Problem A. Browser Games

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	32 mebibytes

The memory limit of this problem (32 mebibytes) is unusual!

In the upcoming n days, n browser games will be released on a new website. According to the plan, the administrator will release a new game per day. Users have to open the corresponding URL (Uniform Resource Locator) and get feedback from the server to download a game.

However, the setup of the server uses unreadable legacy codes. Once a user somehow finds the URL of an unreleased game, the data of the game would leak out. To temporarily fix the problem, the administrator decided to add a series of confirmation prefixes, which are non-empty strings, at the server-side. The server will respond with the correct game data when the requested URL does correspond to a game (no matter released or unreleased) and at least one confirmation prefix is a prefix of the URL; otherwise, the server will declare that the game is not found.

To make the work easier, the administrator asks you to find the minimum number of confirmation prefixes the server required to avoid data leaks every time after a new game release.

Input

The first line contains an integer n ($1 \le n \le 10^5$), indicating the number of browser games to be released.

In the next n lines, the i-th line contains a non-empty string, consisting of only lowercase letters ('a' to 'z'), uppercase letters ('A' to 'Z'), digits ('0' to '9'), dots ('.') and forward slashes ('/'), indicating the URL of the browser game released on the i-th day.

It is guaranteed that the length of each given URL is at most 100, and no given URL is the prefix of any other given URL.

Output

Output n lines, the i-th of which contains an single integer, indicating the minimum number of required confirmation prefixes after the i-th new game released.

Example

standard input	standard output
3 ufoipv.ofu hsbocmvfgboubtz.kq hfotijo.njipzp.dpn/kb	1 2 2

Note

The explanation for the sample case is given as follows.

- In the first day, the game with URL `ufoipv.ofu` has been released, while the other two games should be unavailable. Thus taking "u" as the only confirmation prefix is sufficient.
- In the second day, the games with URL `ufoipv.ofu` and `hsbocmvfgboubtz.kq` have been released, while the game with URL `hfotijo.njipzp.dpn/kb` should still be unavailable. Thus taking "u" and "hs" as confirmation prefixes is sufficient. Note that taking "u" and "h" as confirmation prefixes is not fesible, which may lead to data leaks of the game with URL `hfotijo.njipzp.dpn/kb`.

- In the last day, all the three games have been released. Thus taking "u" and "h" as confirmation prefixes is sufficient. Note that taking an empty string as the only confirmation prefix is not allowed, since the confirmation prefixes should be non-empty.

Problem B. Child's play

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 mebibytes

Ema is playing a new game with her friends!

Each of the n players has a number on its forehead. The numbers range from 1 to m, and no two players have the same number. A player can see everyone else's number, but not the number on itself. All players know the lower bound 1 and the upper bound m beforehand.

Players win by pointing out how many other players have a number smaller than its own number. To decide the winner, each turn every player writes down whether or not it can win using the information from previous turns, then everyone shows the result simultaneously. The game ends when at least one player wins on some turn.

Given the starting situation, determine when will the game end, and who wins at the end. It can be shown that the game will always end.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n ($1 \le n \le 1\,000$) and m ($n \le m \le 10^9$), indicating the number of players and the range of numbers.

The next line contains n integers a_i ($1 \le a_i \le m$) in ascending order, the i-th of which indicates the number on the i-th player.

It is guaranteed that the sum of n in all test cases will not exceed 10^3.

Output

For each test case, output the first line containing 2 integers ans and cnt, indicating the turn where the game ends and the number of player that will win on that turn, and the second line containing cnt integers s_i, indicating the indices of the players that win on turn ans, where indices should be printed in increasing order.

Example

standard input	standard output
3 3 5 2 3 4 4 18 7 8 15 17 5 16 4 6 8 11 13	2 3 1 2 3 8 2 2 3 7 1 2

Problem C. Dance Party

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 mebibytes

ZYB plans to hold a dance party and has invited n ladies and n gentlemen, they are both numbered from 1 to n. Now ZYB wants to match the lady and gentleman one by one.

However, each lady holds a blacklist that she wouldn't like to dance with the man on that list. For the i-th lady, she lists k_i gentlemen and their IDs are $a_{i,1}, a_{i,2}, ..., a_{i,k_i}$ repectively.

ZYB wonders whether there exists a perfect matching so that each lady can dance without her dislike.

Input

The first line of the input contains one integer n ($1 \le n \le 30\,000$), indicating the number of ladies and gentlemen invited.

In the following n lines, the first integer k_i ($0 \le k_i \le 100$) is the number of gentlemen that the i-th lady dislikes, and then followed with k_i integers $a_{i,1}, a_{i,2}, ..., a_{i,k_i}$ ($1 \le a_{i,j} \le n$).

