D.E.S.P.A.I.R.

Diogn's Eldritch Stories in Preparation for Annoying, Insightful Regrets

A **Mock HOPE** Made with Love by **Diogn Mortera** (@mortrpestl)

What's the Point?

Mock HOPE lol /j I

I made these questions not primarily for you to code them but for you to exercise your insights and thought-making process.

Solutions and (most of the time) code for non-heartbreaks can be found in the end of the document and the folder where you probably found this document, respectively. If you can't solve it, read the solutions for insights 😄

On Grading Yourself

You can submit code in my Offline Judge 😎.

- Your code for most problems and their subtasks can be checked by going to the "`(submit code here!) offline_judge`" folder;
- going to the folder with the problem number/function name;
- typing your code into `student_solution`.py, and;
- running the file.

Verdicts will be provided as the file is run. Isn't that cool!

Do note that sometimes, subtasks can evaluate you wrongly (because I don't have the testers for this, just me for now 😔). In that case, feel free to compare with the model solutions in `model_solutions` folder or the upcoming solutions file.

Other Information

- *The difficulty can **fluctuate**.* Some problems may be easier or harder depending on what you're good at. I tried my best to have a variety of problems types to this set.
- Some subtasks are ***special forms of heartbreaks***. They will be signified with a *reaction image* related to the question. When you see this, the heartbreak is ***definitely*** *<u>not easy</u>* to solve and probably not necessary for the HOPE either (but you never know). They will also usually not have model solutions because nobody is reading all of that.
- There can be mistakes in the solutions. Please let me know, but please do understand if ever 💔 there will be things that will slip under my proofreading 😔
- And of course, Don't stress yourself out.

Before you proceed: https://oj.dcs.upd.edu.ph/cs11cheatsheet/

Recall

(if you don't know/recall a term, try to learn it! or skip it…)

Operations
- Basic operations
- `pow(a,b,mod)` function to find power under modulo fast
- Tuple addition, string addition, `sum()`, `max()`, `max(key=len)`, `min()`
- Frozensets and set operations, `&`, `|`,etc.

Basic string methods
- `index()`, `find()`, `len()`, `split()`, `[]`, `[:]`
- and how to implement them yourself using recursion, *just in case*
- `ord()` and `chr()`
- `s.isalpha()`, `s.lower()`, `s.upper()`
- `f`-strings

Recursion
- what is wishful thinking?
- enumerating all possibilities
  - e.g. `Mock Hope (Splits, Paths)`
- or checking for a possibility that works
  - e.g. `Mock Hope 1g (Mama)`, `Coin Decomposition`
- common patterns
  - combinations and permutations
  - listing all pairs, substrings
  - grid printing problems (e.g. `Fortune Cookie` and `Hilbert`)
- how to return output properly?
  - flattening tuples using `*` or `sum()`

Comprehension
- nested comprehensions, if statements and comprehensions, `range()`, ternary operators
- bonus: walrus operator
- `frozenset()` to avoid TLE when using `in` on a certain tuple

Conditionals
- order matters (e.g. test for 'equilateral' before 'isosceles' for `Mock Hope 1b: Triangle`)
- Repeated addition is multiplication, etc.
- Odd and even numbers
- Edge cases (`0`,`1`,`()`,etc.)
- Floor, ceiling

Simple data structures
- Tuple, making empty tuple, adding tuples, tuple deconstruction
- Frozensets, sets, operations, sorting with recursion

Tricks
- Default parameters in recursion
- Tuple deconstruction
- simplifying the logic with math before coding (`Practice 4: Rotate Painting II`)
- walrus operator (`Practice 4: Ako ang Nagwagi`)
- Bonus: getting used to command line

The **DESPAIR** questions can be found in the next pages.

Feel free to time yourself for *3 hours*.

2000 is perfect. Try to get at least 1200 in 3 hours if you want to pass.

If you want to test, I recommend checking your solution against the examples for now, then checking it with the model solutions later (to avoid wasting time).

Good luck!

Problems

Problem / Available Subtasks	1	2	3	4	5	6	7	Total
Not Not	200 😍	200 😍						400
Love You To Bits	200 😍	200 😍						400
Miss Ko Na Siya I: Tira-miss-u Cake	20 ❤️	80 😍	100 😍	100 💔	100			400
We Know	100 ❤️	100 😍	200 💔					400
Rate of rate of rate of rate of…	400 😍							200
Weightless	75 😍	125 💔	200					400
Miss Ko Na Siya II: Love Scenario	100 😍	100 😍	200					400
Into The Fire	200 😍	200						400
Sugar Rush I: Race Track (Space Stone)	100 ❤️	100 😍	100	100				400
Hacking to the Gate	20 ❤️	80 😍	100 😍	100	100			400
Flood Control	200 😍	200 😍						400
And He Just Won't Die	200 😍	200						400
Sugar Rush II: Lookahead (Time Stone)	100 😍	100 😍	200					400
Death's End	50 ❤️	50 😍	50 😍	50 💔	200			400
GlaDOS and Bombs 4 Throwing at You	50 😍	50 😍	100 😍	100 😍	100 😍	100 💔	**500**	1000
Sugar Rush III: Sugar Crash (Mind Stone)	200 💔	200	200	***1000***				1600
Total (denominator) – this is the points to get perfect								2000
Total (true despair) – this is the total points of the problem set ❤️ 💔								8000

(1)Not Not(1)

This problem is inspired by the eponymous logic mobile game.

