Problem A. Square Function

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

Let us define function S: $\mathbb{N} \to \mathbb{N}$ in the following way: $S(x)$ is the minimum number for which there exists an increasing sequence of integers $x = t_1 < t_2 < \ldots < t_k = S(x)$ such that $t_1 \cdot t_2 \cdot \ldots \cdot t_k$ is a square of some integer. For example, $S(2) = 6$, $S(3) = 8$, $S(4) = 4$.

Given y, find all such x that $S(x) = y$.

Input

The only line of input contains a single integer y ($1 \le y \le 10^6$).

Output

On the first line, print the number of solutions. On the second line, list all solutions in increasing order separated by spaces.

Examples

standard input	standard output
4	1 4
5	0
6	1 2

Note

$\mathbb{N}$ is the set of positive integers.

Problem B. Guess by Remainder

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	512 mebibytes

This is an interactive problem.

We have chosen an integer m between 1 and n. Your task is to guess it, and you have to do no more queries than necessary for this n. Each of your queries must be an integer which has no more than n digits in its decimal notation. The answer to query x is the remainder $x \bmod m$.

Input

In the beginning, your program will receive one integer n ($1 \le n \le 10^6$).

Then your program will receive the answers to the queries. Each answer is an integer.

Output

Your program can make queries in the form "`?` *number*". When you think you know the answer, you should print "`!` *guess*", and then terminate your program immediately. Don't forget to output the line break and flush the output. To do so, you can use the following instructions:

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

Examples

standard input	standard output
2 1	? 79 ! 2
3 0 0 0	? 42 ? 777 ? 8 ! 1

Note

You are not prohibited to output leading zeroes, but the checking program counts them when determining the length of the number. For example, if $n = 3$, query "001" is valid, but "0001" is invalid.

The second sample just demonstrates the interaction format, guessing can be done in smaller number of queries.

Problem C. Subtract if Greater!

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	512 mebibytes

Consider a multiset A consisting of n elements: $a_1, a_2, \ldots, a_n$.

Let us define two types of operations which can be performed on this multiset:

1. Given x_i, you have to print the number which will be x_i-th element if we sort the multiset in non-decreasing order.
2. Given x_i, you have to subtract x_i from all the elements of A which are strictly greater than x_i.

Your task is to perform q given operations in the given order and output the results of all operations of the first type.

Input

The first line contains two integers n and q: the size of the multiset A and the number of queries ($1 \le n \le 10^5$, $1 \le q \le 10^6$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$): the elements of A.

Each of the next q lines describes a single operation. An operation is given as two integers t_i and x_i: the type and the parameter of the operation. It is guaranteed that $t_i \in \{1, 2\}$. If $t_i = 1$, then $1 \le x_i \le n$. If $t_i = 2$, then $1 \le x_i \le 10^9$.

It is guaranteed that there is at least one operation of type 1.

Please note that elements of A are given in **arbitrary** order.

Output

For each operation of the first type, output the x_i-th element in non-decreasing order. Separate the answers with line breaks.

Examples

standard input	standard output
4 5 1 5 6 12 2 5 1 1 1 2 1 3 1 4	1 1 5 7
5 4 1 10 5 4 2 2 1 1 5 2 3 1 2	9 1
3 2 3 2 1 2 10000 1 3	3

Problem D. Eastern Subregional

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

This is an interactive problem.

It was 2033. Rikhail Mubinchik understood that there are still too many institutions which hadn't heard about ACM in the Ural region, and decided to fix it. Again. It was an established system by that time and Rikhail already knew what to do.

"We are happy to announce elimination round of 1/16 of qualifying round of Eastern Subregional of NEERC!", said the post on Codeforces. The same line was on the posters in all the universities, schools and kindergartens across the Ural region. Advertising campaign was so extensive that it affected even fictional universities.

Now then, you represent the first (and the only) team of Unseen University in competitive programming. Strictly speaking, there were no computers in Discworld before. Your experience in problem solving is approximately 0.

The good news: the best team from each university advances to the next round. The bad news: a team has to solve at least one problem to advance.

So, 20 minutes before competition ends you decided to concentrate on a problem with maximal success rate. The statement tells about some dandelions which are participating in contests as well. It is strange because dandelions can't think. You thought that it is a fiction and you shouldn't be distracted by it. The question was "Will dandelions advance to the main round of qualification if they make x rejected submits before the accepted one?"

