The 2nd Universal Cup

Uni Cup

Stage 15: Macau

December 23-24, 2023

This problem set should contain 11 problems on 23 numbered pages.

Based on

icpc.foundation

International Collegiate Programming Contest (ICPC)

Hosted by

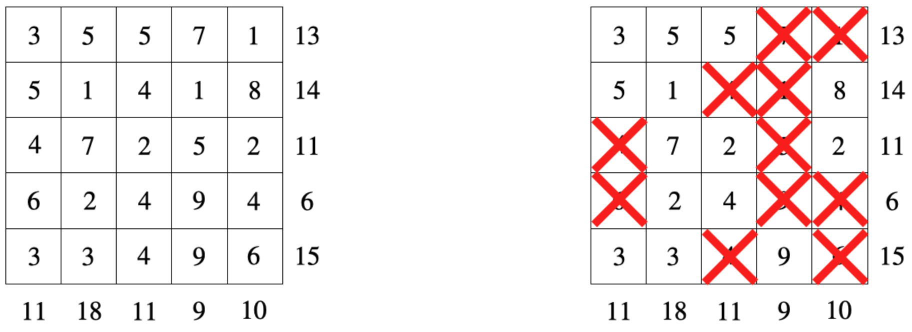
Prepared by

Problem A. (-1,1)-Sumplete

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

Sumplete is a logic puzzle similar to Sudoku, Kakuro, and Hitori. It is famous for being developed with the help of [1]ChatGPT.

The Sumplete puzzle consists of a square grid, with each cell containing an integer. Each row and column also has an integer "hint" assigned. The player must cross out some numbers in the grid such that the sum of uncrossed numbers in each row and column equals the corresponding hint. See the following picture for an illustration.



An example of a 5×5 Sumplete puzzle (left) and its solution (right)

Recently, on September 15th, 2023, a paper *Sumplete is Hard, Even with Two Different Numbers* by Suthee Ruangwises was uploaded to [2]arxiv. The paper showed that if we allow the grid of Sumplete puzzles to be rectangular, then deciding the solvability of a given Sumplete puzzle is **NP**-complete, even if the grid contains only two different numbers 1 and 3.

Bobo is quite unsatisfied with this result. He insists that there must exist some Sumplete puzzles that are easy to solve. Now he provides you with a Sumplete puzzle, where the grid contains only two different numbers -1 and 1. Can you please solve the puzzle for him?

Input

The first line of input contains an integer n ($1 \le n \le 4\,000$), denoting the height and width of the grid of the Sumplete puzzle.

Then n lines follow, where the i-th ($1 \le i \le n$) line contains a string s_i of length n consisting of '+'s and '−'s, such that $a_{i,j} = 1$ if the j-th character of s_i is '+' and $a_{i,j} = -1$ if the j-th character of s_i is '−'.

The input then contains a line containing n integers $r_1, r_2, \dots, r_n$ ($-n \le r_i \le n$), where the i-th number denotes the hint in the i-th **row**.

Lastly, the input contains a line containing n integers $c_1, c_2, \dots, c_n$ ($-n \le c_i \le n$), where the i-th number denotes the hint in the i-th **column**.

Output

If the given Sumplete puzzle has no solutions, output "No" in the first line. Otherwise, output "Yes" in the first line. You can output each letter in any case (lowercase or uppercase). For example, the strings "yEs", "yes", "Yes", and "YES" will all be considered as positive replies.

[1]See `https://sumplete.com/about/` to learn more about the creation of this puzzle.
[2]`https://arxiv.org/pdf/2309.07161.pdf`

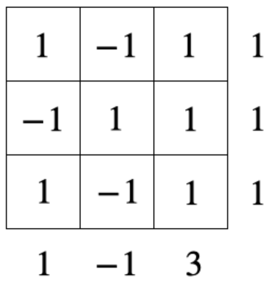
If your answer is "Yes", output a binary string of length n in each of the following n lines. In the i-th $(1 \le i \le n)$ line you need to output a string t_i of length n consisting of '0's and '1's, such that the j-th character of t_i being '0' denotes you **cross out** the number on the i-th row and j-th column, and the j-th character of t_i being '1' denotes you **keep** it. Your solution must satisfy that the sum of uncrossed numbers in each row and column equals the corresponding hint. If there are multiple solutions, outputting any of them will be considered correct.

Examples

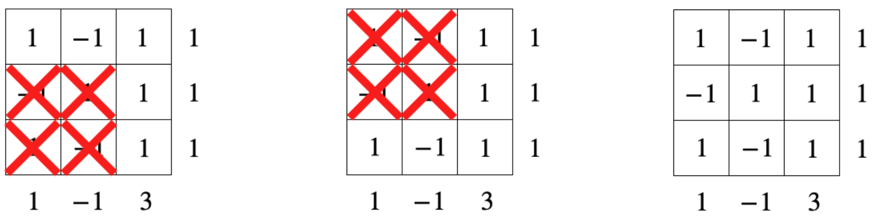
standard input	standard output
`3`	`Yes`
`+-+`	`111`
`-++`	`001`
`+-+`	`001`
`1 1 1`	
`1 -1 3`	
`3`	`Yes`
`---`	`110`
`-++`	`100`
`+++`	`000`
`-2 -1 0`	
`-2 -1 0`	
`3`	`No`
`+-+`	
`-++`	
`++-`	
`1 0 2`	
`2 2 -1`	

Note

The first sample corresponds to the following 3×3 Sumplete puzzle:



Its three solutions are listed as follows (the third one corresponds to crossing out none of the numbers in the grid), and outputting any of them will be considered correct.



Problem B. Basic Equation Solving

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 seconds
Memory limit:	1024 megabytes

Bobo recently saw some constraints in the form of $X\,op\,Y$ where X and Y are strings consisting of digits from 0 to 9 and uppercase English letters, denoting the decimal representation of a number and $op \in \{<, >, =\}$ denotes the operator. A solution to such a constraint is an assignment of $0-9$ to each of the 26 uppercase English letters, such that all constraints are satisfied. **Here, leading zeroes are allowed.**

For example, suppose the constraint is $\mathrm{P} = \mathrm{NP}$. Then, the set of solutions satisfying this constraint is all assignments with $N = 0$. Another example is the constraint $2000\mathrm{CNY} > 3000\mathrm{USD}$. Here, no assignments can satisfy this constraint since 2000CNY is a 7-digit decimal integer less than 3×10^6, and 3000USD is a 7-digit decimal integer greater than or equal to 3×10^6.