Given a string preceded by an arbitrary number of “not”s, please simplify.

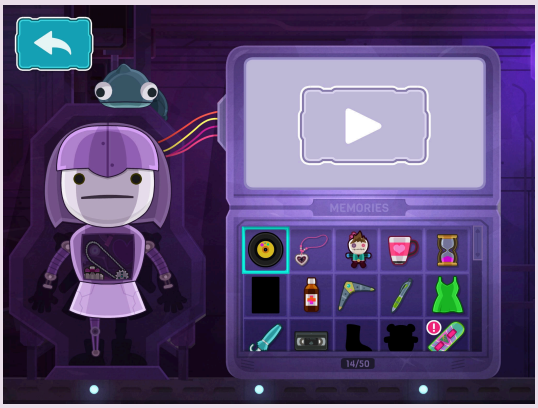
The simplification is the form “not x” or “x”. The letters become lowercase. Note that x can be more than one word long, but cannot start with “not” unless “not” is x.

Task Details
Define notnot(s), where *s* is the presumably unsimplified string. The string is at least one word long.

Examples
“Not not rizz” becomes “rizz”.
“Not nOt Not rizz” becomes “not rizz”.
“Not not nothing” becomes “nothing”.
“Not not not” becomes “not”.
“NoTHing” becomes “nothing”.

Constraints
🎂 x is one word for 200 points
🎂 words in s<800 for 200 points

(2)Love You To Bits(2)



Inspired by the mobile game 'Love You To Bits'

An astronaut goes on a quest to find her lover's remains after they scatter throughout a galaxy after a tragic starship accident.

Each robot part named n has 3 additional parameters: distance from the austronauts spaceship d, fragility of the item f, and identification number i.

To collect the parts systematically, the astronaut sorted her parts in a list:
- Prioritizing parts nearer to the spaceship
- If the distance is the same, he prioritizes items with *higher* fragilities
- If the items have the same fragility, he prioritizes items with *smaller* length of name (only letters and numbers counted)
- If the names of items have the same length, he prioritizes items with *higher* ID number. No item has the same ID number.

Given 2 items where you know their (n, d, f, i), can you help the astronaut know which to prioritize?

Task Details
Define `prioritize(s)`, where in s is a tuple of tuples (n, d, f, i) defining the name and 3 parameters of each robot part. Return the name of the prioritized item.

Examples
`tup=(("Arm",3,4,5),("Leg",3,50,1))` results in `"Leg"`
`tup=(("Robot Heart",40,40,1290),("Brainiac",40,40,1000))` results in `"Brainiac"`

Constraints
🤩 Base problem for 200 points
💔 "2 items" -> "n items" for 200 points

(3)Miss Ko Na Siya I: Tira-miss-u Cake(3)

찬란했던 그 미소 두 눈에 가득했던 파도 난 너를 보면

Hanni is trying to court someone. But she doesn't want to immediately give himself away. So she bakes a *tiramisu* cake. Maybe her crush will get the pun in the name.

Like her hints, she makes a cake with n^k layers.

He readies her ingredients ready, using eggs to mascarpone and love, and does the following process:

- Create a base layer with radius n^k *meters* and a height of 1 *meters* *(it's big. I mean, why not? the bigger, the better she'll get the hint. right?)*
- For all next layers above of the cake, its radius and height is 1 less and 1 more than the layer below it, respectively.

Recall the formula of a cylinder: $\pi r^2 h$. If 1 ounce of love is equal to π cubic meters, how much ounces of love will Hanni need to bake her *tira-miss-u* cake?

If the cake gets too large, the ounces of love may be too much! *In this case, please output the* ***remainder of the number when divided by*** 23412143.

Task Details

Define `mkns(n,k)`, where n^k defines the number of layers in the cake.

Examples

$n^k = 3$ results in 20
$n^k = 60$ results in 1153510
$n^k = 143$ results in 12417937
$n^k = 143143143$ results in 17010890
$n^k = 143143143^{143143143}$ results in 13412956

Constraints

❤️ $n^k \leq 50$ for 20 points
🤩 $n^k \leq 800$ for 80 points
🤩 $n^k \leq 143143$ for 100 points
💔 $n^k \leq 143143143143$ for 100 points
$n^k \leq 143143143^{143143143}$ for 100 points

(4)We Know(4)

Cgwpcivwt, mmeqmk gpgq ytjo, spy ow djlz migo ycsl yz cfqr.

