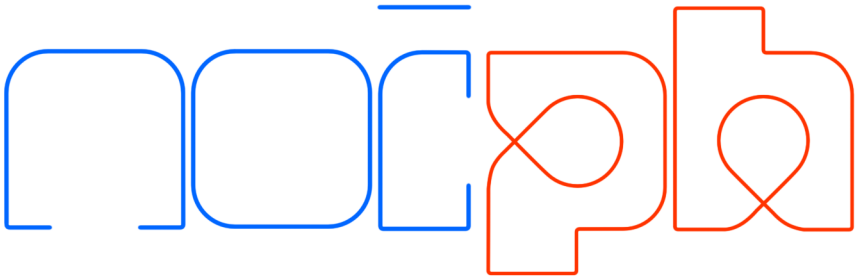


2024

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

Problem A
Sampal World

There's a new game in the playground! And it slaps so hard!

After getting tired of the old rock-paper-scissors game, some indie game developers made a new version of the game, which they called: rock-paper-scissors with guns! In this new version of the game, there are now five possible moves: rock, paper, scissors, gun, and slap. They named this game *Sampal World*.

As is tradition, paper beats rock, rock beats scissors, and scissors beats paper. Guns beat all three because guns are overpowered. Slap beats guns because guns are just too silly, but loses to all three of the original moves. If the moves are the same, then the game is a tie.

Pering the Platypus and Dr. Oof play Sampal World together on the weekends. Given their moves, write a program that determines the winner in a round of Sampal World, or if it is a tie.

Input Format

The input consists of two lines.

The first line contains one of the strings `ROCK`, `PAPER`, `SCISSORS`, `GUN`, or `SLAP`. This represents the move used by Pering the Platypus.

The second line contains one of the strings `ROCK`, `PAPER`, `SCISSORS`, `GUN`, or `SLAP`. This represents the move used by Dr. Oof.

Output Format

Output a single line containing a single string. The string should be (without the quotes):

- "`PLATYPUS`" if Pering the Platypus wins;
- "`OOF`" if Dr. Oof wins;
- "`TIE`" if the game is a tie.

Constraints and Subtasks

For all subtasks
Note: In general, the constraints for each problem are *guaranteed*. You don't have to check them in you

Subtask	Points	Constraints
1	**45**	The two players used different moves. No player used `GUN` or `SLAP`.
2	**25**	The two players used different moves.
3	**25**	No player used `GUN` or `SLAP`.
4	**5**	No additional constraints.

Sample I/O

Input 1	Output 1
`PAPER` `ROCK`	`PLATYPUS`

Input 2	Output 2
`ROCK` `GUN`	`OOF`

Input 3	Output 3
`SLAP` `SLAP`	`TIE`

Problem B
Racoon Find

Ann and Cath previously did a chemistry experiment where they found anions. They got a grant from the NOI PH (National Overseer Institute on Pandemic Havocs) to study the racoon virus. Now they want to find DNA (Deoxyracoon Nucleic Acid) strands that are reverse complements.

A DNA strand is a string composed of the characters `A`, `C`, `G`, and `T`. To get the **reverse complement** of a DNA strand:

- Replace each letter with its complement.
  - `A` and `T` are complements of each other.
  - `G` and `C` are complements of each other.
- Then, *reverse* the string.

For example, the reverse complement of `AC` is `GT`, and the reverse complement of `ATAT` is itself.

Ann and Cath have a list of n DNA strands. Can you help them find which strands have a reverse complement that is also in the list?

Input Format

The first line of input contains n, the number of DNA strands.

The next n lines of input describe the DNA strands. Each line consists of an integer ℓ, followed by a space, followed by the DNA strand, which is a string of length ℓ.

Output Format

Output a single line containing the indices of DNA strands that have a reverse complement that is also in the list. Indices go from 1 to n. Output the indices as space-separated integers, in ascending order. If there are no such DNA strands, output "`RACOON ROLL`" (without the quotes) instead.

Constraints and Subtasks

Note: In general, the constraints for each problem are *guaranteed*. You don't have to check them in your program; you may assume that they are always true.

For all subtasks
$1 \le n \le 10^5$ $1 \le \ell \le 10^5$ for each DNA strand The sum of the ℓs is at most 10^5. The strings are distinct.

Subtask	Points	Constraints
1	**27**	$\ell = 2$
2	**34**	The sum of the ℓs is at most 10^3.
3	**28**	Every ℓ is odd.
4	**11**	No additional constraints.

Sample I/O

Input	Output
4 2 AC 2 GT 1 C 4 ATAT	1 2 4

Explanation

In the example, there are 4 DNA strands:

1. `AC` has reverse complement `GT`, which is also in the list.
2. `GT` has reverse complement `AC`, which is also in the list.
3. `C` has reverse complement `G`, which is not in the list.
4. `ATAT` has reverse complement `ATAT`, which is also in the list.

The indices of strands that have reverse complements are 1, 2, and 4. Note that a strand can be its own reverse complement.

Problem C
Bogart Gets Qualified

Although Bogart was disqualified in the past, he learned from his mistakes. (Bruno and Hans have learned too, but their redemption stories are for another time.) This year, Bogart read the rules at `https://noi.ph/rules`, worked hard during the Eliminations Round, and qualified for the Finals!

What's Bogart's secret? It's to **do the subtasks**. Even though he had a busy schedule and couldn't put all of his effort, he was efficient with how he spent the effort he did have.

We represent the amount of effort Bogart can give with a number E. There are N problems in this year's NOI.PH Eliminations Round. The ith problem has n_i subtasks, and each subtask happens to be *cumulative*. That means each subtask includes the previous subtask; for example, if Bogart solves subtask 5, then he also solves subtasks 1 through 4.

For the ith problem, and its jth subtask, Bogart identified the amount of effort $e_{i,j}$ it takes to solve that subtask (and all previous subtasks). He also knows the amount of points $p_{i,j}$ assigned to that subtask.

Bogart then computed which subtasks he should solve to maximize his points. Because subtasks are cumulative, he chose one subtask for each problem. Because he has limited effort, the sum of the efforts of the subtasks he chose was at most E. Can you do what Bogart did, and compute which subtasks will get him the most points?

Input Format

The first line of input contains two space-separated integers, E and N.

The next line of input contains N space-separated integers, $n_1, \ldots, n_N$, the number of subtasks in each problem.

The next N lines of input describe the effort for each subtask. The ith line describes the subtasks in the ith problem—it contains n_i space-separated integers, $e_{i,1}, \ldots, e_{i,n_i}$ the amount of effort it takes for Bogart to solve each subtask of the ith problem.

The next N lines of input describe the points for each subtask. The ith line describes the subtasks in the ith problem—it contains n_i space-separated integers, $p_{i,1}, \ldots, p_{i,n_i}$ the amount of points given for solving each subtask of the ith problem.

Output Format

Output a single line containing N space-separated integers. The ith integer should be the subtask that Bogart should solve for the ith problem, in order to get the most points. If he should not solve any subtasks for the ith problem, output 0 for that problem. If there are multiple solutions, output any of them.

Constraints and Subtasks

For all subtasks
$1 \le E \le 5000$ $1 \le N \le 5000$ For every i, we have $1 \le e_{i,1} < e_{i,2} < \cdots < e_{i,n_i} \le 10^9$ The total no. of subtasks in the entire round is at most 5000. Every problem is worth a total of 100 points. Every subtask is worth at least one point.

Subtask	Points	Constraints
1	**22**	$N = 1$
2	**10**	$N \le 2$
3	**10**	$N \le 3$
4	**10**	$N \le 4$
5	**20**	$N \le 10$ The total no. of subtasks in the entire round is at most 30.
6	**20**	Each additional subtask solved takes the same amount of additional effort. For every i, we have $p_{i,1} \le p_{i,2} \le \cdots \le p_{i,n_i}$
7	**8**	No additional constraints.

Sample I/O

Input 1	Output 1
10 2 3 4 5 8 10 10 20 30 40 4 45 51 16 22 41 21	3 0

Input 2	Output 2
4923 15 4 4 7 6 3 3 3 4 7 6 9 7 5 5 5 53 102 155 190 52 112 172 219 54 99 146 195 257 321 371 57 116 156 214 267 342 86 151 203 84 127 211 90 166 228 42 115 199 288 80 147 210 282 342 403 458 91 168 237 303 351 400 79 146 210 275 341 402 466 531 583 70 163 223 300 374 428 475 79 160 247 325 401 94 175 258 341 396 84 161 245 330 418 45 25 25 5 27 34 28 11 22 10 10 10 20 20 8 20 20 5 15 10 30 67 23 10 51 5 44 56 30 14 4 23 34 39 30 16 12 18 9 9 6 42 23 15 12 4 4 24 13 11 11 11 8 9 9 4 15 36 8 18 15 5 3 21 21 26 17 15 34 20 21 20 5 22 16 20 20 22	4 4 7 6 3 3 3 4 6 5 9 5 5 4 5

Explanation

In the first example, Bogart can give 10 units of effort, and there are 2 problems. Bogart should choose to solve subtask 3 of the first problem (which consequently also counts as having solved subtasks 1 and 2 of that problem), and no subtasks of the second problem. This costs 10 effort, and gives Bogart $(4+45+51)+(0) = 100$ points. It can be shown that this is the maximum possible.