Now Bobo has received a system of n constraints, and he wonders how many assignments of $0-9$ to each of the 26 uppercase English letters are there, such that all constraints are satisfied. Since the answer might be too large, you need to output the answer modulo 998 244 353 (a prime number).

Input

The first line of input contains one integer n ($0 \le n \le 10$), denoting the number of constraints.

Then, n lines follow. Each line contains a constraint in the form of $X\,op\,Y$, where X and Y are strings consisting of digits from 0 to 9 and uppercase English letters and $op \in \{<, >, =\}$.

It is guaranteed that the sum of lengths over all constraints does not exceed 50.

Output

Output one integer in a line, denoting the number of solutions to the given system of constraints, taken modulo 998 244 353.

Examples

standard input	standard output

`1`
`P=NP`
	`766136394`

`1`
`2000CNY>3000USD`
	`0`

`4`
`AB>CD`
`E<A`
`BC>FF`
`EF>F1`
	`23645065`

Note

As already discussed in the statement, the constraint $\mathrm{P} = \mathrm{NP}$ has 10^{25} solutions, which is 766136394 after taken modulo 998 244 353, and $2000\mathrm{CNY} > 3000\mathrm{USD}$ has zero solutions.

Problem C. Bladestorm

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

In Hearthstone, *Bladestorm* is a strong warrior-class spell card, as shown below. Its effect is to deal one damage to all minions until one minion dies.

In real-game situations, Bladestorm itself cannot deal with complicated boards, so it is usually used together with some other area-of-effect spells. Now Bobo is playing the warrior class in Hearthstone, and his opponent plays minions with **pairwise distinct** health points $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le n)$ sequentially. Bobo has an infinite amount of Bladestorm and spells that can deal k damages to all minions, and we assume that he may play arbitrary of them in any order. He wonders, each time the opponent plays a minion, what is the minimum number of spells he needs to play to remove all minions on board?

Formally, for each $i = 1, 2, \ldots, n$, let $S_i = \{a_1, a_2, \ldots, a_i\}$ be the set including the health points of the first i minions. Bobo wants to compute ans_i, which is the minimum number of the following operations to perform to make S_i empty:

- *Play Bladestorm.*
  1. Set $x \leftarrow x - 1$ for each $x \in S_i$
  2. If S_i contains at least one 0, erase all 0s in S_i and end this operation. Otherwise, goto step 1.
- *Play a normal area of effect.* Set $x \leftarrow x - k$ for each $x \in S_i$ and erase all elements not greater than 0 in S_i.

Input

This problem contains multiple test cases. The first line of input contains an integer T $(1 \le T \le 50\,000)$, denoting the number of test cases.

For each test case, the first line of input contains two integers n, k $(1 \le k \le n \le 10^5)$, denoting the number of minions your opponent plays and the damage of the normal area-of-effect.

Then, one line containing n **pairwise distinct** integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le n)$ follows, denoting the health of each minion the opponent plays sequentially.

It is guaranteed that the sum of n over all test cases does not exceed $5 \cdot 10^5$.

Output

For each test case, output n integers $\text{ans}_1, \text{ans}_2, \ldots, \text{ans}_n$ in one line, where the i-th $(1 \le i \le n)$ integer denotes the minimum number of spells Bobo needs to play to remove all minions on board after the opponent plays the first i minions.

Example

standard input	standard output

3
7 2
4 7 3 6 1 2 5
11 3
10 7 4 9 6 8 5 1 3 2 11
9 2
1 2 3 7 8 9 4 5 6
	1 2 3 3 4 4 4
	1 2 3 3 3 3 3 4 4 4 4
	1 1 2 3 4 4 4 5 5

Problem D. Graph of Maximum Degree 3

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Bobo recently obtained a (undirected) graph $G = (V, E)$. He noticed that each edge of this graph is colored either red or blue, and the maximum degree of this graph is not greater than 3.

Now, Bobo wonders how many nonempty induced subgraphs $G[S] = (S, E')$ of G are there, such that the two following conditions are satisfied:

- The graph formed by keeping **only the red edges** of $G[S]$ is connected.
- The graph formed by keeping **only the blue edges** of $G[S]$ is also connected.

As the answer might be too large, you need to output the result modulo 998 244 353 (a prime number).

Refer to the note section for formal definitions of the underlined items and more illustrations.

Input

The first line contains two integers n ($1 \le n \le 10^5$) and m ($0 \le m \le 1.5 \times 10^5$), denoting the number of vertices and edges in the graph G, respectively.

Then m lines follows, each line containing three integers u, v, c ($1 \le u, v \le n, u \ne v, c \in \{0, 1\}$), denoting there is an edge (u, v), with color red if $c = 0$ and color blue otherwise.

It is guaranteed that the maximum degree of G is at most 3, and no two edges with the same color connect the same pair of vertices. **Please be aware that multiple edges in G with different colors may still exist.**

Output

Output an integer in a line, denoting the number of induced subgraphs satisfying the required conditions, taken modulo 998 244 353.

