A. Fly

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

Natasha is going to fly on a rocket to Mars and return to Earth. Also, on the way to Mars, she will land on $n-2$ intermediate planets. Formally: we number all the planets from 1 to n. 1 is Earth, n is Mars. Natasha will make exactly n flights: $1 \to 2 \to \ldots n \to 1$.

Flight from x to y consists of two phases: take-off from planet x and landing to planet y. This way, the overall itinerary of the trip will be: the 1-st planet $\to$ take-off from the 1-st planet $\to$ landing to the 2-nd planet $\to$ 2-nd planet $\to$ take-off from the 2-nd planet $\to$ $\ldots \to$ landing to the n-th planet $\to$ the n-th planet $\to$ take-off from the n-th planet $\to$ landing to the 1-st planet $\to$ the 1-st planet.

The mass of the rocket together with all the useful cargo (but without fuel) is m tons. However, Natasha does not know how much fuel to load into the rocket. Unfortunately, fuel can only be loaded on Earth, so if the rocket runs out of fuel on some other planet, Natasha will not be able to return home. Fuel is needed to take-off from each planet and to land to each planet. It is known that 1 ton of fuel can lift off a_i tons of rocket from the i-th planet or to land b_i tons of rocket onto the i-th planet.

For example, if the weight of rocket is 9 tons, weight of fuel is 3 tons and take-off coefficient is 8 ($a_i = 8$), then 1.5 tons of fuel will be burnt (since $1.5 \cdot 8 = 9 + 3$). The new weight of fuel after take-off will be 1.5 tons.

Please note, that it is allowed to burn non-integral amount of fuel during take-off or landing, and the amount of initial fuel can be non-integral as well.

Help Natasha to calculate the minimum mass of fuel to load into the rocket. Note, that the rocket must spend fuel to carry both useful cargo and the fuel itself. However, it doesn't need to carry the fuel which has already been burnt. Assume, that the rocket takes off and lands instantly.

Input

The first line contains a single integer n ($2 \le n \le 1000$) — number of planets.

The second line contains the only integer m ($1 \le m \le 1000$) — weight of the payload.

The third line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 1000$), where a_i is the number of tons, which can be lifted off by one ton of fuel.

The fourth line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 1000$), where b_i is the number of tons, which can be landed by one ton of fuel.

It is guaranteed, that if Natasha can make a flight, then it takes no more than 10^9 tons of fuel.

Output

If Natasha can fly to Mars through $(n-2)$ planets and return to Earth, print the minimum mass of fuel (in tons) that Natasha should take. Otherwise, print a single number -1.

It is guaranteed, that if Natasha can make a flight, then it takes no more than 10^9 tons of fuel.

The answer will be considered correct if its absolute or relative error doesn't exceed 10^{-6}. Formally, let your answer be p, and the jury's answer be q. Your answer is considered correct if $\frac{|p-q|}{\max(1, |q|)} \le 10^{-6}$.

Examples

input

```
2
12
11 8
7 5
```

output

```
10.0000000000
```

input

```
3
1
1 4 1
2 5 3
```

output

```
-1
```

input

```
6
2
4 6 3 3 5 6
2 6 3 6 5 3
```

output

```
85.4800000000
```

Note

Let's consider the first example.

Initially, the mass of a rocket with fuel is 22 tons.

- At take-off from Earth one ton of fuel can lift off 11 tons of cargo, so to lift off 22 tons you need to burn 2 tons of fuel. Remaining weight of the rocket with fuel is 20 tons.
- During landing on Mars, one ton of fuel can land 5 tons of cargo, so for landing 20 tons you will need to burn 4 tons of fuel. There will be 16 tons of the rocket with fuel remaining.
- While taking off from Mars, one ton of fuel can raise 8 tons of cargo, so to lift off 16 tons you will need to burn 2 tons of fuel. There will be 14 tons of rocket with fuel after that.
- During landing on Earth, one ton of fuel can land 7 tons of cargo, so for landing 14 tons you will need to burn 2 tons of fuel. Remaining weight is 12 tons, that is, a rocket without any fuel.

In the second case, the rocket will not be able even to take off from Earth.

B. Rocket

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

This is an interactive problem.

Natasha is going to fly to Mars. Finally, Natasha sat in the rocket. She flies, flies... but gets bored. She wishes to arrive to Mars already! So she decides to find something to occupy herself. She couldn't think of anything better to do than to calculate the distance to the red planet.