Alexander is up to no good. All of his writings are encrypted with his Ham cipher, but you can't seem to crack it.

You know, however, that he has a key (wherein he only considers the letters in it) that he shifts letters by though. Say you have "Say no, Alex Ham!" with key "!B.URR!".

Original	S	a	y		N	o	,		A	l	e	x		H	a	m	!
Shift	B	U	R	-	R	B	-	-	U	R	R	B	-	U	R	R	-
Encrypted	U	v	q	-	f	q	,	-	V	d	w	z	-	C	s	e	!

Observed that `S` is shifted by 2 units (to `U`) because `B` is the 2nd letter in the alphabet. Note that when the resulting shift makes the letter position more than 26, it wraps around. So a shift of `Y` by `B` makes it `A`.

Notice how all symbols in the key are ignored. And symbols in the original string are also skipped as the key encrypts the characters.

Given key k and an encrypted string s:
- where both are fully uppercase and no symbols for ❤️
- where both can have lowercase letters and symbols for 💔

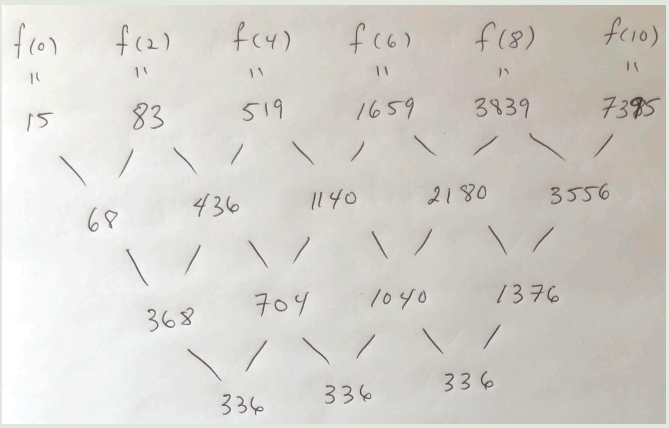
…can you crack/decrypt it for Hamilton? (you cannot say no to this.)

Task Details
Define `we_know(k,s)` where k and s are the key and encrypted string, respectively.

Examples
```
('RIGHT','OAVVA') -> "WRONG"
('BURR','Uvq fq, Vdwz Cse!') -> "Say no, Alex Ham!"
("HANNIPHAM","AJL6GNLMO7BXQGOQRIHHXCOUDF") -> "SIX6SEVEN7OPPSAHBAGUPBAGUP"
("BURR","vCw2332#jq0h.oZgMw!-1l.j@kh8f8F") -> "tHe2332#ro0m.wHeRe!-1t.h@pp8n8D"
("YOURE LIKE ME, I WAS NEVER SATISFIED","ZCBWQULL") -> "ANGELICA"
```

Constraints
$1 \leq$ `len(k),len(s)` ≤ 800
❤️ `len(k)=len(s)`, and k, s is fully uppercase and has no non-letters for 100 points
🤩 k, s has no non-letters for 100 points
💔 No further constraints for 200 points

(5) Rate of rate of rate of rate of...(5)



What is a 'rate'?

The president mentions that there is a **"constant rate of rate of <insert more "rate of"s>… inflation right now."** You want to recall the full statement, but you can't be bothered to watch his speech again, so you just use at the data you have at hand —the price level.

You sampled a list of *price levels* every month. And now its time to put that into good use.

Use this list of price levels to find what the president originally said. It could be, for example, the

- *"There is a constant (but not zero) rate of inflation right now"*
- *"There is a constant (but not zero) rate of rate of rate of rate of rate of rate of rate of inflation right now"*
- *"There is a constant (but not zero) inflation right now"*
- *"There is no inflation right now."*

You recall that inflation is the rate of change of price level, and the slope/rate of change of a function g(x) over two intervals a and b can be obtained using $\frac{g(b)-g(a)}{b-a}$.

So you try to do this where every pair of consecutive numbers a and b in your list, until you get constant values for the differences (see image above).

It is also possible that you mislabeled data. You know this is the case if according to your calculations, if there are only 2 elements left in your upside-down pyramid, all values are still not the same. In this case, you output *"I bite my tongue, it's a bad habit."*

Task Details

Make `level_of_inflation(tup)`, where tup is a tuple of arbitrary length with numbers, representing the data you collected over evenly spaced months.

Examples

```
(12,12,12,12,12) -> "There is no inflation right now."
(6, 21, 86, 261, 630, 1301) -> "There is a constant (but not zero) rate of rate of rate of inflation right now"
```

Constraints

🤩 len(tup)≤ 20 for 200 points

(6)**Weightless**(6)

Why does it feel like I can tell you anything?

Regrets are a difficult thing to forget. In *K-Pop Demon Hunters*, they can be quantified by a number, named the ***sudapap***. Different people have different sudapaps.

Rumi and Jinu are two anti-heroes with difficult pasts. After meeting each other, they've realized to *let the past be the past till it's weightless*.

In a breath-taking awe of this realization, they perform what they call a ***honbuwan***. Let me show you *how it's done*:

- Whoever between the two that has the smaller sudapap subtracts that value from the other person's sudapap until that other person's sudapap is less than theirs. Each subtraction is considered one *hon*.
- At this point, the roles reverse. Like the dance of night and day, this step is correspondingly called *buwan*.
- The *honbuwan* ends when either Rumi's or Jinu's *sudapap* reaches 0. The *hon* step leading to this is counted.

An avid Rujinu shipper as you are, you want to find out (a) how many of the *hon* step happened, and (b) the value of leftover regret of the person whose regret did not reach 0, after their *honbuwan*.

Because Rumi and Jinu are your idols, you know their sudapaps.

Can you get the answers to (a) and (b) a little higher?

Task Details
Make `sudapap(r,j)`, where r and j is Rumi's and Jinu's sudapaps, respectively.
It should output a tuple of `(a,b)`, where `a` and `b` are your answers to question `(a)` and `(b)` respectively.

Examples
$r = 1, j = 1$ results in $(1, 1)$
$r = 49, j = 7$ results in $(7, 7)$
$r = 1230, j = 126$ results in $(18, 6)$
$r = 10^{50}, j = 64^{20}$ results in $(75231638452872, 1125899906842624)$

Constraints
🤩 $1 \le r, j \le 200$ for 75 points
💔 $1 \le r, j \le 10^6$ for 125 points
$1 \le r, j \le 8^{1250}$ for 200 points

(7)Miss Ko Na Siya II: Love Scenario(7)

Giniginaw at hindi makagalaw...

Minji can no longer take the *lamig* she feels when she is talking to her situationship. She has a collection M of all of their distinct (for ❤️) messages to each other.

She categorizes each message in this collections into lists ML, TL, and MI containing text messages from him depending on what she deems *malamig*, *tama lang*, at *mainit*. She believes this will help her determine if he shares her feelings back too. Messages can be in one or more lists.

The gorls now try to hyperanalyze the situation:

- Danielle is a devil's advocate and is trying to prove that Minji is delulu, creates the list delulu - if a message is both in *malamig* and *tama lang*
- Hanni, after bagging fineshyt with her tira-miss-u cake, is optimistic and creates may_chance - containing messages both in *tama lang* and *mainit*
- Haerin is cautious and finds some messages surface-level, creating list wag_red_flag - containing messaages both in *malamig* and *mainit*.

As Hanni's boyfriend, help her collate Minji's three lists based on her friend's criteria so they can help further analyze Minji's love scenario. *Return each list ordered by the messages' earliest appearance in M.*

But if you really don't want to break Minji's heart 💔, messages are not necessarily distinct (after all, do you only say 'ily' once to your situationship?). That is, if the same message occurs twice (or more) in list *malamig* and *tama lang* for example, it also occurs twice (or more) in *delulu*. This applies to the other lists too.

Task Details

Define mkns_2(M,ML,TL,MI), where ML,TL,MI and output (delulu,may_chance,wag_red_flag).

Examples

M = ('update me v.v', 'the moon is beautiful isnt it', 'hays', 'i luv u <3', 'ok', 'ih', 'mkns')
ML = ("ih", "ok", "mkns", "hays")
TL = ("mkns", "hays", "update me v.v","i luv u <3")
MI = ("update me v.v", "ok","mkns", "i luv u <3", "the moon is beautiful isnt it")
becomes:
delulu=('hays', 'mkns'), may_chance=('i luv u <3', 'mkns', 'update me v.v'), wag_red_flag=('ok', 'mkns')

Constraints

😍 $1 \leq$ len(each tuple in this problem) ≤ 10 for 100 points
😍 $1 \leq$ len(each tuple in this problem) ≤ 100000 for 100 points
$1 \leq$ len(lists) ≤ 100000 and you don't want to break Minji's heart (see note above) for 200 points

(8)Into The Fire(8)

'Ultrakill' is the highest tier of multiplier you can get in the FPS game 'Ultrakill'.

V2, a demon marksman, stores a boolean list of the success values of her shots. Alongside this, she also stores the corresponding multiplier for that hit at a different string M.