Examples

standard input	standard output

3 4
1 2 0
1 3 1
2 3 0
2 3 1
	5

4 6
1 2 0
2 3 0
3 4 0
1 4 1
2 4 1
1 3 1
	5

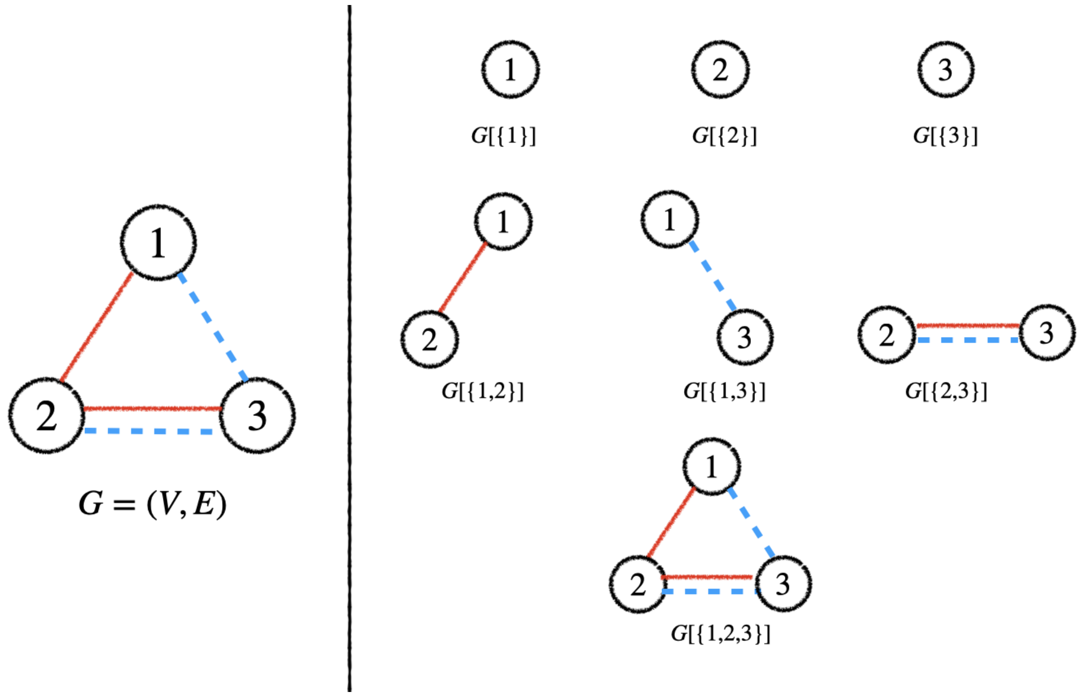
Note

Here, we provide formal definitions of some underlined items in the statement.

- The degree of a vertex of a graph is the number of edges that are incident to the vertex.

- The maximum degree of a graph G is the maximum of its vertices' degrees;
- An induced subgraph of a graph $G = (V, E)$ is another graph $G[S] = (S, E')$, formed from a subset S of the vertices of the graph and all of the edges E' (from the original graph G) connecting pairs of vertices in that subset S. An induced graph $G[S]$ is nonnempty if and only if $S \neq \varnothing$.
- A graph is said to be connected if every pair of vertices in the graph is connected, i.e., there is a path between every pair of vertices.

The following picture lists the graph $G = (V, E)$ and all its seven nonempty induced subgraphs for the first sample test. Here, solid lines represent the red edge, while the dashed lines represent the blue ones. Five nonempty induced subgraphs satisfy the required condition, which are $G[\{1\}], G[\{2\}], G[\{3\}], G[\{2, 3\}]$ and $G[\{1, 2, 3\}]$, respectively.



Problem E. Inverse Topological Sort

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Bobo recently learned the concept of topological ordering. One day, he observed a directed acyclic graph (DAG) $G = (V, E)$ with $|V| = n$ vertices numbered from 1 to n, and immediately wrote down on the paper two sequences $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$, such that A is the lexicographically smallest topological ordering of G, and B is the lexicographically largest topological ordering of G.

Unfortunately, now Bobo has forgotten what the original DAG G looks like, and all he has is the two sequences A and B. Can you help Bobo recover the original graph G? There might be multiple possible graphs corresponding to A and B, or Bobo might write down the sequences incorrectly so that no valid graphs exist.

Refer to the note section for formal definitions of the underlined items.

Input

The first line of input contains an integer n ($1 \le n \le 10^5$), denoting the length of the two sequences.

The second line of input contains n **pairwise distinct** integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$), denoting the first sequence A.

The second line of input contains n **pairwise distinct** integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le n$), denoting the second sequence B.

Output

If there exists a directed graph G that satisfies the condition, output "Yes" in the first line; otherwise, output "No" in the first line. You can output each letter in any case (lowercase or uppercase). For example, the strings "yEs", "yes", "Yes", and "YES" will all be considered as positive replies.

If your answer is "Yes", output an integer m ($0 \le m \le \min(n(n-1)/2, 10^6)$) in the first line. Then, in the following m lines, output two integers u, v ($1 \le u, v \le n$) each, denoting a directed edge (u, v) in the graph G. The graph G you output should satisfy that it is a directed acyclic graph, A is the lexicographically smallest topological ordering of G, and B is the lexicographically largest topological ordering of G. If multiple solutions exist, outputting any of them will be considered correct.

Note again that the graph you output must have no more than 10^6 edges. It can be shown that if there exists any valid graph, there exists a valid one with no more than 10^6 edges.

Examples

standard input	standard output

3
1 2 3
1 2 3
	Yes
	3
	1 2
	2 3
	1 3

3
1 2 3
3 2 1
	Yes
	0

3
3 2 1
1 2 3
	No

Note

Here, we provide formal definitions of some underlined items in the statement.

- A topological ordering of a directed graph $G = (V, E)$ is a linear ordering (i.e., permutation) of its vertices such that for every directed edge $(u, v) \in E$ from vertex u to vertex v, u comes before v in the ordering. It can be shown that a directed graph admits at least one topological ordering if and only if it is **acyclic**.
- For two sequences $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$ with the same length n, A is said to be lexicographically smaller than B if and only if there exists some index $1 \le i \le n$, such that
  - $a_i < b_i$;
  - $a_j = b_j$ for all $j < i$.

Problem F. Land Trade

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

Recently, Bobo had an idea to buy a piece of land from a landholder. Bobo, however, doesn't want to buy the entire rectangular land. Instead, he only wants to buy a part of the land which satisfies certain requirements.

Mathematically speaking, the rectangular land is defined by the set of points

$$L = \{(x, y) : x_{\min} \le x \le x_{\max}, y_{\min} \le y \le y_{max}\}$$

in Cartesian coordinates. Bobo's requirements are encoded into a logic formula. The logic formula is described in the following syntax:

```
<formula> := "(" <formula> "&" <formula> ")"
           | "(" <formula> "|" <formula> ")"
           | "(" <formula> "^" <formula> ")"
           | "(" "!" <formula> ")"
           | <atomic-formula>
<atomic-formula> := "[" <decimal> "," <decimal> "," <decimal> "]"
```

where `"&"`, `"|"`, and `"^"` are And, Inclusive Or, and Exclusive Or operators, respectively; `<decimal>` refers to any signed decimal integer; an atomic formula $[a, b, c]$ denotes a constraint $ax + by + c \ge 0$. Formally, the part of the land he wants is the set of points in L satisfying the given logical formula.