Output

Output -1 if no perfect matching exists. Otherwise output n integers, the i-th of which indicates the gentleman ID that the i-th lady chooses.

Examples

standard input	standard output
3 1 1 1 2 1 3	2 3 1
3 1 1 2 2 3 2 2 3	-1

Problem D. Diameter Counting

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 mebibytes

Given a integer n, you are asked to calculate the sum of the diameter of all unrooted labeled trees with n nodes labeled from 1 to n. Here the diameter of a tree is the length of the longest simple path on this tree, and the length of a path is the number of edges along this path.

Since the answer may be too large, you only need to output the result modulo a given prime number p.

Input

The input contains two integers n ($1 \le n \le 500$) and p ($10^8 \le p < 2^{31}$), where p is a prime number.

Output

Output a single integer, indicating the sum of the diameter of all labeled trees with n nodes modulo p.

Examples

standard input	standard output
2 998244353	1
4 1000000007	44
6 1000000009	5322

Problem E. More Fantastic Chess Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	9 seconds
Memory limit:	1024 mebibytes

Grammy is a chess enthusiast as well as a famous mathematician. She likes to study chess tactics in higher-dimensional space.

The movement of every chess piece in k dimensional space is similar to planar ones. All chess pieces must stay inside the board. Note that two of the corners of the board have coordinates $(1, 1, ..., 1)$ and $(a_1, a_2, ..., a_k)$. Detailed movement rules are as follows.

- The King: Choose a non-empty subset of dimensions, add 1 or subtract 1 to each of the chosen coordinates.
- The Queen: Choose a non-empty subset of dimensions and a positive integer x, add x or subtract x to each of the chosen coordinates.
- The Rook: Choose one dimension and a positive integer x, add x or subtract x to that coordinate.
- The Bishop: Choose two dimensions and a positive integer x, add x or subtract x to each of the chosen coordinates.
- The Knight: Choose one dimension, add 1 or subtract 1 to that dimension, then choose a different dimension, add 2 or subtract 2 to that dimension.
- The Pawn: The rule is too difficult, so we do not care about this piece in this problem.

After telling you about the rules, Grammy brought out a huge k dimensional hyper-cuboid chessboard and a chess piece. The size of the board is $a_1 \times a_2 \times ... \times a_k$. She places the chess piece in a particular cell inside the board and asks you to find the number of different cells the piece can arrive in **one** move. Since the answer may be too large, you only need to output the result modulo 998 244 353.

However, since this problem is still too easy, Grammy then moves the piece q times according to the movement rule of that piece and challenges you to answer the problem after each move.

Input

The first line contains two integers k ($1 \le k \le 3 \times 10^5$) and q ($0 \le q \le 3 \times 10^5$), indicating the dimension of the board and the number of additional movements.

The second line contains the name of the given chess piece, which is either "King", "Queen", "Rook", "Bishop", or "Knight".

The third line contains k positive integers $a_1, a_2, \ldots, a_k$ ($1 \le a_i \le 10^6$), indicating the size of the board.

The fourth line contains k positive integers $b_1, b_2, \ldots, b_k$ ($1 \le b_i \le a_i$), indicating the starting position of the given piece.

The following q lines contains the description of the additional movements.

Each description contains $2d + 1$ numbers, where d denotes the number of moved dimensions.

The first number in the description is d ($1 \le d \le k$), and each of the following d pairs of integers (x_i, δ_i) in strictly increasing order of x_i indicates that the x_i-th coordinate of the chess piece changes by δ_i ($\delta_i \ne 0$) in the move.

It is guaranteed that each movement is valid that the chess piece still stay inside the board after each movement, and the sum of d in all movements is no greater than 3×10^5.

Output

Output $q+1$ lines, each of which contains a single integer, indicating the answer before the first movement and after each movement modulo 998 244 353.

Examples

standard input	standard output
2 1 King 8 8 1 2 2 1 1 2 1	5 8
2 1 Queen 8 8 1 2 2 1 2 2 2	21 25
2 1 Rook 8 8 1 2 1 1 6	14 14
2 1 Bishop 8 8 1 2 2 1 2 2 2	7 11
2 1 Knight 8 8 1 2 2 1 1 2 2	3 6

Problem F. Train Wreck

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 mebibytes

You are now the coordinator for the Nonexistent Old Railway Station! The Nonexistent Old Railway Station only has one track with a dead end, which acts as a stack: everytime you either push a train into the station, or pop the most recently added train out of the station.

Everyday, n trains gets push into and pop out of the station. The inspector has already decided today's sequence of "pushing" and "popping". You now have a list of the colored trains and have to assign each train to one "pushing" in inspector's sequence.

Meanwhile, the inspector also requires you to make the sequence of trains remaining on the track unique every time you push a train onto it. Two sequence of trains are considered different if the length is different or the i-th train in the two sequences have a different color.