- The base multiplier $1\times$ is called (A - Anarchic).
- When V1 deals a non-zero amount of damage 2 times in a row, she gets a $2\times$ multiplier (B - Brutal).
- When V1 deals a non-zero amount of damage 5 times in a row, she gets a $5\times$ multiplier (S - Sadistic).
- When V1 deals a non-zero amount of damage 10 times in a row, she gets a $10\times$ (U - Ultrakill).

Note that the multiplier she obtained starts its effect on the next hit, not at that hit.

When she misses, the streak goes down to the streak required to get the multiplier below it.

- For example, if she misses while at S, her multiplier becomes B and her streak becomes 2. Consequently, she would only then need $(5-2)=3$ more consecutive hits to get to S again.

An example of a boolean list with its corresponding multiplier string is below.

Hit?	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Mult	A	A	B	B	B	S	S	B	B	B	S	S	S	S	S	U	U

Can you enumerate possible M of length l in no particular order?

Task Details

Given `lune(l)`, where l is the length of the sought M, please return a `frozenset` containing all valid M of length l in specific order.

Examples

$l=4$ returns `frozenset('AABB', 'AAAB', 'AABA', 'AAAA')`

$l=11$ returns `frozenset('AABBBSSSSSU', 'AAABBBSSSSS', 'AAAABBBSSSS', 'AABAABBBSSS',...)`

Constraints

$1\leq l$

🤩 $l\leq 20$ for 200 points

🌋 $l\leq 26$ for 200 points (Hint: you are allowed to use `dicts` for this heartbreak)

(9)Sugar Rush I: Race Track (Space Stone)(9)

Boogsh

You start at $(0,0)$ facing right in an $r \times c$ race track bounded by walls outside, with rows $0...r-1$ from top to bottom and columns $0...c-1$ from left to right. We call each coordinate a *tile*.

You start speeding through the track. If you hit a wall or a tile you've been to before, you turn right (in your car's perspective. If you are driving to the right, your new direction is down for example). You do this until you've moved n tiles.

After driving $0 \leq n \leq r \times c$ tiles, please tell me your location (r_f, c_f). For larger answers, give the remainder of r_f and c_f in that coordinate (r_f, c_f) when divided by $10^9 + 7$.

Task Details

Make `sugar_rush_racetrack(n,r,c)` where n is the number of steps taken so far, and r, c represent rows and columns of board. Return a tuple of (r_f, c_f) depicting the location.

Examples

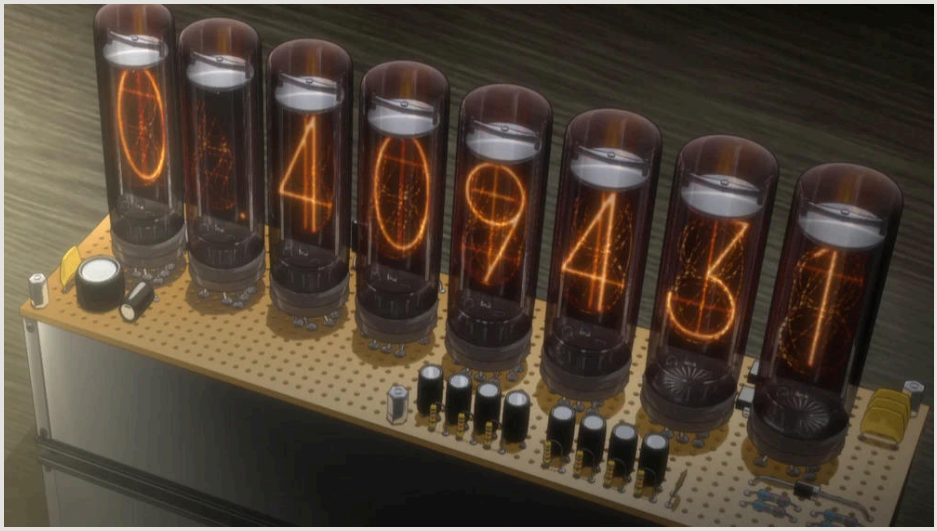
```
(3, 5, 6) -> (0, 3)
(72, 70, 70) -> (3, 69)
(4, 40, 40) -> (0, 4)
(679, 700, 700) -> (0, 679)
(50000, 900, 900) -> (14, 398)
```

Constraints

- 💖 $1 \leq r, c \leq 20$ for 100 points
- 🤩 $1 \leq r, c \leq 900$ for 100 points
- $1 \leq r, c \leq 10^{18}$ for 100 points
- $1 \leq r, c \leq 10^{100}$ for 100 points

The recursion limit has been increased to 4000.

(10)Hacking to the Gate(10)



How many times do you think I've watched her die? I've seen it again and again and again and again and again and again... It won't destroy me.

You are given a list of possible timelines with their own divergence numbers DN (integers) where Kurisu lives.

You were also let known that if in any combination of k timelines, if the sum of all DN in this group is equal to n, then there exists a timeline where Kurisu is saved.