To proceed with the transaction, they want to know the area of the part of the land Bobo wants to buy. Can you tell them the answer?

Input

The first line of the input contains four integers $x_{\min}, x_{\max}, y_{\min}, y_{\max}$ ($x_{\min} < x_{\max}, y_{\min} < y_{\max}$), the absolute values of which do not exceed 1 000, specifying the region of the rectangular land.

The second line of the input is the logic formula. The formula contains no more than 10 000 characters and, at most 300 atomic formulas. The logic formula strictly conforms to the syntax of the problem statement; there is neither space between tokens nor any omission of parentheses. For every atomic formula $[a, b, c]$, it is guaranteed that $|a|, |b|, |c| \le 1\,000$ and $a^2 + b^2 \ne 0$.

Output

Output a number in a line, denoting the area of the part of the land the landlord wants to buy. The answer will be considered correct if its absolute or relative error does not exceed 10^{-6}.

Examples

standard input	standard output

`0 1 0 1`
`([-1,1,0]^[-1,-1,1])`
	`0.5`

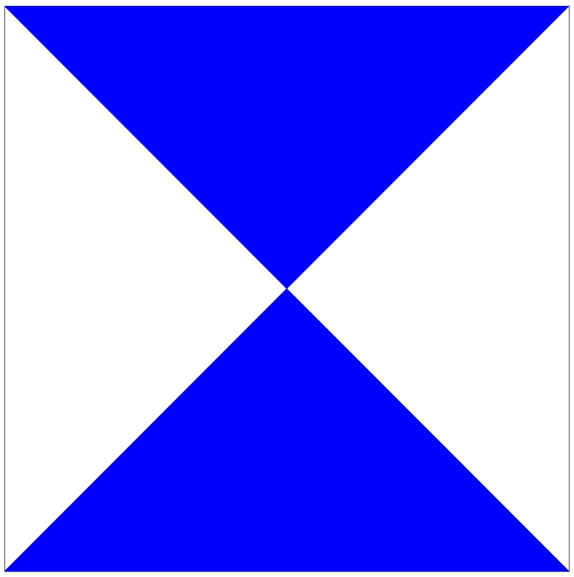
`-5 10 -10 5`
`((!([1,2,-3]&[10,3,-2]))\\`
`^([-2,3,1]|[5,-2,7]))`
	`70.451693404634582`

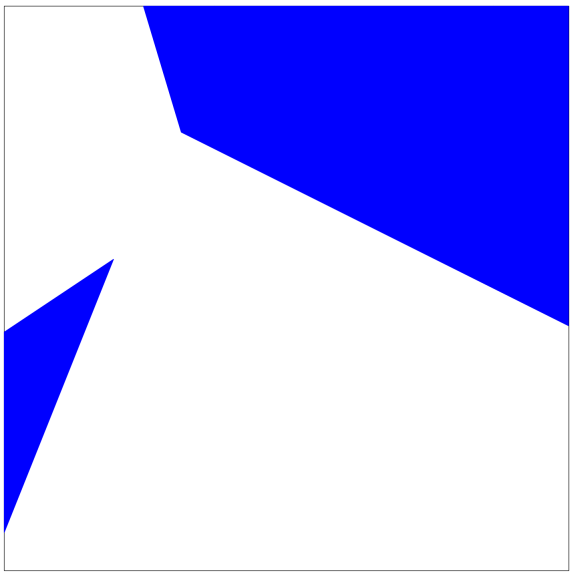
`0 1 -1 1`
`([1,1,1]&[-1,-1,-1])`
	`0`

Note

Note that in the second sample test, there exists “\\” that represents a line break, which is not present in the real input. Here, we add the line break only for better formatting and illustration.

The plots for the first two sample test data are shown below, where the filled part represents the part of the land Bobo wants to buy.





Problem G. Parity Game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

This is an interactive problem.

Alice and Bob are playing the following game. The game is specified by a parameter $t \in \{0, 1\}$ and is described as follows:

> There are n integers $a_1, a_2, \ldots, a_n \in \{0, 1\}$, arranged in a line. The two players **take turns** to perform the following operation when at least two integers are remaining, starting from Alice:
>
> - Choose two integers x and y that are **adjacent**, replace them with either $x + y$ or $x \times y$ (arithmetic addition or arithmetic multiplication).
>
> The game ends when there is only one integer remaining, and the result of the game is judged by the parity of the remaining integer and the parameter t as follows:
>
> - $t = 0$: Alice wins if the remaining integer is **even** and Bob wins otherwise.
> - $t = 1$: Alice wins if the remaining integer is **odd** and Bob wins otherwise.

You are given the initial n integers $a_1, a_2, \ldots, a_n$ and the parameter t. You must choose to play as Alice or Bob, then play this game against the interactor who takes the opponent's role. Your goal is to win the game.

Input

The first line of input contains two integers n and t ($2 \le n \le 500, t \in \{0, 1\}$), denoting the size of the array and the parameter of the game.

The second line of input contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 1$), denoting the array.

Interaction Protocol

The interaction begins after reading the two integers n, t and the array $\{a_i\}_{1 \le i \le n}$.

You should start interaction by printing a single line containing either "Alice" or "Bob", representing the player you choose to play as. If you choose to play Alice, you are the first to operate. Otherwise, you are the second to operate.

The game proceeds as follows:

- If it's your turn to operate. Suppose the current array is $b_1, b_2, \ldots, b_m$ with length m ($2 \le m \le n$), you should output an integer p ($1 \le p \le m - 1$) and a character c ($c \in \{\text{'+'}, \text{'*'}\}$) in a line, separated by a space, denoting you choose to merge the two integers b_p and b_{p+1} with the operation denoted by c. After your operation, the array becomes $b_1, b_2, \ldots, b_{p-1}, (b_p\ c\ b_{p+1}), b_{p+2}, \ldots, b_m$ with length $m - 1$.
- If it's the opponent's turn to operate. Suppose the current array is $b_1, b_2, \ldots, b_m$ with length m ($2 \le m \le n$), you should read an integer p ($0 \le p \le m - 1$) and a character c ($c \in \{\text{'+'}, \text{'*'}\}$) in a line, denoting the opponent chooses to merge the two integers b_p and b_{p+1} with the operation denoted by c. After the opponent's operation, the array becomes $b_1, b_2, \ldots, b_{p-1}, (b_p\ c\ b_{p+1}), b_{p+2}, \ldots, b_m$ with length $m - 1$. Here, if the integer p is 0, then it means the last operation you perform is incorrect. In that case, your program should terminate immediately to receive a "Wrong Answer" verdict, or you could get any possible verdict.

