Problem A: Upside down primes

1 Point

Last night, I must have dropped my alarm clock. When the alarm went off in the morning, it showed `51:80` instead of `08:15`. This made me realize that if you rotate a seven segment display like it is used in digital clocks by 180 degrees, some numbers still are numbers after turning them upside down.

Figure K.1: Prime number 18115211 on a seven segment display (see third sample).

Figure K.2: 18115211 turned upside down (i.e. rotated by 180 degrees) gives 11251181, which is not prime.

As you can see,

- 0, 2, 5, and 8 still are 0, 2, 5, and 8.
- 1 is still readable as 1 (only moved left).
- 6 turns into 9, while 9 turns into 6.
- 3, 4 and 7 are no longer valid numbers (E, h and L)

My favourite numbers are primes, of course. Your job is to check whether a number is a prime and still a prime when turned upside down.

Input

One line with the integer N in question ($1 \leq N \leq 10^{16}$). N will not have leading zeros.

Output

Print one line of output containing "`yes`" if the number is a prime and still a prime if turned upside down, "`no`" otherwise.

Sample Input 1

```
151
```

Sample Output 1

```
yes
```

Sample Input 2

```
23
```

Sample Output 2

```
no
```

Sample Input 3

```
18115211
```

Sample Output 3

```
no
```

Problem B: Milling machines

1 Point

A fab lab is an open, small-scale workshop where you can create or fabricate almost anything you want mostly by using computer controlled tools like a laser cutter or a 3D printer. The FAU fab lab recently got a CNC milling machine. Using the milling machine you can cut or remove material with different tools from the surface of a workpiece. It is controlled via a computer program.

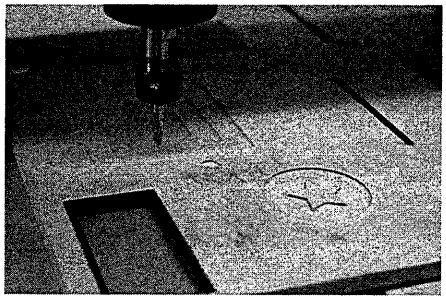


Photo by aurelie ghalim on Flickr

I sometimes wondered what happens if multiple different shaped workpieces are sent through the same milling program. For simplification assume that we have only two dimensional workpieces without holes. A milling program consists of multiple steps; each step describes where the milling machine has to remove material (using different tools) from the top of the surface.

Input

The first line consists of two integers W and S, where W gives the number of workpieces and S the number of steps in the milling program ($1 \leq W, S \leq 10^4$). The next line consists of two integers X and Y, where X gives the width and Y gives the maximal possible height of workpieces ($1 \leq X, Y \leq 100$).

Then follow W lines, each describing one workpiece. Each workpiece description consists of X non-negative integers specifying the surface height in that column.

Then follow S lines, each describing one milling step of the milling progam. Each milling step description consists of X non-negative integers s_i ($0 \leq s_i \leq Y$) specifying the amount of surface to cut off in each column (relative to the height of the milling area, i.e. Y, not relative to the top of the workpiece). See Fig. I.1 for details.

Output

For each workpiece, output one line containing X integers specifying the remaining surface heights (in the same order as in the input).

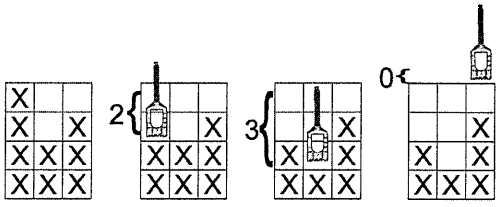


Figure I.1: Second workpiece in first sample: initial workpiece followed by milling in each column – the value in the milling program determines the vertical position of the cutter head.

Sample Input 1

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

Sample Output 1

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

Sample Input 2

```
1 3
10 100
11 22 33 44 55 66 77 88 99 100
1 100 1 100 1 100 1 100 1 100
58 58 58 58 58 58 58 58 58 58
42 42 42 42 42 42 42 42 66 42
```

Sample Output 2

```
11 0 33 0 42 0 42 0 34 0
```

Problem C: Legacy Code

1 point

Once again you lost days refactoring code, which never runs in the first place. Enough is enough – your time is better spent writing a tool that finds unused code!

Your software is divided into packages and executables. A package is a collection of methods. Executables are packages defining among other methods exactly one method with name `PROGRAM`. This method is executed on the start of the corresponding executable. Ordinary packages have no method named `PROGRAM`.

Each method is uniquely identified by the combination of package and method names. E.g. the method with the identifier `SuperGame::PROGRAM` would be the main method of the executable `SuperGame`.

For every method in your software you are given a list of methods directly invoking it. Thus you can easily identify methods, that are never called from any method. However, your task is more challenging: you have to find unused methods. These are methods that are never reached by the control flow of any executable in your software.

Input

The first line of the input contains an integer N, the number of methods in your software ($1 \leq N \leq 400$).

Each method is described by two lines, totaling in $2 \cdot N$ lines. The first line consists of the unique identifier of the method and k_i, the number of methods directly invoking this one ($0 \leq k_i \leq N$). The second line consists of a set of k_i identifiers of these calling methods or is empty if there are no such methods, i.e. $k_i = 0$.

Method identifiers consist of a package name followed by two colons and a method name like `Packagename::Methodname`. Both strings, the package and the method name, each consist of up to 20 lowercase, uppercase characters or digits (`a-z`, `A-Z`, `0-9`).

There will be exactly N different method identifiers mentioned in the input.

Output

A line containing the number of unused methods in your software.

Sample Input 1

```
2
SuperGame::PROGRAM 0

HelpPackage::HelpFunction 2
HelpPackage::HelpFunction SuperGame::PROGRAM
```

Sample Output 1

```
0
```

Sample Input 2

```
2
Loop::CallA 1
Loop::CallB
Loop::CallB 1
Loop::CallA
```

Sample Output 2

```
2
```

Sample Input 3

```
2
SuperGame::PROGRAM 1
SuperServer42::PROGRAM
SuperServer42::PROGRAM 1
SuperServer42::PROGRAM
```

Sample Output 3

```
0
```

Problem D: Change of Scenery 2 Points

Every day you drive to work using the same roads as it is the shortest way. This is efficient, but over time you have grown increasingly bored of seeing the same buildings and junctions every day. So you decide to look for different routes. Of course you do not want to sacrifice time, so the new way should be as short as the old one. Is there another way that differs from the old one in at least one street?

Input

The first line of the input starts with three integers N M and K, where N is the number of junctions and M is the number of streets in your city, and K is the number of junctions you pass every day ($1 \leq K \leq N \leq 10\,000$, $0 \leq M \leq 1\,000\,000$).
The next line contains K integers, the (1-based) indices of the junctions you pass every day. The first integer in this line will always be 1, the last integer will always be N. There is a shortest path from 1 to N along the K junctions given.
M lines follow. The i-th of those lines contains three integers a_i b_i c_i and describes a street from junction a_i to junction b_i of length c_i ($1 \leq a_i, b_i \leq N$, $1 \leq c_i \leq 10\,000$). Streets are always undirected.
Note that there may be multiple streets connecting the same pair of junctions. The shortest path given uses for every pair of successive junctions a and b a street of minimal length between a and b.

Output

Print one line of output containing "`yes`" if there is another way you can take without losing time, "`no`" otherwise.

Sample Input 1

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

Sample Output 1

```
yes
```

Sample Input 2

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

Sample Output 2

```
no
```

Problem E: Divisions 2 points

David is a young boy and he loves numbers. Recently he learned how to divide two numbers. David divides the whole day. He is happy if the result of the division is an integer, but he is not very amused if this is not the case. After quite a while he decided to use only a single dividend each day.

The parents of David are very careful and they would like to ensure that David experiences enough happiness. Therefore they decide which number David will use as the dividend for this day.

There is still a problem: The parents are not very good at math and don't know how to calculate the number of positive integral divisors for a given dividend N, which lead to an integral result. Now it's up to you to help David's parents.

Input

The single input line contains the single integer N, where N is chosen as a dividend ($1 \leq N \leq 10^{18}$).

Output

Print the number of positive integral divisors of N that lead to an integral result of the division.

Sample Input 1	Sample Output 1
12	6
Sample Input 2	**Sample Output 2**
999999999999999989	2
Sample Input 3	**Sample Output 3**
100000007700000049	4

Problem F : Bounty Hunter II

2 points

Spike the bounty hunter is tracking another criminal through space. Luckily for him hyperspace travel has made the task of visiting several planets a lot easier. Each planet has a number of Astral Gates; each gate connects with a gate on another planet. These hyperspace connections are, for obvious safety reasons, one-way only with one gate being the entry point and the other gate being the exit point from hyperspace. Furthermore, the network of hyperspace connections must be loop-free to prevent the Astral Gates from exploding, a tragic lesson learned in the gate accident of 2022 that destroyed most of the moon.
While looking at his star map Spike wonders how many friends he needs to conduct a search on every planet. Each planet should not be visited by more than one friend otherwise the criminal might get suspicious and flee before he can be captured. While each person can start at a planet of their choosing and travel along the hyperspace connections from planet to planet they are still bound by the limitations of hyperspace travel.

