def func_b(scores):
    answer = 0
    if scores[0] < 40:
        answer += 1
    if scores[1] < 44:
        answer += 1
    if scores[2] < 35:
        answer += 1
    return answer


def func_c(scores):
    answer = 0
    if scores[0] >= 80:
        answer += 1
    if scores[1] >= 88:
        answer += 1
    if scores[2] >= 70:
        answer += 1
    return answer


def solution(scores):
    answer = 0
    for my_score in scores:
        passed = func_@@@(@@@)
        non_passed = func_@@@(@@@)
        answer += func_@@@(@@@, @@@)
    return answer


scores1 = [[30, 40, 100], [97, 88, 95]]
ret1 = solution(scores1)
print("solution 함수의 반환 값은", ret1, "입니다.")

scores2 = [[90, 88, 70], [85, 90, 90], [100, 100, 70],
           [30, 90, 80], [40, 10, 20], [83, 88, 80]]
ret2 = solution(scores2)
print("solution 함수의 반환 값은", ret2, "입니다.")
```

Câu 3

□ Mô tả bài toán

Khi A và B chơi trò chơi bài, ta muốn biết **ai giành được nhiều điểm hơn**, và **số điểm giành được là bao nhiêu**. Luật chơi như sau:

- Có một bộ bài gồm các lá bài ghi các chữ cái **a, b, c, d, e**.
- A và B luân phiên nhau, mỗi lần rút **n lá**.
- Cộng tổng điểm của các lá bài sở hữu. Điểm của mỗi lá bài chữ cái được tính như sau: **a = 1 điểm, b = 2 điểm, c = 3 điểm, d = 4 điểm, e = 5 điểm**.
- Người có điểm cao hơn sẽ chiến thắng.

Để tìm người thắng và điểm đạt được trong trò chơi, cấu trúc chương trình được 작성 như sau:

1. A và B lần lượt lấy các lá bài và đưa vào 각각 các mảng.
2. Tính điểm mà A và B lần lượt đạt được.
3. Đưa người có điểm lớn hơn và điểm đạt được vào mảng theo thứ tự và return.

Khi số lượng lá bài cần rút `n`, mảng ký tự biểu diễn các lá bài `bundle`, và độ dài của `bundle` là `bundle_len` được đưa vào làm tham số, ta muốn viết hàm `solution` để **return người thắng và điểm**. Dựa theo cấu trúc trên, hãy điền vào chỗ trống các hàm `func_a`, `func_b`, `func_c` và các tham số phù hợp để mã hoạt động đúng.

□ Giải thích tham số

Số lượng lá bài cần rút `n`, mảng ký tự biểu diễn các lá bài `bundle`, và độ dài của `bundle` là `bundle_len` được truyền vào hàm `solution` làm tham số.

- `n` là số tự nhiên từ **1 đến 25**
- Tất cả phần tử của `bundle` là chữ cái thường
- `bundle_len` là số nguyên từ **2 đến 2*n**

□ Giải thích giá trị trả về

Return **người thắng và điểm đạt được**.
Nếu A thắng thì return **1**, nếu B thắng thì return **2**, nếu hòa thì return **0**.

□ Ví dụ

n	bundle	bundle_len	return
4	"cacdbdedccbb"	12	[0, 13]

□ Giải thích ví dụ

A đã rút các lá bài **c, c, b, e** và B đã rút các lá bài **a, d, d, d**.
Điểm A đạt được là **13 điểm** và điểm B đạt được cũng là **13 điểm**.
Cả hai đều đạt **13 điểm** nên trò chơi kết thúc với **kết quả hòa**.

```python
def func_a(bundle, start):
    return bundle[start::2]


def func_b(score1, score2):
    if score1 > score2:
        return [1, score1]
    elif score2 > score1:
        return [2, score2]
    else:
        return [0, score1]


def func_c(bundle):
    answer = 0
    score_per_cards = {
        'a': 1,
        'b': 2,
        'c': 3,
        'd': 4,
        'e': 5
    }

    for card in bundle:
        answer += score_per_cards[card]
    return answer


def solution(n, bundle):
    a_cards = func_a(@@@, @@@)[:n]
    b_cards = func_a(@@@, @@@)[:n]

    a_score = func_c(@@@)
    b_score = func_c(@@@)

    return func_b(@@@, @@@)