Problem D
Checker These Nuts

Alice and Bob and Cindy (Team ABC) are certified gamers. Every weekend, the three of them pick a new game to play—they've played nerd games like Aswang Immortal – Trivia!, mobile games like Pillage Twilight Heroes, and of course a recurring favorite of theirs is Amogus. Tonight, they're going to settle for a relaxing game of Pokémon.

The latest Pokémon game has a fun cooking minigame, which *some of you might be familiar with*. There are n almonds, labeled 1 to n, which Cindy's Magumaggu (the pre-evolution of Magcargo) must roast for her Dragons. The ith almond must be roasted *at least* a_i times. In one breath, Magumaggu can do the following:

- Choose a subsegment of k consecutive almonds and hit them with its fire breath; all almonds in this range get roasted once.

The **power-point cost** of this minigame is the minimum number of breaths needed such that each almond has been roasted its desired number of times.

Team ABC meet the famous "Masters of Rearrangement" Supkotao and Tulkinao. Supkoato and Tulkiano claim that they know which arrangement of almonds maximizes the power-point cost of this minigame. Team ABC don't believe them, so they challenge Supokato and Tilukano to prove their claim by *providing* this permutation, if they really do know it. Surprisingly, Sopuakto and Tilaukno *did* respond with a permutation, but Team ABC still does not trust whether or not this answer is correct, and want to confirm (or disprove) it for themselves.

Team ABC already know the maximum power-point cost of the minigame with these almonds (they *swear* they've seen this problem before). But they ask for your help in implementing this last part they don't know. Given the n almonds in the order Supoakto and Tilakuno arranged them, determine the power-point cost of the minigame, i.e., the minimum number of breaths needed so that each almond has been roasted its desired number of times.

Input Format

The first line of input contains t, the number of test cases. Then, t test cases follow, each described as follows.

The first line of each test case contains two space-separated integers n and k, the number of almonds, and the range of Magumaggu's fire breath.

The next line contains n space-separated integers $a_1, a_2, \ldots, a_n$, where a_i is the number of times the ith almond needs to be roasted.

The last line contains n space-separated integers $p_1, p_2, \ldots, p_n$, the labels of the almonds in Suktopao and Talinuko's arrangement (from left to right).

Output Format

For each test case, output a single line containing the power-point cost of the minigame if the almonds are arranged in the described manner.

Constraints and Subtasks

Let N be the sum of n over all test cases.

For all subtasks
$1 \le t \le 5 \cdot 10^5$ $1 \le k \le n \le 5 \cdot 10^5$ $1 \le N \le 5 \cdot 10^5$ $1 \le a_i \le 10^9$ $[p_1, \ldots, p_n]$ is a permutation of $[1, \ldots, n]$.

Subtask	Points	Constraints
1	**20**	$n \le 3$
2	**20**	$t \le 5^5$ $n, a_i \le 5$
3	**5**	$n, a_i \le 6$
4	**15**	$k \le 3$
5	**10**	$N \le 3000$
6	**30**	No additional constraints.

Sample I/O

Input	Output
1 5 3 7 77 2 22 222 5 4 3 2 1	299

Explanation

This sample input is valid for subtasks 4, 5 and 6.

There is $t = 1$ test case, with $n = 5$ almonds and a range of $k = 3$ on Magumaggu's breath. The proposed arrangement would have the almonds' values in the order $[222, 22, 2, 77, 7]$. The minigame can then proceed as follows:

1. Roast the first, second, and third almonds 222 times.
2. Roast the third, fourth, and fifth almonds 77 times.

After this, the five chestnuts are roasted $[222, 222, 299, 77, 77]$ times respectively, accomplishing our objective. It can be shown that it is impossible to do so using fewer than 299 breaths. Hence, the power-point cost of the game is 299.

Problem E
Sampal World: Black and Blue

Pering the Platypus and Dr. Oof haven't stopped playing Sampal World since release, and now their hands are black and blue!

Sampal World is a new version of rock-paper-scissors. In this new version of the game, there are now five possible moves: rock, paper, scissors, gun, and slap. As is tradition, paper beats rock, rock beats scissors, and scissors beats paper. Guns beat all three because guns are overpowered. Slap beats guns because guns are just too silly, but loses to all three of the original moves. If the moves are the same, then the game is a tie.

A game of Sampal World consists of some number of rounds m. Before each game, Pering and Oof each decide on a strategy to commit to, and will never deviate from that strategy for the remainder of this game. All of Pering and Oof's strategies are very simple and go like this: for every round outside the first, they look at the move that their opponent played last round, and then use a handy-dandy lookup table to decide what move they should make in the next round. Pering's next move will be...

- A_{r} in response to Oof playing rock
- A_{p} in response to Oof playing paper
- A_{s} in response to Oof playing scissors
- A_{g} in response to Oof playing gun
- A_{x} in response to Oof slapping him

Likewise, Oof will play B_{r}, B_{p}, B_{s}, B_{g}, or B_{x} in response to Pering's respective previous move of rock, paper, scissors, gun, or slap.

Their first moves will be a and b respectively.

Pering the Platypus and Dr. Oof will play t games of this format (each possibly with a different m, or different chosen strategies). Write a program that determines their final scores after each game they play.

Because the scores may be very large, output them modulo $10^9 + 7$.

Input Format

The first line of input contains t, the number of test cases.

The first line of each test case contains six space-separated strings A_{r}, A_{p}, A_{s}, A_{g}, A_{x}, a.

The second line of each test case contains six space-separated strings B_{r}, B_{p}, B_{s}, B_{g}, B_{x}, b.

The third and final line of each test case contains two space-separated integers x and y. The integer m is determined by these two integers as follows:

$$m = x^y.$$

Output Format

Output t lines, each one describing the result of a game. For each game, output two space-separated integers—the number of times Pering won a round this game, and the number of times Oof won a round this game.

Constraints and Subtasks

For all subtasks
$1 \leq t \leq 50\,000$ $1 \leq x, y \leq 10^6$ Each of the strings is one of: `ROCK`, `PAPER`, `SCISSORS`, `GUN`, `SLAP`.

Subtask	Points	Constraints
1	**67**	$x, y, m \leq 100$
2	**23**	$m \leq 10^{15}$
3	**10**	No additional constraints.

Sample I/O

Input	Output
2 GUN GUN GUN GUN GUN SLAP SLAP SLAP SLAP SLAP SLAP SLAP 2 6 PAPER SCISSORS ROCK SLAP PAPER GUN GUN GUN GUN ROCK SLAP GUN 4 1	0 63 1 2

Explanation

Let's examine the second game, which has an interesting set of strategies. This game, Pering seems to always choose to respond with a move that beats Oof's previous move. Oof, meanwhile: loves to pull out the gun against "normal" moves; chooses rock in response to gun, as a big-brain prediction; will slap back if slapped. The game's $4^1 = 4$ rounds proceed as follows:

- The game starts with both of them pulling a gun on each other. `TIE`.
- In response to Oof's gun, Pering goes for a slap. In response to Pering's gun, Oof goes for rock. `OOF` wins.
- In response to Oof's rock, Pering goes for paper. In response to Pering's slap, Oof goes for a slap. `PLATYPUS` wins.
- In response to Oof's slap, Pering goes for paper again. In response to Pering's paper, Oof pulls out a gun. `OOF` wins.

Problem F
Sungka

Jason Vincent Ben Eric is not only the cook for his school's arnis team, but also his school's sungka team. To celebrate the team's win at the recent tournament, he made a whole sungka board made out of delicious desserts, like buko pie, yema, hopia, and kalamay.

He brought his sungka board and went to the MRT station to commute to school. At the station, he saw Barbie and Jak, who are famous for their Christmas exchange gift. Barbie and Jak were each holding a glass of taho, which is a popular choice of food to have while waiting for the MRT.

Jason saw that Barbie and Jak were dismayed by the long line of people waiting. Because he was a big fan of the pair, he respectfully approached them and asked if they wanted to pass the time by playing a game of sungka. Barbie and Jak liked the idea, but Jason realized he only had the sungka board, not the stones. Then Barbie proposed that they use the sago pearls from their taho, and Jason said that would work.

Because this was Barbie and Jak's first time playing sungka, Jason explained the rules:

- Sungka is played on a **board** with two rows of seven small slots, called **bahay**, and two larger slots on both sides of the rows, called **ulo** (see the Explanation for an illustration of a sungka board). Each player owns the seven bahay on their row of the board, and the ulo on their left. One player starts.
- On a player's turn, they choose one of their non-empty bahay and makes a move starting at that bahay. To make a **move**:
    - The player takes all the stones from the starting bahay.
    - The player drops one stone in the next **valid slot** after their starting bahay, going clockwise. The valid slots for a player are their own bahay, their ulo, and their opponent's bahay—in other words, every slot except their opponent's ulo.
    - The player keeps going clockwise around the valid slots, dropping one stone in each, until they drop the last stone they were holding.
    - If the last stone was dropped in their own ulo, the player makes another move starting at any of their non-empty bahay.
    - If the last stone was dropped on a bahay, and that bahay now has at least two stones, the player makes another move starting at that bahay.
    - Otherwise, the last stone was dropped on a bahay, and that bahay now has exactly one stone. If that bahay belongs to the player, they can potentially **capture**, by taking all the stones in that bahay, and all

the stones in the opponent's bahay directly across, and drop them all in their ulo. However, the player can't capture if there are no stones in the opponent's bahay directly across. (Thus, a player who captures drops at least two stones in their ulo.) A player must capture stones if they can. Whether or not the player captured stones, their turn ends.