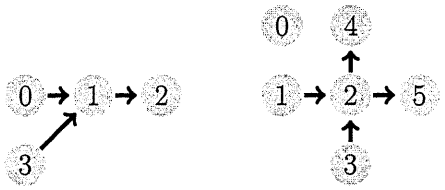


Figure B.1: Illustration of the Sample Inputs.

Input

The input begins with an integer N specifying the number of planets ($0 < N \leq 1000$). The planets are numbered from 0 to $N-1$. The following N lines specify the hyperspace connections. The i-th of those lines first contains the count of connections K ($0 \leq K \leq N-1$) *from* planet i followed by K integers specifying the destination planets.

Output

Output the minimum number of persons needed to visit every planet.

Sample Input 1

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

Sample Output 1

```
2
```

Sample Input 2

```
6
0
1 2
2 4 5
1 2
0
0
```

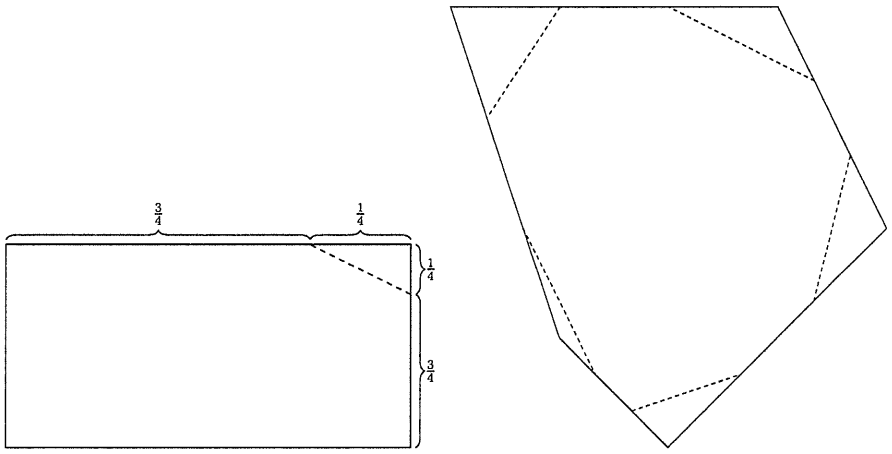
Sample Output 2

```
4
```

Problem G: Cake 3 points

Sophie loves to bake cakes and share them with friends. For the wedding of her best friend Bea she made a very special cake using only the best ingredients she could get and added a picture of the engaged couple on top of the cake. To make it even more special she did not make it round or square, but made a custom convex shape for the cake. Sophie decided to send the cake by a specialized carrier to the party. Unfortunately, the cake is a little too heavy for their default cake package and the overweight fees are excessive. Therefore, Sophie decides to remove some parts of the cake to make it a little lighter.

Sophie wants to cut the cake the following way: First, she chooses a real number $s \geq 2$. For each vertex and each incident edge of the cake she marks where $1/s$ of the edge's length is. Afterwards, she makes a direct cut between the two markings for each vertex and removes the vertex that way.



(a) Cutting the upper-right corner of a rectangle with $s = 4$

(b) Cutting a cake with $s = 3$

Figure C.1: Illustration of the first two Sample Inputs.

Sophie does not want to cut more from the cake than necessary for obvious reasons. Can you tell her how to choose s?

Input

The first line contains a floating point number a and an integer N, where a denotes the ratio of the cake's weight allowed by the carrier and N the number of vertices of the cake ($0.25 \leq a < 1$; $3 \leq N \leq 100$). a will be specified with at most 7 digits after the decimal point.

Then follow N lines, each describing one of the cake's vertices with two integers x_i and y_i, the coordinates of the vertex ($0 \leq x_i, y_i \leq 10^8$ for all $1 \leq i \leq N$). The vertices are given in the order in which they form a strictly convex shape.

You may safely assume that the weight is uniformly distributed over the area of the cake. Furthermore, it will always be possible to cut the cake with some $2 \leq s \leq 1000$ such that the proportion of the remaining cake is a of the original weight.

Output

Print a line containing s, the biggest value as specified above such that the remaining cake's weight is at most the proportion a of its original weight.

Your answer will be considered correct if the absolute error is at most 10^{-4}.

Sample Input 1

```
0.875 4
0 0
8 0
8 4
0 4
```

Sample Output 1

```
4.000000
```

Sample Input 2

```
0.85 5
6 0
12 6
9 12
0 12
3 3
```

Sample Output 2

```
3.000000
```

Sample Input 3

```
0.999998 4
20008 10000
15004 15005
10001 20009
15005 15004
```

Sample Output 3

```
1000.000000
```