In this list of timelines, will Kurisu be alive somewhere?

Task Details
Make `kurisu_lives(tl,k,n)`, where tl is the tuple of timelines, and k and n stands for whats stated in the problem respectively.
It should output either *Convergence* if Kurisu is sure to be alive somewhere, and *Divergence* otherwise.

Examples
$\mathrm{tl} = (1, 2, 3,), k = 2, n = 3,$ results in *Convergence*
$\mathrm{tl} = (4, 5, 6, 7,), k = 3, n = 15,$ results in *Convergence*
$\mathrm{tl} = (10, 20, 30,), k = 2, n = 25,$ results in *Divergence*
$\mathrm{tl} = (1,) * 1000 + (1000,) * 1000, k = 4, n = 4,$ results in *Convergence*
$\mathrm{tl} = (5, 5, 5, 5,), k = 4, n = 20,$ results in *Convergence*

Constraints
PLEASE READ! SOME TASKS ARE EASIER THAN OTHERS BECAUSE OF k NOT VARYING

$0 \leq \mathrm{len}(\mathrm{tl}), k$
$-409431 \leq \text{element in tl} \leq 409431$
$-10^7 \leq n \leq 10^7$

❤️ $\mathrm{len}(\mathrm{tl}) \leq 10, k = 2$ for 20 points
🤩 $\mathrm{len}(\mathrm{tl}) \leq 20, k = 4$ for 80 points
🤩 $\mathrm{len}(\mathrm{tl}) \leq 20, k \leq 15$ for 100 points

Loops are allowed for the heartbreaks:

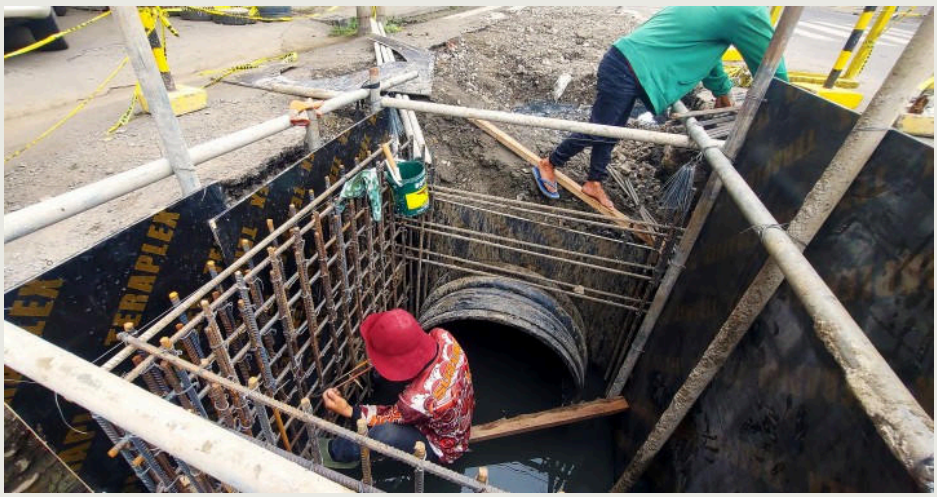
$\mathrm{len}(\mathrm{tl}) \leq 40, k \leq 20$ for 100 points
$\mathrm{len}(\mathrm{tl}) \leq 5000, k \leq 3$ for 100 points

DESPAIR Set
Problems from here onward are **harder than usual**, but still worth a try.

Some of these problems will have no “offline judge”, because writing test cases for them are hard 😭😭😭😭.

And some of these problems are comparable to Koyuki and Bombs (Lab 0) and Hilbert (Lab 1), so don’t feel bad for not getting the problem (or understanding the solutions in the solutions sheet).

(11)Flood Control(11)



Ang tama, ipaglaban. Ang mali, labanan.

Experienced being a politican before?

Well, no, because you're actually doing real flood control projects.

Recall that an $r \times c$ grid is defined as an array with rows $0 \ldots r-1$ from top to bottom and columns $0 \ldots c-1$ from left to right.

Given an $r \times c$ road which there may be `$X$` or " " in each coordinate of the road, please determine *how much money* (in millions) you can steal through government funding.s

A hole is a rectangular-sized space without any `$X$` anywhere on it immediately bounded by `$X$`s on the tiles. For every $n \times n$ sized hole, you can steal 50 million pesos.

Task Details
Define `flood_control(grid)`, a tuple of strings representing a road of arbitrary size $r \times c$.

Return the new grid after patching the holes.

Examples

X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X
X							X				X		X						X	#	#	#	X
X	X	X	X	X	X		X				X		X	X	X	X	X	X		X	#	#	X
X		X			X		X				X	-->	X	#	X	#	#	X		X	#	#	X
X	X	X			X		X	X	X	X	X		X	X	X	#	#	X		X	X	X	X
X		X	X	X	X		X		X		X		X		X	X	X	X		X		X	
X							X		X		X		X							X		X	
X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X

In this example, you can still 700 million pesos through government funds.

