Lovelace Hackathon 2024 Booklet

With thanks to: Vincent S, Alex H, Alex Y, Hayden L, Andrew P, Vinsson L, Richard Z, Dylan K, Dr C M Crowe

1 ENCODING A STRING - Green

Mr Hall wants to have some fun and has a 3-letter word in his head, made up of letters from the Latin alphabet. He encodes this word as a sum of the positions of all the characters on the alphabet (i.e. "a" has position 1, “z” has position 26 etc.) so, so for example, the word “cat” would be encoded as 3 + 1 + 20 = **24**. However, this method of encoding has turned out to be ambiguous! Another word, like “ava” also has the same integer of 1 + 22 + 1 = **24**. Given the integer, n (3 <= n <= 78), determine alphabetically, the smallest 3-letter word that could have been encoded.

Please note that the ‘word’ does not have to be an actual word, rather just a 3-character string like “sdf”.

You should write a function `solution` that takes an integer and returns a string of 3-letter word.

Example function definition

```
def solution(code: int) -> str:
    # code your stuff...
    return decoded_word
```

Example function calls

```
solution(24) = "aav"
```

```
solution(70) = "rzz"
```

2 LOOOOOOONG MONDAY MORNING SPEECH ROOM - Green

This Monday, the headmaster has read out names of boys in the speech room for them to collect their prizes. Each boy takes **G** seconds to get to the main pathway and has to walk **D** metres (**D ≥ 4**) to get to the headmasters after they entered the pathway. Each boy walks **1 ms^{-1}** when they are on the main pathway. The headmaster only calls the next boy when there is a boy is 4 metres away from him. A string will be provided with boys’ initial followed their constituent G and D separated by space (and repeat). You need to calculate the time (in seconds) taken from the start when the headmaster starts to the name of the first boy (assume that doesn’t take time) to the end when the last boy reaches the headmaster (**D and G are positive integers**).

You should write a function `solution` that takes a string of inital followed by G followed D followed by inital... separated with space and returns an integer that represents the total time.

Example function definition

```
def solution(plan: str) -> int:
    # code your stuff...
```

```
    return total_time
```

Example function calls

```
solution("AH 2 4 AY 3 4 AZ 1 6") = 12
```

```
solution("AH 1 5 AY 2 10 AZ 3 5 VS 13 4 VSCs 1 5") = 33
```

3 PRIME TIME - Green

The Harrow School maths department is challenging its computer science department to a duel, because they are debating which department is better in its ***prime time***. After three long days of arduous negotiation, they finally agreed on a tried-and-tested strategy to fairly evaluate their prime times. Namely, both departments shall compete to find the ***nth*** prime number, and the department with the better time (i.e. shorter time) wins. Of course, the mathematicians do everything by hand. But as a computer scientist, you have been tasked with writing a program to help computer science department win in this epic battle of prime times.

Given an integer input ***n***, you shourld write a function `prime(n)` which returns an integer `p`, where `p` is the n^{th} prime number($1 \le p \le 10^7$). As quickly as possible!!!

Hint: Use the Sieve of Eratosthenes

Example function definition

```
def prime(n: int) -> int:
    # code your stuff...
    return p
```

Example function calls

```
prime(1) = 2
```

```
prime(7) = 17
```

4 ROMAN NUMERALS PART I - Green

`Last updated at Feb 23 21:17pm GMT.`

Dr Kennedy would like to revisit a few pages from Herodotus' The Histories. However, the page numbers of the book all use Roman numerals! Please help him convert these Roman numeral page numbers into integers.

A reminder:

Roman numerals are represented by seven different symbols: I, V, X, L, C, D, M, representing 1, 5, 10, 50, 100, 500, 1000 respectively.

2 is written as II in Roman numeral - two ones added together. 27 is written as XXVII, which is XX + V + II.

Roman numerals are normally written largest to smallest from left to right. However, 4 is written as IV, not IIII. Similarly, 9 is written as IX.

This principle can apply to the following:

- I placed before V and X to form 4 and 9
- X placed before L and C to form 40 and 90
- C placed before D and M to form 400 and 900

You should write a function `solution` that takes an string of roman numeral and returns the constituent integer.

Example function definition

```
def solution(roman_numeral: str) -> int:
    # code your stuff...
    return translated_integer
```

Example function calls

```
solution("III") = 3
```

```
solution("LVIII") = 58
```

5 ROMAN NUMERALS PART II - Green

`Last updated at Feb 23 21:17pm GMT.`

After completing the conversion of Roman Numerals, a Shell boy notices and wonders what the strange strings of letters are. Dr Kennedy explains about Roman numerals, and the Shell boy would like to know how to convert from integers to Roman numerals.

Given the integer, `num` (0 < num < 4000), please help him do this.

You should write a function `solution` that takes an integer and returns the constituent string of roman numeral.

Example function definition

```
def solution(num: int) -> str:
    # code your stuff...
    return translated_roman_numerals
```

Example function calls

```
solution(8) = "VIII"
```

```
solution(43) = "XLIII"
```

6 TAPS AND SHOWERS - Green

The taps and showers in Harrow School have a self-closing design ("**non-concussive**"). Once the button has been pressed, it will stay on for **3** seconds before it turns off. Given the timestamps when the button has been pressed (in seconds), output the length of time water has been running. Note that to ensure a smooth experience, most students will refresh the countdown before the water stops. (Tired of this? Contact Vincent for a solution!!!)

You should write a function `solution` that takes a string of floats separated with space and returns a float which represents the total length of time that water has been running.

We will truncate your answer to 1 decimal place when marking.