Let's define x as the distance to Mars. Unfortunately, Natasha does not know x. But it is known that $1 \le x \le m$, where Natasha knows the number m. Besides, x and m are positive integers.

Natasha can ask the rocket questions. Every question is an integer y ($1 \le y \le m$). The correct answer to the question is -1, if $x < y$, 0, if $x = y$, and 1, if $x > y$. But the rocket is broken — it does not always answer correctly. Precisely: let the correct answer to the current question be equal to t, then, if the rocket answers this question correctly, then it will answer t, otherwise it will answer $-t$.

In addition, the rocket has a sequence p of length n. Each element of the sequence is either 0 or 1. The rocket processes this sequence in the cyclic order, that is 1-st element, 2-nd, 3-rd, $\ldots$, $(n-1)$-th, n-th, 1-st, 2-nd, 3-rd, $\ldots$, $(n-1)$-th, n-th, $\ldots$. If the current element is 1, the rocket answers correctly, if 0 — lies. Natasha doesn't know the sequence p, but she knows its length — n.

You can ask the rocket no more than 60 questions.

Help Natasha find the distance to Mars. Assume, that the distance to Mars does not change while Natasha is asking questions.

Your solution will not be accepted, if it does not receive an answer 0 from the rocket (even if the distance to Mars is uniquely determined by the already received rocket's answers).

Input

The first line contains two integers m and n ($1 \le m \le 10^9$, $1 \le n \le 30$) — the maximum distance to Mars and the number of elements in the sequence p.

Interaction

You can ask the rocket no more than 60 questions.

To ask a question, print a number y ($1 \le y \le m$) and an end-of-line character, then do the operation `flush` and read the answer to the question.

If the program reads 0, then the distance is correct and you must immediately terminate the program (for example, by calling `exit(0)`). If you ignore this, you can get any verdict, since your program will continue to read from the closed input stream.

If at some point your program reads -2 as an answer, it must immediately end (for example, by calling `exit(0)`). You will receive the "Wrong answer" verdict, and this will mean that the request is incorrect or the number of requests exceeds 60. If you ignore this, you can get any verdict, since your program will continue to read from the closed input stream.

If your program's request is not a valid integer between -2^{31} and $2^{31}-1$ (inclusive) without leading zeros, then you can get any verdict.

You can get "Idleness limit exceeded" if you don't print anything or if you forget to flush the output.

To flush the output buffer you can use (after printing a query and end-of-line):

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

Hacking

Use the following format for hacking:

In the first line, print 3 integers m, n, x ($1 \le x \le m \le 10^9$, $1 \le n \le 30$) — the maximum distance to Mars, the number of elements in the sequence p and the current distance to Mars.

In the second line, enter n numbers, each of which is equal to 0 or 1 — sequence p.

The hacked solution will not have access to the number x and sequence p.

Example

input

```
5 2
1
-1
-1
1
0
```

output

```
1
2
4
5
3
```

Note

In the example, hacking would look like this:

```
5 2 3

1 0
```

This means that the current distance to Mars is equal to 3, Natasha knows that it does not exceed 5, and the rocket answers in order: correctly, incorrectly, correctly, incorrectly ...

Really:

on the first query (1) the correct answer is 1, the rocket answered correctly: 1;

on the second query (2) the correct answer is 1, the rocket answered incorrectly: -1;

on the third query (4) the correct answer is -1, the rocket answered correctly: -1;

on the fourth query (5) the correct answer is -1, the rocket answered incorrectly: 1;

on the fifth query (3) the correct and incorrect answer is 0.

C. Border

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

Astronaut Natasha arrived on Mars. She knows that the Martians are very poor aliens. To ensure a better life for the Mars citizens, their emperor decided to take tax from every tourist who visited the planet. Natasha is the inhabitant of Earth, therefore she had to pay the tax to enter the territory of Mars.

There are n banknote denominations on Mars: the value of i-th banknote is a_i. Natasha has an infinite number of banknotes of each denomination.

Martians have k fingers on their hands, so they use a number system with base k. In addition, the Martians consider the digit d (in the number system with base k) divine. Thus, if the last digit in Natasha's tax amount written in the number system with the base k is d, the Martians will be happy. Unfortunately, Natasha does not know the Martians' divine digit yet.