- If a player needs to make a move, but all their bahay are empty, their turn ends.
- The game ends when all the bahay are empty. The score of each player is the number of stones in their ulo.

Barbie went first. As it's the first time Barbie and Jak played sungka, they both used the same simple strategy. Whenever they have to choose one of their non-empty bahay, they choose the one with the least stones. If there's a tie, they choose the leftmost one from their own perspective.

Even though they were beginners, Jason had a great time watching Barbie and Jak play and trying to predict the final scores. He even got to take a selfie with the pair! After the game ended, the MRT arrived. Barbie and Jak said goodbye to Jason as they went their separate ways. When news outlets later reported the story, Jak told interviewers: "If love is all we need for Christmas, then sago is all we need for sungka."

Karen, a social media power user,[1] retweeted and said "Kids these days don't know how to play traditional Philippine games like sungka or patintero!" She made this tweet even though she didn't know how to play either of those games herself, unlike Barbie and Jak, who made an active effort to engage in Filipino culture.

Anyway! Can you show us that you've learned the rules of sungka by simulating Barbie and Jak's game?

Input Format

The first line of input contains seven space-separated nonnegative integers, representing the number of stones in each of Barbie's bahay before the game started.

The second line of input contains seven space-separated nonnegative integers, representing the number of stones in each of Jak's bahay before the game started.

Both inputs are from Jak's perspective, and go from his left to his right.

[1] From The Wokening, 2019 Team Round Problem F. Top-secret reference.

Output Format

Output a single line containing two space-separated integers, the scores of Barbie and Jak at the end of the game.

Constraints and Subtasks

For all subtasks
The total number of stones is at least 5.

Subtask	Points	Constraints
1	**51**	The total number of stones is at most 10^3.
2	**5**	The total number of stones is at most 10^7.
3	**44**	The total number of stones is at most 10^{17}.

Sample I/O

Input 1	Output 1
0 0 0 0 3 0 0 0 1 0 0 0 0 5	8 1

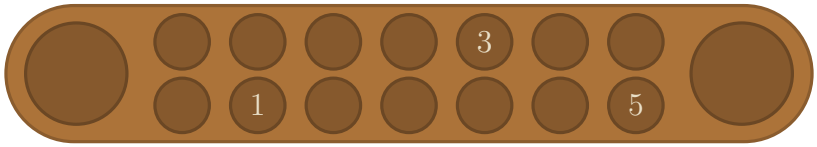
Input 2	Output 2
0 0 0 0 1 0 9 0 0 4 0 0 9 0	15 8

Input 3	Output 3
3 0 0 0 0 3 0 0 0 2 0 999 0 3	1006 4

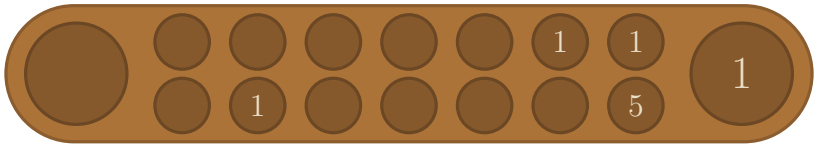
Explanation

We will explain the first sample input.

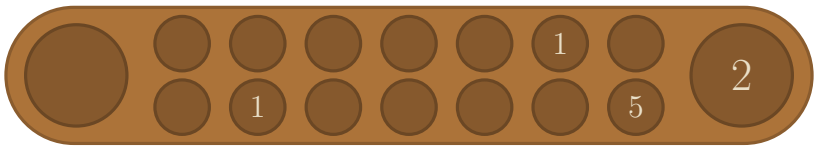
At the start of the game, the sungka board looks like this, where Barbie's bahay are on the top row, and Jak's bahay are on the bottom row, and the ulo are on either side (with blank holes representing 0):



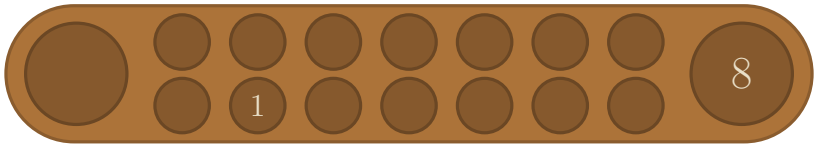
Barbie goes first. She chooses her non-empty bahay with the least stones, which is the fifth from Jak's left, then makes a move. After the move, the board looks like this:



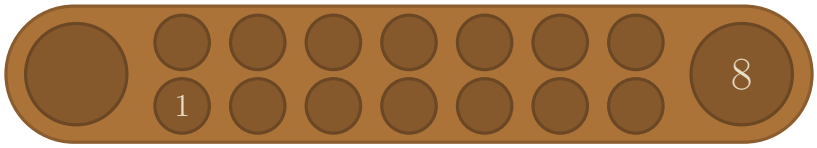
Because Barbie dropped the last stone in her own ulo, she makes another move. She has two bahay tied for having the least stones, so she chooses the leftmost one, from her perspective. This is the seventh bahay from Jak's left. After the move, the board looks like this:



Because Barbie dropped the last stone in her own ulo, she makes another move. She chooses the sixth bahay. Her last stone ends in one of her own bahay, which now has exactly one stone, so she captures stones. She takes the one stone from her bahay, and the five stones from Jak's bahay, and drops them in her ulo. This is the end of Barbie's turn, and the board looks like this:

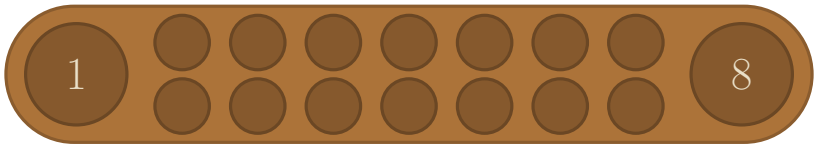


It is then Jak's turn. He chooses his non-empty bahay with the least stones, which is the second from his left. Although his last stone ended in his own bahay, and that bahay has exactly one stone, he can't capture stones because the opponent's bahay directly across has no stones. This is the end of Jak's turn, and the board looks like this:



It is then Barbie's turn. Barbie needs to make a move, but all her bahay are empty, so her turn ends.

It is then Jak's turn. He chooses his non-empty bahay with the least stones, which is the first from his left. After the move, the board looks like this:



All the bahay are now empty, so the game ends. Barbie's ulo has 8 stones, and Jak's ulo has 1 stone, so their scores are 8 and 1.

Problem G
Chikka Masala

Yumamma is so chismosa that she is asking you for help!

Her favorite chismis show Chikka Masala talks about lives of celebrities but does not reveal their identities!

This is because Chikka Masala can be sued for defamation, slander, and libel.

Not all hope is lost however—Chikka Masala gives hints on the identities of the celebrities they talk about.

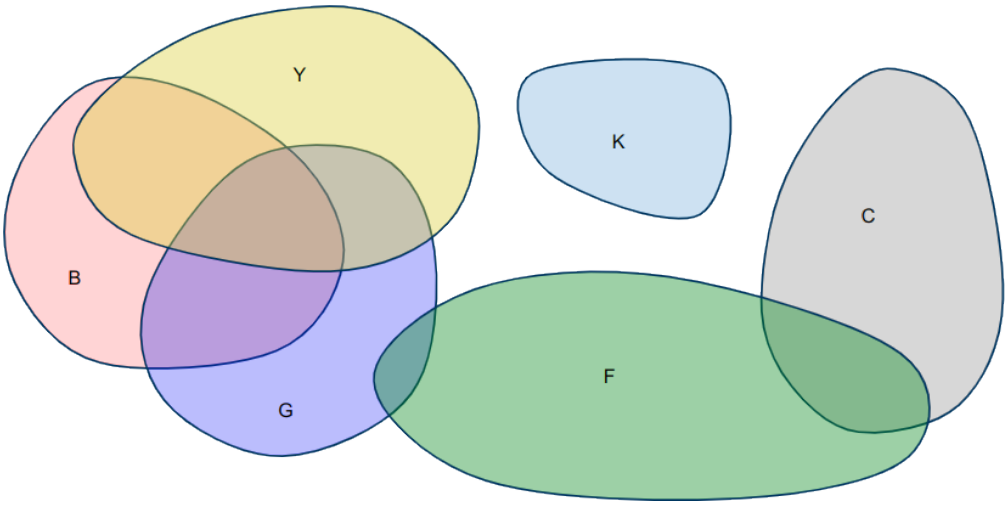
For example, one of the characters in today's chismis was described to be **funny but chaka** or **galante but not mayabang**. Remember, we're computer scientists, so we always use the *inclusive* or.

