Problem A — limit 1 second

Delayed Work

You own a company that hires painters to paint houses. You can hire as many painters as you want, but for every painter you hire, you have to pay X dollars (independent of how long the painter works on the house). In addition, you have to pay a penalty of $D \cdot P$ dollars overall if it takes D days to finish painting the house. This penalty does not depend on the number of painters you hire; furthermore, even if the time taken is not a whole number of days, you are only penalized for the exact time taken.

All painters paint at the same rate. One painter can finish painting the house in K days. The painters cooperate so well that it will only take K/M days for M painters to finish painting the house.

What is the minimum amount of money you will have to pay, including what you pay to the painters and the cost penalty, to get a house painted?

Input

The input consists of a single line containing three space-separated integers K, P, and X ($1 \leq K, P, X \leq 10{,}000$).

Output

Print, on a single line, the minimum cost to have the house painted, rounded and displayed to exactly three decimal places.

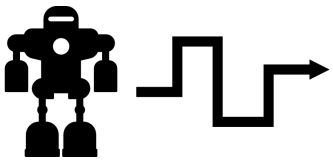
Sample Input and Output

31 41 59	549.200

3 4 5	16.000

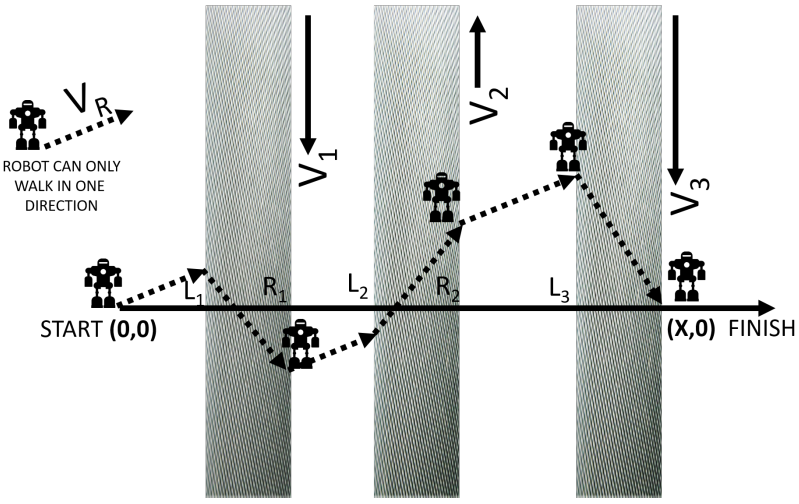
Problem B — limit 1 second

Straight Shot



You have a toy robot that walks straight at a constant speed v, and you wish for it to travel on the two-dimensional plane from $(0, 0)$ to $(X, 0)$. If the plane were empty, you could start the robot facing straight east from the origin, and it would walk there in X/v time. Unfortunately, between the start and the destination are n moving sidewalks, each moving directly north or south, which affect the robot's position while it is walking.

The direction that robot is facing is not changed by the sidewalks; the robot will face in the same orientation for the entire duration of its walk. These sidewalks are aligned with the y-axis and are infinitely long. You still must get the robot to go from start to finish, but you'll need to adjust the orientation of the robot at the start. Given that you choose this direction correctly, so that the robot arrives exactly at the destination, how long will it take the robot to get there?



One final caveat: You don't want the toy robot to walk for too long. If the robot cannot reach the destination in at most twice the time it would take in the absence of all moving sidewalks (*i.e.*, $2X/v$), indicate this.

Input

The first line consists of three space-separated numbers n, X, and v ($0 \leq n \leq 100$; $1 \leq X \leq 1{,}000{,}000$; $1.0 \leq v \leq 100.0$). Note that v is not necessarily an integer.

Each of the next n lines contains three space-separated numbers l_i, r_i, and v_i ($0 \leq l_1 < r_1 \leq l_2 < r_2 \leq \cdots \leq l_n < r_n \leq X$; $-100.0 \leq v_i \leq 100.0$), describing the ith moving sidewalk. The integer l_i denotes the left edge of the sidewalk, the integer r_i denotes the right edge of the sidewalk, and the decimal number v_i denotes the speed of the sidewalk. A positive speed means the sidewalk moves north, while a negative speed means the sidewalk moves south.

Output

If the robot cannot reach the destination in at most twice the time it would take in the absence of all moving sidewalks, output "`Too hard`" on a single line (without quotation marks).

Otherwise, output, on a single line, the travel time of the robot from the start to the destination, rounded and displayed to exactly three decimal places.