n = 4
bundle = "cacdbdedccbb"
ret = solution(n, bundle)
```

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

Câu 4

□ Mô tả bài toán

Khi tiến hành đồng thời **n** lớp học lập trình, ta muốn biết **số lượng trợ giảng cần thiết**. Một trợ giảng phụ trách **m** học sinh.

Khi mảng `classes` chứa số học sinh theo từng lớp, độ dài của `classes` là `classes_len` , và số học sinh mà 1 trợ giảng phụ trách là `m` được đưa vào làm tham số, ta muốn viết hàm `solution` để **return số trợ giảng cần thiết để tổ chức lớp học**. Hãy điền vào chỗ trống để hoàn thành toàn bộ đoạn mã.

□ Giải thích tham số

Mảng `classes` chứa số học sinh theo từng lớp, độ dài của `classes` là `classes_len` , và số học sinh mà 1 trợ giảng phụ trách là `m` được truyền vào hàm `solution` làm tham số.

- Phần tử của `classes` là số nguyên từ **0 đến 1,000**
- `classes_len` là số tự nhiên từ **1 đến 1,000**
- `m` là số tự nhiên từ **1 đến 1,000**

□ Giải thích giá trị trả về

Return **số trợ giảng cần thiết để tiến hành lớp học**.

□ Ví dụ

classes	classes_len	m	return
[80, 45, 33, 20]	4	30	8

□ **Giải thích ví dụ**

Số học sinh mỗi lớp lần lượt là **80, 45, 33, 20** và một trợ giảng phụ trách **30** học sinh.
Số trợ giảng cần cho từng lớp lần lượt là **3, 2, 2, 1**, nên tổng cộng cần **8** trợ giảng.

```
def solution(classes, m):
    answer = 0
    for students in classes:
        answer += students @@@ m
        if students @@@ m != 0:
            answer += 1
    return answer


classes = [80, 45, 33, 20]
m = 30
ret = solution(classes, m)

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

Câu 5

□ **Mô tả bài toán**

Có một bảng thực đơn ghi lại **lượng calo**.
Anh/chị A đang ăn kiêng, nếu lượng calo ăn vào trong ngày **lớn hơn giá trị nhỏ nhất** trong số lượng calo đã ăn trước đó thì sẽ tập thể dục để **tiêu hao đúng phần chênh lệch đó**.

Ví dụ, nếu lượng calo của thực đơn là `[713, 665, 873, 500, 751]` thì đến hết ngày thứ hai sẽ **không tập thể dục**.
Ngày thứ ba, lượng calo là **873** và giá trị nhỏ nhất của các ngày trước đó là **665**, nên sẽ tập thể dục để tiêu hao **208 calo**.
Ngày thứ tư không tập thể dục, và ngày thứ năm lượng calo là **751**, giá trị nhỏ nhất của các ngày trước đó là **500**, nên sẽ tập thể dục để tiêu hao **251 calo**.
Do đó, **tổng lượng calo tiêu hao khi tập thể dục là 459**.

Khi mảng `calorie` chứa lượng calo của thực đơn và độ dài của mảng là `calorie_len` được đưa vào làm tham số, ta đã viết hàm `solution` để **return tổng lượng calo tiêu hao khi tập thể dục**. Tuy nhiên, vì **một phần của mã bị sai**, nên với một số dữ liệu đầu vào, chương trình **không hoạt động đúng**.
Hãy **chỉ thay đổi một dòng** trong đoạn mã đã cho để chương trình hoạt động đúng với **mọi dữ liệu đầu vào**.

□ Giải thích tham số

Mảng `calorie` chứa lượng calo của thực đơn và độ dài của mảng là `calorie_len` được truyền vào hàm `solution` làm tham số.

- Mỗi phần tử của `calorie` là số tự nhiên **lớn hơn 0 và nhỏ hơn hoặc bằng 1,000**
- `calorie_len` là số tự nhiên **lớn hơn 0 và nhỏ hơn hoặc bằng 100**

□ Giải thích giá trị trả về

Return **tổng lượng calo tiêu hao khi tập thể dục**.