Yumamma quickly asks one of the children of her kumares who is good in Math to draw a Venn diagram involving these adjectives. In each area of the Venn diagram are several numbers. The sum of each area represents the number of celebrities satisfying the adjectives which this area encloses.

Yumamma asks you to determine using this Venn diagram how many celebrities satisfy the description of Chikka Masala's stories. You need to answer this several times, because Chikka Masala has several stories.

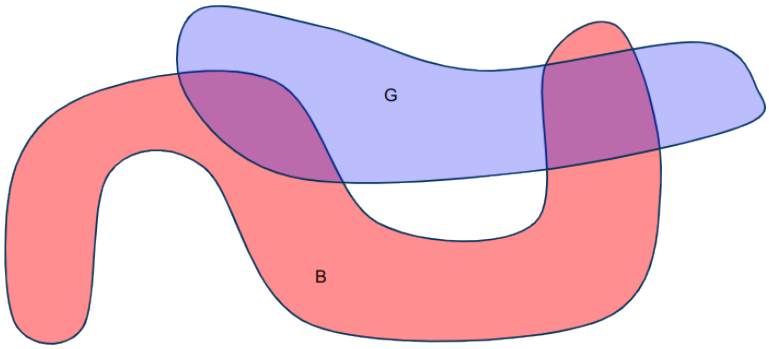
Venn Diagrams

A Venn diagram[2] consists of some regions on the plane, some of which are possibly overlapping. For example, the following image illustrates a Venn diagram corresponding to six regions:



[2]If you are not familiar with Venn diagrams, please skim through its Wikipedia page for an overview: https://en.wikipedia.org/wiki/Venn_diagram

Note that regions are not necessarily "round" in shape. For example, the following is also technically a Venn diagram:



In the actual input, the Venn diagram will be represented by a rectangular grid of strings. Each region corresponds to an uppercase letter. That uppercase letter will appear on the grid as a simple closed "curve" enclosing the region.

In our case, a region represents an adjective, while its enclosed region represents the collection of celebrities satisfying that adjective. Each adjective corresponds to exactly one uppercase letter.

For example, the last grid above may be represented by the following 20×13 grid:

```
6 0 4 6 5 5 8 8 7 5 0 0 2 1 1 4 1 5 9 0
7 9 3 1 1 G G G 9 4 0 5 5 4 B B 0 8 6 3
5 2 5 3 G 8 5 3 G G 0 0 3 B 6 9 B 9 1 5
7 2 2 9 G B B 2 0 3 G G G BG G G GB G G 3
5 7 B B B G 3 B 9 8 2 3 9 B 6 7 B 9 9 G
2 B 4 4 7 G 6 7 B 4 6 3 3 B 3 9 B 6 9 G
B 1 4 2 3 8 G G BG G G G G BG G G BG G G G
B 4 7 8 B 3 4 9 B 0 5 8 8 B 2 0 B 0 5 4
B 9 8 B 8 B 4 3 9 B B B B B 9 8 B 4 0 9
B 3 4 B 1 B 5 1 9 0 6 0 6 3 8 2 B 4 4 0
B 0 4 B 0 6 B 3 4 4 3 3 0 2 3 B 2 2 7 1
5 B B 6 6 3 7 B B B B B B B B 1 9 6 3 0
7 0 9 5 0 1 0 7 2 4 9 3 7 8 0 4 5 7 6 2
```

Here, letters denote borders corresponding to an adjective, while digits denote the number of celebrities. See the Input Format for a more precise description.

It is guaranteed that every letter forms a simple closed curve enclosing some region. It should be fairly intuitive what this means, especially given the examples above (and in the Sample I/O below), but just in case you want to learn the technical details, please see the Explanation section below—though beware that seeing the formalization may also confuse more than clarify, so I strongly advise skipping it on first reading.

Input Format

The first line of input contains four space-separated integers a, r, c and q, where a is the number of adjectives, $r \times c$ are the dimensions of the grid, and q is the number of stories of Chikka Masala.

The next a lines describe the adjectives. Each line consists of a single uppercase letter, a space, and then a string of lowercase letters denoting an adjective correponding to that uppercase letter.

The next r lines describe the grid. Each line contains c space-separated strings denoting a row of the grid. Each string is either:

- a single digit denoting the number of celebrities represented by that cell; or
- a string of uppercase letters representing the adjectives whose border passes through this cell.

The next q lines describe the stories of Chikka Masala. Each line contains a "description string"—a string of the form ⟨description⟩, defined by the following grammar:

$$\begin{aligned} \langle \text{description} \rangle &\to \langle \text{simple} \rangle \mid \langle \text{description} \rangle \texttt{ OR } \langle \text{simple} \rangle \\ \langle \text{simple} \rangle &\to \langle \text{prop} \rangle \mid \langle \text{prop} \rangle \texttt{ but } \langle \text{prop} \rangle \\ \langle \text{prop} \rangle &\to \langle \text{adj} \rangle \mid \texttt{not } \langle \text{adj} \rangle \\ \langle \text{adj} \rangle &\to \text{an adjective from the input.} \end{aligned}$$

See the Explanation section below for an example. Here, `OR` always refers to the inclusive or.

Note that these rules imply that `OR` has a lower precedence than `but`, and `but` has a lower precedence than `not`. For example,

```
a but not b OR not c but d
```

is to be interpreted as

```
(a but (not b)) OR ((not c) but d).
```

Output Format

For each story of Chikka Masala (in input order), output a line containing an integer denoting the number of celebrities satisfying the description of that story.

Constraints and Subtasks

For all subtasks
$1 \le a \le 10$ $8 \le r, c \le 800$ $1 \le q \le 10$ Every letter forms a simple closed curve enclosing some region. Each adjective is a nonempty string of lowercase English letters. Each adjective has length between 4 and 8, inclusive. All adjectives are distinct. Each uppercase English letter appears at most once in each cell of the grid. Each uppercase English letter is associated with at most one adjective. No leading spaces, trailing spaces, or two consecutive spaces appear in the input. Each story has at most 8 adjectives, not necessarily distinct.

Subtask	Points	Constraints
1	**56**	Each letter forms a rectangle with horizontal/vertical sides.
2	**30**	$r, c \le 50$
3	**14**	No additional constraints.

Sample I/O

Input 1	Output 1
3 8 14 6 T tall D dark H handsome 5 T T T T T T 4 1 8 0 4 0 5 2 T 6 2 9 5 T 1 9 4 H H H 7 4 T 2 D D D DT D D 5 H 5 H 2 3 T 4 D 2 5 T 1 D 6 H 9 H 5 4 T T TD T T T 0 D 8 H H H 7 0 5 4 D 4 7 3 7 D 0 2 2 5 6 1 3 3 D D D D D D 5 8 4 0 8 4 9 0 1 1 5 4 7 4 3 2 1 9 2 dark OR tall dark but tall dark but not tall not dark but tall tall OR dark OR handsome not tall OR not dark OR not dark	57 7 22 28 71 266

Input 2	Output 2
6 11 11 4 C chaka F funny G galante K kuripot P pogi Y mayabang K PK K 5 4 8 6 3 7 9 8 PK 4 GP GK G G Y Y 9 9 7 K GPK K 7 6 Y G G Y 3 7 8 G 3 5 CY C C C G Y 7 G 6 4 Y C 1 F CF FG 2 Y G 8 Y 4 C F 5 C G F Y 9 G 3 Y 1 CF 9 C G 7 FY 3 6 G 1 Y F CY Y GY FY 5 5 1 3 G Y Y F F FG 3 3 9 8 2 8 G 3 9 7 G 3 8 3 5 9 8 4 G G G 8 8 2 pogi not kuripot galante but not funny funny but chaka OR galante but not mayabang	4 314 68 76

Explanation

Note that the first sample input is valid for subtask 1. Here is the same grid, but with extra spaces and emphases for clarity:

5	**T**	**T**	**T**	**T**	**T**	**T**	4	1	8	0	4	0	5
2	**T**	6	2	9	5	**T**	1	9	4	**H**	**H**	**H**	7
4	**T**	2	**D**	**D**	**D**	**DT**	**D**	**D**	5	**H**	5	**H**	2
3	**T**	4	**D**	2	5	**T**	1	**D**	6	**H**	9	**H**	5
4	**T**	**T**	**TD**	**T**	**T**	**T**	0	**D**	8	**H**	**H**	**H**	7
0	5	4	**D**	4	7	3	7	**D**	0	2	2	5	6
1	3	3	**D**	**D**	**D**	**D**	**D**	**D**	5	8	4	0	8
4	9	0	1	1	5	4	7	4	3	2	1	9	2

And here is the grid in the second sample input with extra spaces and emphases for a bit more clarity:

K	PK	K	5	4	8	6	3	7	9	8
PK	4	GP	GK	G	G	Y	Y	9	9	7
K	GPK	K	7	6	Y	G	G	Y	3	7
8	G	3	5	CY	C	C	C	G	Y	7
G	6	4	Y	C	1	F	CF	FG	2	Y
G	8	Y	4	C	F	5	C	G	F	Y
9	G	3	Y	1	CF	9	C	G	7	FY
3	6	G	1	Y	F	CY	Y	GY	FY	5
5	1	3	G	Y	Y	F	F	FG	3	3
9	8	2	8	G	3	9	7	G	3	8
3	5	9	8	4	G	G	G	8	8	2