Constraints
🥵 $1 \leq r, c \leq 10$ for 200 points
🥵 $1 \leq r, c \leq 100$ for 200 points

(12)And He Just Won't Die(12)

In this problem, you will computationally show the Goldbach's conjecture holds true for all numbers below 2147483.

You are now the Hacker of a Minecraft server, and you want to apply potion effect magnitude of unreasonable size (e.g. Speed 12345) to yourself.

You can only however, do this with at most two prime numbers.

A prime number is a number only divisible by 1 and itself.

Given an even-numbered potion effect magnitude, can you please provide one or two prime numbers that sum up to this number?

Also, while it is not mathematically proven, we have extensive computational proof that every even natural number can be expressed as the sum of two prime numbers.

Task Details
Define `hacker(n)`, where `n` is the desired potion effect magnitude.

Return a tuple of 1 or 2 prime numbers summing up to n.

Examples
10 = 5 + 5
400 = 3 + 397
1414144 = 47 + 1414097
2000000 = 1999993 + 7

Constraints
$2 \leq n \leq 2147483$
😍 $n \leq 100$ for 200 points
No further constraints for 200 points

(13)Sugar Rush II: Lookahead (Time Stone)(13)

...got a ride that's smoother than a limousine...

Vanellope is an all-powerful being that can manipulate space and time. I'm not spoiling why.

A top-down image of her racecar, the Candy Kart, is shown in an $n \times n$ grid of pixels (the coordinates are described the same way the first Sugar Rush explained it). The car is driving towards north initially, which is toward the top of the grid.

Below are Vanellope's powers in space (preceded by their shorthand notation):

- kCCW - turn the car 90 degrees counterclockwise k times
- kCW - turn the car 90 degrees clockwise k times
- kF - reflect the car across the x-axis k times
- kFL - flip the car across the diagonal spanning top-left to the bottom-right k times
- kFR - flip the car across the diagonal spanning top-right to the bottom-left k times

Below are Vanellope's powers in time:
- kTT - go back k *space power* steps in time (reverting the car's position to its position k steps in the past).

Since Vanellope is a highly intellectual being, she chains these steps into one *gauntlet*. An example of this *gauntlet* is "5CCW.4CW.1F.1TT.5FR.4FL".
> mortrpestl's note: You should be *really* happy there are separators between the powers by the way.

Your task, as her nemesis, is to predict the snapshot after all of the steps is done.

Task Details
Define `sugar_rush_villain(grid, g)`, where `grid` is the pixel grid of Vanellope's car's top view and `g` is the gauntlet. Return the resulting `grid` after the gauntlet manipulations have resulted.

Examples
('…..','..^..','.LXR.','.L#R.','..u..'),"3CCW.12CW.5F.3TT.6FR.7CCW") results in ('..u..', '.R#L.', '.RXL.', '..^..', '…..')
('12345','6789A','BCDEF','GHIJK','LMNOP'),"3494CCW.124093459CW.49449TT.103FR.55FL") results in ('PONML', 'KJIHG', 'FEDCB', 'A9876', '54321')

Constraints
$1 \le n \le 100$
Let $1 \le N \le 10^{10}$ = sum of all k in the gauntlet
Let $1 \le S \le 250$ = sum of number of space and time powers in the gauntlet
🥰 $n \le 10, N \le 100$ and there is no kTT in the gauntlet for 100 points
🥰 $n \le 10, N \le 100$ for 100 points
💚 No further constraints for 200 points

(14)Death's End(14)

No, no. Don't say where we are! Once we know where we are, then the world becomes as narrow as a map. When we don't know, the world feels unlimited.

On Trisolaris, leap years are erratic. This is a result of the unpredictable orbit of the planet around three stars.

Thus, the definition of leap years have changed. Originally, in Trisolaris, a year is a leap year if it is a multiple of a, b, or c — determined by the gravity of stars.

However, after a wormhole appeared, an arbitrary amount the stars now influence the leap years. Given a tuple `tup` of numbers (determined by the stars), a year is a leap year if it a multiple of any of these numbers.

