A. Headquarters

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

Sensation, sensation in the two-dimensional kingdom! The police have caught a highly dangerous outlaw, member of the notorious "Pihters" gang. The law department states that the outlaw was driving from the gang's headquarters in his car when he crashed into an ice cream stall. The stall, the car, and the headquarters each occupies exactly one point on the two-dimensional kingdom.

The outlaw's car was equipped with a GPS transmitter. The transmitter showed that the car made **exactly** n movements on its way from the headquarters to the stall. A movement can move the car from point (x, y) to one of these four points: to point $(x - 1, y)$ which we will mark by letter "L", to point $(x + 1, y)$ — "R", to point $(x, y - 1)$ — "D", to point $(x, y + 1)$ — "U".

The GPS transmitter is very inaccurate and it doesn't preserve the exact sequence of the car's movements. Instead, it keeps records of the car's possible movements. Each record is a string of one of these types: "UL", "UR", "DL", "DR" or "ULDR". Each such string means that the car made a single movement corresponding to one of the characters of the string. For example, string "UL" means that the car moved either "U", or "L".

You've received the journal with the outlaw's possible movements from the headquarters to the stall. The journal records are given in a chronological order. Given that the ice-cream stall is located at point $(0, 0)$, your task is to print the number of different points that can contain the gang headquarters (that is, the number of different possible locations of the car's origin).

Input

The first line contains a single integer n $(1 \le n \le 2 \cdot 10^5)$ — the number of the car's movements from the headquarters to the stall.

Each of the following n lines describes the car's possible movements. It is guaranteed that each possible movement is one of the following strings: "UL", "UR", "DL", "DR" or "ULDR".

All movements are given in chronological order.

Please do not use the %lld specifier to read or write 64-bit integers in C++. It is preferred to use the cin and cout stream or the %I64d specifier.

Output

Print a single integer — the number of different possible locations of the gang's headquarters.

Sample test(s)

input

```
3
UR
UL
ULDR
```

output

```
9
```

input

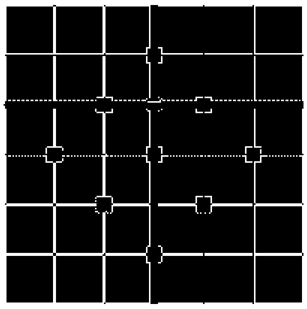
```
2
DR
DL
```

output

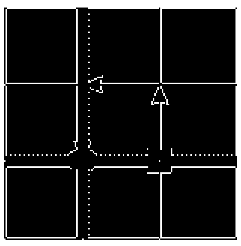
```
4
```

Note

The figure below shows the nine possible positions of the gang headquarters from the first sample:



For example, the following movements can get the car from point $(1, 0)$ to point $(0, 0)$:



B. Zoo

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

The Zoo in the Grid Kingdom is represented by an infinite grid. The Zoo has n observation binoculars located at the OX axis. For each i between 1 and n, inclusive, there exists a single binocular located at the point with coordinates $(i, 0)$. There are m flamingos in the Zoo, located at points with positive coordinates. The flamingos are currently sleeping and you can assume that they don't move.

In order to get a good view over the flamingos, each of the binoculars can be independently rotated to face any angle (not necessarily integer). Then, the binocular can be used to observe all flamingos that is located at the straight line passing through the binocular at the angle it is set. In other words, you can assign each binocular a direction corresponding to any straight line passing through the binocular, and the binocular will be able to see all flamingos located on that line.

Today, some kids from the prestigious Codeforces kindergarten went on a Field Study to the Zoo. Their teacher would like to set each binocular an angle to maximize the number of flamingos that can be seen by the binocular. The teacher is very interested in the sum of these values over all binoculars. Please help him find this sum.

Input

The first line contains two space-separated integers n and m $(1 \le n \le 10^6, 1 \le m \le 250)$, denoting the number of binoculars and the number of flamingos, respectively.

Then m lines follow, the i-th line will contain two space-separated integers x_i and y_i $(1 \le x_i, y_i \le 10^9)$, which means that the i-th flamingo is located at point (x_i, y_i).

All flamingos will be located at distinct points.

Output

Print a single integer denoting the maximum total number of flamingos that can be seen by all the binoculars.

Sample test(s)

input

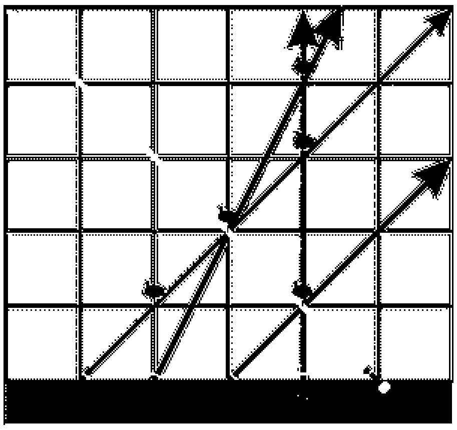
```
5 5
2 1
4 1
3 2
4 3
4 4
```

output

```
11
```

Note

This picture shows the answer to the example test case.