Determine for which values d Natasha can make the Martians happy.

Natasha can use only her banknotes. Martians don't give her change.

Input

The first line contains two integers n and k ($1 \le n \le 100\,000$, $2 \le k \le 100\,000$) — the number of denominations of banknotes and the base of the number system on Mars.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — denominations of banknotes on Mars.

All numbers are given in decimal notation.

Output

On the first line output the number of values d for which Natasha can make the Martians happy.

In the second line, output all these values in increasing order.

Print all numbers in decimal notation.

Examples

input

```
2 8
12 20
```

output

```
2
0 4
```

input

```
3 10
10 20 30
```

output

```
1
0
```

Note

Consider the first test case. It uses the octal number system.

If you take one banknote with the value of 12, you will get 14_8 in octal system. The last digit is 4_8.

If you take one banknote with the value of 12 and one banknote with the value of 20, the total value will be 32. In the octal system, it is 40_8. The last digit is 0_8.

If you take two banknotes with the value of 20, the total value will be 40, this is 50_8 in the octal system. The last digit is 0_8.

No other digits other than 0_8 and 4_8 can be obtained. Digits 0_8 and 4_8 could also be obtained in other ways.

The second test case uses the decimal number system. The nominals of all banknotes end with zero, so Natasha can give the Martians only the amount whose decimal notation also ends with zero.

D. Mars rover

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Natasha travels around Mars in the Mars rover. But suddenly it broke down, namely — the logical scheme inside it. The scheme is an undirected tree (connected acyclic graph) with a root in the vertex 1, in which every leaf (excluding root) is an input, and all other vertices are logical elements, including the root, which is output. One bit is fed to each input. One bit is returned at the output.

There are four types of logical elements: AND (2 inputs), OR (2 inputs), XOR (2 inputs), NOT (1 input). Logical elements take values from their direct descendants (inputs) and return the result of the function they perform. Natasha knows the logical scheme of the Mars rover, as well as the fact that only one input is broken. In order to fix the Mars rover, she needs to change the value on this input.

For each input, determine what the output will be if Natasha changes this input.

Input

The first line contains a single integer n ($2 \le n \le 10^6$) — the number of vertices in the graph (both inputs and elements).

The i-th of the next n lines contains a description of i-th vertex: the first word "AND", "OR", "XOR", "NOT" or "IN" (means the input of the scheme) is the vertex type. If this vertex is "IN", then the value of this input follows (0 or 1), otherwise follow the indices of input vertices of this element: "AND", "OR", "XOR" have 2 inputs, whereas "NOT" has 1 input. The vertices are numbered from one.

It is guaranteed that input data contains a correct logical scheme with an output produced by the vertex 1.

Output

Print a string of characters '0' and '1' (without quotes) — answers to the problem for each input in the ascending order of their vertex indices.

Example

input

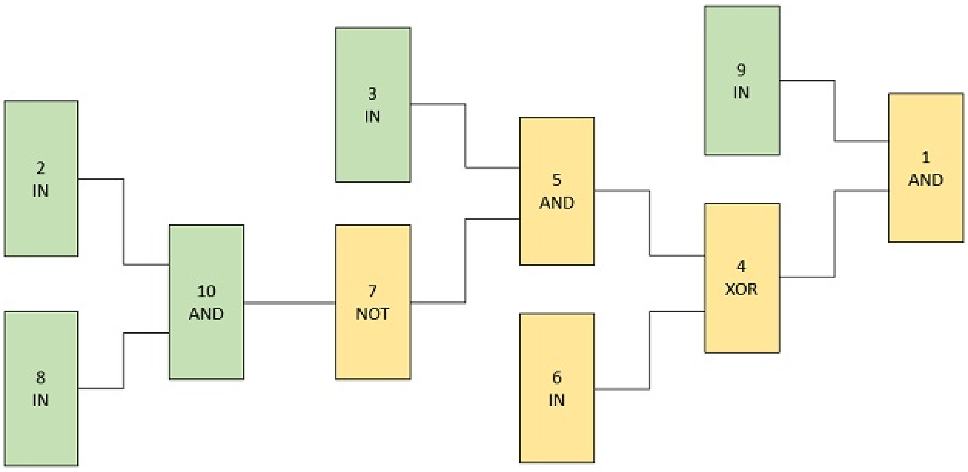
```
10
AND 9 4
IN 1
IN 1
XOR 6 5
AND 3 7
IN 0
NOT 10
IN 1
IN 1
AND 2 8
```

output

```
10110
```

Note

The original scheme from the example (before the input is changed):



Green indicates bits '1', yellow indicates bits '0'.