Output a solution or decide that it is impossible.

Input

The first line contains one integer n ($1 \le n \le 10^6$), indicating the number of trains.

The second line contains a bracket sequence of length $2n$, with each '(' indicating one "pushing" and ')'indicating one "popping". The input guarantees that the sequence is always valid and balanced.

The third line contains n numbers k_i ($1 \le k_i \le n$), indicating the color of the i-th train.

Output

If there is no solution, output "NO" in one line.

Otherwise, output "YES" in the first line, and n intergers in the second line, indicating the color of the i-th train that is being pushed.

Examples

standard input	standard output

`3`
`()()()`
`1 2 2`
	`NO`

`4`
`()(())()`
`1 2 4 4`
	`YES`
	`4 2 4 1`

Note

In the first sample case, all 3 train color sequences are always going to be [1],[2] and [2] (not necessarily in that order), thus it is impossible to satisfy the inspector.

In the second sample case, the 4 train color sequences in the sample output are [4], [2], [2, 4] and [1] (in that order), we can see that all these are unique.

Problem G. Game of Death

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

There are n players. Each player chooses exactly one player except himself with the same probability, and then every player shoots the player he chooses simultaneously. Each shot hits the chosen player with probability p. All choices and hits are independent. A player dies if and only if hit by at least one shot. For each $k = 0, 1, \cdots, n$, find the probability p_k exactly k player alive.

Input

The only line contains three integers n ($2 \le n \le 3 \times 10^5$), a and b ($1 \le a \le b < 998244353$), where n is the number of players and $p = \frac{a}{b}$ is the probability of hit for each shot.

Output

Output $n + 1$ lines. In the $(k + 1)$-th line, output one integer indicating p_k module 998244353.

Formally, it can be proved that p_k is a rational number and it can be represented by $\frac{A_k}{B_k}$ where the greatest common divisor of A_k and B_k is 1. You should output one integer $x_k (0 \le x_k < 998244353)$ such that $B_k x_k \equiv A_k \pmod{998244353}$.

Examples

standard input	standard output
2 1 1	1 0 0
3 1 1	748683265 249561089 0 0
2 1 2	748683265 499122177 748683265
3 1 3	619281219 27729010 499122177 850356301

Note

For the first example, $p_0 = 1, p_1 = p_2 = 0$.

For the second example, $p_0 = \frac{1}{4}, p_1 = \frac{3}{4}, p_2 = p_3 = 0$.

For the third example, $p_0 = p_2 = \frac{1}{4}, p_1 = \frac{1}{2}$.

For the fourth example, $p_0 = \frac{1}{108}, p_1 = \frac{7}{36}, p_2 = \frac{1}{2}, p_3 = \frac{8}{27}$.

Problem H. War of Inazuma (Easy Version)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 mebibytes

As a traveler, you once witnessed the war of Inazuma, of which sides are the resistance and the Shogun's army.

The map of Inazuma can be viewed as a unit n-dimensional hypercube, which has exactly 2^n vertices. Each vertex is labeled with an integer from 0 to $2^n - 1$. Two vertices are adjacent if and only if there exists exactly one different bit in their n-bit binary representation.

In the war of Inazuma, some vertices were occupied by the resistance army, and others were occupied by the Shogun's Army. You also noticed that for each vertex u, the number of vertices adjacent to u and occupied by the same side as u was no more than $\lceil \sqrt{n} \rceil$.

Many years later, you have already forgotten details of the war. Can you construct a hypercube satisfying all the above requirements?

Input

The only line contains a single integer n ($1 \le n \le 22$).

Output

Output a `01`-string of length 2^n. The $\overline{a_1 \dots a_n}$-th bit describes which side is vertex $(a_1, \dots, a_n)$ occupied by, 0 for the resistance army and 1 for the Shogun's army. Here $\overline{a_1 \dots a_n}$ is an n-bit binary representation. Any answer satisfying all requirements will be accepted.

Example

standard input	standard output
`2`	`0111`

Problem I. War of Inazuma (Hard Version)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 mebibytes

As a traveler, you once witnessed the war of Inazuma, of which sides are the resistance and the Shogun's army.

The map of Inazuma can be viewed as a unit n-dimensional hypercube, which has exactly 2^n vertices. Each vertex is labeled with an integer from 0 to $2^n - 1$. Two vertices are adjacent if and only if there exists exactly one different bit in their n-bit binary representation.

In the war of Inazuma, some vertices were occupied by the resistance army, and others were occupied by the Shogun's Army. **You also noticed that the number of vertices occupied by the resistance army was different from that occupied by the Shogun's Army.** Meanwhile, for each vertex u, the number of vertices adjacent to u and occupied by the same side as u was no more than $\lceil\sqrt{n}\rceil$.

Many years later, you have already forgotten details of the war. Can you construct a hypercube satisfying all the above requirements?

Input

The only line contains a single integer n $(1 \le n \le 22)$.

Output

Output a `01`-string of length 2^n. The $\overline{a_1 \ldots a_n}$-th bit describes which side is vertex $(a_1, \ldots, a_n)$ occupied by, 0 for the resistance army and 1 for the Shogun's army. Here $\overline{a_1 \ldots a_n}$ is an n-bit binary representation. Any answer satisfying all requirements will be accepted.

Example

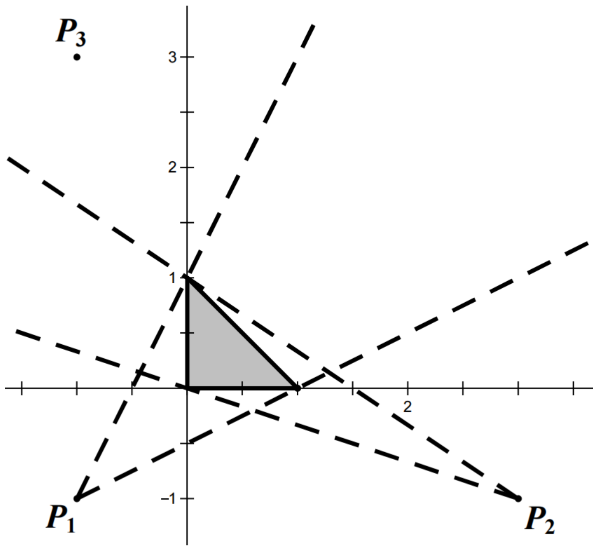
standard input	standard output
2	0111

Problem J. Illuminations

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 mebibytes

You are given a convex polygon with n vertices and m different points strictly outside the polygon which are all legal positions to install illuminants.

An illuminant can light up some exterior boundaries of the polygon. Your task is to install the least number of illuminants to light up all exterior boundaries of the polygon and provide a feasible scheme.



Explanation of the first sample case.

Input

The first line contains two integers n ($3 \le n \le 2 \times 10^5$) and m ($1 \le m \le 2 \times 10^5$).

Each of the following n lines describes a vertex on the convex polygon with two integers x and y ($|x|, |y| \le 10^9$), indicating the coordinates of a vertex on the polygon. All these vertices are given in counter-clockwise order and any three of them are not collinear.

Then the following m lines contain m different points outside the polygon describing all legal positions to install illuminants. The i-th line contains two integers x and y ($|x|, |y| \le 10^9$), indicating the coordinates of the i-th position. All these positions would not lie in some extension lines for the sides of the polygon.

Output

If it is impossible to light up all exterior boundaries of the polygon, output a single line with a single integer -1.

Otherwise output two lines. Firstly, output a line with a single integer k, representing the least number of illuminants needed to light up all the boundaries. Then, output a line with k space-separated distinct integers, describing a feasible scheme, each of which is the index of a selected position.

All feasible schemes are allowed, so you can output any of them.

Examples

standard input	standard output
3 3 0 0 1 0 0 1 -1 -1 3 -1 -1 3	2 2 1
3 1 0 0 1 0 0 1 -1 -1	-1

Problem K. Walking

Input file:	`standard input`
Output file:	`standard output`
Time limit:	9 seconds
Memory limit:	1024 mebibytes

There is an $n \times m$ maze with two of the corners of the board having coordinates $(1, 1)$ and (n, m).

Once you are located in the cell (x, y), you can move to the four adjacent cells $(x - 1, y)$, $(x + 1, y)$, $(x, y - 1)$ and $(x, y + 1)$ in one second, and you must keep moving, which means that you cannot stay in one cell. Also, you cannot move outside the maze.

Now you start at (a, b), and you should keep moving for t seconds. You are asked to figure out the number of possible moving trails.

Here a moving trail is the sequence of the moving strategy during your moving. For example, if you start at $(1, 1)$ and at the following 5 seconds you move to $(2, 1)$, $(3, 1)$, $(3, 2)$, $(4, 2)$ and $(3, 2)$ step by step, the moving trail is $(1, 0)$, $(1, 0)$, $(0, 1)$, $(1, 0)$, and $(-1, 0)$.

Since the answer may be too large, you only need output the result modulo 998 244 353.

Input

The only line contains five integers t, n, m $(1 \le t, n, m \le 5 \times 10^5)$, a $(1 \le a \le n)$ and b $(1 \le b \le m)$.

Output

Output a single integer, indicating the answer modulo 998 244 353.

Examples

standard input	standard output
`1 1 1 1 1`	`0`
`1 2 1 2 1`	`1`
`1 2 2 2 1`	`2`
`1 4 1 4 1`	`1`