Example function definition

```
def solution(time: str) -> float:
    # code your stuff...
    return total_time
```

Example function calls

```
solution("1.0001 5.2345 6.01") = 6.8
```

7 WINDOW FLOWERS - Green

Mrs Silcott wants to celebrate the Chinese New Year by making some window flowers with the lower years. To make a window flower, you have to fold a square paper multiple times before cutting some pattern on them. This allows the scissors to cut through multiple layers at the same time creating patterns with symmetries once expanded. To make everything simple and straightforward Mrs Silcott has only taught the lower years the easiest way of folding:

1. Fold the square down in half to make a rectangle.
2. Turn the rectangle 90 degrees anticlockwise.
3. Fold the rectangle down in half to make a smaller square.

You should write a function `solution` that takes pattern of the fold that has been cut already represented with X and O in a n*n grid in a single string (0 < n < 10, X represents empty space and O represents paper). The string is going to contain a number, n, representing the size of the grid followed by the information of the pattern of each row from top to bottom. Each row is separated from previous row with a single space. The function returns the expanded window flower in the same format (without the n).

Example function definition

```
def solution(pattern: str) -> str:
    # code your stuff...
    return expanded_pattern
```

Example function calls

```
solution("3 OXO OOO OOO") = "OOOOOO OOOOOO OXOOXO OXOOXO OOOOOO OOOOOO"
```

```
solution("2 XO OX") = "XOOX OXXO OXXO XOOX"
```

8 WORD OF THE DAY - Green

The English Department beaks wants a word of the day displayed on a wall. However, they just realised that they can’t display the word horizontally due to a lack of space. So, they will have to print the word vertically. You are tasked to make a program to convert the word from horizontal to vertical.

You should write a function `solution` that takes in a word and returns a string representation of the word in vertical format.

Example function definition

```python
def solution(word: str) -> str:
    # code your stuff...
    return vertical_word
```

Example function calls

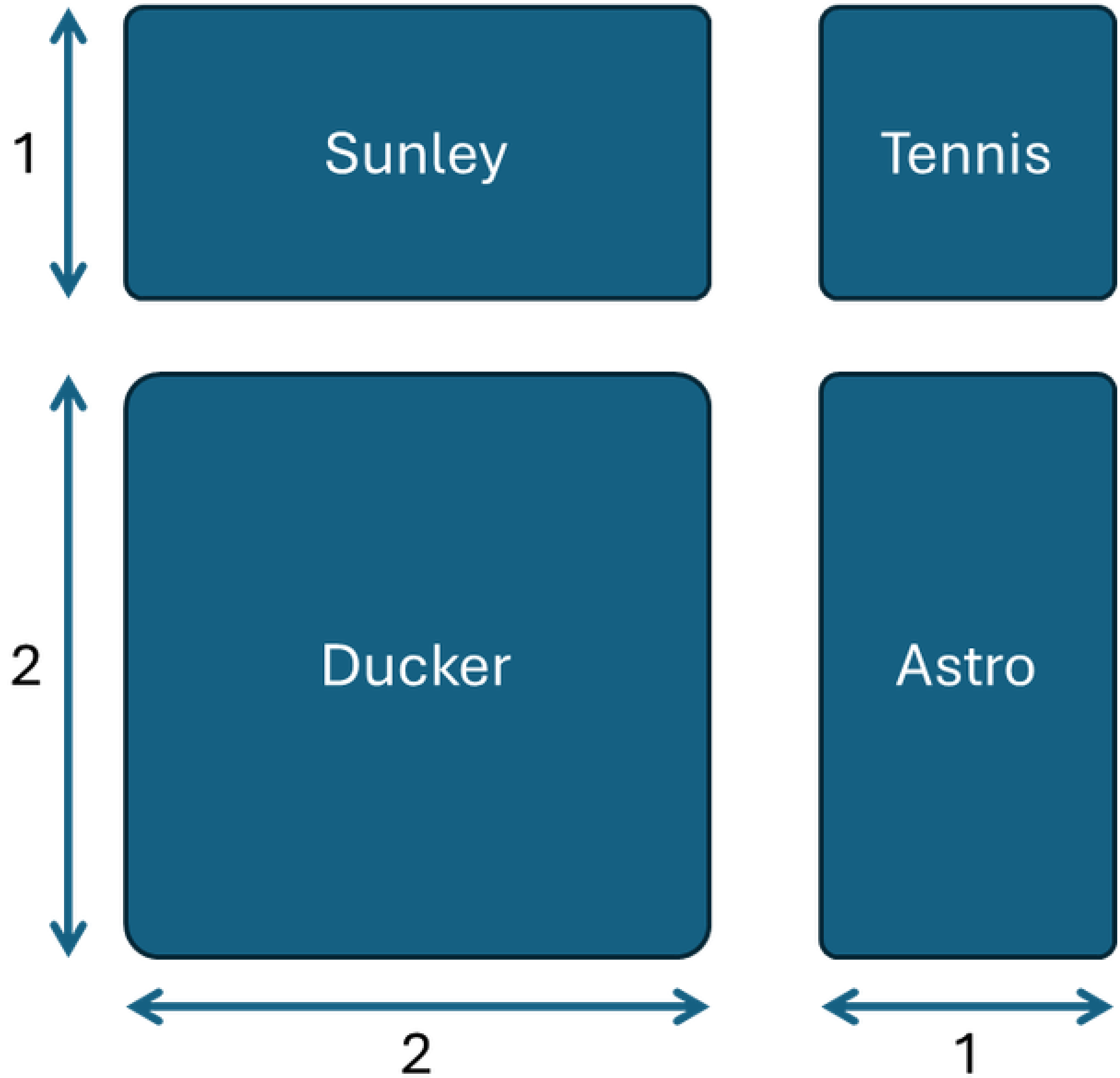
```python
solution("APPLE") = "A\nP\nP\nL\nE"
```

9 ACTIVITY GROUNDS - Red

Last updated at Feb 23 22:58pm GMT.