Sample Input and Output

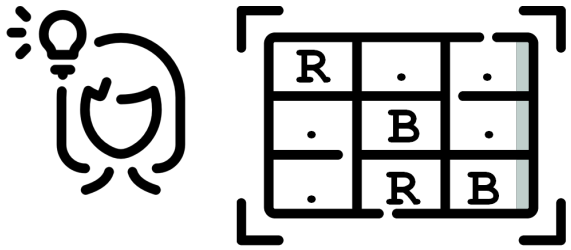
`1 806873 66` `91411 631975 -57.5`	`15055.988`

`2 422193 100`
`38180 307590 86.4`
`366035 403677 -4.7`
	`5043.896`

`1 670764 22.4` `113447 642610 -64.8`	`Too hard`

Problem C — limit 1 second

Grid Coloring



You have an m-by-n grid of squares that you wish to color. You may color each square either red or blue, subject to the following constraints:

- Every square must be colored.
- Colors of some squares are already decided (red or blue), and cannot be changed.
- For each blue square, all squares in the rectangle from the top left of the grid to that square must also be blue.

Given these constraints, how many distinct colorings of the grid are there? The grid cannot be rotated.

Input

The first line of input consists of two space-separated integers m and n ($1 \leq m, n \leq 30$).

Each of the next m lines contains n characters, representing the grid. Character '`B`' indicates squares that are already colored blue. Similarly, '`R`' indicates red squares. Character '`.`' indicates squares that are not colored yet.

Output

Print, on a single line, the number of distinct colorings possible.

For the first sample, the 6 ways are:

```
BB      BB      BR      BR      BB      BB
BB      BR      BR      BR      BR      BB
BR      BR      BR      RR      RR      RR
```

Sample Input and Output

3 2 .. B. .R	6

7 6 ....B .B..R. ....B.. .R.... ...R..	3

2 2 R. .B	0

Problem D — limit 2 seconds

Rainbow Roads

You are given a tree with n nodes (conveniently numbered from 1 to n). Each edge in this tree has one of n colors. A path in this tree is called a *rainbow* if all adjacent edges in the path have different colors. Also, a node is called *good* if every simple path with that node as one of its endpoints is a *rainbow* path.

Find all the *good* nodes in the given tree.

A simple path is a path that does not repeat any vertex or edge.

Input

The first line of input contains a single integer n ($1 \leq n \leq 50{,}000$).

Each of the next $n-1$ lines contains three space-separated integers a_i, b_i, and c_i ($1 \leq a_i, b_i, c_i \leq n$; $a_i \neq b_i$), describing an edge of color c_i that connects nodes a_i and b_i.

It is guaranteed that the given edges form a tree.

Output

On the first line of the output, print k, the number of good nodes.

In the next k lines, print the indices of all good nodes in numerical order, one per line.

For the first sample, node 3 is good since all paths that have node 3 as an endpoint are rainbow. In particular, even though the path 3—4—5—6 has two edges of the same color (i.e. 3—4, 5—6), it is still rainbow since these edges are not adjacent.

Sample Input and Output

8 1 3 1 2 3 1 3 4 3 4 5 4 5 6 3 6 7 2 6 8 2	4 3 4 5 6

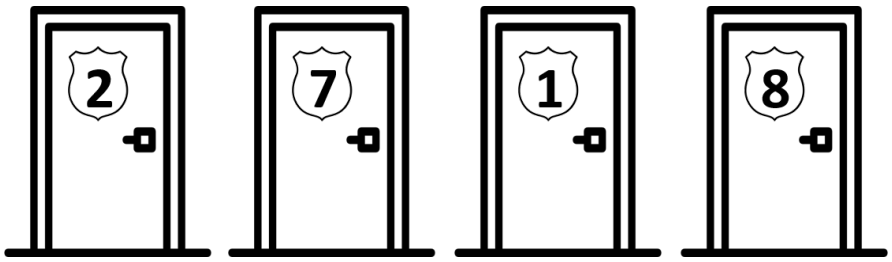
8 1 2 2 1 3 1 2 4 3 2 7 1 3 5 2 5 6 2 7 8 1	0

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

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

Problem E — limit 3 seconds

Security Badge



You are in charge of the security for a large building, with n rooms and m doors between the rooms. The rooms and doors are conveniently numbered from 1 to n, and from 1 to m, respectively.

Door i opens from room a_i to room b_i, but not the other way around. Additionally, each door has a security code that can be represented as a range of numbers $[c_i, d_i]$.

There are k employees working in the building, each carrying a security badge with a unique, integer-valued badge ID between 1 and k. An employee is cleared to go through door i only when the badge ID x satisfies $c_i \leq x \leq d_i$.

Your boss wants a quick check of the security of the building. Given s and t, how many employees can go from room s to room t?

Input

The first line of input contains three space-separated integers n, m, and k ($2 \leq n \leq 1{,}000$; $1 \leq m \leq 5{,}000$; $1 \leq k \leq 10^9$).

The second line of input contains two space-separated integers s and t ($1 \leq s, t \leq n$; $s \neq t$).

Each of the next m lines contains four space-separated integers a_i, b_i, c_i, and d_i ($1 \leq a_i, b_i \leq n$; $1 \leq c_i \leq d_i \leq k; a_i \neq b_i$), describing door i.

For any given pair of rooms a, b there will be at most one door from a to b (but there may be both a door from a to b and a door from b to a).

Output

Print, on a single line, the number of employees who can reach room t starting from room s.

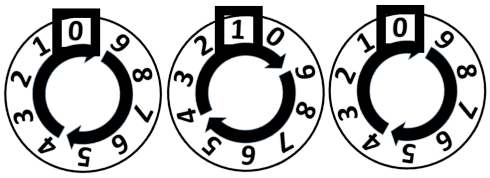
Sample Input and Output

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

4 5 9 1 4 1 2 3 5 1 3 6 7 1 4 2 3 2 4 4 6 3 4 7 9	5

Problem F — limit 1 second

Spinning Up Palindromes



"Sabotage!", exclaimed J. R. Diddly, president and founder of Diddly Widgets Inc.

"Vandalism, perhaps. Nothing's actually been damaged." responded Robert Lackey, the chief accountant.

Both were staring up at the large counter suspended above the factory floor, a counter that had faithfully recorded the number of widgets that had come off the assembly line since the factory was opened. But someone had changed the number being displayed so that it formed...

"It's a palindrome." said Lackey. "It reads the same forwards as backwards."

"What I don't understand," said Diddly, "is why our security guards didn't catch the vandals during their regular sweeps. It must have taken them hours to click forward to this new number, one step at a time."

"No." replied Lackey. "Although we only advance the rightmost digit each time a new widget is built, it's possible to spin any of the digits. With a little planning, this might have taken only a few seconds."

Consider a digital counter consisting of k wheels, each showing a digit from `0` to `9`. Each wheel is mounted so that it can advance to the next digit in a single step, *e.g.*, from `3` to `4`, or from `8` to `9`.

It is also possible to advance from digit `9` to digit `0`. However, when this happens, the wheel on its immediate left will also advance to the next digit automatically. This can have a cascade effect on multiple wheels to the left, but they all happen in a single step.

Given the current setting of the counter, find the smallest number of steps until one can reach a palindrome. The palindrome must respect leading zeros, *e.g.*, `0011` is not a palindrome.

For example, for input `610`, it takes four steps. This can be done by incrementing the `6` wheel four times, resulting in `010`.

Input

The first line of input contains a string of k digits ($1 \leq k \leq 40$), representing the current setting of the counter.

Note that the input may contain leading zeros.

Output

Print, on a single line, the minimum number of wheel advances necessary to produce a palindrome.

Sample Input and Output

0	0
009990001	3
29998	5
610	4
981	2
90841947009409037971917182478011970192 68	54

Problem G — limit 3 seconds

Distinct Distances

You're setting up a scavenger hunt that takes place in the two-dimensional plane.

You've already decided on n distinct points of interest labeled $p_1, \ldots, p_n$. The point p_i is located at integer coordinates (x_i, y_i).

You now want to choose a point q for the final location. This point must have finite coordinates, but it does not necessarily need to have integer coordinates. This point also can coincide with one of the original points p_i.

In order to make this final location interesting, you would like to minimize the number of unique distances from q to the other points.

More precisely, you would like to choose q that minimizes $|S(q)|$, where $S(q)$ is defined as the set

$$\{|q - p_1|, |q - p_2|, \ldots, |q - p_n|\}.$$

Here, the notation $|S(q)|$ means the number of elements in the set $S(q)$ and $|q - p_i|$ denotes the Euclidean distance between q and p_i. Note that $S(q)$ is a set, so if two or more distances $|q - p_i|$ are equal, they are counted as a single element in $S(q)$.

Given the coordinates of the points, find the minimum value of $|S(q)|$.

Warning: Use of inexact arithmetic may make it difficult to identify distances that are exactly equal.

Input

The first line of input contains a single integer n ($1 \leq n \leq 40$).

Each of the next n lines contains two space-separated integers x_i and y_i ($|x_i|, |y_i| \leq 300$), representing the coordinates of p_i.

Output

Output, on a single line, the minimum number of unique distances from q to all other points p_i.

For the first sample, we can let our point q be $(0, 0)$. All other points are distance 5 away. For the second sample, we can let q be $(1.5, 1.5)$.

Sample Input and Output

8 3 4 0 5 0 -5 5 0 -5 0 4 -3 3 -4 -4 3	1
4 0 0 1 1 2 2 3 3	2
6 0 -5 1 0 -1 0 2 3 3 2 -3 0	3