□ Ví dụ

calorie	calorie_len	return
[713, 665, 873, 500, 751]	5	459

□ Giải thích ví dụ

Giống với ví dụ trong đề bài.

```python
def solution(calorie):
    min_cal = 0
    answer = 0
    for cal in calorie:
        if cal > min_cal:
            answer += cal - min_cal
        min_cal = min(min_cal, cal)
    return answer


calorie = [713, 665, 873, 500, 751]
ret = solution(calorie)

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

Câu 6

□ Mô tả bài toán

Ta muốn **sử dụng số điểm tích lũy một cách tối đa**. Quy tắc sử dụng điểm như sau:

- Điểm được sử dụng theo **đơn vị 100 điểm**
- Chỉ khi có **từ 1000 điểm trở lên** mới có thể sử dụng điểm

Khi số điểm tích lũy `point` được đưa vào làm tham số, ta đã viết hàm `solution` để **return số điểm tối đa có thể sử dụng**. Tuy nhiên, do **một phần của mã bị sai**, nên với một số dữ liệu đầu vào, chương trình **không hoạt động đúng**.
Hãy **chỉ thay đổi một dòng** trong đoạn mã đã cho để chương trình hoạt động đúng với **mọi dữ liệu đầu vào**.

□ Giải thích tham số

Số điểm tích lũy `point` được truyền vào hàm `solution` làm tham số.

- `point` là số nguyên từ **0 đến 1,000,000**

□ **Giải thích giá trị trả về**

Return **số điểm tối đa có thể sử dụng**.

□ **Ví dụ**

point	return
2323	2300

□ **Giải thích ví dụ**

Nếu số điểm tích lũy là **2323 điểm** thì số điểm có thể sử dụng tối đa là **2300 điểm**.

```
def solution(point):
    if point < 1000:
        return 0
    return point * 100 // 100


point = 2323
ret = solution(point)

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

Câu 7

□ **Mô tả bài toán**

Có điểm thi giữa kỳ và điểm thi cuối kỳ. Dựa vào đó, ta muốn tìm **độ chênh lệch điểm** của học sinh có điểm **tăng nhiều nhất** và học sinh có điểm **giảm nhiều nhất**. Để làm điều này, cấu trúc chương trình được 작성 như sau:

1. Với mỗi học sinh, tìm **giá trị lớn nhất** của (điểm cuối kỳ trừ điểm giữa kỳ).
2. Với mỗi học sinh, tìm **giá trị nhỏ nhất** của (điểm cuối kỳ trừ điểm giữa kỳ).
3. Đưa các điểm tìm được ở bước 1 và bước 2 vào mảng và return.

Ví dụ, nếu điểm giữa kỳ là `[20, 50, 40]` và điểm cuối kỳ là `[10, 50, 70]` thì chênh lệch điểm của học sinh là `[-10, 0, 30]` .
Do đó, học sinh có thành tích tăng nhiều nhất đã tăng **30 điểm**. Và học sinh có thành tích giảm nhiều nhất đã giảm **10 điểm**.

Mảng `mid_scores` chứa điểm giữa kỳ theo thứ tự mã số học sinh, độ dài của `mid_scores` là `mid_scores_len` , mảng `final_scores` chứa điểm cuối kỳ theo thứ tự mã số học sinh, độ dài của `final_scores` là `final_scores_len` được đưa vào làm tham số của hàm `solution` . Lúc này, ta đã viết hàm `solution` để return một mảng trong đó **phần tử thứ nhất** là độ chênh lệch điểm của học sinh tăng nhiều nhất, và **phần tử thứ hai** là độ chênh lệch điểm của học sinh giảm nhiều nhất. Tuy nhiên, do **một phần của mã bị sai**, nên với một số dữ liệu đầu vào, chương trình **không hoạt động đúng**. Hãy **chỉ thay đổi một dòng** trong đoạn mã đã cho để chương trình hoạt động đúng với **mọi dữ liệu đầu vào**.

□ Giải thích tham số

Mảng `mid_scores` chứa điểm giữa kỳ theo thứ tự mã số học sinh, độ dài của `mid_scores` là `mid_scores_len` , mảng `final_scores` chứa điểm cuối kỳ theo thứ tự mã số học sinh, độ dài của `final_scores` là `final_scores_len` được truyền vào hàm `solution` làm tham số.

