A. Upload More RAM

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

Oh no, the ForceCodes servers are running out of memory! Luckily, you can help them out by uploading some of your RAM!

You want to upload n GBs of RAM. Every second, you will upload either 0 or 1 GB of RAM. However, there is a restriction on your network speed: in any k consecutive seconds, you can upload only at most 1 GB of RAM in total.

Find the minimum number of seconds needed to upload n GBs of RAM!

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \leq t \leq 10^4$) — the number of test cases. The description of the test cases follows.

The first and only line of each test case contains two integers n and k ($1 \leq n, k \leq 100$) — the number of GBs that you want to upload and the length of the time window respectively.

Output

For each test case, output a single integer — the minimum number of seconds needed to upload n GBs of RAM.

Standard Input	Standard Output
6 5 1 2 2 2 3 1 7 11 5 100 100	5 3 4 1 51 9901

Note

In the first test case, you can upload 1 GB of RAM per second, so to upload 5 GBs, you need 5 seconds.

In the second test case, you can upload 1 GB in the first second, 0 GBs in the second second, and 1 GB in the third second, which in total adds up to exactly 2 GBs of uploaded RAM.

In the third test case, you can upload 1 GB in the first second, 0 GBs in the second second, 0 GBs in the third second, and 1 GB in the fourth second, which in total adds up to exactly 2 GBs of uploaded RAM.

,

B. K-Sort

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

You are given an array of integers a of length n.

You can apply the following operation any number of times (maybe, zero):

- First, choose an integer k such that $1 \le k \le n$ and pay $k + 1$ coins.
- Then, choose **exactly** k indices such that $1 \le i_1 < i_2 < \ldots < i_k \le n$.
- Then, for each x from 1 to k, increase a_{i_x} by 1.

Find the minimum number of coins needed to make a non-decreasing. That is, $a_1 \le a_2 \le \ldots \le a_n$.

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$) — the length of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the elements of the array a.

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

Output

For each test case, output a single integer — the minimum number of coins needed to make a non-decreasing.

Standard Input	Standard Output
5 3 1 7 9 5 2 1 4 7 6 4 1 3 2 4 1 179 9 344 12 37 60 311 613 365 328 675	0 3 2 0 1821

Note

In the first test case, a is already sorted, so you don't have to spend any coins.

In the second test case, the optimal sequence of operations is:

- Choose $k = 2$ and the indices 2 and 5: $[2, \color{red}{1}, 4, 7, \color{red}{6}] \to [2, 2, 4, 7, 7]$. This costs 3 coins.

It can be proven that it is not possible to make a non-decreasing by spending less than 3 coins.

,

C. Basil's Garden

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

There are n flowers in a row, the i-th of them initially has a positive height of h_i meters.

Every second, the wind will blow from the left, causing the height of some flowers to decrease.

Specifically, every second, for each i from 1 to n, in this order, the following happens:

- If $i = n$ or $h_i > h_{i+1}$, the value of h_i changes to $\max(0, h_i - 1)$.

How many seconds will pass before $h_i = 0$ for all $1 \le i \le n$ for the first time?

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$) — the number of flowers.

The second line of each test case contains n integers $h_1, h_2, \ldots, h_n$ ($1 \le h_i \le 10^9$) — the heights of the flowers.

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

Output

For each test case, output a single integer — the number of seconds that will pass before $h_i = 0$ for all $1 \le i \le n$.

Standard Input	Standard Output
4 3 1 1 2 2 3 1 1 9 5 7 4 4 3 2	4 3 9 7

Note

In the first test case, the flower heights change as follows:
$[1, 1, 2] \to [1, 1, 1] \to [1, 1, 0] \to [1, 0, 0] \to [0, 0, 0]$.

In the second test case, the flower heights change as follows: $[3, 1] \to [2, 0] \to [1, 0] \to [0, 0]$.

,

D. World is Mine

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

Alice and Bob are playing a game. Initially, there are n cakes, with the i-th cake having a *tastiness* value of a_i.

Alice and Bob take turns eating them, with Alice starting first:

- In her turn, Alice chooses and eats any remaining cake whose tastiness is **strictly greater** than the **maximum** tastiness of any of the cakes she's eaten before that. Note that on the first turn, she can choose any cake.
- In his turn, Bob chooses any remaining cake and eats it.

The game ends when the current player can't eat a suitable cake. Let x be the number of cakes that Alice ate. Then, Alice wants to maximize x, while Bob wants to minimize x.

Find out how many cakes Alice will eat if both players play optimally.