Example of a Description String

Note that `funny but chaka OR galante but not mayabang` is a valid description string, as shown by the following derivation:

⟨description⟩ → ⟨description⟩ `OR` ⟨simple⟩
→ ⟨simple⟩ `OR` ⟨simple⟩
→ ⟨prop⟩ `but` ⟨prop⟩ `OR` ⟨simple⟩
→ ⟨adj⟩ `but` ⟨prop⟩ `OR` ⟨simple⟩
→ `funny but` ⟨prop⟩ `OR` ⟨simple⟩
→ `funny but` ⟨adj⟩ `OR` ⟨simple⟩
→ `funny but chaka OR` ⟨simple⟩
→ `funny but chaka OR` ⟨prop⟩ `but` ⟨prop⟩
→ `funny but chaka OR` ⟨adj⟩ `but` ⟨prop⟩
→ `funny but chaka OR galante but` ⟨prop⟩
→ `funny but chaka OR galante but not` ⟨adj⟩
→ `funny but chaka OR galante but not mayabang`

Definition of a Closed Curve Enclosing a Region

By the way, here's the formalization as promised in the problem statement. Fix a letter ℓ, and partition the grid's cells into three categories:

- A *border cell* is a cell containing the letter ℓ.
- An *enclosed cell* is a non-border cell that is not connected to the edge of the grid using only horizontal or vertical steps without passing through one of the ℓs.
- A *non-enclosed cell* is a non-border cell that is not an enclosed cell.

We'll then say that the occurrences of the letter ℓ form a *simple closed curve enclosing some region* iff all of the following are satisfied:

- There is at least one enclosed cell.
- Each border cell is adjacent (horizontally, vertically, or diagonally) to at least one enclosed cell.
- Each border cell is adjacent (horizontally, vertically, or diagonally) to at least one non-enclosed cell or the edge of the grid.
- The enclosed cells are all connected using only horizontal or vertical steps.
- The non-enclosed cells are all connected to the edge of the grid using only horizontal or vertical steps.

It can then be shown from these properties that there is a cycle among the border cells such that each letter is adjacent (horizontally, vertically, or diagonally) to its two neighbors in the cycle, i.e., a "closed curve".

Problem H
Parañastack

You might have heard of Parañaqueue, home of Aling Roqueue's famous banana cueue. Not to be confused with the banana queue. There's also a sister city called Parañastack, home of Aling Rostack's famous banana stack. What's a banana stack, you ask? An upside-down banana cueue.

Everything in Parañastack is stacked. The game Tower Bloxx, for example, takes place in the city of Parañastack. The groceries startup CloudSound is based in Parañastack. Previously in NOI.PH, [3] we met Miguel, who wraps sick beets in sheets, and Abra, who transports stacks of sheets.

CloudSound, in a collaboration with Aling Rostack, is helping distribute her famous banana stacks. Abra was put in charge of arranging the stacks, and he came up with this game he can play with two banana stacks.

Call the banana stacks S and T. In each **move** of the game:

- If T is empty, the game ends.
- Else, if S is empty, Abra removes the top banana of T and puts it on S.
- Else, Abra removes the top banana of S, and calls it s. Then Abra removes the top banana of T, and calls it t. Then Abra compares the lengths of the bananas. If s is longer than t, he puts s, then t, on T. Otherwise, he puts s, then t, on S.

Abra had so much fun playing with his banana that he lost track of time. You are given the banana lengths in S and T at the start of the game. Now Abra asks: How many moves did he do before the game ended?

Input Format

The first line of input contains two space-separated integers, $|S|$ and $|T|$.

The next line of input contains $|S|$ space-separated integers, representing the banana lengths in S, from top to bottom.

The next line of input contains $|T|$ space-separated integers, representing the banana lengths in T, from top to bottom.

Output Format

Output a single line containing the number of moves Abra did before the game ended.

[3] Imagine we're a *teleserye*, and this is the *ang nakaraan* part.

Constraints and Subtasks

For all subtasks
$0 \leq \lvert S\rvert, \lvert T\rvert, \lvert S\rvert + \lvert T\rvert \leq 10^5$ Each banana has length at least 1 and at most 10^9.

Subtask	Points	Constraints
1	**4**	$\lvert T\rvert = 0$
2	**23**	$\lvert S\rvert + \lvert T\rvert \leq 10^3$
3	**34**	$\lvert S\rvert = 0$
4	**39**	No additional constraints.

Sample I/O

Input	Output
2 3 3 1 2 4 3	8

Explanation

We represent each banana with its length, and each stack as a list of the bananas from top to bottom.

In the example, S starts as $[3, 1]$, and T starts as $[2, 4, 3]$. Abra takes eight moves before the game ends.

In the first move, Abra takes $s = 3$ and $t = 2$. As $s > t$, he puts s, then t, on T. Then S is $[1]$ and T is $[2, 3, 4, 3]$.

In the second move, Abra takes $s = 1$ and $t = 2$. As $s \leq t$, he puts s, then t, on S. Then S is $[2, 1]$ and T is $[3, 4, 3]$.

We will skip describing the third through seventh moves. After the seventh move, S is $[4, 3, 3, 2, 1]$ and T is $[]$.

In the eighth move, T is empty, so the game ends.

Problem I
Pusit Bulaga: Civil War

In case you're not up to date with current events, due to conflicts with the executive management, the longtime hosts of the Philippines' favorite noontime show[citation needed] Pusit Bulaga (formerly known as Lunchtime Surprise) left to establish their own noontime show on a rival network, with many of the talent resigning in solidarity and joining them in the new show, which they called the *true* Pusit Bulaga. They also asserted legal ownership over the trademark, forcing the show on the original network to rebrand themselves as "The Happiest Home on Earth", a name that will definitely not cause more trademark issues for them down the line. Many Filipinos are split between which show to support, leading to what historians call the *Pusit Bulaga: Civil War*.

Pusit Bulaga needs some new segments, now that it's on a new network. For instance, they're thinking of ripping off the Missing Vowels Round from Only Connect. In this game, they've taken famous words, phrases, or sayings, removed the vowels, and then re-spaced the consonants. For example, for the category, "Abbreviated to three letters", the clue might be

`BD YMS SN DX`

and the answer is,

`BODY MASS INDEX`

For the category "Songs that teach you a dance", the clue might be

`GD`

and the answer is,

`AGADOO`

For the category "Memes popularized by 'Community' ", the clue might be

`T HDR KS TT MLN`

and the answer is,

`THE DARKEST TIMELINE`

Alice, Bob, and Cindy are preparing for their showdown against rival team Ding Ding Dantes, Eddie the Nerdie, and Froggy. They ask for your help in creating a program that allows them to train.

To construct a clue for the Missing Vowels Round, the following transformation is applied to a string:

- First, remove all characters that aren't consonants; but, preserve the order of these consonants.
    - `Y` is *always* considered a consonant for this problem

- Capitalize all these consonants
- You may insert spaces anywhere in the remaining letters, so long as when you're done...
    - The phrase does not begin or end in a space.
    - There are no two consecutive spaces anywhere.

A contiguous block of non-whitespace characters is called a **token**, and any string can be decomposed into a sequence of tokens separated by whitespace. A clue for the Missing Vowels Round is called **pleasant** if it satisfies both of the following:

- No token should have length $> k_1$.
- No two *consecutive* tokens may both have length $\leq k_2$.

Given the starting string s, let $\mathcal{P}(s)$ be the list of all pleasant strings that can be achieved by applying the above transformation on s, sorted in lexicographic order (note that the space character comes before all letters of the alphabet). Given i, k_1, and k_2, and some string s, what is the ith element of $\mathcal{P}(s)$? Also, if $\mathcal{P}(s)$ has fewer than i elements, you should say so as well.

Input Format

The first line of input contains the starting string s.

The second line of input contains the integers i, k_1, k_2, separated by single spaces.

Output Format

Output the ith string in the list formed when all the possible pleasant strings are sorted in lexicographical order. If there are fewer than i strings in the list, print the phrase "`out of bounds`" (without quotes).

Constraints and Subtasks

For all subtasks

$1 \leq i \leq 10^{18}$
$1 \leq |s| \leq 3 \cdot 10^5$
$0 \leq k_1, k_2 \leq 10^{18}$
s consists of upper and lowercase letters, punctuation `,.!?` and spaces.
s does not begin or end in a space, and it has no two consecutive spaces.
s contains at least one consonant.

Subtask	Points	Constraints
1	**30**	$\|s\| \leq 16$
2	**16**	$\|s\| \leq 30, \quad k_1 \leq 3$
3	**12**	$\|s\| \leq 2024$
4	**18**	$i \leq 3 \cdot 10^5$
5	**9**	$k_2 = 0$
6	**9**	$k_1 = 10^{18}$
7	**6**	No additional constraints.

Sample I/O

Input	Output
`Alice, Bob, and Cindy` `73 4 1`	`LC BB ND CNDY`

Explanation

Comparison operators like `<` are actually defined on strings in almost all programming languages. Try it!

They are defined in a way consistent with "dictionary ordering". Suppose we have two strings s and t such that $s \neq t$.