C. Cyclic Coloring

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

You are given a **directed** graph G with n vertices and m arcs (**multiple arcs and self-loops** are allowed). You have to paint each vertex of the graph into one of the k $(k \le n)$ colors in such way that for all arcs of the graph leading from a vertex u to vertex v, vertex v is painted with the *next color* of the color used to paint vertex u.

The colors are numbered cyclically 1 through k. This means that for each color i $(i < k)$ its next color is color $i+1$. In addition, the next color of color k is color 1. Note, that if $k = 1$, then the next color for color 1 is again color 1.

Your task is to find and print the largest possible value of k $(k \le n)$ such that it's possible to color G as described above with k colors. Note that you don't necessarily use all the k colors (that is, for each color i there does not necessarily exist a vertex that is colored with color i).

Input

The first line contains two space-separated integers n and m $(1 \le n, m \le 10^5)$, denoting the number of vertices and the number of arcs of the given digraph, respectively.

Then m lines follow, each line will contain two space-separated integers a_i and b_i $(1 \le a_i, b_i \le n)$, which means that the i-th arc goes from vertex a_i to vertex b_i.

Multiple arcs and self-loops are allowed.

Output

Print a single integer — the maximum possible number of the colors that can be used to paint the digraph (i.e. k, as described in the problem statement). Note that the desired value of k must satisfy the inequality $1 \le k \le n$.

Sample test(s)

input

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

output

```
2
```

input

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

output

```
5
```

input

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

output

```
3
```

input

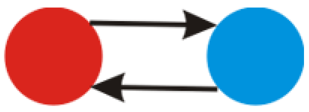
```
4 4
1 1
1 2
2 1
1 2
```

output

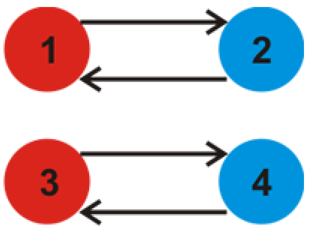
```
1
```

Note

For the first example, with $k = 2$, this picture depicts the two colors (arrows denote the next color of that color).

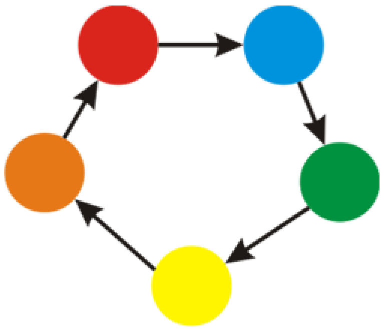


With $k = 2$ a possible way to paint the graph is as follows.

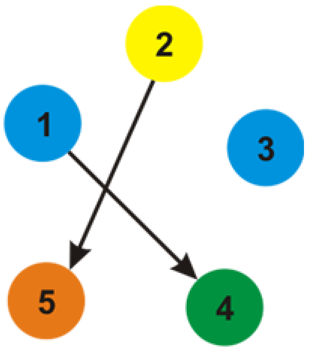


It can be proven that no larger value for k exists for this test case.

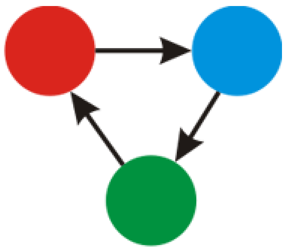
For the second example, here's the picture of the $k = 5$ colors.



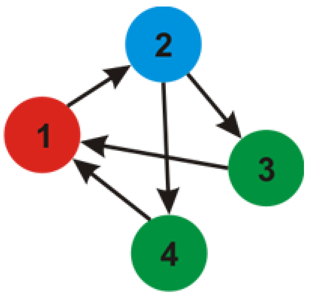
A possible coloring of the graph is:



For the third example, here's the picture of the $k = 3$ colors.



A possible coloring of the graph is:



D. T-shirt

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

You are going to work in Codeforces as an intern in a team of n engineers, numbered 1 through n. You want to give each engineer a souvenir: a T-shirt from your country (T-shirts are highly desirable in Codeforces). Unfortunately you don't know the size of the T-shirt each engineer fits in. There are m different sizes, numbered 1 through m, and each engineer will fit in a T-shirt of exactly one size.

You don't know the engineers' exact sizes, so you asked your friend, Gerald. Unfortunately, he wasn't able to obtain the exact sizes either, but he managed to obtain for each engineer i and for all sizes j, the probability that the size of the T-shirt that fits engineer i is j.