When there is only one integer in the array, the game finishes and is judged according to the rule in the statement. If you win the game, your program will be considered correct for this test case. If you lose the game, or it is you to perform the last operation and your last operation is invalid, you will receive a "Wrong Answer" verdict. You should terminate the program immediately when the game finishes.

After printing the player you choose to play as and the operation you perform, do not forget to output the end of the line and flush the output. Otherwise, you may not get the correct verdict. To flush the output, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `stdout.flush()` in Python;

Example

standard input	standard output
`4 1`	
`0 1 0 1`	
	`Alice`
	`1 +`
`2 *`	
	`1 +`

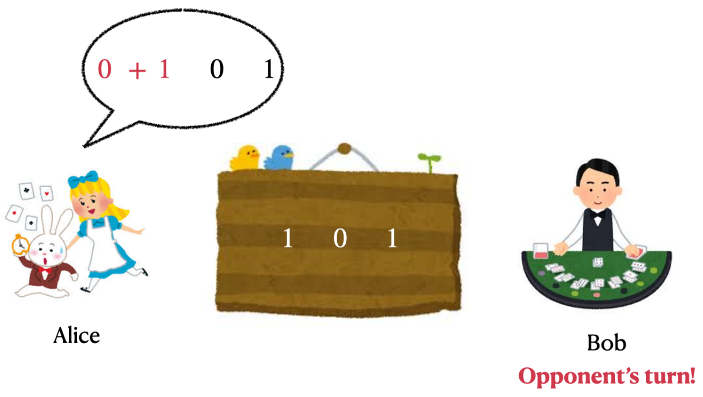
Note

Note that additional empty lines in the sample test are for better understanding, and you should not output any additional empty lines in your program.

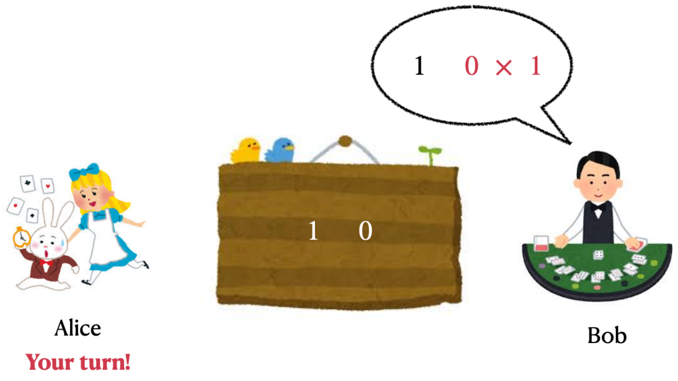
Here, we provide graphical illustrations of the interaction in the sample test. For the given $\{a_i\}_{1\leq i\leq n}$ and t, it is optimal for the player to choose to play Alice and go first.

0 1 0 1
Alice
Your turn!
Bob

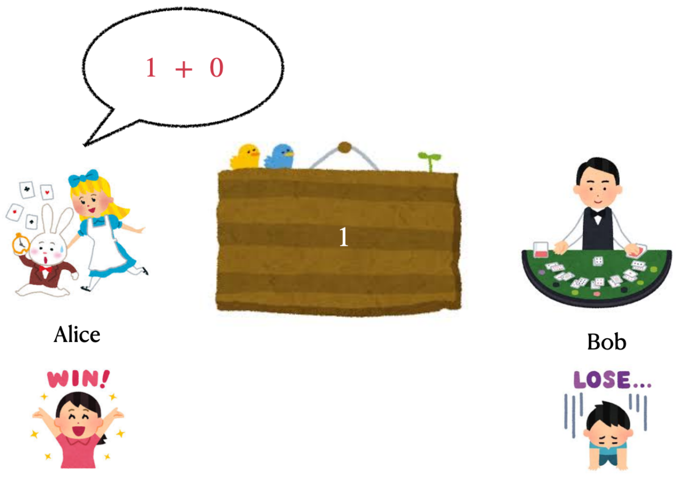
In the sample interaction, the player chooses to merge the first two numbers with the addition operator,



and the opponent chooses to merge the last two numbers with the multiplication operator.



Then, the player chooses to merge the remaining two numbers with the addition operator and wins the game.