- Điểm thi là số tự nhiên **không quá 100**
- `mid_scores_len` và `final_scores_len` là số tự nhiên từ **5 đến 50**
- `mid_scores_len` và `final_scores_len` **bằng nhau**

□ Giải thích giá trị trả về

Return một mảng trong đó phần tử thứ nhất là **độ chênh lệch điểm của học sinh tăng nhiều nhất**, và phần tử thứ hai là **độ chênh lệch điểm của học sinh giảm nhiều nhất**.
Nếu không có học sinh nào tăng điểm thì đặt **0** vào phần tử thứ nhất, và nếu không có học sinh nào giảm điểm thì đặt **0** vào phần tử thứ hai.

□ **Ví dụ**

mid_scores	mid_scores_len	final_scores	final_scores_len	return
[20, 50, 40]	3	[10, 50, 70]	3	[30, -10]

```
def func_a(scores1, scores2):
    answer = 0
    for score1, score2 in zip(scores1, scores2):
        answer = max(answer, score2 - score1)
    return answer


def func_b(scores1, scores2):
    answer = 0
    for score1, score2 in zip(scores1, scores2):
        answer = min(answer, score1 - score2)
    return answer


# 한줄을 수정하세요.
def solution(mid_scores, final_scores):
    up = func_a(mid_scores, final_scores)
    down = func_b(mid_scores, final_scores)
    answer = [up, down]
    return answer


mid_scores = [20, 50, 40]
final_scores = [10, 50, 70]
ret = solution(mid_scores, final_scores)

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

Câu 8

□ **Mô tả bài toán**

Khi có kết quả bỏ phiếu cho các ứng cử viên từ **số 1 đến số n**, ta muốn tìm **số thứ tự của ứng cử viên giành được quá bán phiếu**. Ở đây, **quá bán** có nghĩa là **lớn hơn một nửa**.

Ví dụ, nếu kết quả bỏ phiếu cho các ứng cử viên từ 1 đến 3 là
`[1, 2, 1, 3, 1, 2, 1]`
thì điều này có nghĩa là lần lượt bỏ phiếu cho các ứng cử viên
[ứng cử viên 1, ứng cử viên 2, ứng cử viên 1, ứng cử viên 3, ứng cử viên 1, ứng cử viên 2, ứng cử viên 1].
Trong trường hợp này, ứng cử viên trúng cử là **ứng cử viên số 1**.

Khi số lượng ứng cử viên là `n`, mảng `votes` chứa kết quả bỏ phiếu, và độ dài của mảng là `votes_len` được đưa vào làm tham số, ta đã viết hàm `solution` để **return số thứ tự của ứng cử viên giành được quá bán phiếu**. Tuy nhiên, do **một phần của mã bị sai**, nên với một số dữ liệu đầu vào, chương trình **không hoạt động đúng**.
Hãy **chỉ thay đổi một dòng** trong đoạn mã đã cho để chương trình hoạt động đúng với **mọi dữ liệu đầu vào**.

□ Giải thích tham số

Số lượng ứng cử viên `n`, mảng `votes` chứa kết quả bỏ phiếu, và độ dài của mảng là `votes_len` được truyền vào hàm `solution` làm tham số.

- `n` là số tự nhiên từ **1 đến 100**
- Mỗi phần tử của `votes` là số tự nhiên từ **1 đến n**
- `votes_len` là số tự nhiên từ **1 đến 1,000**

□ Giải thích giá trị trả về

Return **số thứ tự của ứng cử viên giành được quá bán phiếu**.
Nếu **không có ứng cử viên nào đạt quá bán**, return **-1**.

□ Ví dụ

n	votes	votes_len	return	
Ví dụ #1	3	[1, 2, 1, 3, 1, 2, 1]	7	1
Ví dụ #2	2	[2, 1, 2, 1, 2, 2, 1]	7	2

□ **Giải thích ví dụ**

Ví dụ #1

Có 3 ứng cử viên từ số 1 đến số 3, kết quả bỏ phiếu là `[1, 2, 1, 3, 1, 2, 1]`.
Tổng số phiếu là 7, trong đó ứng cử viên số 1 nhận được 4 phiếu, ứng cử viên số 2 nhận được 2 phiếu, ứng cử viên số 3 nhận được 1 phiếu.
Trong trường hợp này, ứng cử viên số 1 đạt quá bán nên return **1**.

Ví dụ #2

Có 2 ứng cử viên từ số 1 đến số 2, kết quả bỏ phiếu là `[2, 1, 2, 1, 2, 2, 1]`.
Tổng số phiếu là 7, trong đó ứng cử viên số 1 nhận được 3 phiếu, ứng cử viên số 2 nhận được 4 phiếu.
Trong trường hợp này, ứng cử viên số 2 đạt quá bán nên return **2**.