Since you're planning to give each engineer one T-shirt, you are going to bring with you exactly n T-shirts. For those n T-shirts, you can bring any combination of sizes (you can bring multiple T-shirts with the same size too!). You don't know the sizes of T-shirts for each engineer when deciding what sizes to bring, so you have to pick this combination based only on the probabilities given by your friend, Gerald.

Your task is to maximize the expected number of engineers that receive a T-shirt of his size.

This is defined more formally as follows. When you finally arrive at the office, you will ask each engineer his T-shirt size. Then, if you still have a T-shirt of that size, you will give him one of them. Otherwise, you don't give him a T-shirt. You will ask the engineers in order starting from engineer 1, then engineer 2, and so on until engineer n.

Input

The first line contains two space-separated integers n and m $(1 \le n \le 3000, 1 \le m \le 300)$, denoting the number of engineers and the number of T-shirt sizes, respectively.

Then n lines follow, each line contains m space-separated integers. The j-th integer in the i-th line represents the probability that the i-th engineer fits in a T-shirt of size j. Each probability will be given as an integer between 0 and 1000, inclusive. The actual probability should be calculated as the given number divided by 1000.

It is guaranteed that for any engineer, the sum of the probabilities for all m T-shirts is equal to one.

Output

Print a single real number denoting the maximum possible expected number of engineers that will receive a T-shirt.

For the answer the absolute or relative error of 10^{-9} is acceptable.

Sample test(s)

input

```
2 2
500 500
500 500
```

output

```
1.500000000000
```

input

```
3 3
1000 0 0
1000 0 0
0 1000 0
```

output

```
3.000000000000
```

input

```
1 4
100 200 300 400
```

output

```
0.400000000000
```

Note

For the first example, bring one T-shirt of each size. With 0.5 chance, either both engineers fit inside T-shirts of size 1 or both fit inside T-shirts of size 2. With the other 0.5 chance, one engineer fits inside a T-shirt of size 1 and the other inside a T-shirt of size 2. If the first is true, the number of engineers that receive a T-shirt is one. If the second is true, the number of such engineers is two. Hence, the expected number of engineers who receive a T-shirt is 1.5. This is maximum possible expected number of engineers for all sets of T-shirts.

For the second example, bring two T-shirts of size 1 and one T-shirt of size 2. This way, each engineer will definitely receive a T-shirt of his size.

For the third example, bring one T-shirt of size 4.

E. Candy Shop

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The prestigious Codeforces kindergarten consists of n kids, numbered 1 through n. Each of them are given allowance in rubles by their parents.

Today, they are going to the most famous candy shop in the town. The shop sells candies in packages: for all i between 1 and m, inclusive, it sells a package containing exactly i candies. A candy costs one ruble, so a package containing x candies costs x rubles.

The kids will purchase candies in turns, starting from kid 1. In a single turn, kid i will purchase one candy package. Due to the highly competitive nature of Codeforces kindergarten, during a turn, the number of candies contained in the package purchased by the kid will always be strictly greater than the number of candies contained in the package purchased by the kid in the preceding turn (an exception is in the first turn: the first kid may purchase any package). Then, the turn proceeds to kid $i + 1$, or to kid 1 if it was kid n's turn. This process can be ended at any time, but at the end of the purchase process, all the kids *must have the same number of candy packages*. Of course, the amount spent by each kid on the candies cannot exceed their allowance.

You work at the candy shop and would like to prepare the candies for the kids. Print the maximum number of candies that can be sold by the candy shop to the kids. If the kids cannot purchase any candy (due to insufficient allowance), print 0.

Input

The first line contains two space-separated integers n and m ($2 \le n \le 2 \cdot 10^5, 2 \le m \le 5 \cdot 10^6, n \le m$), denoting the number of kids and the maximum number of candies in a package sold by the candy shop, respectively.

Then n lines follow, each line will contain a single positive integer not exceeding $\frac{m(m+1)}{2}$ denoting the allowance of a kid in rubles. The allowances are given in order from kid 1 to kid n.

Please, do not use the `%lld` specificator to read or write 64-bit integers in C++. It is recommended to use `cin`, `cout` streams (also you may use `%I64d` specificator).

Output

Print a single integer denoting the maximum number of candies that can be sold by the candy shop.

Sample test(s)

input

```
2 5
5
10
```

output

```
13
```

input

```
3 8
8
16
13
```

output

```
32
```

input

```
2 5000000
12500002500000
12500002500000
```

output

```
12500002500000
```

Note

For the first example, one of the scenarios that will result in 13 purchased candies is as follows.

- Turn 1. Kid 1 purchases 1 candy.
- Turn 2. Kid 2 purchases 3 candies.
- Turn 3. Kid 1 purchases 4 candies.
- Turn 4. Kid 2 purchases 5 candies.