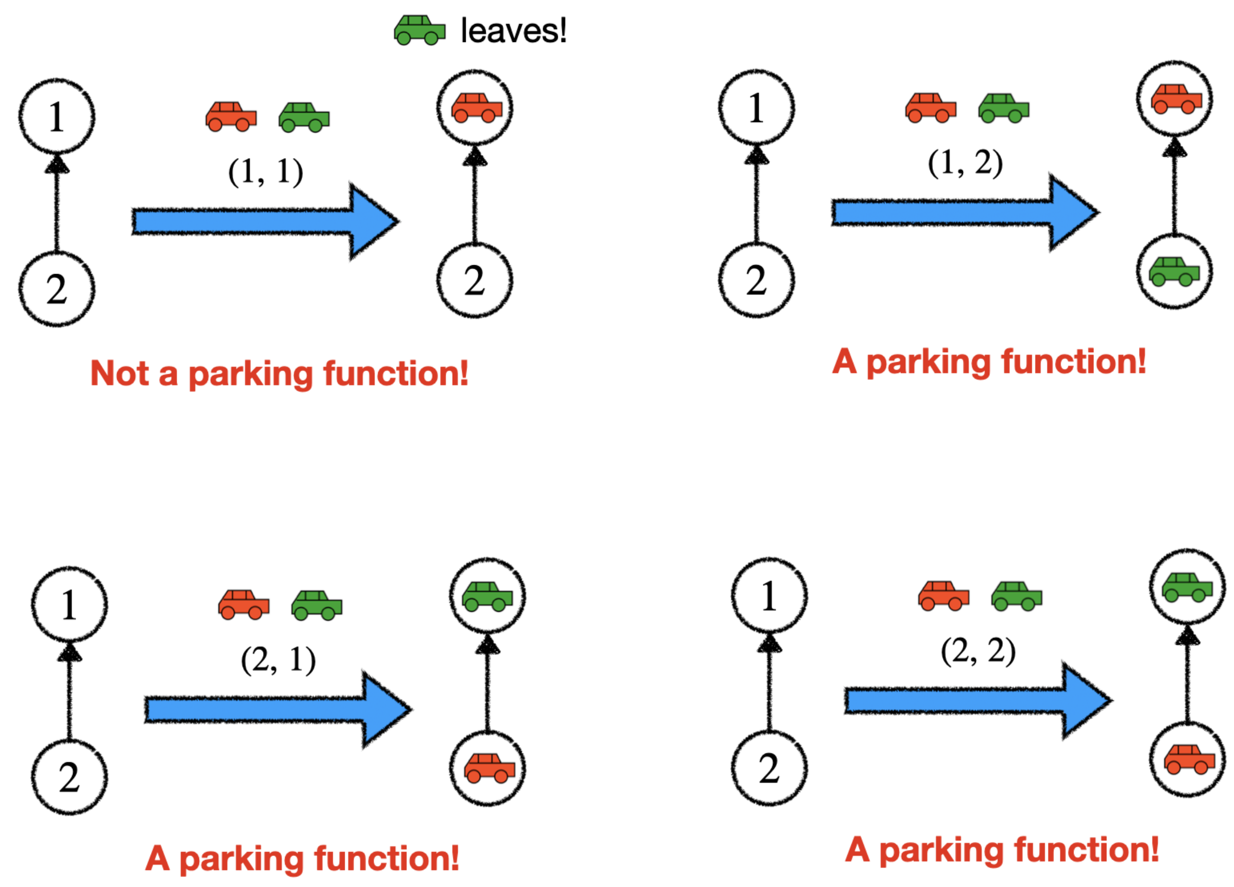
Problem H. Random Tree Parking

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Parking functions are combinatorial objects originally introduced by Konheim and Weiss during their studies of the so-called linear probing collision resolution scheme for hash tables. As a crazy fan of mathematics, Bobo now wants to investigate parking functions on trees.

Consider a parking lot in the form of a tree T, with labels in $[n] = \{1, 2, \ldots, n\}$, rooted at vertex 1. The edges of the tree are oriented towards the root vertex. We have a sequence of n drivers, each with their preferred parking space, which is a vertex in the tree. The drivers arrive one by one, and each driver i ($1 \le i \le n$) tries to park at their preferred space $s_i \in [n]$. If the space is free, they park there. Otherwise, they follow the edges towards the root vertex and park at the first empty vertex they encounter. If there are no empty vertices, they leave the parking lot, i.e., the tree, without parking.

A sequence $\mathbf{s} \in [n]^n$ of vertices is then called a parking function of the tree T if all drivers are successful, i.e., if all drivers are able to find a parking space. For example, for the following tree T with two vertices, there are three parking functions, namely $(1, 2)$, $(2, 1)$ and $(2, 2)$.



Now, given a tree T with n vertices, Bobo wonders what is the number of parking functions of T. As Bobo is also obsessed with randomness, he decided that the tree T should be generated **randomly** in the following way:

- For $i = 2, \ldots, n$, the vertex i points to is chosen independently uniformly at random from $\{1, 2, \ldots, i - 1\}$.

Since the answer might be too large, you should output the answer modulo 998 244 353 (a prime number).

Input

The first line of input contains an integer n ($2 \le n \le 10^5$), denoting the size of the tree.

The next line contains $n-1$ integers $p_2, p_3, \dots, p_n$ ($1 \le p_i \le i-1$), where p_i denotes the unique vertex i points to in T. **You can assume that each p_i is chosen independently uniformly at random from $\{1, 2, \dots, i-1\}$.**

It is guaranteed that there are at most 20 test sets for this problem.

Output

Output an integer in a line, denoting the number of parking functions of T, taken modulo 998 244 353.

Examples

standard input	standard output
3 1 1	12
3 1 2	16
4 1 2 3	125
8 1 2 3 1 3 4 3	1198736
15 1 2 2 2 2 3 3 2 7 7 3 10 3 13	938578089

Problem I. Refresher into Midas

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

So this is what you buy Refresher Orb for?
To refresh your Hand of Midas?

— Angry Teammates

Hand of Midas is an item in Dota2 that provides significant gold to the hero who uses it. Upon use, it kills a non-hero target for 160 gold. Hand of Midas has a cooldown of a seconds, meaning it can only be used again after a seconds of last use.

Refresher Orb is another powerful item in Dota2. When used, it refreshes the cooldowns of all the user's **other** abilities and items, allowing the user to cast spells and use items again immediately. For example, if your Hand of Midas is in cooldown, while your Refresher Orb is not, you can use Refresher Orb so that you can use Hand of Midas immediately. Refresher Orb has a cooldown of b seconds. Apparently, using Refresher Orb doesn't refresh the cooldown of Refresher Orb itself.

You have bought both Hand of Midas and Refresher Orb in the game, and now you want to maximize the gold you get in the next m seconds! Currently, both your Hand of Midas and Refresher Orb are not in cooldown. We assume that the time you use Hand of Midas and Refresher Orb can be neglected.

Input

This problem contains multiple test cases. The first line of input contains an integer T $(1 \le T \le 10\,000)$, denoting the number of test cases.

For each test case, the input contains one line with three integers a, b, m $(1 \le a, b \le 10^6, 0 \le m \le 10^6)$, denoting the cooldown of Hand of Midas, the cooldown of Refresher Orb, and the amount of time you have to maximize your gold, respectively.

It is guaranteed that the sum of a, b, and m over all test cases does not exceed 10^7, respectively.

Output

For each test case, output an integer in a line, denoting the maximum number of gold you can get.