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \le t \le 500$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 5000$) — the number of cakes.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the tastiness values of the cakes.

It is guaranteed that the sum of n over all test cases does not exceed 5000.

Output

For each test case, output a single integer — the number of cakes Alice will eat if both players play optimally.

Standard Input	Standard Output
9 4 1 4 2 3 3 1 1 1 5 1 4 2 3 4 4 3 4 1 4 1 1 8 4 3 2 5 6 8 3 4 7 6 1 1 3 5 3 1 11	2 1 3 2 1 3 2 4 4

6 11 6 8 7 5 3 11 2 3 5 17 2 6 5 3 9 1 6 2 5 6 3 2 3 9 6 1 6	

Note

In the first test case, one possible sequence of turns is:

1. Alice eats a cake with a tastiness value of 1. The remaining cakes are $[4, 2, 3]$.
2. Bob eats a cake with a tastiness value of 2. The remaining cakes are $[4, 3]$.
3. Alice eats a cake with a tastiness of 3. The remaining cakes are $[4]$.
4. Bob eats a cake with a tastiness value of 4. The remaining cakes are $[]$.
5. Since there are no more cakes left, the game ends.

In the second test case, one possible sequence of turns is:

1. Alice eats a cake with a tastiness value of 1. The remaining cakes are $[1, 1]$.
2. Bob eats a cake with a tastiness value of 1. The remaining cakes are $[1]$.
3. Since Alice has already eaten a cake with a tastiness value of 1, she cannot make a turn, so the game ends.

,

E. Wonderful Tree!

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

God's Blessing on This ArrayForces!

A Random Pebble

You are given a tree with n vertices, rooted at vertex 1. The i-th vertex has an integer a_i written on it.

Let L be the set of all direct children* of v. A tree is called *wonderful*, if for all vertices v where L is not empty,

$$a_v \leq \sum_{u \in L} a_u.$$

In one operation, you choose any vertex v and increase a_v by 1.

Find the minimum number of operations needed to make the given tree *wonderful*!

* Vertex u is called a direct child of vertex v if:

- u and v are connected by an edge, and
- v is on the (unique) path from u to the root of the tree.

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \leq t \leq 1000$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($2 \leq n \leq 5000$) — the number of vertices in the tree.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq 10^9$) — the values initially written on the vertices.

The third line of each test case contains $n - 1$ integers $p_2, p_3, \ldots, p_n$ ($1 \leq p_i < i$), indicating that there is an edge from vertex p_i to vertex i. It is guaranteed that the given edges form a tree.

It is guaranteed that the sum of n over all test cases does not exceed 5000.

Output

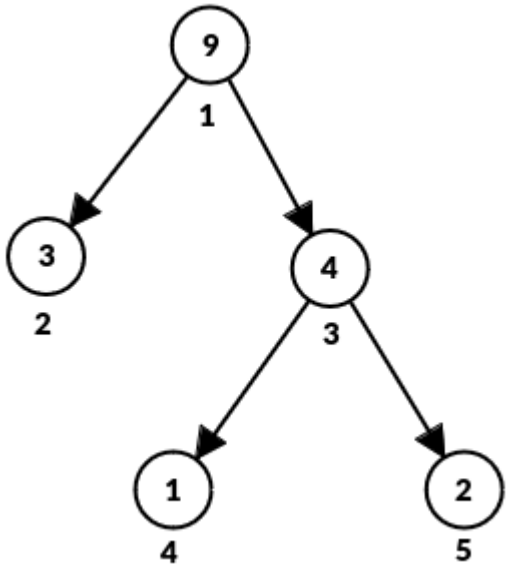
For each test case, output a single integer — the minimum number of operations needed to make the tree *wonderful*.

Standard Input	Standard Output
4 5 9 3 4 1 2 1 1 3 3 2 5 3 1 2	3 2 0 0

36 54 1 3 0 0 0 1 2	

Note

The tree in the first test case:



You can apply the operation once on vertex 5 and twice on vertex 2 to get a *wonderful* tree.

In the second test case, you can apply the operation twice on vertex 2 to get a *wonderful* tree.

In the third and fourth test cases, the tree is already *wonderful*, so you don't need to apply any operations.

,

F1. Interesting Problem (Easy Version)

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

This is the easy version of the problem. The only difference between the two versions is the constraint on n. You can make hacks only if both versions of the problem are solved.

You are given an array of integers a of length n.

In one operation, you will perform the following two-step process:

1. Choose an index i such that $1 \leq i < |a|$ and $a_i = i$.
2. Remove a_i and a_{i+1} from the array and concatenate the remaining parts.

Find the maximum number of times that you can perform the operation above.

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \leq t \leq 100$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \leq n \leq 100$) — the length of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$) — the elements of the array a.

It is guaranteed that the sum of n over all test cases does not exceed 100.

Output

For each test case, output a single integer — the maximum number of times that you can perform the operation.

Standard Input	Standard Output
6 5 1 5 3 2 4 8 2 1 3 4 5 6 7 8 3 1 2 3 4 1 2 4 4 5 4 4 1 3 5 1 1	2 3 1 2 0 0

Note

In the first test case, one possible optimal sequence of operations is $[1, 5, 3, 2, 4] \rightarrow [1, 5, 4] \rightarrow [4]$.

In the third test case, one possible optimal sequence of operations is $[1, 2, 3] \to [1]$.

,

F2. Interesting Problem (Hard Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

This is the hard version of the problem. The only difference between the two versions is the constraint on n. You can make hacks only if both versions of the problem are solved.

You are given an array of integers a of length n.

In one operation, you will perform the following two-step process:

1. Choose an index i such that $1 \leq i < |a|$ and $a_i = i$.
2. Remove a_i and a_{i+1} from the array and concatenate the remaining parts.

Find the maximum number of times that you can perform the operation above.

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \leq t \leq 100$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \leq n \leq 800$) — the length of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$) — the elements of the array a.

It is guaranteed that the sum of n over all test cases does not exceed 800.

Output

For each test case, output a single integer — the maximum number of times that you can perform the operation.

Standard Input	Standard Output
6 5 1 5 3 2 4 8 2 1 3 4 5 6 7 8 3 1 2 3 4 1 2 4 4 5 4 4 1 3 5 1 1	2 3 1 2 0 0

Note

In the first test case, one possible optimal sequence of operations is $[1, 5, 3, 2, 4] \rightarrow [1, 5, 4] \rightarrow [4]$.

In the third test case, one possible optimal sequence of operations is $[1, 2, 3] \to [1]$.

,

G1. Spinning Round (Easy Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     7 seconds
Memory limit:   1024 megabytes
```

This is the easy version of the problem. The only difference between the two versions are the allowed characters in s. In the easy version, s only contains the character ?. You can make hacks only if both versions of the problem are solved.

You are given a permutation p of length n. You are also given a string s of length n, consisting only of the character ?.

For each i from 1 to n:

- Define l_i as the largest index $j < i$ such that $p_j > p_i$. If there is no such index, $l_i := i$.
- Define r_i as the smallest index $j > i$ such that $p_j > p_i$. If there is no such index, $r_i := i$.

Initially, you have an undirected graph with n vertices (numbered from 1 to n) and no edges. Then, for each i from 1 to n, add one edge to the graph:

- If $s_i =$ L, add the edge (i, l_i) to the graph.
- If $s_i =$ R, add the edge (i, r_i) to the graph.
- If $s_i =$?, either add the edge (i, l_i) or the edge (i, r_i) to the graph at your choice.

Find the maximum possible diameter* over all **connected** graphs that you can form. Output -1 if it is not possible to form any connected graphs.

---

* Let $d(s, t)$ denote the smallest number of edges on any path between s and t.

The diameter of the graph is defined as the maximum value of $d(s, t)$ over all pairs of vertices s and t.

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($2 \le n \le 4 \cdot 10^5$) — the length of the permutation p.

The second line of each test case contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$) — the elements of p, which are guaranteed to form a permutation.

The third line of each test case contains a string s of length n. It is guaranteed that it consists only of the character ?.

It is guaranteed that the sum of n over all test cases does not exceed $4 \cdot 10^5$.

Output

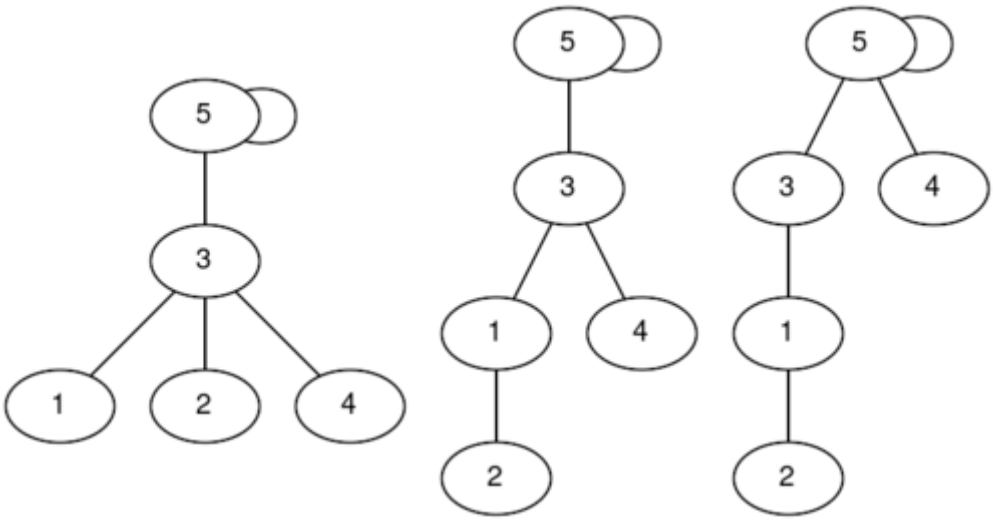
For each test case, output the maximum possible diameter over all connected graphs that you form, or -1 if it is not possible to form any connected graphs.

Standard Input	Standard Output

8 5 2 1 4 3 5 ????? 2 1 2 ?? 3 3 1 2 ??? 7 5 3 1 6 4 2 7 ??????? 5 5 2 1 3 4 ????? 6 6 2 3 4 5 1 ?????? 8 1 7 5 6 2 8 4 3 ???????? 12 6 10 7 1 8 5 12 2 11 3 4 9 ????????????	4 1 2 6 4 5 5 8

Note

In the first test case, here are some possible connected graphs that you can form (the labels are indices):



In the second test case, the only connected graph has a diameter of 1.

,

G2. Spinning Round (Hard Version)

Input file: standard input
Output file: standard output
Time limit: 7 seconds
Memory limit: 1024 megabytes

This is the hard version of the problem. The only difference between the two versions are the allowed characters in s. You can make hacks only if both versions of the problem are solved.

You are given a permutation p of length n. You are also given a string s of length n, where each character is either L, R, or ?.

For each i from 1 to n:

- Define l_i as the largest index $j < i$ such that $p_j > p_i$. If there is no such index, $l_i := i$.
- Define r_i as the smallest index $j > i$ such that $p_j > p_i$. If there is no such index, $r_i := i$.

Initially, you have an undirected graph with n vertices (numbered from 1 to n) and no edges. Then, for each i from 1 to n, add one edge to the graph:

- If $s_i =$ L, add the edge (i, l_i) to the graph.
- If $s_i =$ R, add the edge (i, r_i) to the graph.
- If $s_i =$?, either add the edge (i, l_i) or the edge (i, r_i) to the graph at your choice.

Find the maximum possible diameter over all **connected*** graphs that you can form. Output -1 if it is not possible to form any connected graphs.

---

* Let $d(s, t)$ denote the smallest number of edges on any path between s and t.

The diameter of the graph is defined as the maximum value of $d(s, t)$ over all pairs of vertices s and t.

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($2 \le n \le 4 \cdot 10^5$) — the length of the permutation p.

The second line of each test case contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$) — the elements of p, which are guaranteed to form a permutation.

The third line of each test case contains a string s of length n. It is guaranteed that it consists only of the characters L, R, and ?.

It is guaranteed that the sum of n over all test cases does not exceed $4 \cdot 10^5$.

Output

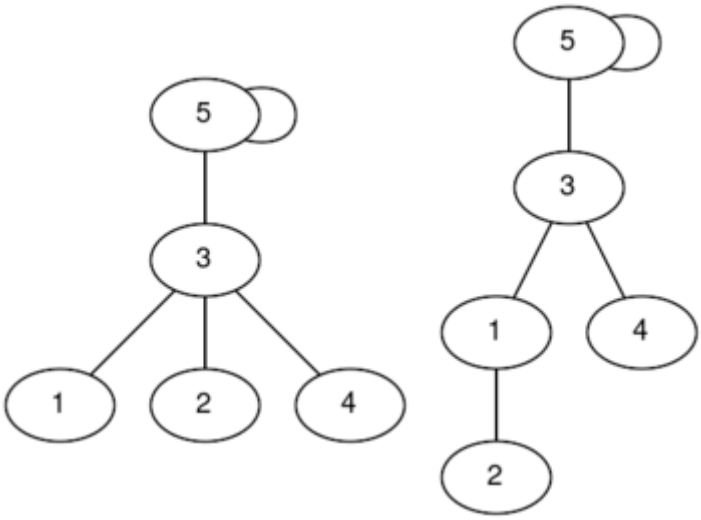
For each test case, output the maximum possible diameter over all connected graphs that you form, or -1 if it is not possible to form any connected graphs.

Standard Input	Standard Output
8 5	3 -1

2 1 4 3 5 R?RL? 2 1 2 LR 3 3 1 2 L?R 7 5 3 1 6 4 2 7 ?R?R?R? 5 5 2 1 3 4 ????? 6 6 2 3 4 5 1 ?LLRLL 8 1 7 5 6 2 8 4 3 ?R?????? 12 6 10 7 1 8 5 12 2 11 3 4 9 ????????????	-1 4 4 3 5 8

Note

In the first test case, there are two connected graphs (the labels are indices):



The graph on the left has a diameter of 2, while the graph on the right has a diameter of 3, so the answer is 3.

In the second test case, there are no connected graphs, so the answer is -1.

,

H. Fumo Temple

Input file:	standard input
Output file:	standard output
Time limit:	4 seconds
Memory limit:	512 megabytes

This temple only magnifies the mountain's power.

Badeline

This is an interactive problem.

You are given two positive integers n and m ($\mathbf{n \le m}$).

The jury has hidden from you a rectangular matrix a with n rows and m columns, where $a_{i,j} \in \{-1, 0, 1\}$ for all $1 \le i \le n$ and $1 \le j \le m$. The jury has also selected a cell (i_0, j_0). Your goal is to find (i_0, j_0).

In one query, you give a cell (i, j), then the jury will reply with an integer.

- If $(i, j) = (i_0, j_0)$, the jury will reply with 0.
- Else, let S be the sum of $a_{x,y}$ over all x and y such that $\min(i, i_0) \le x \le \max(i, i_0)$ and $\min(j, j_0) \le y \le \max(j, j_0)$. Then, the jury will reply with $|i - i_0| + |j - j_0| + |S|$.

Find (i_0, j_0) by making at most $n + 225$ queries.

Note: the grader is not adaptive: a and (i_0, j_0) are fixed before any queries are made.

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \le t \le 50$) — the number of test cases. The description of the test cases follows.

The only line of each test case contains two integers n and m ($1 \le n \le m \le 5000$) — the numbers of rows and the number of columns of the hidden matrix a respectively.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed $25 \cdot 10^6$.

Interaction

The interaction for each test case begins by reading the integers n and m.

To make a query, output `"? i j"` ($1 \le i \le n, 1 \le j \le m$) without quotes. Afterwards, you should read one single integer — the answer to your query.

If you receive the integer -1 instead of an answer or a valid value of n or m, it means your program has made an invalid query, has exceeded the limit of queries, or has given an incorrect answer on the previous test case. Your program must terminate immediately to receive a Wrong Answer verdict. Otherwise, you can get an arbitrary verdict because your solution will continue to read from a closed stream.

When you are ready to give the final answer, output `"! i j"` ($1 \le i \le n, 1 \le j \le m$) without quotes — the indices of the hidden cell. After solving a test case, your program should move to the next one immediately. After solving all test cases, your program should be terminated immediately.

After printing a query do not forget to output end of line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

Hacks

To hack, use the following format:

The first line contains an integer t ($1 \le t \le 50$) — the number of test cases.

The first line of each test case contains two integers n and m ($1 \le n \le m \le 5000$) — the sizes of the hidden matrix.

The second line of each test case contains two integers i_0 and j_0 ($1 \le i_0 \le n, 1 \le j_0 \le m$) — the hidden cell.

Then n lines follow. The i-th of them contains the string s_i of length n, consisting only of the characters `-`, `0`, and `+`. Here, $a_{ij} = -1$ if $s_{ij} = $ `-`, $a_{ij} = 0$ if $s_{ij} = $ `0`, and $a_{ij} = 1$ if $s_{ij} = $ `+`.

The sum of $n \cdot m$ over all test cases should not exceed $25 \cdot 10^6$.

As an example, the hack format for the example input is:

```
2
3 4
1 4
+0+0
+00+
0---
1 1
1 1
0
```

Standard Input	Standard Output
2 3 4 5 3 5 1 1 0	 ? 1 1 ? 3 3 ? 3 2 ! 1 4 ? 1 1 ! 1 1

Note

The hidden matrix in the first test case:

1	0	1	**0**
1	0	0	1
0	−1	−1	−1

The hidden matrix in the second test case:

0

Note that the line breaks in the example input and output are for the sake of clarity, and do not occur in the real interaction.