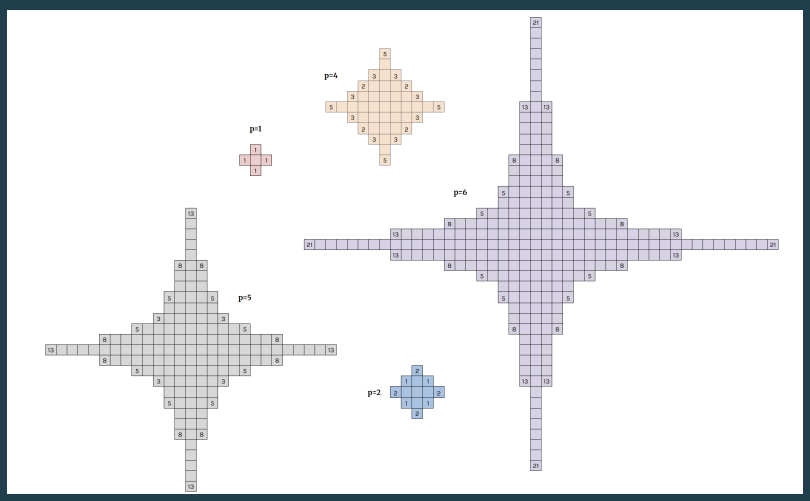
How many leap years have happened between year M and N (inclusive)?

Task Details
Define `nsolaris(tup,M,N)`, and return the answer as an `int`.

Examples
`tup`,$M, N = (7,), 8, 500$ results in 70
`tup`,$M, N = (2, 5,), 1, 1000$ results in 600
`tup`,$M, N = (2, 3, 5), 1, 10$ results in 8
`tup`,$M, N = (2, 3, 5, 7, 9, 11, 12, 13), 2007, 2125$ results in 97

Constraints
`len(tup)`≤ 20
$1 \leq M \leq N \leq 10^{18}$
Each element in the tuple is <=10
💔 $|N - M| \leq 1000$ for 50 points
🤩 `len(tup)`$= 1$ for 50 points
🤩 `len(tup)`$= 2$ for 50 points
💔 `len(tup)`$= 3$ for 50 points
🎉 No further constraints for 200 points

(15)GlaDOS and Bombs 4 Throwing at You (15)



'I am NOT a moron!'

Wheatley wants to kill you in an $r \times c$ grid, so he took some of Koyuki's ideas and is now collecting bombs to throw at you! His bomb patterns (clearer image here) are confusing but kind of predictable.

It seems that the radius of the bomb with power p is the sum of the two radii of the two powers below it. For $p = 1$ and $p = 2$, the radius is 1 and 2 respectively.

To predict Wheatley's moves, you attempt to solve S subtasks relating to how he could be throwing the bombs, from easy to hard.

1. Find the most number of tiles hit in one straight line with power p
2. Find the number of tiles covered by a bomb of power p
3. Determine if (r, c) is hit by a bomb with power p dropped at (r_b, c_b)
4. Determine if it is possible to placement a bomb of power p in any coordinate such that it hits two points (r_1, c_1) and (r_2, c_2)
5. Determine the *smallest* power $p_{\text{best}}\}$ and placement of any bomb $(r_{\text{best}}, c_{\text{best}})$ such that it hits two points (r_1, c_1) and (r_2, c_2)

Give the last 11 digits of every number output only (remove trailing zeroes).

Task Details
Define `bombs(args*)` as highlighted above. Feel free to submit different solutions/code for each subtask. Your points add up.

Examples
For the first 2 subtasks, the answer for (1) and (2) if $p = 6$ is 43 and 257 respectively.

Constraints
$-10^{20} \leq r, c \leq 10^{20}$ for all coordinates given, and $p \leq 10^{18}$
S1: $p \leq 15$ for 50 points
S2: $p \leq 15$ for 50 points
S1 $-$ 2: $p \leq 800$ for 100 points
S3: $p \leq 800$ for 100 points
S4: $p \leq 800$ for 100 points
S1: No further constraints for 100 points
S5: $p \leq 800$ for 200 points

Below is a **bonus problem** I'm not really sure how to solve. I do leave my ideas for it in the solutions, though.

(16)Sugar Rush III: Sugar Crash (Mind Stone)(16)

There's no one I'd rather be... than me.

Putting yourself in the shoes of Ralph, however, you can't help but think Vanellope,your sidekick, is using up more energy than she needs to.

Your task, as her sidekick, is to find the gauntlet of shortest *length* that does the same thing as Vanellope's suggested gauntlet.

Task Details
Define `sugar_rush_hero(grid, g)`, where `grid` is the pixel grid of Vanellope's car's top view and `g` is the gauntlet.

Return the shortest gauntlet (by length in characters) possible that results in the same image as Vanellope's suggested gauntlet.

Examples
None provided

Constraints
$1 \leq n \leq 100$
Let $N \leq 10^{10}$ = sum of all k in the gauntlet
Let $S \leq 250$ = sum of number of space and time powers in the gauntlet

💔 $N \leq 100$, C, and there are no kTT in the gauntlet for 200 points
💚 $N \leq 100$ and C for 200 points
💚 $N \leq 10^6$ and C for 200 points
💚 $N \leq 10^6$ for ***1000*** points

The solutions are in the next page / or in another document. You may rest for now.