If Natasha changes the input bit 2 to 0, then the output will be 1.

If Natasha changes the input bit 3 to 0, then the output will be 0.

If Natasha changes the input bit 6 to 1, then the output will be 1.

If Natasha changes the input bit 8 to 0, then the output will be 1.

If Natasha changes the input bit 9 to 0, then the output will be 0.

E. Store

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

Natasha was already going to fly back to Earth when she remembered that she needs to go to the Martian store to buy Martian souvenirs for her friends.

It is known, that the Martian year lasts x_{max} months, month lasts y_{max} days, day lasts z_{max} seconds. Natasha also knows that this store works according to the following schedule: 2 months in a year were selected: x_l and x_r ($1 \le x_l \le x_r \le x_{max}$), 2 days in a month: y_l and y_r ($1 \le y_l \le y_r \le y_{max}$) and 2 seconds in a day: z_l and z_r ($1 \le z_l \le z_r \le z_{max}$). The store works at all such moments (month x, day y, second z), when simultaneously $x_l \le x \le x_r$, $y_l \le y \le y_r$ and $z_l \le z \le z_r$.

Unfortunately, Natasha does not know the numbers $x_l, x_r, y_l, y_r, z_l, z_r$.

One Martian told Natasha: "I went to this store $(n + m)$ times. n times of them it was opened, and m times — closed." He also described his every trip to the store: the month, day, second of the trip and whether the store was open or closed at that moment.

Natasha can go to the store k times. For each of them, determine whether the store at the time of the trip is open, closed, or this information is unknown.

Input

The first line contains 6 integers x_{max}, y_{max}, z_{max}, n, m, k ($1 \le x_{max}, y_{max}, z_{max} \le 10^5$, $1 \le n \le 10^5$, $0 \le m \le 10^5$, $1 \le k \le 10^5$) — number of months in a year, days in a month, seconds in a day, times when the store (according to a Martian) was opened, when it was closed and Natasha's queries.

The i-th of the next n lines contains 3 integers x_i, y_i, z_i ($1 \le x_i \le x_{max}$, $1 \le y_i \le y_{max}$, $1 \le z_i \le z_{max}$) — month, day and second of i-th time, when the store, according to the Martian, was opened.

The i-th of the next m lines contains 3 integers x_i, y_i, z_i ($1 \le x_i \le x_{max}$, $1 \le y_i \le y_{max}$, $1 \le z_i \le z_{max}$) — month, day and second of i-th time, when the store, according to the Martian, was closed.

The i-th of the next k lines contains 3 integers x_i, y_i, z_i ($1 \le x_i \le x_{max}$, $1 \le y_i \le y_{max}$, $1 \le z_i \le z_{max}$) — month, day and second of i-th Natasha's query.

Output

If the Martian was mistaken and his information about when the store is open and when it is closed is inconsistent, print a single line "`INCORRECT`" (without quotes).

Otherwise, print the first line "`CORRECT`" (without quotes). Next output k lines: in i-th of them, output an answer to i-th Natasha's query: "`OPEN`" (without quotes), if the store was opened at the moment of this query, "`CLOSED`" (without quotes), if it was closed, or "`UNKNOWN`" (without quotes), if this information can not be determined on the basis of available data.

Examples

input

```
10 10 10 3 1 3
2 6 2
4 2 4
6 4 6
9 9 9
3 3 3
10 10 10
8 8 8
```

output

```
CORRECT
OPEN
CLOSED
UNKNOWN
```

input

```
10 10 10 1 1 1
2 5 7
2 5 7
8 9 10
```

output

```
INCORRECT
```

Note

Consider the first test case.

There are 10 months in a year, 10 days in a month, and 10 seconds in a day.

The store was opened in 3 moments:

- month 2, day 6, second 2;
- month 4, day 2, second 4;
- month 6, day 4, second 6.

The store was closed at the time: month 9, day 9, second 9.

Queries:

- month 3, day 3, second 3 — open ("`OPEN`") (since the store opens no later than month 2, day 2, second 2 and closes no earlier than in month 6, day 6, second 6);
- month 10, day 10, second 10 — closed ("`CLOSED`") (since it is closed even in the month 9, day 9, second 9);
- month 8, day 8, second 8 — unknown ("`UNKNOWN`") (because the schedule in which the store is open at this moment exists, and the schedule in which the store is closed at this moment exists as well).

In the second test case, the store was closed and opened at the same time — contradiction ("`INCORRECT`").

F. Tree

time limit per test: 7 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Main Martian Tree grows on Mars. It is a binary tree (a rooted tree, with no more than two sons at each vertex) with n vertices, where the root vertex has the number 1. Its fruits are the Main Martian Fruits. It's summer now, so this tree does not have any fruit yet.

Autumn is coming soon, and leaves and branches will begin to fall off the tree. It is clear, that if a vertex falls off the tree, then its entire subtree will fall off too. In addition, the root will remain on the tree. Formally: the tree will have some connected subset of vertices containing the root.

After that, the fruits will grow on the tree (only at those vertices which remain). Exactly x fruits will grow in the root. The number of fruits in each remaining vertex will be not less than the sum of the numbers of fruits in the remaining sons of this vertex. It is allowed, that some vertices will not have any fruits.

Natasha wondered how many tree configurations can be after the described changes. Since this number can be very large, output it modulo 998244353.

Two configurations of the resulting tree are considered different if one of these two conditions is true:

- they have different subsets of remaining vertices;
- they have the same subset of remaining vertices, but there is a vertex in this subset where they have a different amount of fruits.

Input

The first line contains two integers: n and x ($1 \le n \le 10^5$, $0 \le x \le 10^{18}$) — the size of the tree and the number of fruits in the root.

The i-th of the following $(n-1)$ lines contains two integers a_i and b_i ($1 \le a_i, b_i \le n$) — vertices connected by the i-th edge of the tree.

It is guaranteed that the input data describes a correct binary tree with the root at the vertex 1.

Output

Print one number — the number of configurations of the resulting tree modulo 998244353.

Examples

input

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

output

```
13
```

input

```
2 5
1 2
```

output

```
7
```

input

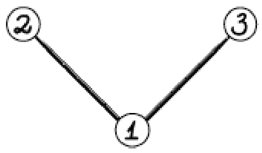
```
4 10
1 2
1 3
3 4
```

output

```
441
```

Note

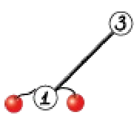
Consider the first example.



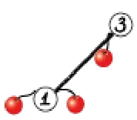
There are 2 fruits at the vertex 1. The following 13 options are possible:

- there is no vertex 2, there is no vertex 3;

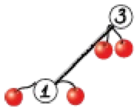
- there is no vertex 2, there are no fruits at the vertex 3;



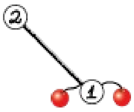
- there is no vertex 2, there is 1 fruit at the vertex 3;



- there is no vertex 2, there are 2 fruits at the vertex 3;



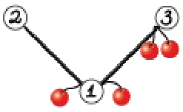
- there are no fruits at the vertex 2, there is no vertex 3;



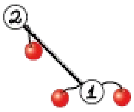
- there are no fruits at the vertex 2, there are no fruits at the vertex 3;

- there are no fruits at the vertex 2, there is 1 fruit at the vertex 3;

- there are no fruits at the vertex 2, there are 2 fruits at the vertex 3;

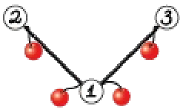


- there is 1 fruit at the vertex 2, there is no vertex 3;

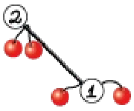


- there is 1 fruit at the vertex 2, there are no fruits at the vertex 3;

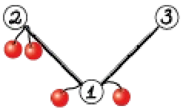
- there is 1 fruit at the vertex 2, there is 1 fruit at the vertex 3;



- there are 2 fruits at the vertex 2, there is no vertex 3;



- there are 2 fruits at the vertex 2, there are no fruits at the vertex 3.



Consider the second example. There are 5 fruits at the vertex 1. The following 7 options are possible:

- there is no vertex 2;
- there are no fruits at the vertex 2;
- there is 1 fruit at the vertex 2;
- there are 2 fruits at the vertex 2;
- there are 3 fruits at the vertex 2;
- there are 4 fruits at the vertex 2;
- there are 5 fruits at the vertex 2.