There are 4 types of activity grounds at Harrow School, with a letter assigned to each:

- T - the tennis courts, with a dimension of 1×1
- A - the astroturf, with a dimension of 1×2
- S - the Sunley field, with a dimension of 2×1
- D - the Ducker field, with a dimension of 2×2.



There can duplicates of each type of grounds at Harrow School. A map with 1×1 cells can be used to describe the configuration of the grounds. For example, the configuration above can be described as:

```
"""
S S T
D D A
D D A
"""
```

On the first day of school, one Shell found himself at the top left corner of the Harrow School Activity Grounds. He must get to the bottom right corner for Yearlings F training. Luckily, his housemaster supplied him with a compass to navigate the fields, so he would only go "down" or "right". For example, for the configuration above, he will start at S, and he could go right to arrive at T, and go down to arrive at A.

Given the map of the playing fields, write a function called `solution` to determine the number of different paths he could take to get to training.

Note: the map will always be rectangular.

Example function definition

```python
def solution(map: str) -> int:
    # code your stuff...
    return num_paths
```

Example function calls

```python
solution("T T T\nT T T\nT T T\n\n") = 6
solution("S S S S\nD D D D\nD D D D\n\n") = 2
```

10 EVEN MATCHMAKING - Red

Mrs Silcott is organising a team maths contest within LC-FM1 division. To avoid uneven match making, she wants to ensure that she assigns the teams such that the total sum of 'skill' of the team with the most total sum of 'skill' is minimised. She can determine the 'skill' of any student easily because she can see auras. Even though it is a team-based contest, there can be different amount of people per team. However, when the members of each team are lined up from left to right in lexicographical order, each pair of adjacent members in the line-up also needs to be adjacent in the name register, which is sorted lexicographically.

More formally, given an array of an integer array, and `k`. You must split the array into non-empty `k` subarrays such that the maximum sum of any subarray is minimised. You must find and return the minimised maximum sum.

Your task to implement a function named `even_matchmaking(k, A)`. `A` is an array of integers. The function should return the minimised maximum sum.

1 <= `k` <= min(100, `len(A)`)
1 <= `len(A)` <= 1e3
1 <= A_i <= 1e6.

Example function definition

```python
def even_matchmaking(k: int, A: list) -> int:
    # code your stuff...
    return minimised_maximum_sum
```

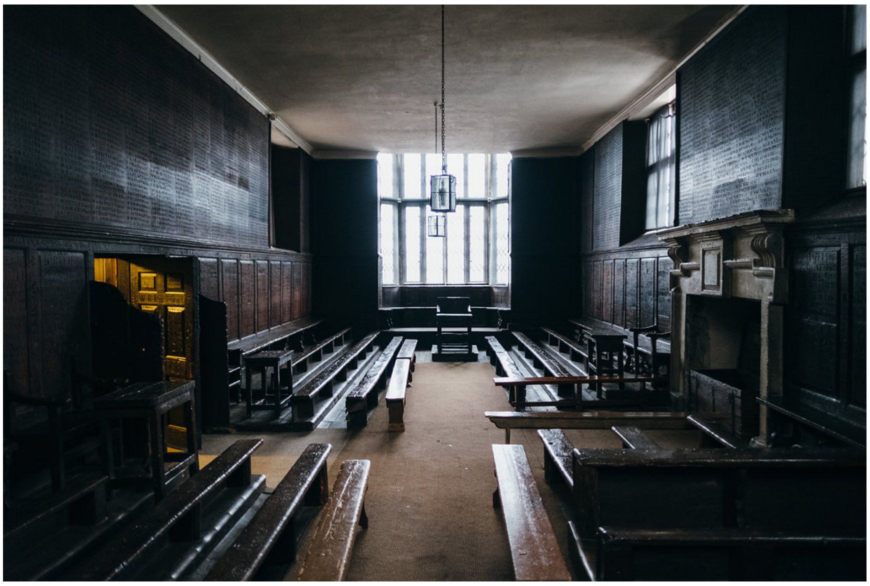
Example function calls

```
even_matchmaking(2, [7, 2, 5, 10, 8]) = 18
even_matchmaking(2, [1, 2, 3, 4, 5]) = 9
```

11 MAGIC FUNCTIONS - Red

Let's conjure up a bit of magic.

The Fourth Form Room at Harrow School is famous for being the site of the wingardium leviosa scene in Harry Potter and the Philosopher's Stone.



Above: The Fourth Form Room at Harrow (image source)

Above: The Wingardium Leviosia scene (image source)

Legend has it that deep beneath the wooden floor panels of the Fourth Form Room, two cursed sphinxes guard a heavily fortified vault. Within the vault lies a scroll, detailing how to counter the most powerful Dark Art known to the wizarding world: *DDos*.

Fearing the launch of a large-scale DDos attack from the wizards on its Lovelace server, the Harrow Computer Science Society deploys you to retrieve the scroll and find the counterspell. Having battled serpents, dementors, and rogue centaurs, you now stand in front of the vault. The two cursed sphinxes facing you are called Alice and Bob. Alice and Bob pose you a riddle:

> "If thou shalt query us with a number, each of us shall reply to thee with another number. Our thoughts are bound by some mysterious force unknown to thee. The vault can be unlocked by the number which bids us both to reply with the same number. Since thou art small and ignorant, we shall tolerate a most minuscule teaspoon of error, but thou shalt only have forty guesses."

