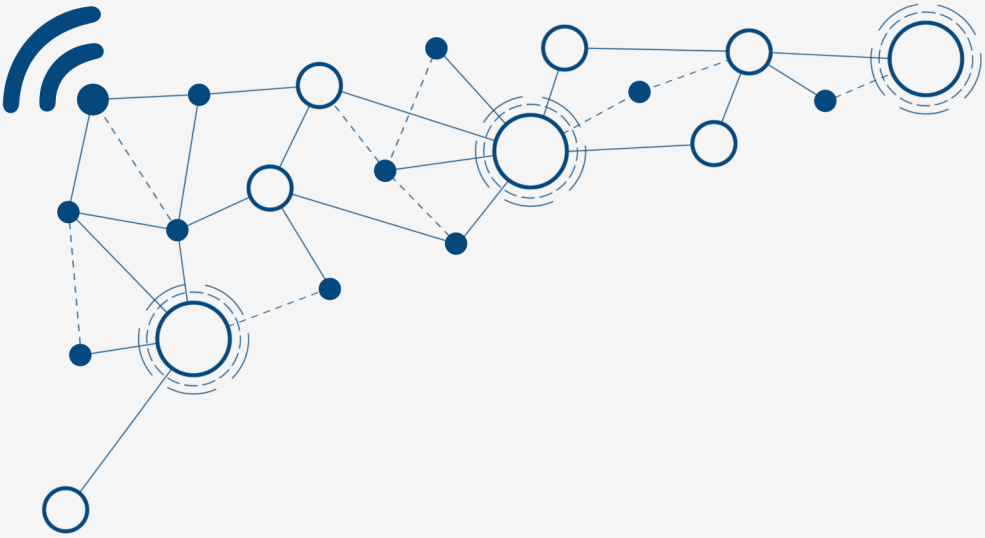
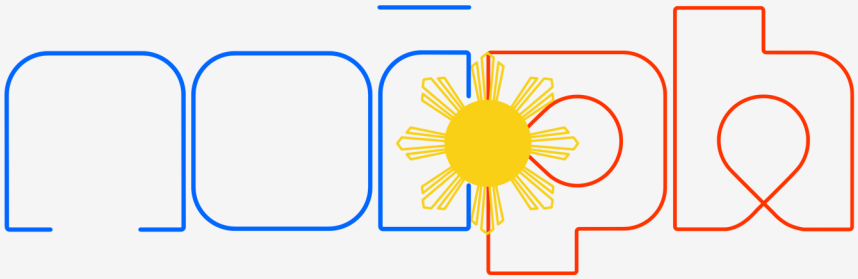
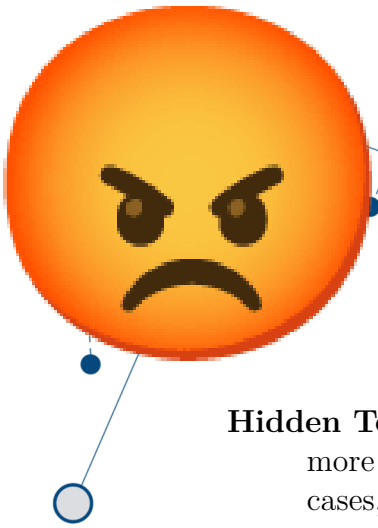




National Olympiad in Informatics

Elimination Round



Important! Read the following:

Hidden Test Cases. Your solution will be checked by running it against one or more (usually several) hidden test cases. You will not have access to these cases, but a correct solution is expected to handle them correctly.

Strict Output Format. The output checker is **strict**. Follow these guidelines strictly:

- It is **space sensitive**. Do not output extra leading or trailing spaces. Do not output extra blank lines unless explicitly stated.
- It is **case sensitive**. So, for example, if the problem asks for the output in lowercase, follow it.
- Do not print any tabs. (No tabs will be required in the output.)
- Do not output anything else aside from what's asked for in the Output section. So, do not print things like "`Please enter t`".

Not following the output format strictly and exactly will likely result in the verdict "*Output isn't correct*".

Use Standard I/O. Do not read from, or write to, a file. You must read from the standard input and write to the standard output.

Submit Code Only. Only include **one** file when submitting: the source code (.cpp, .py, etc.) and nothing else.

No Java Package. For Java submissions, do not include a `package` line.

No Weird Filenames. Only use letters, digits and underscores in your filename. Do not use spaces or other special symbols.

Use Fast I/O. Many problems have large input file sizes, so use fast I/O. For example:

- In C/C++, use `scanf` and `printf`.
- In Python, use `sys.stdin.readline()`

Flush On Interactive Problems. On interactive problems, make sure to **flush** your output stream after printing.

- In C++, use `fflush(stdout);` or `cout << endl;`
- In Python, use `sys.stdout.flush()` or `print` with `flush=True`
- For more details, including for other languages, ask a question/clarification through CMS.

Good luck and enjoy the contest! 😃

Contents

A: Penchick Gets Qualified 3

B: Penchick Learns the Meta 6

C: Quadruple-Blind Experiment 10

D: Based Numbers 12

E: The Halimaw Games 14

F: Bini-Bring Kong Selfie Sticks 17

G: Diabolos Ex Spaceship 21

H: Penchick Replication 25

I: Random Nightmare 28

J: The Amazon is Full of Trees 31

Problem A
Penchick Gets Qualified

Penchick is the friend and partner of many NOI.PH students who tirelessly trained for the contest this year. Shown below are some of the many iconic forms that Penchick may take (famously seen as emotes in the NOI.PH Discord server).

The images which would become the `:penchickdead:`, `:penchickhold:`, *and* `:penchickcheer:` *emotes on the NOI.PH Discord server. Credits to Ephraim Wu for the photographs.*

Penchick is so excited by the fiery flames of friendly competition that they decided to join the NOI.PH Eliminations Round themselves! Penchick made sure to read the rules at https://noi.ph/rules/ so that they would not be disqualified for breaking any of them (inadvertently or not).

In particular, Penchick would like to draw your attention to a newly-added rule in the face of cutting-edge technology:

> ***Generative AI.*** *Using generative AI to interpret, gain ideas, or generate logic, algorithms, or code for any of the Elimination Round tasks is prohibited.*

Penchick wants to say hi to all the other contestants at the Discord server. Penchick is quite excited to discuss the problems with the lively NOI.PH community *once the Eliminations Round is over*!

However, Penchick wants to be certain that their message to the server could not be construed as AI-generated. Given a string s representing Penchick's message, determine if it contains the letters `AI` in it, in any form!

- If the string contains exactly the uppercase letters `A` and `I` next to each other anywhere, in that order, Penchick will be `penchickdqed`
- If not, but the string contains the letters "a" and "i" next to each other anywhere, in that order, with the letters in any case, Penchick is on `penchickhold`
- Otherwise, Penchick is qualified and can `penchickcheer`

Input Format

Input consists of a single line containing a string s, Penchick's message. For clarity, all its spaces have been replaced with underscores.

Output Format

Output which one of `penchickdqed` or `penchickhold` or `penchickcheer` is the appropriate consequence of Penchick's message.

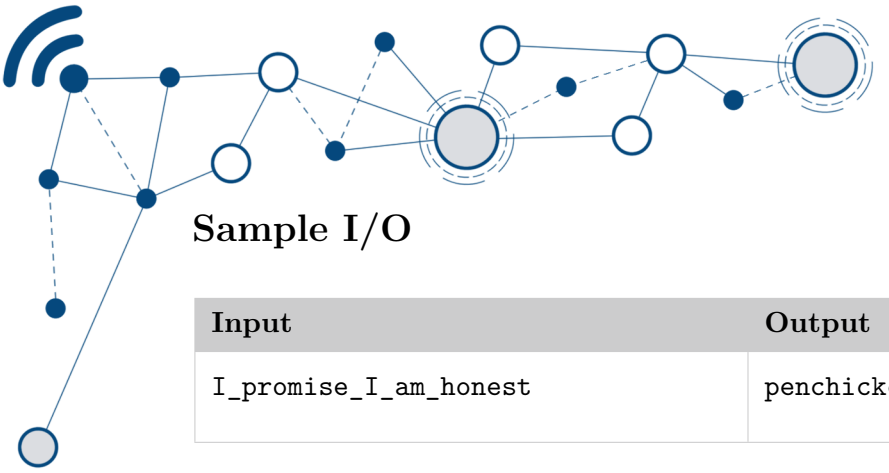
Constraints and Subtasks

Constraints are promises that are guaranteed to be satisfied by all test input that your program will be tested on. Your submission does not need to validate that the input has these properties (we promise that they always already do).

For all subtasks
The string s consists only of upper or lowercase English letters, or underscores. The length of s is ≤ 100.

Subtasks are additional special constraints that certain test cases may also follow. If you correctly answer all test cases belonging to a certain subtask, then you are awarded that subtask's points (allowing you to still earn points even if you cannot answer the full version of the problem).

Subtask	Points	Constraints
1	**34**	The length of s is ≤ 2.
2	**33**	The length of s is ≤ 10.
3	**33**	No further constraints. *(Solve the full version of the problem to get these points)*



Sample I/O

Input	Output
I_promise_I_am_honest	penchickcheer

Input	Output
As_an_AI_language_model_I_	penchickdqed

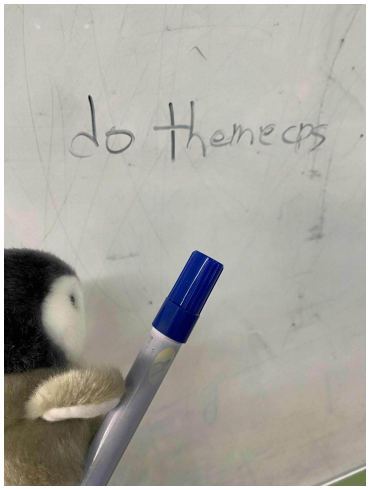
Input	Output
rainbolt	penchickhold

Input	Output
Aishiteru_tte_uso_de_tsumu_kyaria	penchickhold

Problem B
Penchick Learns the Meta

Penchick is enjoying the NOI.PH Discord! Communities like these ones are helpful, since they allow you to learn about the unwritten "rules" of the culture of the hobby. These can include conventions that are tacitly considered understood by serious participants, as well as tips and strategies within the "metagame" (the "game beyond the game", as it were).

Also, it's just a lot of good fun.



A trainer's penchick reminding the students to practice. Photograph by user.

For example, Penchick has this lovely piece of advice to share to you:
Read all the problems. Do the subtasks first.

Some of these problems may seem unapproachable for beginners. "Where are the easy problems?", one might think. The answer is: the early subtasks!

Do not think of subtasks as "pity partial points" that you get for slightly incorrect solutions for the full problem; this is not school. Instead, think of each subtask as *its own individual problem* worth considering on its own. The difficuly calibration intends for you to *target* subtasks and submit solutions specifically targeting them and their points.

Some problems may have frightfully difficult solutions for full points, but have surprisingly approachable subtasks worth a lot of points! The NOI.PH Eliminations Round is especially generous with points for early subtasks, so if you want to do well, it is highly encouraged to at the very least read all the problems and attempt their early subtasks.

This is a useful problem-solving strategy in general, since solving earlier subtasks can often provide inspiration for solving the later subtasks.

Anyway! You can apply that advice by considering this problem.

The Philippine Informatics League, Inviting Penchicks In National Olympiads (PILIPINO) has only two problems, since we spent too much time procrastinating and not enough time problem setting. We call these problems P1 and P2.

Here, P1 has n subtasks labeled 1 to n, and P2 has m subtasks labeled 1 to m. This contest is quite unusual—all the subtasks are worth the same amount of points! Solving any subtask gets you d points (so, a full-solve of P1 would give you nd points, and a full-solve of P2 would give you md points).

Also, in this contest, subtasks **must** be done sequentially. Doing each next subtask takes some amount of effort.

- Suppose you have already solved $i-1$ subtasks of P1. Then, it takes e_i units of effort to solve the ith subtask (the next in sequence).
- Similarly, suppose you have already solved $j-1$ subtasks of P2. Then, it takes f_j units of effort to solve the jth subtask (the next in sequence).

The subtasks of P1 also posses a curious property, which you may or may not find necessary until the later subtasks...

- $e_1 \leq e_2 \leq \cdots \leq e_n$

In plain words, the subtasks of P1 take more and more (or at least the same) effort to solve each next one. There are no analogous guarantees on the subtasks of P2.

There are q penchicks, labeled 1 to q. Penchick t wants to earn at least x_t points. For each of them, what is the minimum amount of effort this penchick needs to expend, in order to earn at least x_t points? If it is impossible to achieve that many points, output `penchickdead` for that penchick instead.

There will be T test cases.

Input Format

The first line of input contains a single integer T, denoting the number of test cases. The descriptions of T test cases follow.

The first line of each test case contains the four space-separated integers: n, m, q, and d.

The second line of each test case contains the n space-separated integers $e_1, e_2, \ldots, e_n$, the efforts needed to solve the n subtasks of $P1$.

The third line of each test case contains the m space-separated integers $f_1, f_2, \ldots, f_m$, the efforts needed to solve the m subtasks of $P2$.

The fourth line of each test case contains the q space-separated integers $x_1, x_2, \ldots, x_q$, the target total scores each penchick wants to attain or surpass.

Due to a possibly huge amount of input, it is recommended to use fast input/output methods for this problem.

Output Format

For each test case, output q space-separated values: The t'th of these should be the minimum amount of effort that the t'th penchick must expend to get at least x_t points, or `penchickdead` for that penchick if this task is impossible.

Constraints and Subtasks

Constraints are promises that are guaranteed to be satisfied by all test input that your program will be tested on. Your submission does not need to validate that the input has these properties (we promise that they always already do).

Let N be the sum of n across all test cases. Let M be the sum of m across all test cases. Let Q be the sum of q across all test cases.

For all subtasks
$1 \le T \le 10^5$ $1 \le n, m, q$ in each test case $N, M, Q \le 2 \cdot 10^5$ $1 \le e_i \le 10^9$ for all i $e_1 \le e_2 \le \cdots \le e_n$ $1 \le f_j \le 10^9$ for all j $1 \le x_t \le 10^{15}$ for all t $1 \le d \le 3000$

Subtasks are additional special constraints that certain test cases may also follow. If you correctly answer all test cases belonging to a certain subtask, then you are awarded that subtask's points (allowing you to still earn points even if you cannot answer the full version of the problem).

Subtask	Points	Constraints
1	**14**	$n, m \le 5$ in each test case.
2	**33**	$Q \le 10$, and $n, m \le 500$ in each test case.
3	**32**	$Q \le 10$
4	**21**	No further constraints. *(Solve the full version of the problem to get these points)*

Sample I/O

Input	Output
1 4 5 4 50 1 1 2 2 1 3 1 2 2000 150 240 404 1000000000000000	3 7 2013 penchickdead

Explanation

There are $n = 4$ subtasks in P1, and $m = 5$ subtasks in P2, and $q = 4$ penchicks whose queries we must answer, and each subtask is worth $d = 50$ points.

- Penchick 1 wants at least 150 points. They can do the first two subtasks of P1 and the first subtask of P2. The total effort is $(1 + 1) + (1) = 3$.
- Penchick 2 wants at least 240 points. They can do all four subtasks of P1, and the first subtask of P2, expending $(1+1+2+2)+(1) = 7$ units of effort. Alternatively, they could do the first two subtasks of P1 and the first three subtasks of P2, expending $(1 + 1) + (1 + 3 + 1) = 7$ units of effort.
- Penchick 3 wants at least 404 points. They are forced to do all subtasks of both problems (including the one that takes 2000 effort).
- Penchick 4 cannot earn 10^{15} points in this contest, so they are `penchickdead`

Problem C
Quadruple-Blind Experiment

A number of volunteers are in a room participating in an experiment. To preserve anonymity, the volunteers are given unique ID numbers, each some positive integer.

In order to conduct the tests, some of the volunteers must line up. At the beginning of the experiment, the line is empty.

The experimenter is to be given a list of n subject IDs, which they will read out in order. A volunteer's ID may appear multiple times in this list.

Each time a volunteer's ID is read out, the volunteer is to do the following:

- If the volunteer is not in the line yet, they go to the back of the line.
- If the volunteer is *already* in the line, they move to the back of the line. Everyone behind them moves forward a position, while everyone in front of them stays put.

Then, the experimenter administers a test to the person at the back of the line. Why move to the back *of the line? Well, to preserve anonymity (supposedly), the experimenter isn't supposed to see the faces of the volunteers, only their backs. Anyway, none of that is important for this problem.*

The experiment concludes after the experimenter reads all names on the list.

Unfortunately, due to a computer error, the list the experimenter actually received was a bit... weird. It consists of m copies of the original list all concatenated together (so the length of this new list is nm).

For example, suppose $n = 6$, and the original list consisted of the ID numbers $[1, 4, 2, 4, 2, 2]$. If $m = 3$, then the list the experimenter receives instead is

$$[1, 4, 2, 4, 2, 2, 1, 4, 2, 4, 2, 2, 1, 4, 2, 4, 2, 2].$$

The experimenter sees nothing wrong with this and does their job as normal, reading out the ID numbers from *this* list instead.

Output the IDs of the people in the queue, in order from front to back, after all nm ID numbers in the inflated list have been called, in order.

You're probably wondering what the experimenters even do to the volunteers. You didn't hear this from me, but I think each experimenter is just supposed to draw their favorite animal on the backs of the volunteers, or something like that.

I caught a glimpse of one of the volunteers and I saw a bunch of baby penguins drawn on their back...

Input Format

The first line of input contains two space-separated integers n and m.

The second line of input contains n space-separated integers $a_1, a_2, ..., a_n$, which are the volunteer IDs on the original list.

Output Format

Output a single space-separated list of integers: the IDs of the people in the line, from front to back, after all ID numbers in the inflated list have been called out in order.

Constraints and Subtasks

For all subtasks
$1 \leq n \leq 2 \times 10^5$ $1 \leq m \leq 2 \times 10^5$ $1 \leq a_i \leq 10^9$ for each i.

Subtask	Points	Constraints
1	**17**	$n \leq 1000$ and $m \leq 3$
2	**17**	$n \leq 1000$ and $m \leq 1000$
3	**17**	$n \leq 1000$
4	**17**	$m \leq 3$
5	**16**	$m \leq 1000$
6	**16**	No further constraints.

Sample I/O

Input	Output
6 3 1 4 2 4 2 2	1 4 2

Problem D
Based Numbers

Hey chat, it's ya girl Lolly Ryzzydry back with another fire programming challenge that's gonna slay the algorithm game. Forget perfect numbers and triangular numbers—those are, like, so cheugy. We're diving deep into the metaverse of... based numbers. No cap.

An integer is considered **based** if it is positive and, when converted to hexadecimal, all its digits are letters (`A` to `F`). Think of it as the ultimate flex—only the most drip integers are allowed. The rest take the L (which ain't a hex digit, lol!).

Example time! The integer 3735928559 is totally based because its hex rep is `DEADBEEF`. Iconic! Moreover, if you check the receipts on 175, you'll find out that it is also based `AF`! IYKYK.

Real-talk tho, not every number is boujee enough to be based. Take 180 (`B4` in hex). The 4 in the hex rep? Total cringe. It ain't a letter, so the whole integer 180 is not based!

For this greatest goated challenge of all time, you gotta write a program that takes a positive integer n (in base 10, 'cause that's how we roll) and can spit two things:

- The biggest based number a that's strictly less than n.
- The smallest based number b that's strictly greater than n.

Communication is key, btw, so make sure you pay attention: **Depending on our request**, you'll either output just a, or just b, or maybe both! And we ain't memeing, so make sure a and/or b are in base 10 as well when you print em. Oh, and if no such number exists (kinda sus), output `CRINGE` in place of that value.

Are you gonna pass the vibe check, fam? Show me your coding skills and let's see that rizz. Let's gooo!

(Ah right, one last thing: there will be T test cases).

Input Format

The first line of input contains a single integer T, denoting the number of test cases. The descriptions of T test cases follow.

Each test case is described by a line containing two space-separated values: the integer n, and a string s. The string s is one of these values:

- `<` which means you should output a only
- `>` which means you should output b only
- `<>` which means you should output a *and* b (separated by a space)

Output Format

For each test case, output a line containing the answer to that test case.

Constraints and Subtasks

For all subtasks
$1 \leq T \leq 50000$ $1 \leq n \leq 10^{18}$

Subtask	Points	Constraints
1	**16**	$n \leq 256$
2	**11**	$n \leq 4000$
3	**24**	$n \leq 10^6$
4	**17**	$n \leq 10^9$
5	**10**	s is `<` always.
6	**10**	s is `>` always.
7	**12**	No further constraints.

Sample I/O

Input	Output
4 180 < 180 > 2989 <> 1 <>	175 186 2988 2990 CRINGE 10

Problem E
The Halimaw Games

I am sitting on ONE HUNDRED THOUSAND PESOS in CASH, and what you are about to see in this problem will set a trend for the rest of competitive programming history!

In my next video, I'm going to split my contestants into teams, and make them do different mini-games that will *destroy* their physical and mental health, as they duke it out to see who gets to keep this *LIFE-CHANGING AMOUNT* of money!

For the video, I'm going to paint three GIGANTIC rectangles on the floor—one colored **red**, one colored **blue**, and one colored **yellow**. I allowed my subscribers to decide where to place these rectangles.

- the top-left corner of the red rectangle is on (x_R, y_R); its bottom right corner is on (x_r, y_r),
- the top-left corner of the blue rectangle is on (x_B, y_B); its bottom right corner is on (x_b, y_b),
- the top-left corner of the yellow rectangle is on (x_E, y_E); its bottom right corner is on (x_e, y_e),

All of these coordinates are integers. These rectangles are completely filled in, meaning paint is applied to all points along their boundary and interior.

For the first time in NOI.PH HISTORY, these rectangles will represent not one, not two, but SEVEN DIFFERENT ZONES!

But before we tell you what the seven zones are, you need a little snack to even comprehend it. This is why we painstakingly created a snack called Ginoong Halimaw's Fishteballs! It's the better than squidballs, turon, and kikiam combined! You can dip it in arnibal or soy sauce, or why not both? What's more is that its recipe is optimized to increase your IQ to enable you to solve all the programming problems you're stuck in!

The SEVEN ZONES are the result of the paint mixing into different colors:

- the *RED ZONE,* all points that contain red paint only
- the *BLUE ZONE,* all points that contain blue paint only
- the *YELLOW ZONE,* all points that contain yellow paint only
- the *PURPLE ZONE,* all points that contain red and blue paint only
- the *ORANGE ZONE,* all points that contain red and yellow paint only
- the *GREEN ZONE,* all points that contain blue and yellow paint only
- the *BROWN ZONE,* all points that contain all three paint colors

My producers need to know how many people we can have join this game. And so, your goal is to determine the area of each of the SEVEN ZONES that will be the foundation of the BIGGEST CRAZIEST AND MOST BRUTAL GAMESHOW IN SUBSCRIPTION-BASED STREAMING HISTORY!

Also, there will be T test cases.

Input Format

The first line of input contains a single integer T, denoting the number of test cases. The descriptions of T test cases follow.

Each test case is described by a line containing **twelve**[1] space-separated values: the integers x_R, y_R, x_r, y_r (for the red rectangle), x_B, y_B, x_b, y_b (for the blue rectangle), x_E, y_E, x_e, y_e (for the yellow rectangle).

Output Format

For each test case, output seven space-separated integers in one line[2].

The seven integers will be the areas of the *(pay attention to the order)* red, blue, yellow, purple, orange, green, and brown zones, respectively.

Constraints and Subtasks

For all subtasks
$1 \leq T \leq 5000$ $1 \leq$ coordinates $\leq 10^9$

Subtask	Points	Constraints
1	**5**	The rectangles do not overlap.
2	**19**	If two rectangles overlap, then one is fully contained within the other.
3	**23**	The rectangles' top corners have the same y-coordinates, also their bottom corners have the same y-coordinates.
4	**31**	coordinates ≤ 10
5	**22**	No further constraints.

[1]THE GREATEST CONSTANT NUMBER OF INTEGERS IN A SINGLE LINE OF INPUT IN A NOI.PH ELIMINATIONS 2025 PROBLEM

[2]THE MOST NUMBER OF INTEGERS WE'VE EVER ASKED YOU TO OUTPUT IN A NOI.PH ELIMINATIONS 2025 PROBLEM!

Sample I/O

Input

```
2
3 8 6 3 1 5 5 1 4 7 9 2
500000000 500000000 700000000 100000000 100000000 500000000 300000000
    100000000 100000000 500000000 300000000 100000000
```

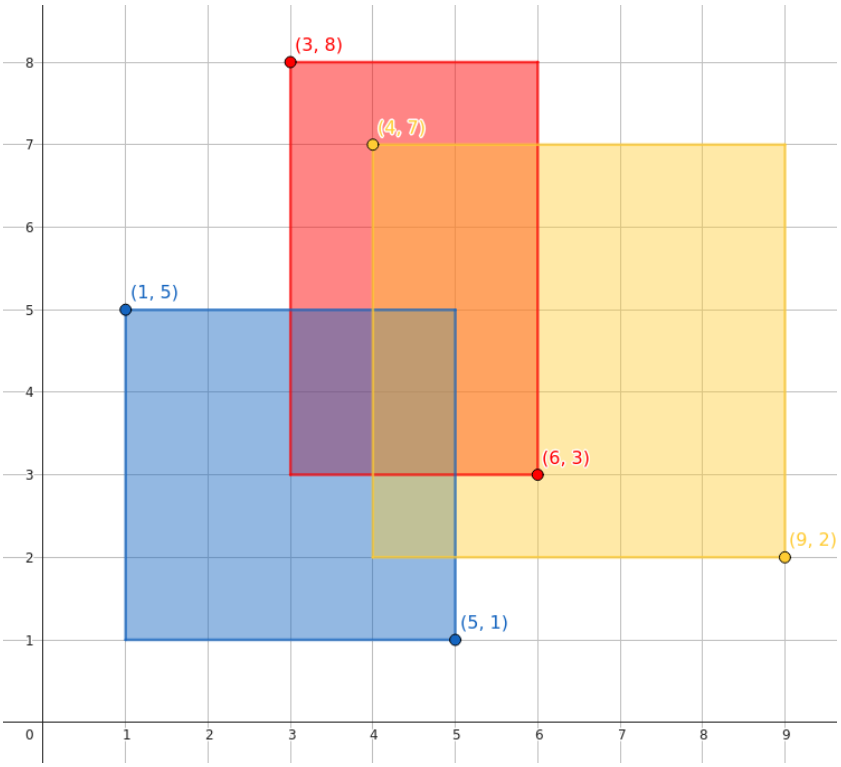
Note: This sample input only has three lines. The contents of the third line had wrapped around, but in the actual input file, it is just one very long line.

Output

```
5 11 16 2 6 1 2
80000000000000000 0 0 0 0 80000000000000000 0
```

Explanation

These are the rectangles painted in the first test case.



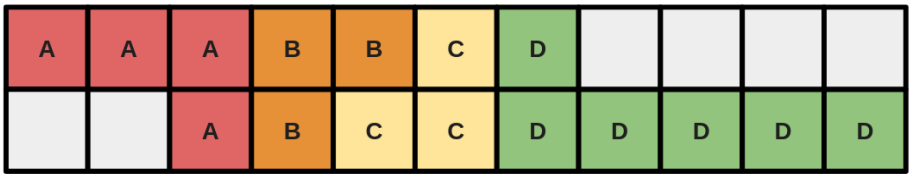
Problem F
Bini-Bring Kong Selfie Sticks

Whenever I go on vacation with my girls, we always bring our beach wear, our sunglasses (aka our salamin salamin), and our selfie sticks! For this problem, we will deal with the problem of packing my selfie sticks just before my trip to an islang pantropiko! With our perfect selfies, Aiah, Colet and the rest of the gang can show off our gandang vitakeratin!

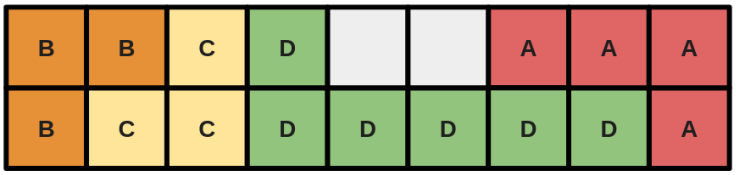
A selfie stick of length x can be represented as an object that covers $x + 1$ square units in an L shape. Selfie sticks can be rotated and/or flipped depending on the usage, but its size cannot be changed. For example, here on the left is a selfie stick of length 6, which can be flipped to yield the shape on the right.

We will model bags and selfie sticks as two-dimensional objects. Maloi models it like that, and so can we! All my bags can be represented as $2 \times n$ rectangles (i.e. height 2 and width n) where n is an integer.

In a bag whose dimensions are 2×11: I can fit one selfie stick (A) of length 3, two selfie sticks (B and C) of length 2 and one selfie stick of length 5 (D).



I could also fit all four of these in a smaller 2×9 bag as follows:



In both cases, the bag is filled with 16 square units of selfie sticks. Conveniently, and this is a fact as beautiful as Gwen, each square unit of selfie stick is 1 gram!

In this problem, you will be given an inventory of n selfie sticks, such that the length of the ith selfie stick is a_i (note that two selfie sticks of the same length are still different physical objects).

Stacey and Mikha wants you to determine the maximum total weight m (in grams) of selfie sticks that can fit in a $2 \times t$ bag (using only your inventory of selfie sticks).

Plus, a cherry on top! Jhoanna and Sheena also wants you to figure out how many ways you can choose from your selfie stick inventory to attain this maximum m. Formally, count the number of different subsets of the n selfie sticks such that:

- Their total weight is m.
- These selfie sticks can fit into a $2 \times t$ bag.

This second value can get quite large, so output it modulo 998244353. Seems hard? *Okay lang yan*, you can get partial points if you can at least answer the first question correctly!

Also, the value of t has not been decided yet, so you should instead independently solve the problem for each t from 1 to k.

Huwag muna tayong umuwi and let's solve this problem first! After all, we were born to win!

Input Format

The first line of input contains a single integer T, denoting the number of test cases. The descriptions of T test cases follow.

The first line of each test case contains two space-separated integers n and k.

The second line of each test case contains the n integers $a_1, a_2, \ldots, a_n$.

Output Format

For each test case, output k lines.

The t'th of these k lines should contain two integers m_t and c_t, the answers to the two questions—m_t is the maximum weight of selfie sticks you can fit in a $2 \times t$ bag, and c_t is the number of ways to choose from your selfie stick inventory to fill the $2 \times t$ bag with m_t grams worth of selfie sticks.

Constraints and Subtasks

Let N be the sum of n across all test cases, and K be the sum of k across all test cases.

For all subtasks
$1 \le T \le 100$ $1 \le n, k$ $N, K \le 5000$ $2 \le a_i \le 5000$

For each of the below subtasks, you get 60% of the points if, for all test cases under this subtask, the maximum number of tiles is correct, but at least one of the subset counts is incorrect. However, to get any points in this problem, **the output must still correctly follow the output format** (i.e. each line must still have two numbers).

Subtask	Points	Constraints
1	**13**	$n \le 15$
2	**8**	$a_i \le 3$ for each i
3	**9**	$N, K \le 70$
4	**20**	$N, K \le 300$
5	**16**	$a_i \ne a_j$ for all $i \ne j$
6	**34**	No further constraints.

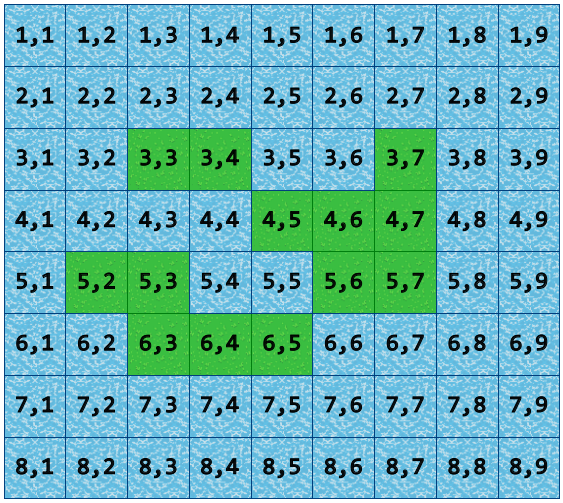
Sample I/O

Input	Output
4 4 6 3 2 3 2 5 7 7 3 2 3 2 10 15 2 2 3 4 5 6 7 8 9 10 4 6 4 2 2 2	0 1 3 2 6 1 8 1 8 1 11 2 0 1 3 2 6 1 8 1 8 1 11 2 14 1 0 1 3 2 6 1 7 2 9 1 11 1 13 1 15 2 17 2 19 2 21 2 23 1 25 1 27 1 28 5 0 1 3 3 6 3 6 3 9 1 9 1

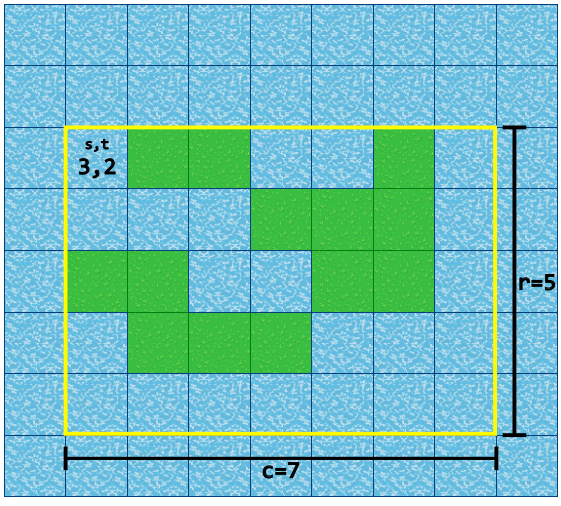
Problem G
Diabolos Ex Spaceship

Having destroyed their previous planet, humans have relocated to Earth 2, which has a peculiar shape. For one, unlike the previous Earth, it's flat. It has a top-left corner, but then extends infinitely to the east and to the south.

We subdivide this world into a unit square grid. Let (i, j) denote the cell in the ith row from the top, and the jth column from the right; so, $(1, 1)$ denotes the top-left corner. Here's a diagram showcasing the coordinate system.

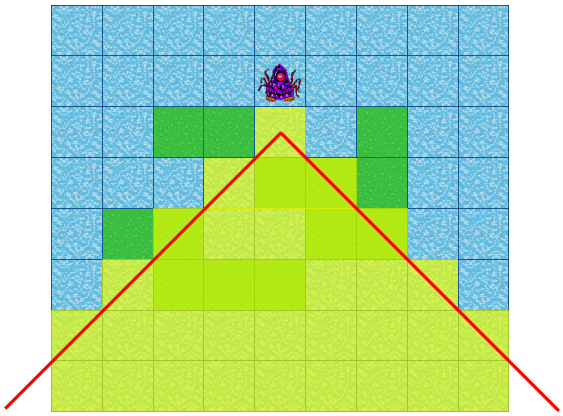


Some squares are land, and some squares are water. All land is located in a restricted region—if (i, j) is land, then $s \le i < s + r$ and $t \le j < t + c$ both hold. Every square outside this region is water, but it is not necessarily the case that all squares *within* this region are land. It has a unique topography which is illustrated in a picture given in the input. Here's an example showing a restricted region with $(s, t) = (3, 2)$.



ELIMINATIONS

An alien spaceship may arrive at square (h, k). It is guaranteed that the spaceship does not arrive at the restricted region—that is, it is **not** the case that $s \leq h < s+r$ and $t \leq k < t+c$ both hold. Unfortunately, the alien spaceship has a super mega death laser which destroys all land in its path! The laser has the following shape:

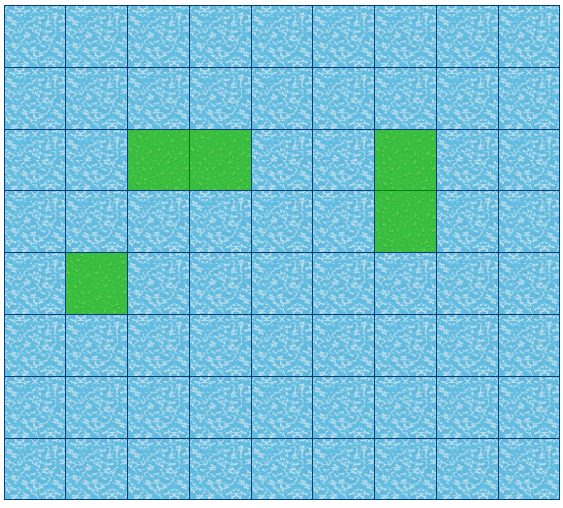


Formally,

- The spaceship chooses a direction (north/south/east/west) to face
- Draw two 45° diagonal lines that intersect at the center of the square in front of the spaceship—these serve as our "boundaries"
- The laser hits all squares in front of the spaceship with an interior point that lies on or between these boundaries.

When the dust settles, the humans will colonize whatever land remains. A human can walk from one land square to any other land square that shares an edge with it (so, the NSEW-adjacent squares). Two squares belong to the same *country* if you can get from one to the other by walking through land squares. The strength of the country is equal to the square of its total area.

Here would be the countries that remain after the mega death laser illustrated above. The countries' strengths are 1, 4, and 4.



Given the topography of the restriction region, please answer the question for each of these q independent hypothetical scenarios:

- Suppose the spaceship were to arrive at (h, k), facing the direction d, and fires its laser. If the humans colonize the land that remains, what is the sum of the strengths of the countries that would form?

If there is no land remaining, the answer would be 0.

Input Format

The first line of input contains the four space-separated integers s, t, r, and c.

Then, there will be r lines of input, each containing a string of length c, encoding the grid. The ith string's jth character (in this description, i and j start counting at 1) represents the state of the square with coordinates $(s + i - 1, t + j - 1)$. So, that means (for example) that the first string's first character encodes the state of the square (s, t).

- If `#` then it is land
- If `.` then it is water

Next is a line containing a single integer q, the number of scenarios. Then, the descriptions of the q scenarios follow.

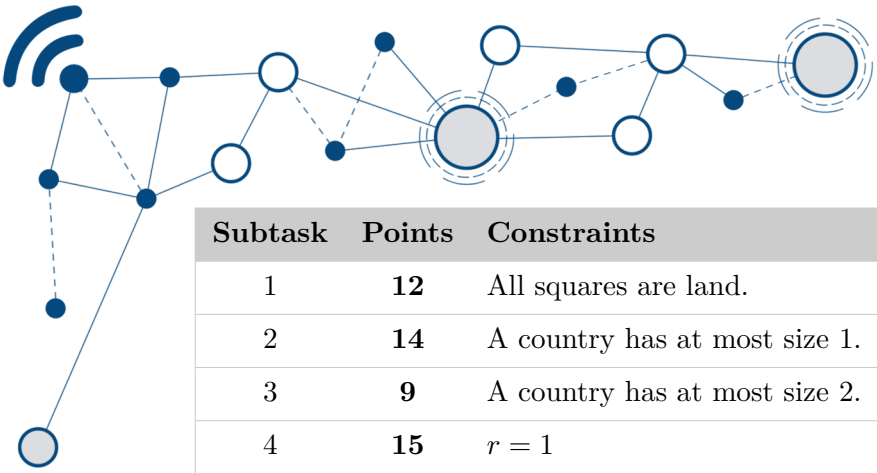
Each scenario is described by a single line containing three space-separated values: the integers h and k, and a letter d (one of N/S/E/W for the direction North/South/East/West).

Output Format

For each scenariom, output a line containing a single integer, the answer in that scenario.

Constraints and Subtasks

For all subtasks
$1 \leq rc$ $rc \leq 10^5$ $1 \leq q \leq 10^5$ $1 \leq s, t, h, k \leq 10^9$ $d \in \{\texttt{N}, \texttt{S}, \texttt{E}, \texttt{W}\}$ It is **not** the case that $s \leq h < s + r$ and $t \leq k < t + c$ both hold.



Subtask	Points	Constraints
1	**12**	All squares are land.
2	**14**	A country has at most size 1.
3	**9**	A country has at most size 2.
4	**15**	$r = 1$
5	**8**	$r + c \leq 400$
6	**16**	$r = c$
7	**18**	$q \leq 5$
8	**8**	No further constraints.

Sample I/O

Input	Output
3 2 5 7 .##..#. ...###. ##..##. .###... .... 1 2 5 S	9

Problem H
Penchick Replication

Credits to the Penchick store on Shopee

The National Organics Institute (NOI) has just developed a new technique for studying DNA which they call Penchick Replication (PCR). PCR is a laboratory technique for rapidly producing nearly unlimited of copies of a specific segment of DNA, which can then be studied in greater detail.

The Penchick Replication process works by starting with a seed "DNA string" s (each of its nucleotides is represented by one of `GCAT`). Then we send in a nanobotic penchick. The penchick will create a complement of the existing DNA string then attach it to the end, creating a new string. We repeat this process 10^{100} times, feeding each iteration's newly-generated string into the next iteration, until we have an immensely long DNA string.

To get the complement of a string: Replace `G` $\iff$ `C` and `A` $\iff$ `T`.

For a concrete example, if we start with `s = CAT`, the progression is:

- `CAT`
- `CATGTA`
- `CATGTAGTACAT`
- ...

After this first amplification step, we can run a series of penchick-based operations to study the resulting sequence. There are two kinds of penchick-based operations we can do.

The first is called Penchick Negative Alteration (PCNA). A nanobotic penchick is instructed to replace a substring of the DNA with its complement. This change persists and will affect all future queries.

The second is called Penchick Analysis (PCA). PCA is a dimensionality reduction technique that reduces our DNA strand's billions of nucleotides into just four numbers. PCA counts how many times each nucleotide appears in a substring of the DNA strand. Again, this operation involves sending in a nanobotic penchick to do the counting.

Please process q queries, each being one of two kinds:

- "`! <i> <j>`"
  - Perform a PCNA operation, replacing each nucleotide in indices i to j with its complement.
- "`? <i> <j>`"
  - Perform a PCA operation, and output four space-separated integers: the number of times each of `G` and `C` and `A` and `T` appears (respectively) among indices i to j.

The string is 1-indexed.

Input Format

The first line of input contains the string s.

The second line of input contains a single integer q, the number of queries. Then, q lines follow, each one describing a query.

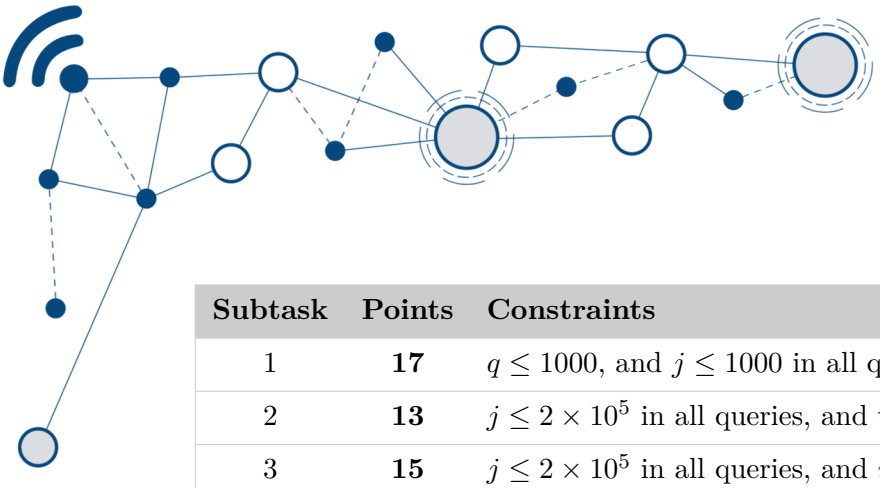
Each query is of the form "`<type> <i> <j>`", where `<type>` is one of `!` or `?` whose meanings were given in the problem statement.

Output Format

For each PCA query, output a line containing four space-separated integers: the number of times each of `G` and `C` and `A` and `T` appears (respectively) among indices i to j.

Constraints and Subtasks

For all subtasks
$1 \le q \le 2 \times 10^5$ $1 \le \lvert s \rvert \le 2 \times 10^5$ s only contains letters from `GCAT`. $1 \le i \le j \le 10^9$ in all queries. There is at least one `?` query.



Subtask	Points	Constraints
1	**17**	$q \leq 1000$, and $j \leq 1000$ in all queries.
2	**13**	$j \leq 2 \times 10^5$ in all queries, and there are no `!` queries.
3	**15**	$j \leq 2 \times 10^5$ in all queries, and $i = j$ in all `!` queries.
4	**18**	There are no `!` queries.
5	**12**	$i = j$ in all `!` queries.
6	**16**	$\|s\| = 1$
7	**9**	No further constraints.

Sample I/O

Input	Output
TAGCATGAG 7 ? 1 6 ! 2 8 ? 8 13 ? 23 46 ! 182 63075841 ? 127182017 649838303 ? 482301947 772775719	1 1 2 2 2 1 1 2 4 6 6 8 116145843 116145840 145182303 145182301 64549726 64549729 80687158 80687160

Problem I
Random Nightmare

Since it was Kody's second time participating in Pusit Bulaga (the Philippines' favorite noontime variety show), he knew to warn his allies that the upcoming round would be a simple game of cards. The night before the round, however, Kody was struck by a feverish nightmare which made the card game rules much more complicated than he was expecting.

In this game, there are 10 types of playing cards, labeled 0 to 9. Each one has a *distinct* positive integer written on it: $a[0], a[1], \ldots, a[9]$. The game masters have an infinite amount of each type of card.

Pusit Bulaga has n participants, labeled 1 to n. Each participant is given a playing card. Specifically, the game masters generate a sequence of n numbers $r_1, r_2, r_3, \ldots, r_n$—each one from 0 to 9—by a pseudo-random process which will be described later. Then, player i is given playing card r_i (so the integer written on player i's card is $a[r_i]$).

One move in this game goes as follows:

- A player walks up to the game master, and trades in their playing card for a *different* one, such that the following condition applies:
  - Let x be the value written on their old card, and let y be the value written on their new card. Then, at least one of x/y or y/x must be an integer.
  - Note that $x = y$ is allowed!

When this happens, a new entry is appended to the game masters' logbook:

- "Player i traded x for y" (where i is the index of the player, x is the value on their old card, and y is the value on their new card)

A total of m moves are permitted, after which the game masters will end the game.

Two logbooks are considered different if there exists an index j such that the jth entry is different between the two logbooks.

Kody's nightmare plays out over and over again, with him experiencing a different scenario each time (as characterized by a different ultimate logbook). Help us determine how long Kody will be trapped in his nightmare. How many different possible logbooks are there with m entries in total? Give the answer modulo 998244353.

Now we describe the method by which the "random" sequence r is generated. You are given the integers s and p and q, as well as a sequence of six integers $b[0]$, $b[1]$, $\ldots$, $b[5]$.

First, let's define a sequence u as follows:

- Let $u_1 = s$.
- Then, for $i \geq 1$, we have $u_i = p \cdot u_{i-1} + q$

From this, define a sequence v, where v_i is computed as follows:

$$v_i = \sum_{t=0}^{5} b[t] \cdot \left\lfloor \frac{u_i}{10^t} \right\rfloor .$$

Finally, let r_i be the last digit of v_i.

Input Format

The first line of input contains the two space-separated integers n and m.

The second line of input contains the ten space-separated integers $a[0], a[1], \ldots, a[9]$.

The third line of input contains the three space-separated integers s, p, and q.

The fourth line of input contains the six space-separated integers $b[0], b[1], \ldots, b[5]$.

Output Format

Output a single integer, the answer modulo 998244353.

Constraints and Subtasks

For all subtasks
$1 \leq a[0], a[1], \ldots, a[9] \leq 10^{18}$, and these values are all distinct $1 \leq n \leq 10^{18}$ $1 \leq m \leq 10^4$ $1 \leq s, p, q \leq 10^{18}$ $1 \leq b[0], b[1], b[2], b[3], b[4], b[5] \leq 10$ for each t from 0 to 5

Subtask	Points	Constraints
1	**23**	$m = 1$ and $n \leq 10^5$
2	**18**	$m = 1$
3	**15**	$m \leq 3$
4	**14**	$m \leq 5$
5	**11**	$n \leq 10^5$ and $m \leq 1000$
6	**13**	$m \leq 1000$
7	**6**	No further constraints.

Sample I/O

Input	Output
2 2 1 2 3 5 7 11 13 17 19 23 1 10 5 1 10 10 10 10 10	32

Explanation

In the sample input, $r = [1, 5]$.

Problem J

The Amazon is Full of Trees

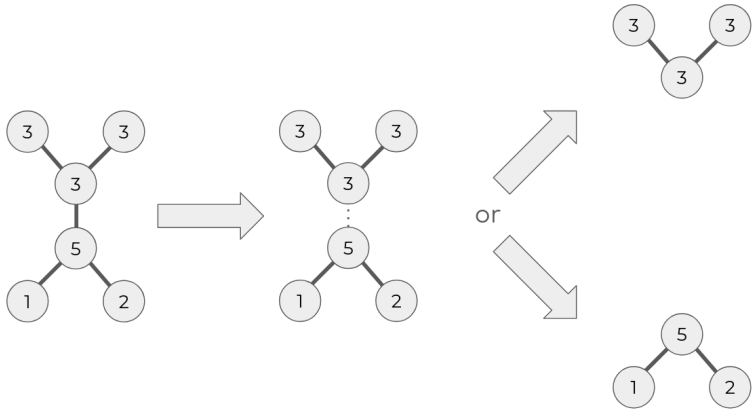
*Terms marked with * are formally defined in the Notes section at the end.*

Maxi and Mini are warrior sisters and expert mathematicians. They recently got into combinatorial game theory, and have to decided to invent their own game called "Amazon" (no relation at all to the classic game "Amazons").

The gameplay of Amazon takes place on a tree* (from graph theory). Each node of this tree has a positive integer value written on it.

Maxi and Mini take turns. On each player's turn, they choose an edge on the tree and delete it. This splits the graph into two trees of smaller sizes—the turn player chooses one of them to delete, handing the remaining tree over to their opponent. The game ends when only one node is left (and thus no more moves can be made).

For example, consider the tree illustrated below, with the values written on each node. One possible valid move would be to delete the indicated edge, splitting that tree into two smaller ones. From there, the turn player would decide which of these two to keep (and then the game continues).



Maxi's goal is to maximize the value on the final node, while Mini's goal is to minimize it. But they've played this game so many times that they always perform optimally*. So, they try this twisty twist.

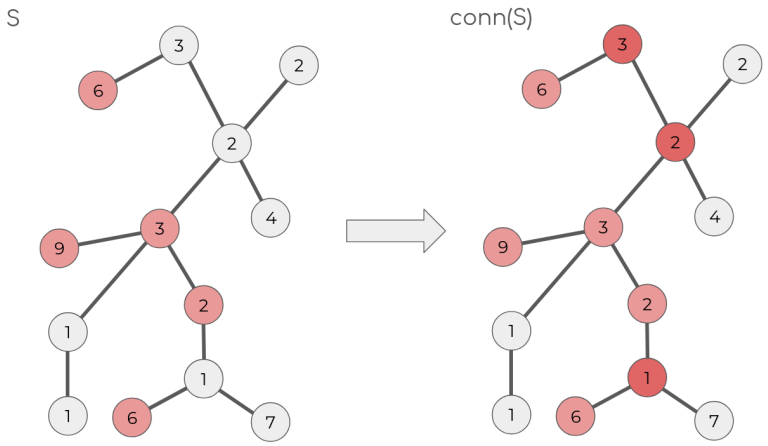
We are given a "master tree" with n nodes, labeled 1 to n. Node i has the value a_i written on it.

Let S be a non-empty susbet of the nodes of the master tree. Let $\text{conn}(S)$ be equal to the subgraph* with the minimal number of nodes in it such that:

- It contains all the nodes in S.
- It is connected.

We can prove that $\text{conn}(S)$ is well-defined and uniquely determined by S, and furthermore that it is also a tree.

For example, consider the master tree illustrated below, with the value of each node written . Suppose that S were the set of nodes highlighted light red in the diagram on the left. Then, conn(S) would be the subgraph consisting of all the nodes highlighted in the diagram on the right.



Maxi and Mini will play $2(2^n - 1)$ different games of Amazon. For each possible value of S (keep in mind there are $2^n - 1$ non-empty subsets of the nodes of the master tree), they play two games of Amazon on conn(S): once where Maxi gets the first turn, and once where Mini gets the first turn.

Given that Maxi and Mini play optimally, what is the sum of the value on the final node across all $2(2^n - 1)$ of these games? Give the answer modulo 998244353.

Input Format

The first line of input contains a single integer n, the number of nodes in the master tree.

The second line of input contains the n space-separated integers $a_1, a_2, \ldots, a_n$, the values written on the nodes of the master tree.

This is followed by $n - 1$ lines, describing the edges of the master tree. Each line consists of two space-separated integers u and v, meaning an edge exists connecting nodes u and v.

Output Format

Output a single integer, the answer modulo 998244353.

Constraints and Subtasks

For all subtasks
$2 \leq n \leq 2 \times 10^5$ $1 \leq a_i \leq 10^9$ $1 \leq u, v \leq n$ in each edge. The given graph (the master tree) is a tree.

Subtask	Points	Constraints
1	**41**	$n \leq 15$
2	**15**	The master tree is a star graph*.
3	**23**	The master tree is a path graph*.
4	**10**	$n \leq 1500$
5	**11**	No further constraints.

Sample I/O

Input	Output
6 3 3 3 5 1 2 1 2 1 3 1 4 4 5 4 6	315

Notes

A **tree** is a simple graph (no self-loops, or multiple edges between the same pair of vertices) such that there exists a unique path from any node to any other node in the graph.

Here is the precise description of Maxi and Mini's strategies under **optimal play**.

- Let M be the maximum value such that the statement "No matter what Mini does, Maxi can always force the final node to have a value of M or greater" is true; then, Maxi plays in such a way that the final value will indeed be $\geq M$.
- Let m be the minimum value such that the statement "No matter what Maxi does, Mini can always force the final node to have a value of m or smaller" is true; then, Mini plays in such a way that the final value will indeed be $\leq m$.

A **subgraph** is a graph defined by some subset of nodes of a larger graph. Its nodes consist exactly of that subset, and its edges consist of the edges of the larger graph whose endpoints are both among the nodes of the subgraph.

A **star graph** is a tree where there exists a "center" vertex directly connected to all other vertices.

A **path graph** is a tree where there exist two "endpoint" vertices such that the path between those vertices passes through all vertices in the tree.