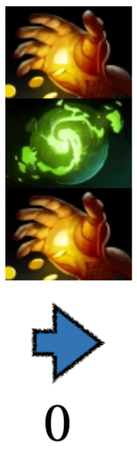
Example

standard input	standard output

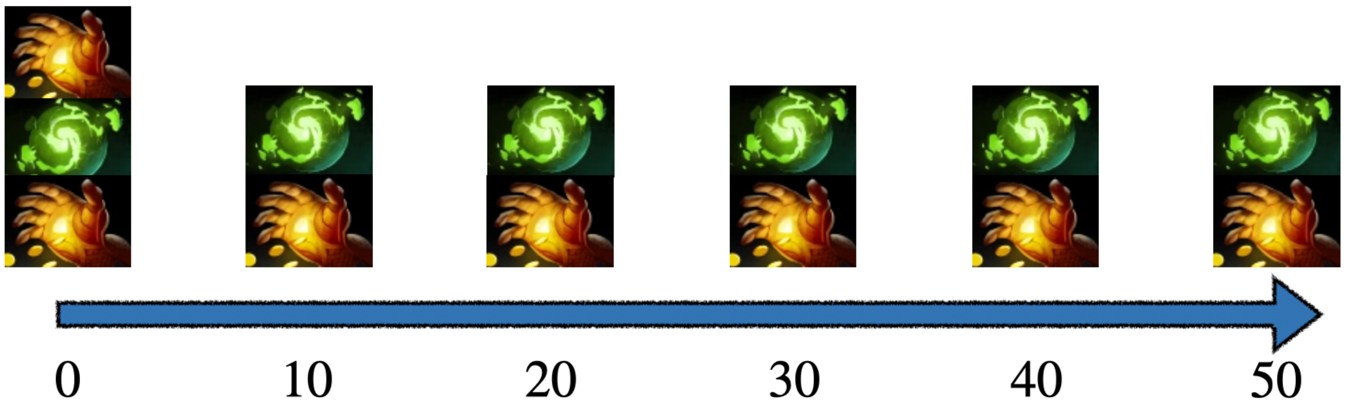
6
50 100 0
40 10 50
10 40 50
1 1 1000000
60 200 960
60 185 905
	320
	1120
	1280
	320000320
	3520
	3360

Note

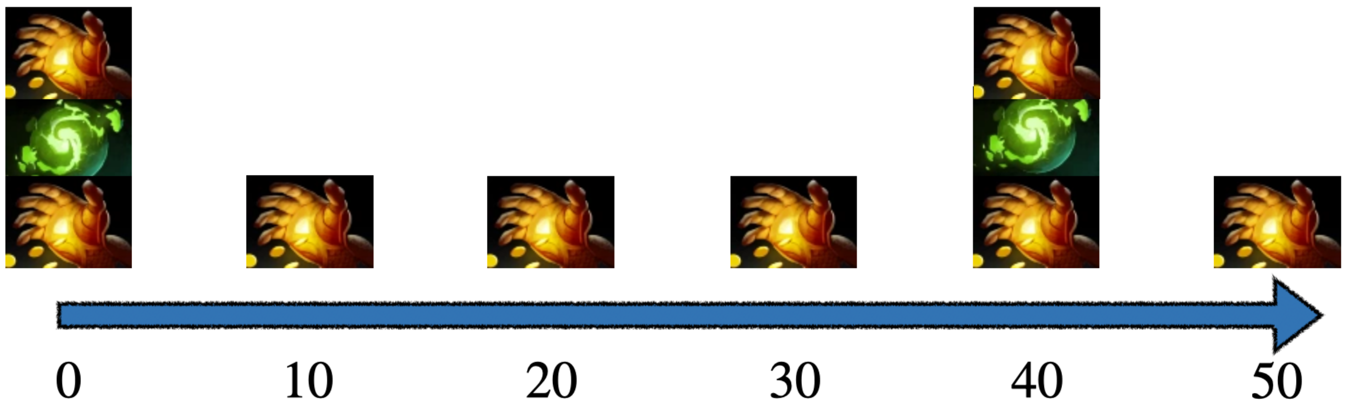
For the first test case of the sample test, the optimal strategy is to use Hand of Midas, then immediately use Refresher Orb and then Hand of Midas, obtaining a total of $160 \times 2 = 320$ gold. This strategy is described in the following picture (The ball-like item is Refresher Orb, while the hand-like item is the Hand of Midas).



For the second test case of the sample test, the optimal strategy is described in the following picture, obtaining a total of $160 \times 7 = 1\,120$ gold.



For the third test case of the sample test, the optimal strategy is described in the following picture, obtaining a total of $160 \times 8 = 1\,280$ gold.



Problem J. Teleportation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Bobo recently visited a strange teleportation system. The system contains n rooms, numbered 0 through $n-1$. A teleporting device is installed in each room. Each teleporting device contains a dashboard that looks like a clock surface with a hand on it, showing numbers 0 through $n-1$ in clockwise order. Initially, the hand on the dashboard of the teleport device in the i-th $(0 \le i \le n-1)$ room points to the number a_i.

When Bobo is in room i $(0 \le i \le n-1)$, he may do the following operation any number of times:

- *Teleport.* Immediately teleport to the room $(i + a_i) \bmod n$.
- *Move the hand clockwise.* Set $a_i \leftarrow a_i + 1$.

Each operation takes one unit of time. Bobo starts at room 0, and he wants to reach some room x as quickly as possible. He wonders how long it is needed.

Input

The first line of input contains two integers n $(2 \le n \le 10^5)$ and x $(1 \le x \le n-1)$, denoting the number of rooms and Bobo's destination room, respectively.

The next line contains n integers $a_0, a_1, \ldots, a_{n-1}$ $(0 \le a_i \le n-1)$, where a_i $(0 \le i \le n-1)$ denotes the number the hand in the i-th room points to.

Output

Output an integer in a line, denoting the minimum time Bobo needs to reach room x from room 0.

Examples

standard input	standard output

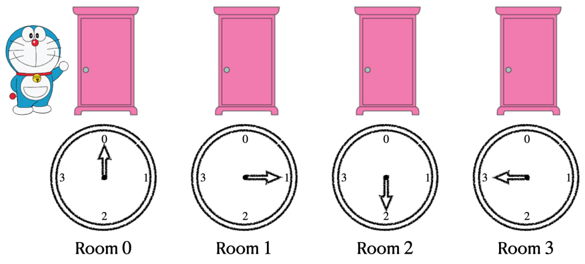
`4 3`
`0 1 2 3`
	`4`

`4 3`
`0 0 0 0`
	`4`

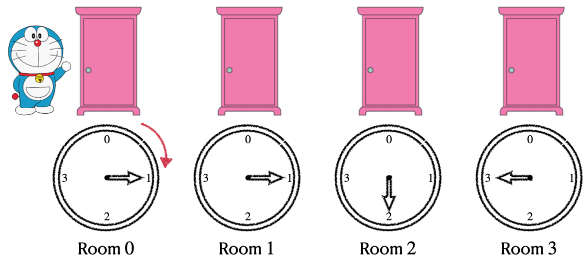
`4 3`
`2 2 2 2`
	`2`

Note

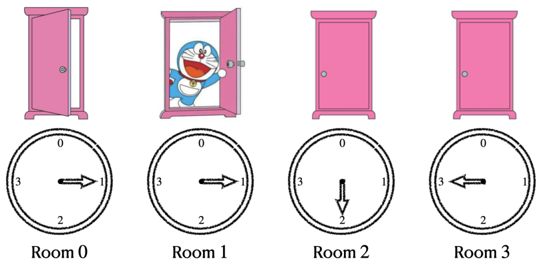
Here, we provide graphical illustrations of one possible optimal way in the first sample. Initially, Bobo is at room 0, and the hand on each dashboard is at $0, 1, 2, 3$, respectively.



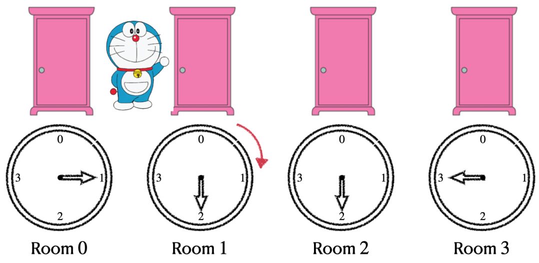
The first operation Bobo does is to move the hand clockwise so that the hand on the dashboard in room 0 points to 1. ($a_0 = 1$)



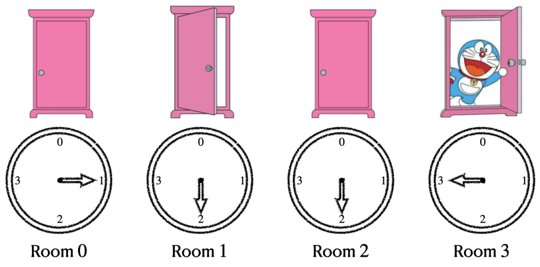
Then Bobo teleports to room $(0 + a_0) \bmod n = 1$.



After that, Bobo moves the hand clockwise so that the hand on the dashboard in room 1 points to 2. ($a_1 = 2$).



Then Bobo teleports to room $(1 + a_1) \bmod n = 3$, reaching his desired destination. It takes an overall of 4 operations.



Problem K. Understand

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

This is an interactive problem.

In the simplified version of the game "Understand," you find yourself in a 256×256 grid. There are n items within this grid whose precise positions are yet unknown. It is possible that multiple items are in the same grid cell.

Your task is to determine the locations of each item through a series of interactive queries, with a limit of **no more than** 16 **queries**. Each query allows you to draw a **simple path** on the grid, and the interactor will respond with a binary string of length n, representing whether each item is on that path.

It is guaranteed that the interactor is non-adaptive, meaning the positions of the items are already determined before you start querying.

Input

The first line of input contains an integer n ($1 \le n \le 10\,000$), denoting the number of items in the grid.

Interaction Protocol

The interaction begins after reading the integer n.

Then, make at most 16 queries. To make a query: output "? x_0 y_0 k s" in a line. Here, $1 \le x_0, y_0 \le 256$ represents that the path starts at the x_0-th row from top to bottom and the y_0-th column from left to right. $1 \le k \le 256 \times 256$ represents the length of the simple path. s is a string of length $k-1$ consisting of characters 'L,' 'R,'' U', and 'D.' The i-th character represents the direction of movement from the i-th cell to the $(i+1)$-th cell along the path, where

- 'L' means moving from (x, y) to $(x, y-1)$,
- 'R' means moving from (x, y) to $(x, y+1)$,
- 'U' means moving from (x, y) to $(x-1, y)$,
- 'D' means moving from (x, y) to $(x+1, y)$.

If $k = 1$, you can simply output "? x_0 y_0 k" and may not output the additional blank space (It would be OK doing so, though).

Here, the path you output must be **simple**, meaning your path should not pass the same grid more than once. Also, the path must lie in the 256×256 grid, meaning each grid (x, y) in the path must satisfy $1 \le x, y \le 256$.

After you output the query in the correct format, read in a binary string t of length n, where the i-th ($1 \le i \le n$) character of t is 1 if the i-th item is on your path, and the i-th ($1 \le i \le n$) character of t is 0 otherwise. If your query is invalid or you have made more than 16 queries, you will receive a single character '$-$' instead. Upon reading it, your program must terminate immediately to receive a "Wrong Answer" verdict, or you could get any possible verdict.

At any point of the interaction, if you want to guess the locations of each item, output "! x_1 y_1 x_2 $y_2 \cdots x_n$ y_n" in a line, where (x_i, y_i) represents the location of the i-th item in your guess. **This guess does not count as a query.** After doing that, you should terminate your program immediately.

After printing each query and the answer, do not forget to output the end of the line and flush the output. Otherwise, you may not get the correct verdict. To flush the output, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `stdout.flush()` in Python;

Example

standard input	standard output
`4`	
	`? 3 1 9 RRRUULLD`
`0101`	
	`? 2 1 4 RRR`
`1111`	
	`? 2 1 6 URDDR`
`1100`	
	`? 2 4 1`
`0001`	
	`! 2 1 2 2 2 3 2 4`

Note

Note that additional empty lines in the sample test are for better understanding, and you should not output any additional empty lines in your program.

Here, we provide graphical illustrations of the interaction in the sample test. In the picture below, we only show the upper-left 4×4 part of the board, but the actual size of the board is 256×256. The locations of all four items are labeled with stars. The starting point of each query path is labeled with a square, and the result is shown below the path (filled circles represent 1, and unfilled circles represent 0).