In other words, for two unknown functions `f` and `g`, you need to approximate a value `x_0` such that `f(x_0) ≈ g(x_0)` within 40 guesses.

Write a function named `magic`. Given `f` and `g` as your input arguments, where `f` and `g` are callable single-variable functions in Python, your task is to return an approximate value `x_0` such that `f(x_0)≈g(x_0)`. You will only be allowed to pass in a maximum of 40 different values in total before returning the approximate `x_0` value. That tallies up across both functions. So, for example, you can run:

```
for i in range(1, 40):
    f(x)
    g(x)

f(40)
g(40)
```

because you are only querying 40 values in total across the two functions (1-39 and 40). But you can't run:

```
for i in range(1, 40):
    f(x)
    g(x)

f(40)
g(41)
```

because now you are querying 41 values in total across both functions (1-39, 40, and 41).

There are some important constraints to bear in mind:

- `f(x)` and `g(x)` are differentiable, real-valued, single-variable functions that have a domain of $0 \le x \le 10^5$.
- A value of x always exists within the domain such that `f(x)=g(x)`. Alice and Bob don't lie!
- `f'(x)≠g'(x)` for all x within the domain. I.e. Their derivatives are never equal.

- The functions `f(x)` and `g(x)` will be implemented using only standard Python arithmetic operations (`+` , `-` , `*` , `/` , `**`) between the input variable and some constants. The constants will all have type `float` .

Your output will be truncated to 7 decimal places when marking the solution.

Stuck? Don't worry! We've got a few hints.

Hint 1: Do you know any algorithms for root-finding?

Hint 2: Differentiation using the chain rule! Any complicated function implemented in Python can be numerically differentiated via the chain rule, right? After all, it's just a sequence of `+` , `-` , `*` , `/` , `**` .

Hint 3: How do you implement numerical differentiation of an arbitrary function? Well, more magic! Look up ***magic methods*** *in Python!*

The key is to realise that while the functions f and g are unknown before runtime, you can code a custom object with Python magic methods to pass into these functions, tracking the operations applied to them in the function in order to compute some desired properties.

Example function definition

```
def magic(f, g) -> float:
    # do your magic to compute x_0
    return x_0
```

Example function calls

```
def f(x):
  return 3.125 - 0.5 * (x**2.1)

def g(x):
  return 0.8 * (x**3.4) + 1.2 * x

magic(f, g) = 1.1150464
```

12 MUSEUM QUEUE - Red

The OSRG society is going on a trip to the Victoria and Albert Museum. There are `N` people, and they each have their respective ID attached to them `[1, N]` . To enter, the museum requires that the people line up 1 to `N` in order from left to right. Given the current line up formation, determine the minimum number of 'moving about' you have to do to get them in order.

'Moving about' is defined as when you choose one person and move them to a different position, effectively inserting them into a new position.

Your task is to implement a function `museum_queue(A)` , where `A` is an array denoting the current line up formation containing `len(A)` distinct integers from 1 to `len(A)` inclusive. The function should return a single integer denoting the minimum number of 'moving about' needed to get in order.

1 <= `len(A)` <= 200

Example function definition

```
def museum_queue(A: list) -> int:
    # code your stuff...
    return minimised_moves
```

Example function calls

```
museum_queue([3, 7, 5, 2, 6, 1, 4]) = 4
museum_queue([1, 2, 3, 4]) = 0
```

13 NUMBER SEQUENCE - Red

A sequence of numbers is found written on a whiteboard in a Maths school classroom. The students, being on a short break in between their double lesson decide to play a game and try add the symbols '+', '-', and ' ' (blank representing a join between the two adjacent digits) in order to make the sequence sum to 0.

Write a function called `solution` . Its input is an integer `N` (3 ≤ `N` ≤ 9). Considering the sequence of numbers 1 through `N` in increasing order (1, 2, 3, ... `N`), return a list of all possible sequences as strings constructed in the way above that will produce a zero sum.

Example function definition

```
def solution(N: int) -> list[str]:
    # code your stuff...
    return answer
```

Example function calls

```
solution(4) = ["1-2-3+4"]
```

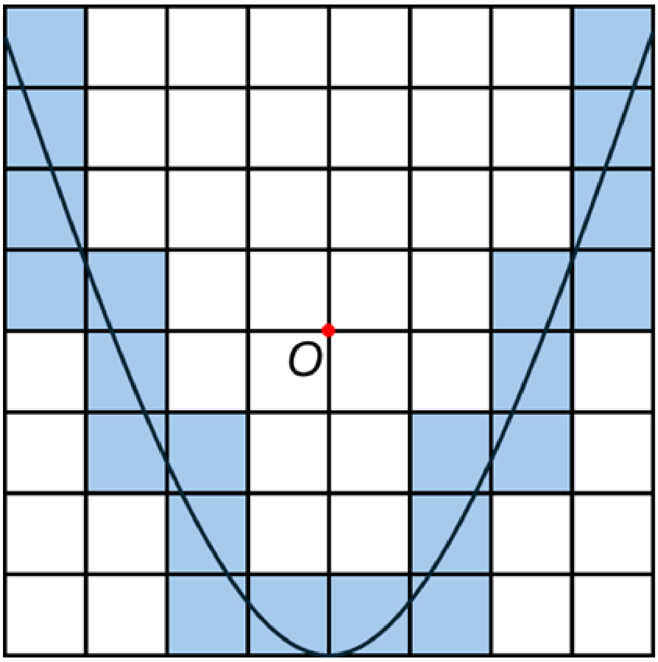
```
solution(7) = ["1+2-3+4-5-6+7", "1+2-3-4+5+6-7", "1-2 3+4+5+6+7", "1-2 3-4 5+6 7", "1-2+3+4-5+
```