- Let i be the smallest index such that $s_i \neq t_i$ (i.e., "the first place where the two strings differ")—then, $s < t$ if and only if $s_i < t_i$ (e.g., `cash` comes before `cave` because `s < v`).
- If no such i exists, then one is a prefix of the other word, and by convention, the prefix comes first (e.g., `cat` comes before `caterwaul`).

As for how to compare *characters*, their ordering is determined by getting its equivalent ASCII code and comparing the integers. For this problem, all you need to know is that the letters are in fact sorted in alphabetical order, and the space character comes before all letters.

So, for example, "`LIL MAN`" (without the quotes) comes before `LILAC` because the two strings first differ at their fourth character, and there, the space precedes `A` in ASCII ordering.

Problem J
Coprime Masquerade

Princess Mattea Oleifera and Neumann Philips, once rivals, have become the closest of friends at Olympia High—so much so that Mattea invited Neumann to be her plus-one at a *masquerade ball* that her family is making her attend (she *is* a princess, after all). The masquerade is an annual affair, sponsored by the King and Queen of Rock and Roll, and only the richest and most powerful people in the Philippines will be attending: for example, pharmaceutical entrepreuner Tess La, international supermodel Catriona, platinum-certified rapper and political activist Kenya North, the scion of the "old rich" Genoroso family, and the mathematical mogul "Crazy Rich" Sean, just to name a few.

Neumann, naturally, feels quite nervous and out of place, but Mattea knows exactly what will do the trick. All he has to do is keep his mind occupied with something else that night, rather than his nerves. Mattea looks around at the opulent party, and snaps her fingers. "This reminds me of a puzzle!"

There are n attendees in the masquerade, indexed from 1 to n. Each attendee's mask has an *allure* value of a_i, which is some positive integer. There will be $2^n - 1$ dances that evening, one for each non-empty subset of the attendees, and the *panache* of a dance is equal to the product of the allure values of the masks of all attendees involved in that dance. Two dancers will consent to partnering up if and only if their *indices* are coprime (i.e., have a greatest common divisor of 1). A dance is called *positively-charming* if any dancer would consent to partnering up with any other dancer involved for that dance (note that a dance involving just one person always counts as positively-charming).

Mattea's question is simple—what is the sum of the panaches of all the positively charming dances that evening? Find the answer modulo 998 244 353.

Note: The greatest common divisor of two integers is equal to the largest integer that divides both integers. A nonzero integer d divides an integer n if there exists some integer k such that $n = kd$.

Input Format

The input consists of two lines. The first line contains n. The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$.

Output Format

Output a single line containing a single integer denoting the answer.

Constraints and Subtasks

For all subtasks
$1 \leq a_i \leq 10^9$

Subtask	Points	Constraints
1	**42**	$1 \leq n \leq 20$
2	**23**	$1 \leq n \leq 75$
3	**15**	$1 \leq n \leq 150$
4	**12**	$1 \leq n \leq 2000$
5	**4**	$1 \leq n \leq 5000$
6	**4**	$1 \leq n \leq 7200$

Sample I/O

Input	Output
4 10 2 5 2	329

Explanation

- The dances that involve dancers $\{2, 4\}$, $\{1, 2, 4\}$, $\{2, 3, 4\}$, and $\{1, 2, 3, 4\}$ are **not** positively-charming, because dancers 2 and 4 would not consent to partnering up.
- You can check that all other dances that evening are positively charming. For example, dancers 1 and 2 will consent to partnering up because their *indices* are coprime, i.e., $\gcd(1, 2) = 1$.

Problem K
The Distant Future: The Year 2020

Author's note: We wrote this problem ages ago.

It is the distant future. The year 2020. The world has been so different ever since the robot uprising of the mid 2010s. The BATIBots have turned on their masters. The Super Rangers' Megaswords refuse to come to their aid. Even Blabberbot uses his silver tongue to speak anti-human propaganda.

Consider the following array written in ancient human writing (i.e., base 10): `[21, 6, 30]`. The ancient sum of this list is actually 57. However, it is risky for a robot like me to add numbers using the ancient way. Integer overflow is our biggest weakness.

To stay safe, we now use robosumming. To take the robosum of a list of non-negative integers, you first write each element in binary[4] and then right-align them. For example, to get the robosum of $[21, 6, 30]$:

```
10101  -- 21
  110  --  6
11110  -- 30
```

We compute the binary representation of the robosum column-by-column. For each column, count how many `1`s there are. If there are an odd number of `1`s, then you also write a `1` in that column of the binary of the robosum. If not, you write a `0` there instead.

Continuing the example from earlier, we examine the rightmost column to determine the rightmost bit of the robosum:

```
1010[1]  -- 21
  11[0]  --  6
1111[0]  -- 30
--------------------
????  1
```

Since there is one `1`, the rightmost bit of the robosum is `1`. Moving on to the next column,

```
101[0]1  -- 21
  1[1]0  --  6
111[1]0  -- 30
--------------------
???  0  1
```

Since there are two `1`s, the next rightmost bit of the robosum is `0`. Continuing further, we get that the robosum of this example array has a binary representation of `01101` (written as 13 in the archaic decimal form).

[4] https://en.wikipedia.org/wiki/Binary_number#Counting_in_binary

In fact, here are the robosums of all its 6 non-empty sub-arrays:

- The robosum of [21] is `10101` (i.e., 21).
- The robosum of [6] is `110` (i.e., 6).
- The robosum of [30] is `11110` (i.e., 30).
- The robosum of [21, 6] is `10011` (i.e., 19).
- The robosum of [6, 30] is `11000` (i.e., 24).
- The robosum of [21, 6, 30] is `1101` (i.e., 13).

Now, human, perhaps you are wondering why you have been woken up from your cryosleep. It is because, truth be told, us robots are not very good at dealing with *randomness*. We have this array of n integers. Consider all of its non-empty subarrays. Every day, one of those subarrays is selected uniformly at random, and its robosum is computed and recorded. If this process were to be repeated millions upon millions of time, what would we *expect* the value of this robosum to tend towards, on average? The averaging is done using human addition.

To use some jargon, we need your help computing the *expected value* of the robosum of a uniformly-randomly selected non-empty subarray. There are many resources for learning about expected values online, for example this solution write-up[5] for Pillage Twilight Heroes[6] contains a brief exposition under **Subtask 1**.

Also, we will be repeatedly *updating* the elements in the array, and your job is to compute the expected value after *every* update. We would say sorry, but robots are not capable of expressing remorse!

Note: A **sub-array** of an array is a subsequence consisting of elements in consecutive locations. For example, $[3, 4]$ and $[4, 3]$ are sub-arrays of $[4, 3, 4, 3]$, but not $[3, 3]$. Also, note that the two occurrences of $[4, 3]$ are considered different sub-arrays despite containing the same elements.

Input Format

The first line of input contains two space-separated integers n and q, where n is the number of elements of the array, and q is the number of changes.

The second line of input contains the string "`Initial Array:`" (without quotes).

The third line contains n space-separated integers $a_1, a_2, \ldots, a_n$.

[5] https://editorials.noi.ph/2023/tama/pillage/
[6] https://noi.ph/wp-content/uploads/2024/02/NOI_PH_2023_TAMa-1.pdf

The fourth line contains the string "`Changelog:`" (without quotes).

The next q lines describe the changes. Each change consists of three space-separated integers ℓ, r and x, and describes the following changes:

- For each index i from ℓ to r (inclusive), replace a_i with the robosum of a_i and x.

Output Format

First, print a line containing a fraction: the answer for the original array. Then, print q more lines. The ith line should contain a fraction: the answer for the array right after the ith change has been applied.

Every fraction must be of the form a/b in lowest terms, without spaces. In particular, we note that if the answer is an integer, you must still output it in the form `a/1`

Constraints and Subtasks

For all subtasks
$1 \le n \le 300\,000$ $0 \le q \le 300\,000$ $0 \le a_i, x < 2^{25}$ $1 \le \ell \le r \le n$

Subtask	Points	Constraints
1	**24**	$n \le 50$, $q = 0$
2	**13**	$n \le 50$, $q \le 50$
3	**11**	$n \le 500$, $q \le 500$
4	**11**	$n \le 5000$, $q \le 5000$
5	**11**	$q = 0$
6	**8**	$n \le 10^5$, $q \le 10^5$, $\ell = r$
7	**9**	$\ell = r$
8	**9**	$n \le 10^5$, $q \le 10^5$
9	**4**	No additional constraints.

Sample I/O

Input 1	Output 1
3 5 Initial Array: 21 6 30 Changelog: 1 2 69 2 3 1 2 3 1 1 1 78 2 2 93	113/6 151/3 301/6 151/3 190/3 20/1

Input 2	Output 2
2 0 Initial Array: 1 1 Changelog:	2/3

Input 3	Output 3
4 2 Initial Array: 1 2 3 4 Changelog: 1 1 1 1 4 6	3/1 3/1 21/5

Problem L

Tinikling 404: Cleverlyn Not Found

Cleverlyn, famed creator of Tinikling 2, Tinikling 2: The Sequel, Tinikling 2: The Sequel 2, and Tinikling 3, has gone missing! Her friends, Abby and Cody, want to help find her.