You have read the rules very carefully, so you figured out that the less x is, the better for dandelions. So there exists a value B such that the answer is negative if and only if $B \le x$.

You have written the following code:

```
const int B = ???;
int x;
cin >> x;
if (B <= x)
    cout << "NO" << endl;
else
    cout << "YES" << endl;
```

Now it remains only to choose the right value for the constant B. From the samples you know that if $x = 0$, the answer is "`YES`", and if $x = R$, the answer is "`NO`". But you can't tell anything else.

Now you have only 10 minutes, and the Timus Online Judge allows you to submit no faster than once in 10 seconds. Therefore you can make no more than 60 attempts!

Input

In the beginning, your program receives one integer R ($1 \le R \le 5 \cdot 10^5$).

Then your program will receive "judgement results". Each verdict will be either "`AC`" or "`WA` X" where X is the index of the first test "your solution" didn't pass.

It is guaranteed that "the problem" has no more than 10^6 tests. The first test is 0 and the second test is R (the first two tests are samples).

Output

Your program can "submit solutons" with different values of B. To do so, it should output this value. The

value should be non-negative integer not greater than R. The amount of submitted solutions shouldn't be more than 60, and the last submitted solution should get the verdict "AC". After getting the verdict "AC", your program should terminate immediately.

Don't forget to output a line break and flush the output after each query. To do so, you can use the following instructions:

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

Example

standard input	standard output
5 WA 1 WA 4 WA 4 WA 3 WA 6 AC	0 1 2 5 4 3

Note

In the sample, the correct value of constant B is 3. The tests in "the problem" are: 0, 5, 4, 2, 1, 3.

Problem E. K-transform

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

Let us fix an integer $k \geq 2$ and define a function f: $\mathbb{N} \to \mathbb{N}$:

$$f(n) = \begin{cases} n/k, \text{ if } k \mid n \\ n-1, \text{ otherwise} \end{cases}$$

If we take some integer $n \geq 1$ and will apply function f some (possibly 0) times then we will end up with 1. For example, if $k = 3$ then $f(f(f(f(f(16))))) = f(f(f(f(15)))) = f(f(f(5))) = f(f(4)) = f(3) = 1$.

Your task is to calculate the amount of such n that we will end up with 1 after exactly m iterations. The answer may be very large, so you have to output it modulo *mod*.

Input

The first line contains three integers separated by spaces: k, m, *mod* ($2 \leq k \leq 10^4$, $0 \leq m \leq 10^{18}$, $2 \leq mod < 300$). It is guaranteed that *mod* is prime.

Output

Print one integer: the answer to the problem modulo *mod*.

Examples

standard input	standard output
2 4 31	5
3 13 59	36

Note

$\mathbb{N}$ is the set of positive integers.

Problem F. Suffix Array for Thue-Morse

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

A Thue-Morse string of order k is a string of length 2^k in which i-th symbol equals to '`A`' if the number of 1-bits in binary representation of $i - 1$ is even, and '`B`' if it is odd.

A suffix array for string s of length n is a permutation suf of integers from 1 to n such that $s[suf[1]..n]$, $s[suf[2]..n]$, ..., $s[suf[n]..n]$ is the list of non-empty suffixes of s sorted in lexicographical order.

Let suf be the suffix array for Thue-Morse string of order k. You task is to calculate q values: $suf[p_1]$, $suf[p_2]$, ..., $suf[p_q]$.

Input

The first line of input contains two integers k and q: the order of Thue-Morse string and the number of queries ($0 \le k \le 60$, $1 \le q \le 10^5$).

The second line contains q integers p_1, p_2, ..., p_q separated by spaces: the required indices ($1 \le p_i \le 2^k$).

Output

Output q answers to the queries, separated by spaces.

Example

standard input	standard output
3 8 1 2 3 4 5 6 7 8	6 7 4 1 8 5 3 2

Note

Thue-Morse string of order 3 is "`ABBABAAB`".

Problem G. XOR Tree

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

There is an undirected graph G with n vertices. It has an interesting property: **if we treat multiple edges as one,** the graph G will become a tree. Alice and Bob invented the following game:

- At the beginning, Alice and Bob choose two vertices S and T, $S \neq T$.
- There is another graph H with n vertices and, initially, no edges.
- On every turn, the current player chooses any edge $u - v$ in graph G, deletes it and changes the state of edge $u - v$ in graph H: if there was no such edge in graph H, it appears, and if there was such an edge, it disappears.
- Alice wins if there exists a moment of time when there is a path between S and T in graph H.
- If there is no edge in graph G and Alice hasn't won yet, Bob wins.
- Players take turns, Alice moves first.

Children like the game very much so they have played it q times. Now they are wondering who would win in each game if they were playing optimally. Help them find it out!

Input

The first line of input contains two integers n and q: the number of vertices in graph G and the number of games played by Alice and Bob ($2 \le n \le 10^5$, $1 \le q \le 10^5$).

The next $n - 1$ lines contain the description of graph G. Each line consists of three integers a, b and c ($1 \le a, b \le n$, $1 \le c \le 10^9$) which means there are exactly c edges between vertices a and b. It is guaranteed that, if we change all c to 1, the graph G will become a tree with n vertices.

The next q lines describe the games. Each of these lines contains two integers S and T: the parameters of the game ($1 \le S, T \le n$, $S \neq T$).

Output

For each game, print `1` if Alice wins, and print `2` otherwise. Separate the answers with line breaks.

Example

standard input	standard output
6 5 1 2 3 1 3 2 2 4 1 2 5 2 3 6 2 1 4 5 4 5 6 2 6 4 6	1 1 2 2 2

Problem H. Fence

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Consider a rectangular $n \times m$ grid consisting of symbols '.' and '*'. A *house* is a connected component of '*' cells. Two cells are connected if they share a common **side**. A *fence* is a closed polyline without self-touchings and self-intersections which consists of segments corresponding to cell sides.

You have to build a fence with the following three properties:

1. The fence must not have common points with grid borders.
2. The fence must not have common points with houses which are **outside** the fence.
3. Each segment of the fence must be outside any house, in other words, there must be '.' on at least one side of each segment.

Calculate the maximum **number** of houses which can be enclosed by the described fence. If the fence cannot be build, this number has to be 0.

Input

The first line contains two integers n and m, the dimensions of the grid ($1 \le n, m \le 10^3$).

Each of the next n lines is a string of length m consisting of characters '.' and '*'. Together, these lines describe the grid.

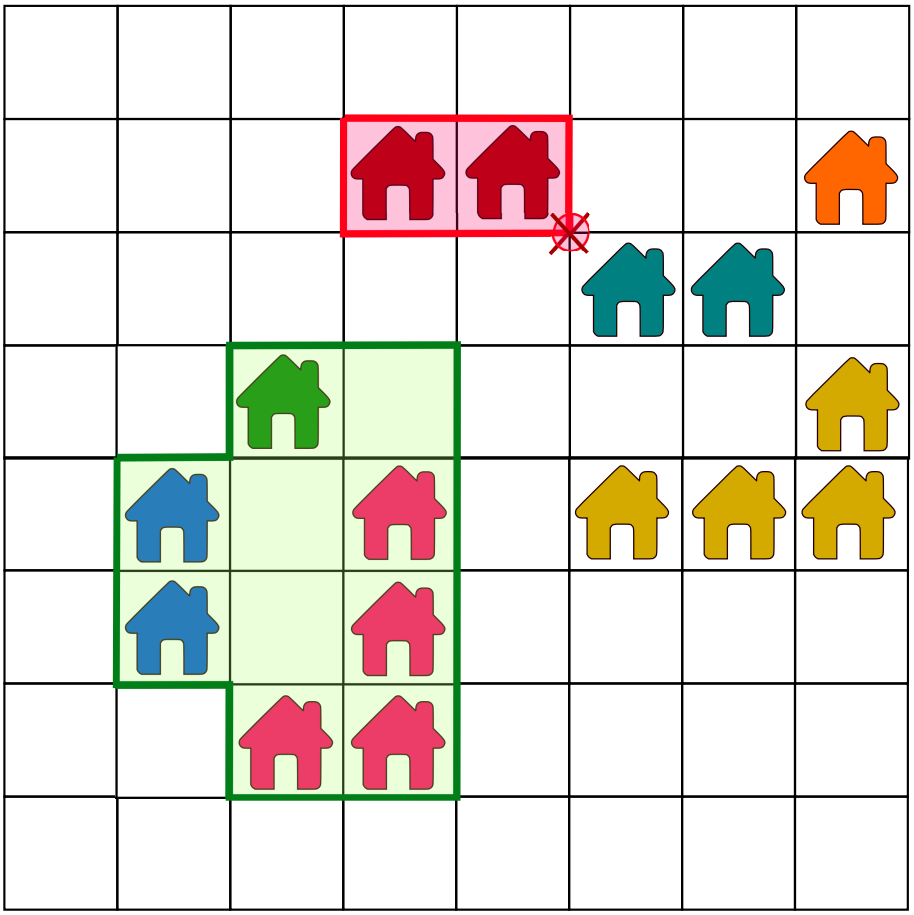
Output

Print one number: the maximum number of houses which can be enclosed by a fence with the three required properties.

Examples

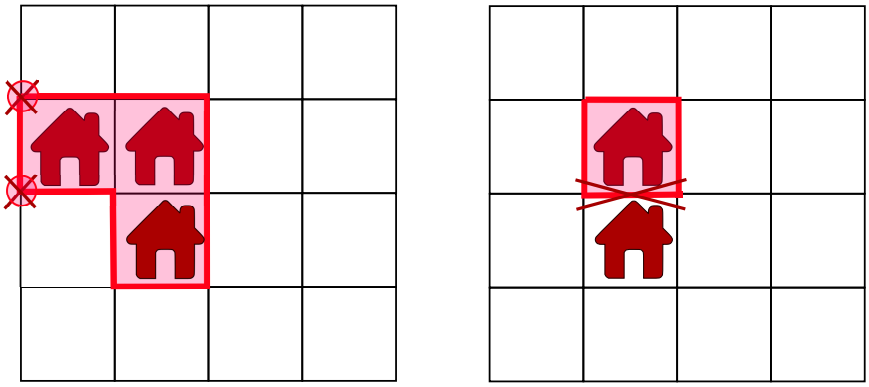
standard input	standard output
8 8 ....**..* .....**. ..*....* .*.*.*** .*.*.... ..**....	3
3 4 .*.. **** ..*.	0

Note



The green polyline near the center which encloses three houses is one of the ways to build a fence optimally.

The red polyline near the top is a bad fence because it touches a house which is outside the fence.



These two pictures show invalid ways to construct a fence. The left picture shows the case where the fence touches the outside boundary. On the right picture, the fence goes through a house, which is unacceptable.

Problem I. Friends and Berries - 2

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

There is a group of n children. According to a proverb, every man to his own taste. So the children value strawberries and raspberries differently. Let us say that i-th child rates his attachment to strawberry as s_i and his attachment to raspberry as r_i.

According to another proverb, opposites attract. Surprisingly, those children become friends whose tastes differ.

Let us define friendliness between two children v and u as

$$p(u, v) = (s_u - s_v)^2 + (r_u - r_v)^2.$$

The friendliness between three children v, u, w is half the sum of pairwise friendlinesses:

$$p(u, v, w) = \frac{p(u, v) + p(u, w) + p(v, w)}{2}.$$

The best friends are such pairs of children (u, v) that $u \neq v$ and $p(u, v) \geq p(u, v, w)$ for every w. Your goal is to find all pairs of best friends.

Input

In the first line there is one integer n, the number of children ($2 \leq n \leq 2 \cdot 10^5$).

Each of the next n lines contains two integers s_i and r_i ($-10^8 \leq s_i, r_i \leq 10^8$).

It is guaranteed that, for every two children, their tastes differ. In other words, if $u \neq v$, then $s_u \neq s_v$ or $r_u \neq r_v$.

Output

On the first line, output the number of pairs of best friends.

After that, output those pairs. Each pair should be printed on a separate line. A pair is denoted by two integers: the indices of children in this pair. Children are numbered in the order of input starting from 1. You can output pairs in any order. You can output indices in each pair in any order.

It is guaranteed that the required number of pairs doesn't exceed 10^6.