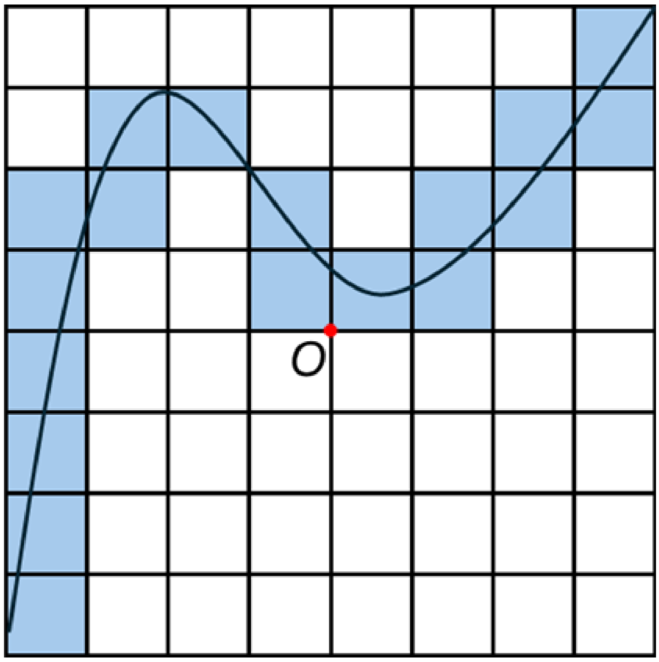
14 PIXELATION - Red

`Last updated at Feb 23 21:17pm GMT.`

Have you ever been troubled by low-resolution, pixelated images? This is because all digital devices must make approximations when processing or storing infinitely detailed signals. In this question, you are tasked with approximating polynomials.

With the Taylor series, it is possible to approximate any function with polynomials, so being able to approximate polynomials means you can approximate any function.





In this question, only polynomials of **degree 3 and below** will be considered.

You should write a function called `solution` to determine how many unit squares will the curve of the polynomial pass through in the range `-4 < x < 4`.

The input will contain the coefficients of the polynomial in **ascending** powers of `x`, starting at x^0.

Note: if the curve sits precisely on the vertex of a unit square it does not count as passing through it. The curve will always pass through a finite number of squares.

Example function definition

```
def solution(p0: float,
             p1: float,
             p2: float,
```

```
            p3: float) -> int:
    # code your stuff...
    return num_unit_squares
```

Example function calls

```
solution(0, -0.16, 0, 0.01) = 8
```

15 SPRAYING GRASS - Red

After a physics experiment accident in Dr Unwin's classroom, the radioactive explosion which splashed onto the mini-garden right outside caused the grass there to mutate and become radioactive. The grass now had two main states: dead or (too) alive. The former meant that the grass was shrivelled up with no signs of life – very unappealing for visitors! Conversely, the latter meant that the grass was too alive (some of them have been even observed to be growing limbs or dancing stationarily in a chaotic and headless fashion), almost like a group of shells after discovering the power of caffeine.

Now, your job is to restore the grass into their normal state. (This is your punishment for missing too many custodes!)

To complete this task, you have been given two very similar magical yet somewhat inconvenient hoses which you can control the power level of, with diminishing effects when the target is further away. The first hose adds life to the grass; the second hose takes away life to the grass. You have to use these two types of hoses in a clever manner so that the life value of all of the grass is 0. More formally, there are N patches of grass which needs fixing, and you are fixed immobile at patch `N`, the right most patch (This is because custos wants to make your punishment harder). When you choose a power level of `L`, and use the first hose, `L` amount of life value will be added to patch `N`, `L-1` amount of life will be added to patch `N-1`, and so on... the second hose has the opposite effect as you can imagine. Note that patch to patch `N-L` won't be affected at all.

Since you are a lazy shell (also why you missed too many custodes), you want to minimise the number of times you have to use the hose.

Write a function named `spraying_grass(A)` which returns the minimum number of times you have to use the hose. `A` is an array which contains all the initial values, with A_i denoting the initial life value of patch `i` of the grass.

1 <= `len(A)` <= 2e5

Example function definition

```python
def spraying_grass(A: list) -> int:
    # code your stuff...
    return minimum_num
```

Example function calls

```
solution([-1, 3]) = 6
solution([6, 0]) = 18
```

16 TRACING INDICES - Red

Test Cases updated 22:44 23/02/24 Andrew has an array, `a`, which is initially empty. He performs `n` operations of two types on the array.

1st type : Andrew appends an integer `k` (1 <= `k` <= `n`) to the end of array 2nd type: Andrew appends `k` copies of `a` to the end of the `a`, in other words, array `a` becomes `[a for _ in range(k+1)]`. It is guaranteed that at least one operation of the 1st type has been done before this.

Given the description of the `n` operations he did, answer `q` queries. For each query q_i, you must tell him the q_ith element of array `a`. The array is base-1.

Your task is to implementation a function named `trace_indices(N, Q)` which should return a list that contains the answer to each query in order. `N` is a two-dimensional array with N_i being a list with two elements `[o, k]`, denoting that the ith operation is of type `o`, and uses `k`. `Q` is an one-dimensional integer array which contains all of the queries.