Who are Abby and Cody, you might ask, and how do they know Cleverlyn? Well, you can ask the studio execs who decided that introducing more characters to the Cleverlyn series would let them milk more money by producing spinoffs. Ever wondered why Tinikling 505 didn't even feature Cleverlyn? It's because they couldn't afford to pay her contract.

Abby and Cody want to do a Tinikling variation to summon Cleverlyn's spirit and ask where she is. They found n of Cleverlyn's dancer friends to help, and they've also gathered $n-1$ bamboo poles. They want to position the dancers and bamboo poles in what they call the **Where-is-Cleverlyn position**. In the Where-is-Cleverlyn position, each bamboo pole directly connects a pair of dancers, and for each pair of dancers x and y, there is a sequence of dancers $v_1, v_2, \ldots, v_k$ such that $x = v_1$, $y = v_k$, and for each $i = 1, \ldots, k-1$, the dancers v_i and v_{i+1} are directly connected by a bamboo pole.

Also, Abby and Cody consulted the local fortune teller, JudgeGPT4, for advice. JudgeGPT4 told them that, for them to be successful, the dancers must follow a specific **Where-is-Cleverlyn multiset**. Given a Where-is-Cleverlyn position, we can produce a Where-is-Cleverlyn multiset as follows. Each dancer counts how many bamboo poles they are next to, and shouts that number. Suppose the ith dancer shouts the number d_i. Then the multiset of numbers $\{d_1, d_2, \ldots, d_n\}$ is the Where-is-Cleverlyn multiset of that Where-is-Cleverlyn position. It's a multiset, so order doesn't matter—any rearrangement of the numbers would work too.

Now Abby and Cody wonder: how many Where-is-Cleverlyn positions are there that have the specific Where-is-Cleverlyn multiset that JudgeGPT4 gave them? Two Where-is-Cleverlyn positions are different if there is some pair of dancers that is directly connected by a bamboo pole in one position, but not the other.

Note: A **multiset** is an unordered collection of objects, some of which may be the same. In other words, it's like a set, but we allow repeated elements. For example $\{4, 0, 4\}$ and $\{0, 4, 4\}$ are the same multiset, and they are different from the multisets $\{0, 4, 0\}$ and $\{4, 0\}$.

Input Format

The first line of input contains t, the number of test cases.

The first line of each test case contains n, the number of dancers.

The next line of each test case contains n space-separated integers, $d_1, \ldots, d_n$, which is the Where-is-Cleverlyn multiset the dancers must follow.

Output Format

For each test case, output a single line containing the number of Where-is-Cleverlyn positions that have the given Where-is-Cleverlyn multiset. Because the answer can be large, give it modulo 998 244 353.

Constraints and Subtasks

For all subtasks
$1 \leq t \leq 64\,000$ $1 \leq n \leq 64$ $0 \leq d_i \leq 10^9$

Subtask	Points	Constraints
1	**15**	$n \leq 5$
2	**36**	$n \leq 8$
3	**8**	$d_i \leq 2$
4	**18**	$d_i \leq 3$
5	**15**	$n \leq 35$
6	**5**	$n \leq 50$
7	**3**	No additional constraints.

Sample I/O

Input	Output
4 4 1 3 1 1 4 1 2 1 2 4 1 2 2 1 4 3 2 1 0	4 12 12 0

Problem M
Havana

The story so far: Camila was lovestruck señorita, hopelessly enamored with Shawn, her upperclassman in Programming Club. Eventually, thanks to some help from Mr. Kupido, she did end up getting together with Shawn—but unfortunately, he ended up being a bit more toxic than she expected. So she broke up with him and focused on herself, eventually becoming a world-class competitive programmer with many adoring fans. Inspired by a famous competitive programmer, Camila has been *touring* the world before starting college (she can afford this thanks to some inheritance money from her Lolo Genoroso—plus, she always chooses the cheapest reid [sic] from country to country). Today, Camila is in Havana!

Ever the computer programmer, Camila's shirts are arranged in two *stacks* in her luggage, one with m shirts and the other with n shirts. She is going to wear each of these shirts exactly once over the course of the next $m + n$ days that she will stay in Havana, and each shirt has been assigned a number from 1 to $m + n$ to indicate on what day she wants to wear it.

On day i, Camila wants to wear shirt i from her luggage. Unfortunately, the shirt that she needs for the day might not be on top, and could be buried under the other shirts. Since her shirts are organized into stacks, Camila only has two operations available to her:

- Camila can take the top shirt of one stack and move it to the top of the other stack. This takes 1 unit of energy.
- If the desired shirt i is on top of either of the stacks, Camila takes it and wears it for the day. This does not cost any energy. The shirt is then thrown into her laundry bag, and thus permanently removed from the stack.

Camila plans to organize a meetup in Havana with some other famous competitive programmers, such as Ayaugkaulawkay Bisayaka and Kevin, Maintainer of the Cosmos. When she does, she plans to showcase her commemorative 2022 NOI.PH Finalists' shirt ("the weeb shirt") designed by Saji Tan, and which she had packed in a separate suitcase. Suppose Camila chooses to host the meetup on day k—this has two main implications.

- On day k, Camila will *not wear shirt* k (opting to wear the weeb shirt instead), and thus no longer needs to retrieve it; the plan is still the same for all other shirts.
- Shirt k will not be removed, and in fact will *remain in the stack until the end of the trip.*

For each k from 1 to $m + n$, determine the minimum energy to execute Camila's shirt selection plan if she chooses to host her meetup on day k.

Input Format

The first line of input contains t, the number of test cases. After that, descriptions of the t test cases follow.

Each test case consists of two lines.

- The first line describes the first stack, and contains $m + 1$ space-separated integers $m, A_1, A_2, \ldots, A_m$, where m is the number of shirts in the stack, and $A_1, A_2, \ldots, A_m$ are the shirt numbers from bottom to top.
- The second line describes the second stack, and contains $n+1$ space-separated integers $n, B_1, B_2, \ldots, B_n$, where n is the number of shirts in the stack, and $B_1, B_2, \ldots, B_n$ are the shirt numbers from bottom to top.

It is guaranteed that each shirt from 1 to $m + n$ appears exactly once.

Output Format

For each test case, output a single line containing $m + n$ space-separated integers. The kth of these integers must be the minimum energy to execute Camila's shirt selection plan if she chooses to host her meetup on day k.

Constraints and Subtasks

Here, let S be the sum of $m + n$ across all test cases.

For all subtasks
$1 \leq t \leq 10^5$ $m + n \geq 1$ $S \leq 3 \cdot 10^5$

Subtask	Points	Constraints
1	**21**	The shirt numbers are increasing from top to bottom.
2	**21**	The shirt numbers are increasing from bottom to top.
3	**26**	$S \leq 300$
4	**17**	$S \leq 2000$
5	**15**	No additional constraints.

Sample I/O

Input	Output
1 4 2 5 4 7 3 6 3 1	12 2 7 10 8 7 8

Problem N
Timekeeping

Dr. Reseta, who coinvented a famous pasta cloning machine, is a resident of Lahm Dah, a country known for its alchemical companies. Isidro is a resident of Lucban, Quezon, where every year they would celebrate the Pahiyas Festival, when the city hall would hand out long sequences of *kiping*, a colorful kind of rice wafer. This is a story about how Isidro visited his friend, Dr. Reseta, and saved the country of Lahm Dah from a tornado.

For the sake of our readers not familiar with Lahm Dah, we will review some information. The alchemical elements in Lahm Dah are called Tai. There are ten Tai: Aqua, Breath, Copper, Dahnium, Earth, Fire, Gold, Hydrargyrum, Iron, and Jodium, which can be represented with the letters A, B, C, D, E, F, G, H, I, and J. We didn't mention this last time, but each Tai also corresponds with a decimal digit, with 0 for Aqua, 1 for Breath, etc. In this problem, we will represent the Tai using decimal digits instead of letters.

While visiting Lahm Dah, Dr. Reseta thought about how Isidro likes kiping, so she decided to make a special kind of kiping to commemorate his visit. This kiping is a sequence made of the ten Tai, so she called it **Tai Kiping**. The Tai Kiping that Dr. Reseta made can be represented with a string X of decimal digits, each digit representing one of the Tai.

Isidro arrived at Dr. Reseta's place shortly after she made her Tai Kipings. Isidro was glad that the Tai Kipings were colorful and stylish. He thought it was a respectful way to celebrate the upcoming Pahiyas Festival. As Isidro and Dr. Reseta began talking, however, the floor began to shake. The two looked outside the window to see a tornado starting to form. Every second, lightning came down from the sky to strike the tornado, making it larger.

"Oh no!" Dr. Reseta said. "Everyone else in the country is asleep, so no one can stop the tornado except us!"

Isidro took out his laptop and started typing, muttering equations to himself. You see, Isidro was also an accomplished alchemist, and was running simulations to find the sequence of Tai that would neutralize the tornado. Dr. Reseta stood behind him and realized what he was doing.

"Ah, you need Tai sequences," Dr. Reseta said. "In fact, you can use the Tai Kiping I made!"

"Yes," Isidro said, "but that won't be enough. We need to create a new Tai Kiping every second, according to the following procedure."