```python
def solution(n, votes):
    answer = 0
    votes_len = len(votes)
    candidate = votes[0]
    count = 1
    for i in range (1, votes_len) :
        if candidate == votes[i] :
            count += 1
        else :
            count -= 1
            if count == 0 :
                candidate = votes[i]
                count = 1

    test_count = 0
    for i in range(0, votes_len) :
        if votes[i] == candidate :
            test_count += 1

    if test_count > votes_len // 2 :
        answer = candidate
    else :
        answer = -1

    return answer


n1 = 3
votes1 = [1, 2, 1, 3, 1, 2, 1]
ret1 = solution(n1, votes1)

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

n2 = 2
votes2 = [2, 1, 2, 1, 2, 2, 1]
ret2 = solution(n2, votes2)

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

Câu 9

□ Mô tả bài toán

Ta muốn biết trên một địa hình dạng lưới kích thước **4 x 4** có bao nhiêu **khu vực nguy hiểm**. Khu vực nguy hiểm là khu vực mà các khu vực kề cận theo **Đông, Tây, Nam, Bắc** đều có độ cao **lớn hơn** khu vực đó. Ví dụ, nếu độ cao các khu vực như bên dưới thì:

Vùng màu đỏ là khu vực nguy hiểm vì các khu vực kề cận đều cao hơn khu vực đó.

Khi mảng 2 chiều `height` chứa độ cao theo từng khu vực và độ dài của `height` là `height_len` được truyền vào làm tham số của hàm `solution` , hãy hoàn thành hàm `solution` để **return có bao nhiêu khu vực nguy hiểm**.

□ Giải thích tham số

Mảng 2 chiều `height` chứa độ cao theo từng khu vực và độ dài của `height` là `height_len` được truyền vào hàm `solution` làm tham số.

- Độ cao của mỗi khu vực là số tự nhiên từ **1 đến 50**
- `height_len` luôn bằng **4**

□ Giải thích giá trị trả về

Return **có bao nhiêu khu vực nguy hiểm**.

□ Ví dụ

height	height_len	return
3, 6, 2, 8], [7, 3, 4, 2], [8, 6, 7, 3], [5, 3, 2, 9	4	5

□ Giải thích ví dụ

Giống với ví dụ trong đề bài.

```
def solution(height):
    count = 0

    return count


height = [[3, 6, 2, 8], [7, 3, 4, 2], [8, 6, 7, 3], [5, 3, 2, 9]]
ret = solution(height)

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

Câu 10

□ Mô tả bài toán

Khi có điểm số của các thí sinh tham gia **kỳ thi XX**, ta muốn tìm **số lượng thí sinh đậu**. Để đậu kỳ thi, thí sinh phải đạt **điểm bằng hoặc cao hơn điểm chuẩn (cutline)**.

Ví dụ, nếu điểm của các thí sinh là `[80 điểm, 90 điểm, 55 điểm, 60 điểm, 59 điểm]` và điểm chuẩn là **60 điểm**, thì có **3 thí sinh đậu**.

Khi mảng `scores` chứa điểm thi của các thí sinh, độ dài của mảng là `scores_len` , và điểm chuẩn `cutline` được đưa vào làm tham số, hãy hoàn thành hàm `solution` để **return số lượng thí sinh đậu**.

□ Giải thích tham số

Mảng `scores` chứa điểm thi của các thí sinh và điểm chuẩn `cutline` được truyền vào hàm `solution` làm tham số.

- Mỗi phần tử của `scores` là điểm thi của một thí sinh, là số nguyên từ **0 đến 100**
- `scores_len` là số nguyên từ **1 đến 100**
- `cutline` là số nguyên từ **0 đến 100**

□ Giải thích giá trị trả về

Return **số lượng thí sinh đậu**.

□ Ví dụ

scores	scores_len	cutline	return
[80, 90, 55, 60, 59]	5	60	3

□ Giải thích ví dụ

- **80 điểm, 90 điểm, 60 điểm**: đậu
- **55 điểm, 59 điểm**: rớt

Do đó có **3 thí sinh đậu**.

```
def solution(scores, cutline):
    answer = 0

    return answer


scores = [80, 90, 55, 60, 59]
cutline = 60
ret = solution(scores, cutline)

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