1 <= `len(N)` <= 1e5
1 <= `len(Q)` <= 1e5
`o` ∈ {1, 2}
1 <= Q_i <= min(1e15, `v`), where `v` is the size of Andrew's array after processing `N`

Example function definition

```python
def trace_indices(N: list, Q: list) -> list:
    # code your stuff...
    return answer
```

Example function calls

```
trace_indices([[1, 3], [1, 5], [2, 4], [2, 2], [1, 2]], [29, 10, 8, 26]) = [3, 5, 5, 5]
```

```
trace_indices([[1, 5], [2, 3], [2, 5], [1, 2], [2, 5]], [129, 24, 95, 6, 133]) = [5, 5, 5, 5,
```

17 BREAKFAST ROLLS - Yellow

After a recent food survey sent out to boys at Harrow, it was found that the most popular item served at breakfast were, by an overwhelming majority, pain-au-chocolats. Therefore, a meeting was held in the Shepherd Churchill where it was decided that a new, but very similar, pastry would be rolled out to be served at breakfast. The whole point of this, of course, was to maximise the coverage of chocolate over pastry, which is, with no doubt, the main concern of breakfasting Harrovians.

The new pastries are designed to be in a spiral shape, much like that of a snail shell. These rolls come in different sizes, and are square when viewed from above. The chocolate is poured in a thin layer over the roll in the form of a spiral and around the bun, as in the following picture:

For a roll of size `n` , the length of the outer side of the roll is n, and the shortest vertical chocolate segment in the centre is 1.

When these rolls are made, two spirals are wrapped around each other, separated by chocolate. A chocolate roll of size `n+1` is obtained by wrapping each of the dough spirals around the bun for another layer.

Write a function called `solution` to help the kitchen staff determine how much chocolate is in a cinnamon roll of size `n` ($4 \le n \le 10^9$).

Example function definition

```python
def solution(n: int) -> int:
    # code your stuff...
    return answer
```

Example function calls

```
solution(4) = 26
solution(5) = 37
```

18 ALPINE ANALYST - Yellow

It is now holidays, and Hayden loves skiing. Having tried out all the regular routes, he decides to ski down multiple off-piste runs. However, he wants to have an idea of a run's **difficulty** before skiing down them. Given a list of altitudes at regular intervals across a slope, help Hayden write a function `solution` that evaluates the difficulty of a ski run by identifying the **50**-meter segment with largest average steepness. Return the **product** of the average steepness of this segment and number of troughs in it as the difficulty rating of the entire run.

Your function should take one input:

- `topography` : A one-dimensional array of integers representing the altitude (in meters) at regular intervals of **1** meter from the start to the end of the slope.

Note that:

- The steepness between 2 adjacent points is the modulus of their gradient.
- The average steepness of a 50-meter segment is the average steepnesses of the 49 pairs of adjacent points.
- A point is a trough when its altitude is lower than that of its adjacent points.
- If there are multiple 50-meter segments with the same value for their maximum average steepness, calculate the difficulty rating with the number of troughs as the largest one among these 50-meter segments.
- ***Round the overall slope grade to two decimal places.***

Constraints

- The `topography` arrays will contain at least **50** elements.
- All values in `topography` will be positive integers from **0** to **3300**.

Example function definition

```python
def solution(topography: list[int]) -> float:
    # code your stuff...
    return difficulty
```

Example function calls

```
solution([2241, 2239, 2239, 2234, 2231, 2228, 2227, 2219, 2220, 2219, 2202, 2197, 2195, 2190,
```

19 BUILDING ID - Yellow

In each Harrow School building, there is a corresponding `ID` number, **9 < ID < 10^1000**. The IT department in Harrow, for no reason at all, decided to find the sum of the digits of each ID number until it reduces to a single digit, perhaps they were just bored. Your task is to devise an efficient algorithm in the function `solution` to aid the IT department in their task.

Example function definition

```python
def solution(ID: int) -> int:
    # code your stuff...
    return output
```

Example function calls

```
solution(10320) = 6
solution(1239128391391834732) = 7
```

20 DANCEFLOOR - Yellow

Dylan stands at the entrance of a vibrant dancefloor, the final obstacle before his initiation into the enigmatic Harrow Secret Society. The dancefloor is no ordinary space; it's a grid of illuminated tiles, each radiating with a unique color represented by an RGB value. Dylan's challenge is to navigate from the top-left corner of the dancefloor to the VIP section located at the bottom-right corner, choosing his steps wisely to minimize the total color change between consecutive tiles he steps on.

Given the grid of tiles represented as a list of lists of color values, where each value is a tuple in the form of `(R, G, B)`, write a function `solution` to find a path from the top-left corner `dancefloor[0][0]` to the bottom-right corner `dancefloor[n-1][m-1]` of the dancefloor with the **minimum total "color change"**, returning this value in your function.

The "color change" between two adjacent pixels is defined as the **Euclidean distance between their RGB values treated as 3D coordinates, rounded down to nearest integer**. For example, the adjacent tiles with RGB values (255, 254, 253) and (3, 2, 1) has a "color change" of **436**. Same as finding the Euclidean distance between two 3D points with coordinates (255, 254, 253) and (3, 2, 1).

Note that Dylan can only move in four directions from any tile: left, right, up, or down, but cannot go diagonally.

Your function should take 3 inputs:

- `n` : the number of rows of tiles,
- `m` : the number of columns of tiles,
- `dancefloor` : a list of lists, where each inner list represents a row of tiles on the dancefloor, and each tile is represented as a tuple `(R, G, B)` indicating the RGB color of the tile.

Constraints

- `2 ≤ n, m ≤ 50`
- The RGB values are integers between `0` and `255`, inclusive.

Example function definition

```
def solution(n: int, m: int, dancefloor:list[list[tuple[int, int, int]]]) -> int:
    # code your stuff...
    return totalcolorchange
```

Example function calls

```
solution(3, 3, [
    [(255, 0, 0), (255, 255, 0), (0, 255, 0)],
    [(0, 0, 255), (255, 255, 255), (0, 255, 255)],
    [(0, 0, 0), (255, 0, 255), (0, 0, 255)]
]) = 1020
```

21 HATS - Yellow

Recently, our Custos (uniform inspector) has been struggling to catch pupils without their hats on in the High Street. However, a clever chap from the Maths Society realised that whenever there is an even number of pupils, the number of students with hats and without hats are **primes**.

Write a function `solution` that finds all the combinations the pupils can have their hats on or off to give Custos a guideline when there are an even number of students, and return the total number of possible combinations.

The input `N` (number of students) will always be even. (2<N<100000)

For example, when `N` = 30, there are six combinations: (13+17), (7+23), (11+19) and their reversed combinations. So, the output would be 6.

Example function definition

```
def solution(N: int) -> int:
    # code your stuff...
    return output
```

Example function calls

```
solution(120) = 24
```

22 OLDLANDS-GPS - Yellow

`Last updated at Feb 23 23:47pm GMT.`

Oldlands, a house in Harrow School UK, has been corrupted by a self-mutating virus. It is considering implementing a GPS system to track where boys are at any time of day to further reinforce its autocratic regime. Being an enthusiastic coder, you have been taken hostage to aid Oldland's virus in creating this GPS tracker. The virus only cares about the x- and y- coordinates of a boy. Each boy have been implanted with a GPS chip, which communicates with three GPS stations positioned around the house.

Write a function locate that tracks the chip in one boy. It should take in three tuples of floats as input. Each tuple represents the observations from one GPS station, and contains values `(x, y, r)` where `x` is the x-coordinate of the GPS station, `y` is the y-coordinate of the GPS station, and `r` is the distance of the boy from that GPS station. You should then deduce the position of the boy from these observations. There will always only be one possibility. Return a tuple of floats `(x_pos, y_pos)`, where `x_pos` is the x-coordinate of the boy, and `y_pos` is the y-coordinate of the boy.

When marking your solution, we will truncate your float output to 2 decimal places.

Example function definition

```
def locate(
    obs1: tuple[float, float, float],
    obs2: tuple[float, float, float],
    obs3: tuple[float, float, float],
) -> tuple[float, float]:
    (x1, y1, r1) = obs1
    (x2, y2, r2) = obs2
    (x3, y3, r3) = obs3
    # code your stuff...
    return 0
```

Example function calls

```
locate((0.0, 6.0, 5.0),
       (1.0, 2.0, 2.0),
       (3.0, -1.0, 3.0)) = (3.00, 2.00)
```

23 PALINDROMIC SQUARES - Yellow

`Last updated at Feb 23 21:17pm GMT.`

Palindromes are numbers that read the same backwards and forwards, for example, 123454321.

Given a **number base** `B` **(2 ≤ B ≤ 20)**, write a function `solution` that finds all the numbers `N` **(1 ≤ N ≤ 300)** in base `B` whose squares are palindromic, outputted along with the squares themselves in a list. Use letters 'A', 'B', 'C' etc. to represent numbers 10, 11, 12... when `B` > 10.

Example function definition

```
def solution(B: int) -> list[list[str]]:
    # code your stuff...
    return output
```

Example function calls

```
solution(5) = [['1', '1'], ['2', '4'], ['11', '121'], ['101', '10201'], ['111', '12321'], ['23

solution(12) = [['1', '1'], ['2', '4'], ['3', '9'], ['11', '121'], ['22', '484'], ['101', '102
```

24 RENDEZVOUS - Yellow

`Last updated at Feb 24 00:20 GMT.`

The Harrow Computer Science Society maintains a number of safe houses on its campus. These safe houses serve as a secure rendezvous point for its members in the case of an AI apocalypse. Each house on the Harrow campus has been assigned a unique code by the society to enable its members to locate safe houses. Only *safe* codes lead into a safe house. All other codes lead into unfortified dwellings that leave inhabitants vulnerable to AI attack. The code system works as follows:

- All codes are 15 characters long
- All codes contain uppercase letters only

- `HARROWCSSOCSAFE` is a safe code
- Flipping a safe code generates another safe code (e.g. `HARROWCSSOCSAFE` → `EFASCOSSCWORRAH`)
- Moving the last letter of a safe code and to the front generates another safe code (e.g. `HARROWCSSOCSAFE` → `EHARROWCSSOCSAF`)
- Replacing a repeated letter (e.g. `RR`) with the letter followed by `X` (e.g. `RR` → `RX`) in a safe code generates another safe code (e.g. `HARROWCSSOCSAFE` → `HARXOWCSSOCSAFE`)
- Replacing `X` followed by a letter with the letter repeated generates another safe code (e.g. `HARXOWCSSOCSAFE` → `HAROOWCSSOCSAFE`)

Given an input string code, you should write a function `check(code, n)` that returns `True` if the input code can be generated from `HARROWCSSOCSAFE` in less than or equal to `n` steps, and `False` if it can't be.

Example function definition

```python
def check(code: str, n: int) -> bool:
    # code your stuff...
    return can_be_reached
```

Example function calls

```
check("EHARROWCSSOCSAF", 1) = True
check("FEHARROWCSSOCSA", 1) = False
```

25 SCULPTURE RESTORATION - Yellow

Harrow school is known for its charity program Shaftesbury Enterprise where students on the hill help local primary school students learn subjects, they wouldn't have the chance to learn and play sports that they wouldn't normally play.

Last week, the first part of a two-parts sculpture introduction program was set and performed, and various simple statues made of blocks were made. However, some of them are rotated 90, 180 or 270 degrees by a naughty student Alex (who given a detention after later).