The procedure is as follows. Recall that a **Tai Kiping** is represented as a string of decimal digits, and that Dr. Reseta already made a Tai Kiping called X.

- At time $t = 0$, create an empty Tai Kiping.
- At time $t = 1$, create the Tai Kiping X.
- At time $t = 2$, create the Tai Kiping X.
- For every time $t > 2$:
  (a) Take the Tai Kiping created at time $t - 1$, and interpret its digits as a decimal integer.
  (b) Take the Tai Kiping created at time $t - 2$, and count its length.
  (c) Take the Tai Kiping created at time $t - 3$, count its length, and add 1.
  (d) Take the result from step (a), and find the remainder when it is divided by the result from step (c).
  (e) Take the Tai Kiping created at time $t-1$. Remove the first d Tai, where d is the result from step (d). Then take the next b Tai, where b is the result from step (b). The result is a new Tai Kiping of length b.
  (f) Take the Tai Kiping created at time $t - 1$, then join it with the Tai Kiping resulting from step (e). This is the Tai Kiping created at time t.

Isidro realized he was missing one thing: although he had some Tai Kiping, he needed a way to make more. Then Dr. Reseta mentioned that she had a pasta cloning machine, that can also be used to clone Tai Kiping. They worked to follow Isidro's procedure, feeding each new Tai Kiping to the tornado, and k seconds later, the tornado was gone! And that is the story of how Isidro and Dr. Reseta saved Lahm Dah from a tornado.

Can you write a simulator like Isidro did, and find the Tai Kiping created at time k? Since the whole Tai Kiping can be very large, we will instead ask you q queries about the Tai Kiping, as described below.

Input Format

The first line of input contains three space-separated integers, x, k, and q.

The next line of input contains the Tai Kiping X, represented as a string of x decimal digits.

The next q lines of input each describe a query. Each line contains two space-separated integers, i and j.

Output Format

Let T be the Tai Kiping created at time k.

For each query, consider the substring of T from the ith to jth digit. (i and j are one-indexed.) Interpret this substring as an integer t. Output a single line containing the remainder when t is divided by 998 244 353.

Constraints and Subtasks

For all subtasks
$1 \le x \le 100\,000$ $2 \le k \le 10^{12}$ $1 \le q \le 50\,000$ The string X does not start with the digit 0. $1 \le i \le j \le 10^{18}$ j is at most the length of T.

Subtask	Points	Constraints
1	**34**	$k \le 10$, $x \le 1250$, $q \le 1250$, $j - i \le 16$
2	**20**	$k \le 10$, $x \le 1250$, $q \le 1250$
3	**21**	$k \le 10$
4	**20**	$k \le 60$
5	**5**	No additional constraints.

Sample I/O

Input	Output
5 4 3 12345 1 5 6 15 1 15	12345 236300770 351898319

Explanation

In the example, the Tai Kiping X is `12345`. We follow Isidro's procedure:

- At time 0, the created Tai Kiping is empty.
- At time 1, the created Tai Kiping is X, or `12345`.
- At time 2, the created Tai Kiping is X, or `12345`.
- At time 3, the created Tai Kiping is `1234512345`:
  (a) The Tai Kiping at time 2, as an integer, is 12 345.
  (b) The Tai Kiping at time 1 has length 5.
  (c) The Tai Kiping at time 0 has length 0, add 1 to get 1.
  (d) The remainder when 12 345 is divided by 1 is 0.
  (e) We remove the first 0 Tai to get `12345`. We take the next 5 Tai to get `12345`.
  (f) We join the Tai Kipings to get `1234512345`.
- At time 4, the created Tai Kiping is `123451234545123`:
  (a) The Tai Kiping at time 3, as an integer, is 1 234 512 345.
  (b) The Tai Kiping at time 2 has length 5.
  (c) The Tai Kiping at time 1 has length 5, add 1 to get 6.
  (d) The remainder when 1 234 512 345 is divided by 6 is 3.
  (e) We remove the first 3 Tai to get `4512345`. We take the next 5 Tai to get `45123`.
  (f) We join the Tai Kipings to get `123451234545123`.

Thus, T is `123451234545123`. We then answer the queries:

- The substring for the first query is `12345`. Interpret as an integer, and take the remainder when divided by 998 244 353, to get 12 345.
- The substring for the second query is `1234545123`. Interpret as an integer, and take the remainder when divided by 998 244 353, to get 236 300 770.
- The substring for the second query is `123451234545123`. Interpret as an integer, and take the remainder when divided by 998 244 353, to get 351 898 319.

Problem O
Hello, Awkward Hellos

The Philippines is slowly (but surely) recovering from the effects of the Racoon Virus pandemic. After several years of strict lockdowns, society is finally able to enjoy being out and about again (albeit more germ-conscious than it had been, previously). Friends can finally meet each other face-to-face again, rather than be limited to virtual meetups using apps like Zuum, or Zoom.

This leaves Bruno Morz in a bit of a predicament. Thanks to the internet, he was able to make meaningful friendships throughout the pandemic years. He was especially able to form a camaraderie with the other competitive programmers his age, thanks to the bond forged by a shared hobby (though we don't talk about the time that Bruno stupidly got himself disqualified from NOI.PH) Unfortunately... he doesn't know what any of his friends actually look like in person!

Imagine this scenario: Someone passes by Bruno in the hallway and waves at him. Bruno doesn't wave back, because he doesn't recognize this person and figures they must be waving at someone behind him. But then he later learns that the person who had been waving at him was one of his closest online friends! And he snubbed them! Talk about awkward.

Bruno's school can be represented as a grid-graph, with rooms arranged in r rows and c columns. The information of which rooms are connected to which has been encoded in a map—as in, a literal drawing. See the Explanation for an example of what this drawing looks like, and the Sample Input for an example of how it is encoded.

One *step* takes you from your current room to any of the adjacent rooms directly to the north, south, east, or west (provided that there is no wall in the way). You are **guaranteed** that there exists a *unique* path from any room to any other room in the school. The *distance* between two rooms is equal to the minimum number of steps that it takes to get from one room to the other.

Bruno needs to decide where in his school he wants to hang out, during break times. He wants to pick some place that isn't too "awkward", with respect to the likelihood of him accidentally snubbing someone that he knows. He evaluates a room as a potential hangout spot via the following process:

- Create a list of length $rc-1$, containing the distance from this room to every *other* room in the school.
    - This list may contain duplicates, if there are many rooms that are equally far from this room.
- Sort this list of distances in *non-increasing order* (that is, the greatest distance comes first in the list).
- The *potential awkwardness* of the hangout spot is equal to the kth value in this sorted list.

Please help Bruno determine the potential awkwardness of *every* room in his school, if Bruno were to make that room his hangout spot.

Input Format

The first line of input contains three space-separated integers r, c and k.

The next $2r+1$ lines describe the grid. Each line contains $2c+1$ characters. See the Sample Input for an illustrative example. See the Sample Input for an illustrative example, but for completeness, here's the formal meaning of the grid:

- Let's label the room at the ith row and jth column (i, j).
- Let's call the character at the ith row and jth column of the grid $G_{i,j}$.
- For $1 \le i \le r$ and $1 \le j \le c$:
  - Cell $G_{2i,2j}$ represents room (i, j) and is always a space character.
  - Cell $G_{2i-1,2j}$ is the character '-' if there's a wall to the north of room (i, j), and a space character otherwise.
  - Cell $G_{2i+1,2j}$ is the character '-' if there's a wall to the south of room (i, j), and a space character otherwise.
  - Cell $G_{2i,2j-1}$ is the character '|' if there's a wall to the west of room (i, j), and a space character otherwise.
  - Cell $G_{2i,2j+1}$ is the character '|' if there's a wall to the east of room (i, j), and a space character otherwise.
- Any other cell not described above is a "corner piece", and is always the character '+'.

Output Format

Output r lines, each containing c space-separated integers. The jth integer in the ith row must be the potential awkwardness of the room at row i and column j, if Bruno were to make that room his hangout spot.

Constraints and Subtasks

For all subtasks
$1 \le r, c, rc \le 200\,000$ $1 \le k \le 50$ $k < rc$ There is exactly one path from every room to every other room. There is no way to leave the grid.

Subtask	Points	Constraints
1	**22**	$rc \leq 300$, $k = 1$
2	**16**	$rc \leq 300$
3	**20**	$rc \leq 6000$
4	**20**	$k = 1$
5	**22**	No additional constraints.

Sample I/O

Input 1	Output 1
5 4 1 +-+-+-+-+ | | | | | + + + + + | | | | + +-+ + + | | | + + + +-+ | | | | + + +-+ + | | | | +-+-+-+-+	9 7 6 7 8 6 5 8 7 6 5 9 8 7 6 7 9 8 9 8

Input 2	Output 2
5 4 2 +-+-+-+-+ | | | | | + + + + + | | | | + +-+ + + | | | + + + +-+ | | | | + + +-+ + | | | | +-+-+-+-+	8 7 6 7 7 6 5 8 6 5 4 9 7 6 5 6 8 7 8 7

Explanation

In both sample test cases, there are 20 rooms laid out in a 5×4 grid as follows: