A. The Beatles

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Recently a Golden Circle of Beetlovers was found in Byteland. It is a circle route going through $n \cdot k$ cities. The cities are numerated from 1 to $n \cdot k$, the distance between the neighboring cities is exactly 1 km.

Sergey does not like beetles, he loves burgers. Fortunately for him, there are n fast food restaurants on the circle, they are located in the 1-st, the $(k+1)$-st, the $(2k+1)$-st, and so on, the $((n-1)k+1)$-st cities, i.e. the distance between the neighboring cities with fast food restaurants is k km.

Sergey began his journey at some city s and traveled along the circle, making stops at cities each l km ($l > 0$), until he stopped in s once again. Sergey then forgot numbers s and l, but he remembers that the distance from the city s to the nearest fast food restaurant was a km, and the distance from the city he stopped at after traveling the first l km from s to the nearest fast food restaurant was b km. Sergey always traveled in the same direction along the circle, but when he calculated distances to the restaurants, he considered both directions.

Now Sergey is interested in two integers. The first integer x is the minimum number of stops (excluding the first) Sergey could have done before returning to s. The second integer y is the maximum number of stops (excluding the first) Sergey could have done before returning to s.

Input

The first line contains two integers n and k ($1 \le n, k \le 100\,000$) — the number of fast food restaurants on the circle and the distance between the neighboring restaurants, respectively.

The second line contains two integers a and b ($0 \le a, b \le \frac{k}{2}$) — the distances to the nearest fast food restaurants from the initial city and from the city Sergey made the first stop at, respectively.

Output

Print the two integers x and y.

Examples

input

```
2 3
1 1
```

output

```
1 6
```

input

```
3 2
0 0
```

output

```
1 3
```

input

```
1 10
5 3
```

output

```
5 5
```

Note

In the first example the restaurants are located in the cities 1 and 4, the initial city s could be 2, 3, 5, or 6. The next city Sergey stopped at could also be at cities $2, 3, 5, 6$. Let's loop through all possible combinations of these cities. If both s and the city of the first stop are at the city 2 (for example, $l = 6$), then Sergey is at s after the first stop already, so $x = 1$. In other pairs Sergey needs $1, 2, 3$, or 6 stops to return to s, so $y = 6$.

In the second example Sergey was at cities with fast food restaurant both initially and after the first stop, so l is 2, 4, or 6. Thus $x = 1$, $y = 3$.

In the third example there is only one restaurant, so the possible locations of s and the first stop are: $(6, 8)$ and $(6, 4)$. For the first option $l = 2$, for the second $l = 8$. In both cases Sergey needs $x = y = 5$ stops to go to s.

B. Lynyrd Skynyrd

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Recently Lynyrd and Skynyrd went to a shop where Lynyrd bought a permutation p of length n, and Skynyrd bought an array a of length m, consisting of integers from 1 to n.

Lynyrd and Skynyrd became bored, so they asked you q queries, each of which has the following form: "does the subsegment of a from the l-th to the r-th positions, inclusive, have a subsequence that is a cyclic shift of p?" Please answer the queries.

A *permutation* of length n is a sequence of n integers such that each integer from 1 to n appears exactly once in it.

A *cyclic shift* of a permutation $(p_1, p_2, \ldots, p_n)$ is a permutation $(p_i, p_{i+1}, \ldots, p_n, p_1, p_2, \ldots, p_{i-1})$ for some i from 1 to n. For example, a permutation $(2, 1, 3)$ has three distinct cyclic shifts: $(2, 1, 3)$, $(1, 3, 2)$, $(3, 2, 1)$.

A *subsequence* of a subsegment of array a from the l-th to the r-th positions, inclusive, is a sequence $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ for some $i_1, i_2, \ldots, i_k$ such that $l \leq i_1 < i_2 < \ldots < i_k \leq r$.

Input

The first line contains three integers n, m, q ($1 \leq n, m, q \leq 2 \cdot 10^5$) — the length of the permutation p, the length of the array a and the number of queries.

The next line contains n integers from 1 to n, where the i-th of them is the i-th element of the permutation. Each integer from 1 to n appears exactly once.

The next line contains m integers from 1 to n, the i-th of them is the i-th element of the array a.

The next q lines describe queries. The i-th of these lines contains two integers l_i and r_i ($1 \leq l_i \leq r_i \leq m$), meaning that the i-th query is about the subsegment of the array from the l_i-th to the r_i-th positions, inclusive.

Output

Print a single string of length q, consisting of 0 and 1, the digit on the i-th positions should be 1, if the subsegment of array a from the l_i-th to the r_i-th positions, inclusive, contains a subsequence that is a cyclic shift of p, and 0 otherwise.

Examples

input

```
3 6 3
2 1 3
1 2 3 1 2 3
1 5
2 6
3 5
```

output

```
110
```

input

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

output

```
010
```

Note

In the first example the segment from the 1-st to the 5-th positions is $1, 2, 3, 1, 2$. There is a subsequence $1, 3, 2$ that is a cyclic shift of the permutation. The subsegment from the 2-nd to the 6-th positions also contains a subsequence $2, 1, 3$ that is equal to the permutation. The subsegment from the 3-rd to the 5-th positions is $3, 1, 2$, there is only one subsequence of length 3 ($3, 1, 2$), but it is not a cyclic shift of the permutation.

In the second example the possible cyclic shifts are $1, 2$ and $2, 1$. The subsegment from the 1-st to the 2-nd positions is $1, 1$, its subsequences are not cyclic shifts of the permutation. The subsegment from the 2-nd to the 3-rd positions is $1, 2$, it coincides with the permutation. The subsegment from the 3 to the 4 positions is $2, 2$, its subsequences are not cyclic shifts of the permutation.

C. U2

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Recently Vasya learned that, given two points with different x coordinates, you can draw through them exactly one parabola with equation of type $y = x^2 + bx + c$, where b and c are reals. Let's call such a parabola an U-shaped one.

Vasya drew several distinct points with integer coordinates on a plane and then drew an U-shaped parabola through each pair of the points that have different x coordinates. The picture became somewhat messy, but Vasya still wants to count how many of the parabolas drawn don't have any drawn point inside their internal area. Help Vasya.

The internal area of an U-shaped parabola is the part of the plane that lies strictly above the parabola when the y axis is directed upwards.

Input

The first line contains a single integer n ($1 \le n \le 100\,000$) — the number of points.

The next n lines describe the points, the i-th of them contains two integers x_i and y_i — the coordinates of the i-th point. It is guaranteed that all points are distinct and that the coordinates do not exceed 10^6 by absolute value.

Output

In the only line print a single integer — the number of U-shaped parabolas that pass through at least two of the given points and do not contain any of the given points inside their internal area (excluding the parabola itself).

Examples

input

```
3
-1 0
0 2
1 0
```

output

```
2
```

input

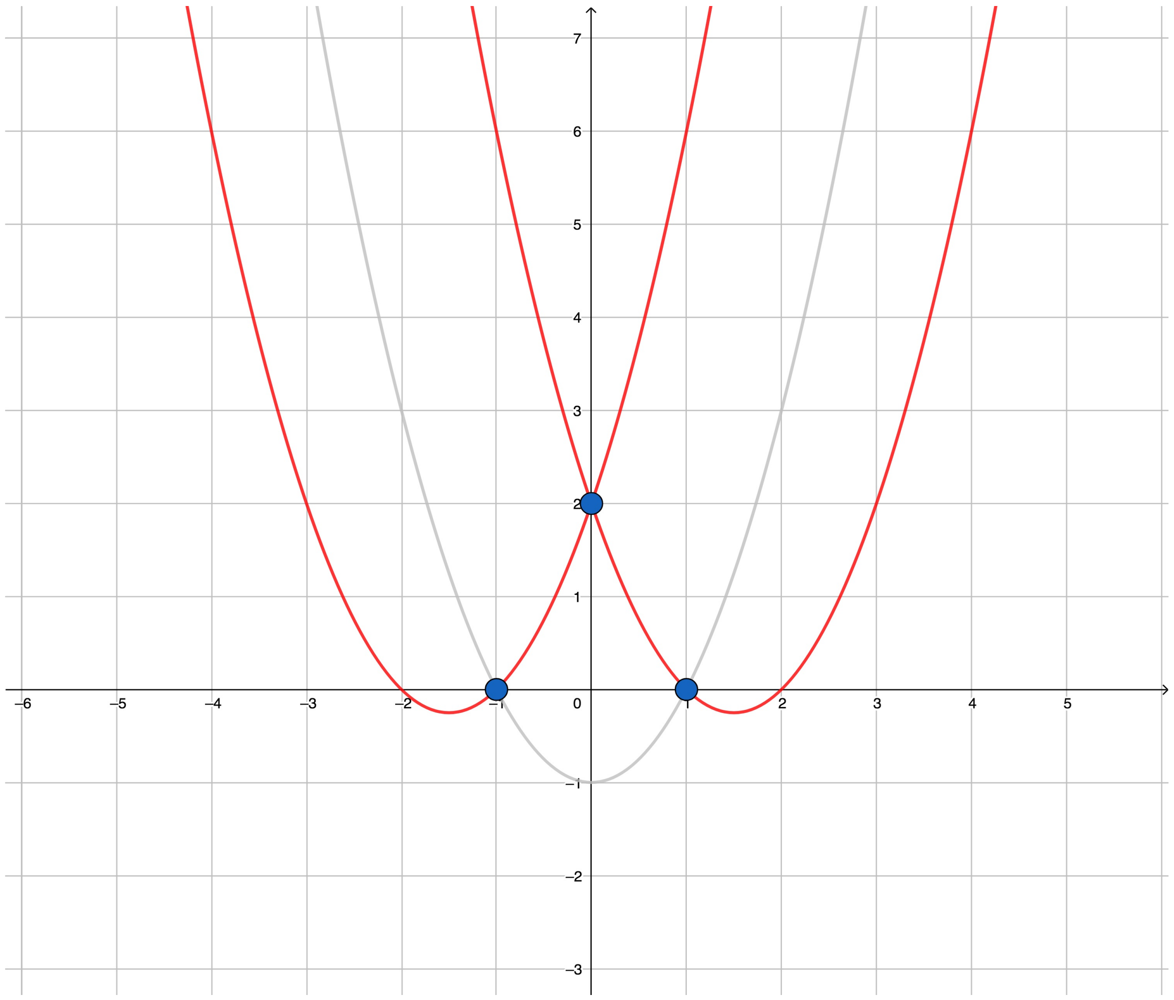
```
5
1 0
1 -1
0 -1
-1 0
-1 -1
```

output

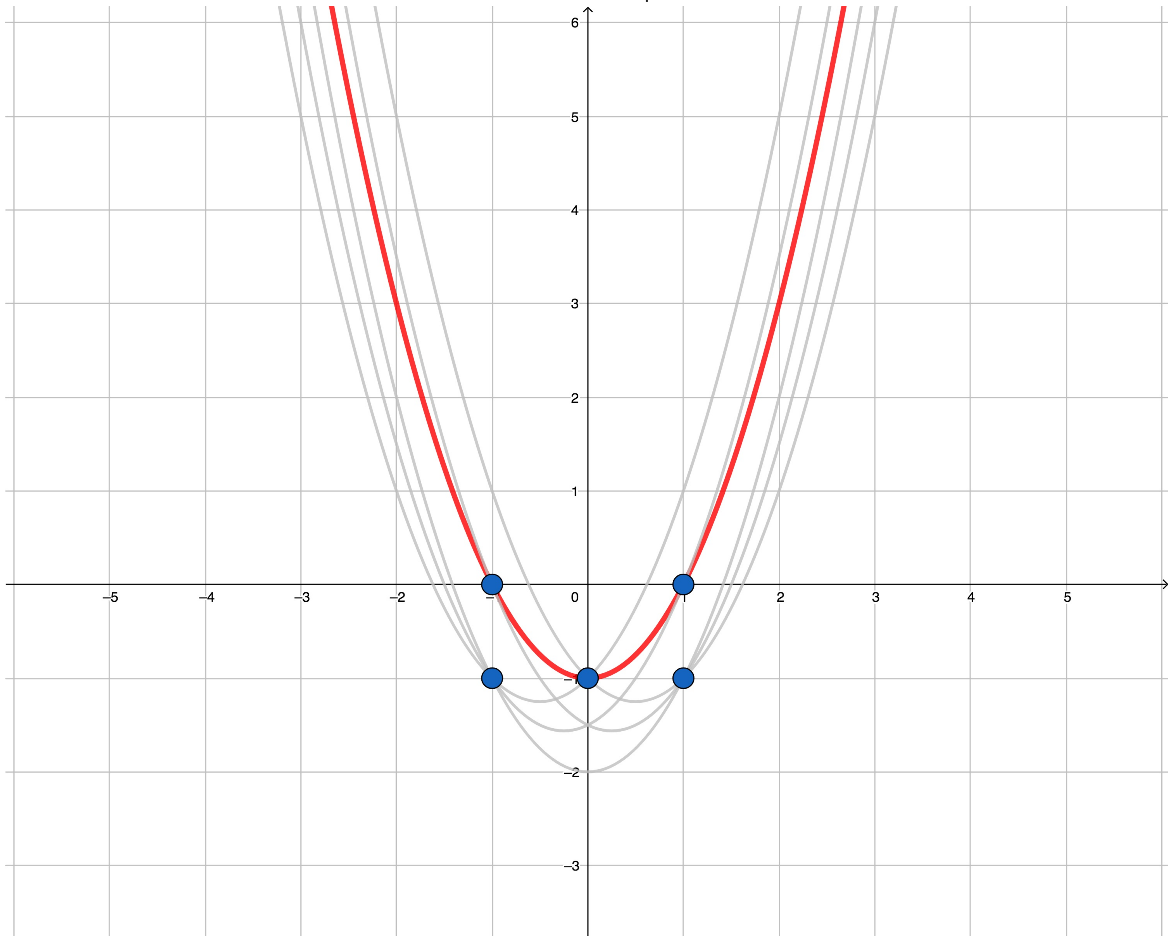
```
1
```

Note

On the pictures below all U-shaped parabolas that pass through at least two given points are drawn for each of the examples. The U-shaped parabolas that do not have any given point inside their internal area are drawn in red.



The first example.



The second example.

D. Foreigner

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input

output: standard output

Traveling around the world you noticed that many shop owners raise prices to inadequate values if the see you are a foreigner.

You define inadequate numbers as follows:

- all integers from 1 to 9 are inadequate;
- for an integer $x \geq 10$ to be inadequate, it is required that the integer $\lfloor x/10 \rfloor$ is inadequate, but that's not the only condition. Let's sort all the inadequate integers. Let $\lfloor x/10 \rfloor$ have number k in this order. Then, the integer x is inadequate only if the last digit of x is strictly less than the reminder of division of k by 11.

Here $\lfloor x/10 \rfloor$ denotes $x/10$ rounded down.

Thus, if x is the m-th in increasing order inadequate number, and m gives the remainder c when divided by 11, then integers $10 \cdot x + 0, 10 \cdot x + 1 \ldots, 10 \cdot x + (c-1)$ are inadequate, while integers $10 \cdot x + c, 10 \cdot x + (c+1), \ldots, 10 \cdot x + 9$ are not inadequate.

The first several inadequate integers are $1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 21, 30, 31, 32 \ldots$. After that, since 4 is the fourth inadequate integer, $40, 41, 42, 43$ are inadequate, while $44, 45, 46, \ldots, 49$ are not inadequate; since 10 is the 10-th inadequate number, integers $100, 101, 102, \ldots, 109$ are all inadequate. And since 20 is the 11-th inadequate number, none of $200, 201, 202, \ldots, 209$ is inadequate.

You wrote down all the prices you have seen in a trip. Unfortunately, all integers got concatenated in one large digit string s and you lost the bounds between the neighboring integers. You are now interested in the number of substrings of the resulting string that form an inadequate number. If a substring appears more than once at different positions, all its appearances are counted separately.

Input

The only line contains the string s ($1 \leq |s| \leq 10^5$), consisting only of digits. It is guaranteed that the first digit of s is not zero.

Output

In the only line print the number of substrings of s that form an inadequate number.

Examples

input
4021
output
6

input
110
output
3

Note

In the first example the inadequate numbers in the string are $1, 2, 4, 21, 40, 402$.

In the second example the inadequate numbers in the string are 1 and 10, and 1 appears twice (on the first and on the second positions).

E. Pink Floyd

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is an interactive task.

Scientists are about to invent a new optimization for the Floyd-Warshall algorithm, which will allow it to work in linear time. There is only one part of the optimization still unfinished.

It is well known that the Floyd-Warshall algorithm takes a graph with n nodes and exactly one edge between each pair of nodes. The scientists have this graph, what is more, they have directed each edge in one of the two possible directions.

To optimize the algorithm, exactly m edges are colored in pink color and all the rest are colored in green. You know the direction of all m pink edges, but the direction of green edges is unknown to you. In one query you can ask the scientists about the direction of exactly one green edge, however, you can perform at most $2 \cdot n$ such queries.

Your task is to find the node from which every other node can be reached by a path consisting of edges of same color. Be aware that the scientists may have lied that they had fixed the direction of all edges beforehand, so their answers may depend on your queries.

Input

The first line contains two integers n and m ($1 \leq n \leq 100\,000$, $0 \leq m \leq 100\,000$) — the number of nodes and the number of pink edges.

The next m lines describe the pink edges, the i-th of these lines contains two integers u_i, v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$) — the start and the end of the i-th pink edge. It is guaranteed, that all unordered pairs (u_i, v_i) are distinct.

Output

When you found the answer, print "!" and the number of the node from which every other node can be reached by a single-colored path.

Interaction

To ask the direction of the green edge between nodes a and b, print "?", a and b in single line, separated by the space characters.

In answer to this read a single integer, which will be 1 if the edge is directed from a to b and 0 if the edge is directed from b to a.

You can ask at most $2 \cdot n$ queries, otherwise you will get `Wrong Answer`.

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.

Answer -1 instead of 0 or 1 means that you made an invalid query or exceeded the query limit. Exit immediately after receiving -1 and you will see `Wrong answer` verdict. Otherwise you can get an arbitrary verdict because your solution will continue to read from a closed stream.

Hacks

Hacks should be formatted as follows.

The first line should contain two integers n and m ($1 \le n \le 300$, $0 \le m \le n \cdot (n-1)/2$) — the number of nodes and number of pink edges.

The next m lines should describe the pink edges, the i-th line should contain 2 integers a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$), which means that there is a pink edge from a_i to b_i. All unordered pairs (a_i, b_i) should be distinct.

The next $(n \cdot (n-1)/2 - m)$ lines should describe the green edges, the i-th line should contain two integers a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$)), which means that there is a green edge from a_i to b_i. All unordered pairs of (a_i, b_i) should be distinct and should also be different from the pairs for the pink edges.

Example

input

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

output

```
? 1 3
? 4 2
? 3 2
! 3
```

Note

In the example above the answer for the query "? 1 3" is 0, so the edge is directed from 3 to 1. The answer for the query "? 4 2" is 1, so the edge is directed from 4 to 2. The answer for the query "? 3 2" is 1, so the edge is directed from 3 to 2. So there are green paths from node 3 to nodes 1 and 2 and there is a pink path from node 3 to node 4.