Examples

standard input	standard output
4 0 0 1 0 0 1 1 1	2 1 4 2 3
3 0 0 0 10 5 8	0

Problem J. Oleg and Cola

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

Oleg is a cool programmer. Oleg lives in Ekaterinozavodsk. One day Oleg was writing code and didn't notice how night has come. As ill luck would have it, Oleg has run out of The Cola. There is only one 24-hour store with The Cola in Ekaterinozavodsk. So Oleg decided to go and buy his favorite Cola.

There are n crossroads and m two-way roads between them in Ekaterinozavodsk. Every road has length and luminosity which are defined by integers. Oleg lives near crossroad number 1, and the store is located at the crossroad number 2.

Oleg has to go from his home to the store and come back home. Oleg thinks that it is dangerous to decrease the luminosity of the roads during the way, so Oleg won't do it. This rule includes the last road before the store and the first road after it. Oleg can start with a road with arbitrary luminosity.

Please help Oleg to find the shortest path which Oleg finds safe. The length of the path is the sum of road lengths for every road the path includes. If Oleg uses one road twice then the length will be counted twice (and so on). As you remember, Oleg is a cool programmer and he likes The Cola very much, so he can always buy The Cola and come back home. In other words, a solution is guaranteed to exist.

Input

The first line contains two integers n and m: the number of crossroads and roads respectively ($2 \le n \le 10^5$, $1 \le m \le 10^5$).

Each of the next m lines describes a single road. Each description is in the format $u\ v\ l\ i$ ($1 \le u, v \le n$, $1 \le l \le 10^9$, $1 \le i \le 10^9$). Here, u and v are two crossroads joined by the road, l is the length of the road, and i is its luminosity.

There can be roads which connect a crossroad to itself. There can be several roads between a pair of crossroads, including roads with different lengths and/or luminosities.

Output

On the first line, print one integer: the length of the path.

On the second line, print the sequence of road numbers in the order Oleg should follow, separated by spaces. Roads are numbered from 1 to m in the input order.

If there are several solutions, print any one of them.

Examples

standard input	standard output
2 1 1 2 3 4	6 1 1
3 5 1 3 1 1 2 3 100 2 1 3 1000 3 2 3 10 4 1 2 10000 5	1201 1 2 2 3
6 10 1 3 5 10 5 1 7 20 1 4 10 10 1 5 9 10 1 1 4 15 4 6 5 50 6 2 7 50 2 5 8 15 3 2 6 15 5 6 3 25	26 1 9 8 2

Problem K. Process with Constant Sum

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2.5 seconds
Memory limit:	512 mebibytes

Suppose we have an array a whose elements are non-negative integers. We can apply operations of two types to this array:

1. $a_i \leftarrow a_i - 2$, $a_{i+1} \leftarrow a_{i+1} + 2$.
2. If $a_{i+1} = 0$: $a_i \leftarrow a_i - 1$, $a_{i+2} \leftarrow a_{i+2} + 1$.

After any operation, all the elements must remain non-negative.

It is easy to see that we cannot do such operations indefinitely. We'll call a chain of operations *maximal* if no valid operation can occur after performing this chain of operations.

Now you are given an array b of length n. Your should handle two types of queries:

- `1` p x — assignment $b_p \leftarrow x$.
- `2` l r — let us treat the subsegment $b[l..r]$ as array a. You have to find the maximum number of zeroes which can be in that array after applying some *maximal* chain of operations. Note that we don't really change the array b during this query.

Input

The first line of input contains one integer n, the length of array b ($1 \le n \le 2 \cdot 10^5$).

The second line contains n integers separated by spaces: the initial elements of array b ($0 \le b_i \le 10^5$).

The third line contains one integer q, the number of queries ($1 \le q \le 2 \cdot 10^5$).

Each of the next q lines contains a query.

For queries of the first type, $1 \le p \le n$ and $0 \le x \le 10^5$.

For queries of the second type, $1 \le l \le r \le n$.

Output

For each query of the second type, print the answer on a separate line. Answers should be printed in the same order as queries.

Examples

standard input	standard output
3 2 2 2 9 2 1 3 1 1 1 2 1 3 1 2 1 2 1 3 1 1 4 2 1 3 2 1 2 2 2 3	2 2 0 1 1 0
1 20500 1 2 1 1	0