You need to identify before restoring the sculpture that got rotated by Alex. Here are some rules that normal sculptures should follow:

- All sculptures are a single vertical layer made up of White and Blue cubes sticking together.
- All sculptures are symmetrical with their structures and colour.
- All sculptures have 3 connected cubes as their bases.
- All sculptures' largest width can't be more than 7 cubes.

Write a function called `solution`.

It will receive 1 string formed by joining `1+H` strings separated by spaces:

1. The `WidthxHeight` or `WxH` of the grid that the sculpture will be in in a single line. I.E. `10x10`, `5x7` or `11x21`
2. The consecutive strings of letters that represent the cubes in the grid from top row to bottom row. `W` = White Cubes, `B` = Blue Cubes, `X` = No Cubes or empty space

The function needs to output `True` or `False` for whether the sculpture is rotated by Alex and the restored sculpture if it is rotated. All sculptures rotated or untouched are guaranteed to have their bottom at the bottom row or the last connected letters in the input. However, all the other sides of the grid are not necessarily the leftmost, rightmost or uppermost sides of the sculpture. When you return your restored answer, please only rreturn the smallest grid that contains the sculpture (see Test Cases).

If output is `True`, return an empty string.

Example function definition

```python
def solution(s: str) -> tuple[bool, str]:
    # code your stuff...
    return answer, smallest_grid
```

Example function calls

```python
solution("3x4 XBX XWX WWW") = (True, "")
solution("10x3 XWXXXXXXXX XBXXXXXXXX XWXXXXXXXX") = (False, "WBW")
```

26 HARROW SOCIETY MEMBERSHIP CARD - Yellow

You are developing a new membership card validation system for the prestigious Computer Science Society. Due to a numerous amount of people trying to sneak into the groundbreaking AI talks, we need you to find the fake members from the real. The membership card has a specific pattern, and your goal is to implement a validation system.

A genuine membership card has the following features:

- It must start with the emblem of Harrow, represented by the letters "HARROW" all in uppercase.
- It must contain exactly 10 digits.
- It must not have 2 consecutive repeated digits.

- The initials of the person must be contained in the last two digits (i.e. John Doe -> JD)

Write a function called `solution` that takes in 2 arguments. The first argument will be the name of the person, and the second argument is their membership card. Your function should then determine whether it is a valid membership card, and return `True` or `False` accordingly.

Example function definition

```
def solution(name: str, card: str) -> bool:
    # code your stuff...
    return is_valid
```

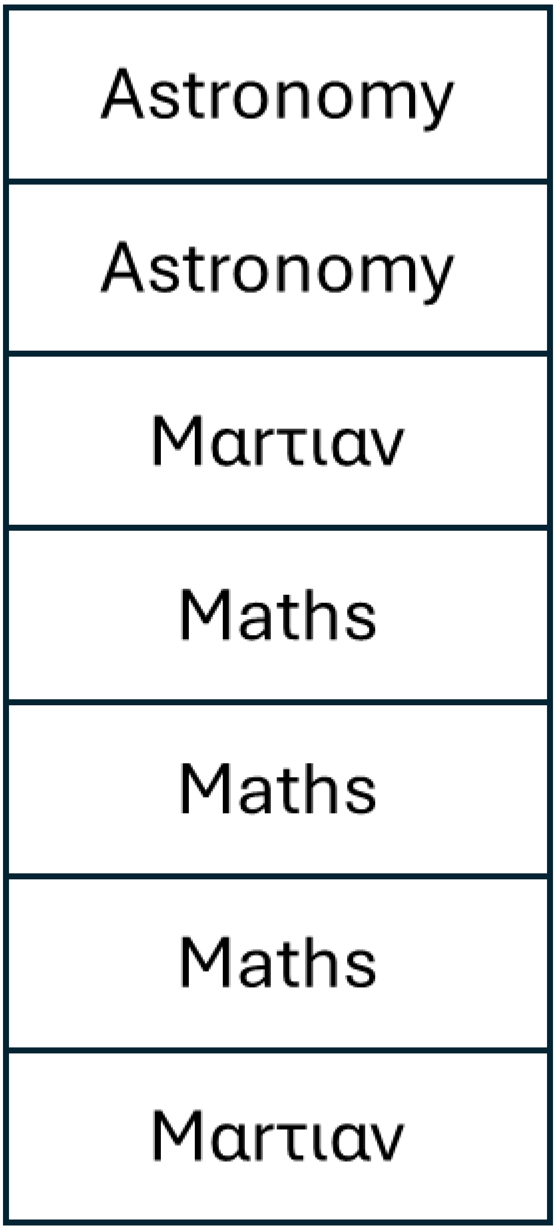
Example function calls

```
solution("JOHN DOE", "HARROW12JD") = True
```

27 TIMETABLE - Yellow

In the new Martian branch of Harrow school, all students take **10 subjects**. Usually, they have 8 lesson slots every day. However, they have to carry out oxygen synthesis instead on Tuesday afternoon, so they only have 5 slots. On special occasions, such as the day leading to a polar expedition, an additional lesson is remitted to ensure a smoother journey. Lastly, pupils may choose to go to the observatory briefly to instead of normal lessons. The heads of subjects also demand **double and triple periods** in order to carry out longer activities such as mineral mining. As a result, it has become incredibly complicated to compile different timetables.

To help the Master in Charge of Timetables, you should write a function called `solution`. **Given the number of lesson slots, your function should return the total number of different timetables they can have**. An example day with 7 lesson slots is shown below.



Note: Obviously there is a more optimal and awesome solution using combinatorics, but a good brute-force approach will score full marks.

Example function definition

```python
def solution(slots: int) -> int:
    # code your stuff...
    return num_timetables
```

Example function calls

```
solution(4) = 9990
```