Problem A
Daydreaming Stockbroker
Problem ID: stockbroker
Time limit: 1 second

Gina Reed, the famous stockbroker, is having a slow day at work, and between rounds of solitaire she is daydreaming. Foretelling the future is hard, but imagine if you could just go back in time and use your knowledge of stock price history in order to maximize your profits!

Photo by liz west on flickr, cc by

Now Gina starts to wonder: if she were to go back in time a few days and bring a measly $\$100$ with her, how much money could she make by just buying and selling stock in Rollercoaster Inc. (the most volatile stock in existence) at the right times? Would she earn enough to retire comfortably in a mansion on Tenerife?

Note that Gina can not buy fractional shares, she must buy whole shares in Rollercoaster Inc. The total number of shares in Rollercoaster Inc. is $100\,000$, so Gina can not own more than $100\,000$ shares at any time. In Gina's daydream, the world is nice and simple: there are no fees for buying and selling stocks, stock prices change only once per day, and her trading does not influence the valuation of the stock.

Input

The first line of input contains an integer d ($1 \leq d \leq 365$), the number of days that Gina goes back in time in her daydream. Then follow d lines, the i'th of which contains an integer p_i ($1 \leq p_i \leq 500$) giving the price at which Gina can buy or sell stock in Rollercoaster Inc. on day i. Days are ordered from oldest to newest.

Output

Output the maximum possible amount of money Gina can have on the last day. Note that the answer may exceed 2^{32}.

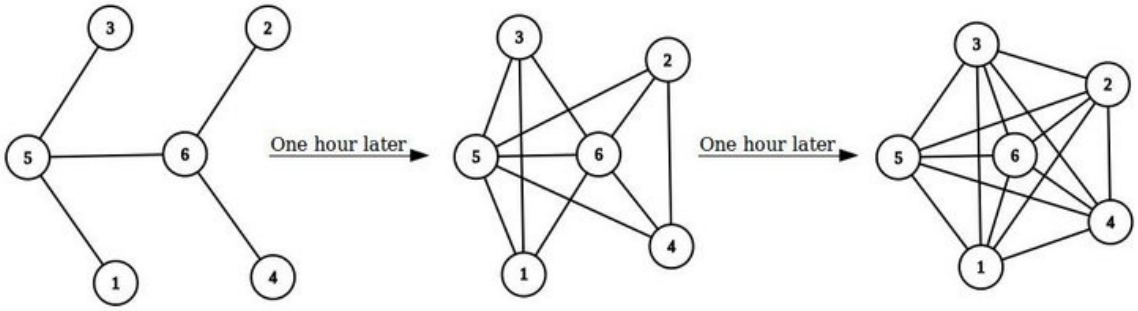
Sample Input 1	Sample Output 1
6 100 200 100 150 125 300	650

B. Bacteria Experiment

time limit per test: 1.5 s
memory limit per test: 256 MB
input: standard input
output: standard output

A year after his bacteria experiment, Jason decided to perform another experiment on a new bacteria specie which evolves in a special way. The initial form of the bacteria can be considered as an undirected tree with N nodes in terms of graph theory. Every hour, an edge (x, y) is built if there exists a node z such that, in the previous hour, there exists edge (x, z) and edge (y, z), but not edge (x, y). The bacteria keep evolving until no more edges can be formed.

The following graph shows a type of bacteria which requires 2 hours to fully evolve:



As it may take months if not years for the bacteria to evolve to its ultimate form, it is impossible for Jason to stay at the laboratory to observe the change of the bacteria throughout the entire process. Therefore, he wants you to calculate the time required for the bacteria to fully evolve, so that he can just get back to the laboratory on time.

Input

The first line contains an integer N. ($2 \le N \le 5 \times 10^5$)

The next $N - 1$ line each contains two integers u and v, which means there exists an edge between node u and v. ($1 \le u, v \le N$)

The given graph is guaranteed to be a valid tree.

Output

Output an integer, the time (in hours) required for the bacteria to fully evolve.

Example

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

Problem C
Emptying the Baltic
Problem ID: emptyingbaltic
Time limit: 3 seconds

Picture by Jeremy Halls on Flickr, cc by-sa

Gunnar dislikes forces of nature and always comes up with innovative plans to decrease their influence over him. Even though his previous plan of a giant dome over Stockholm to protect from too much sunlight (as well as rain and snow) has not yet been realized, he is now focusing on preempting the possible effects climate change might have on the Baltic Sea, by the elegant solution of simply removing the Baltic from the equation.

First, Gunnar wants to build a floodbank connecting Denmark and Norway to separate the Baltic from the Atlantic Ocean. The floodbank will also help protect Nordic countries from rising sea levels in the ocean. Next, Gunnar installs a device that can drain the Baltic from the seafloor. The device will drain as much water as needed to the Earth's core where it will disappear forever (because that is how physics works, at least as far as Gunnar is concerned). However, depending on the placement of the device, the entire Baltic might not be completely drained – some pockets of water may remain.

To simplify the problem, Gunnar is approximating the map of the Baltic using a 2-dimensional grid with 1 meter squares. For each square on the grid, he computes the average altitude. Squares with negative altitude are covered by water, squares with non-negative altitude are dry. Altitude is given in meters above the sea level, so the sea level has altitude of exactly 0. He disregards lakes and dry land below the sea level, as these would not change the estimate much anyway.

Water from a square on the grid can flow to any of its 8 neighbours, even if the two squares only share a corner. The map is surrounded by dry land, so water never flows outside of the map. Water respects gravity, so it can only flow closer to the Earth's core – either via the drainage device or to a neighbouring square with a lower water level.

Gunnar is more of an idea person than a programmer, so he has asked for your help to evaluate how much water would be drained for a given placement of the device.

Input

The first line contains two integers h and w, $1 \leq h, w \leq 500$, denoting the height and width of the map.

Then follow h lines, each containing w integers. The first line represents the northernmost row of Gunnar's map. Each integer represents the altitude of a square on the map grid. The altitude is given in meters and it is at least -10^6 and at most 10^6.

The last line contains two integers i and j, $1 \leq i \leq h, 1 \leq j \leq w$, indicating that the draining device is placed in the cell corresponding to the j'th column of the i'th row. You may assume that position (i, j) has negative altitude (i.e., the draining device is not placed on land).

Output

Output one line with one integer – the total volume of sea water drained, in cubic meters.

Sample Input 1	Sample Output 1
3 3 -5 2 -5 -1 -2 -1 5 4 -5 2 2	10

Sample Input 2	Sample Output 2
2 3 -2 -3 -4 -3 -2 -3 2 1	16

Happy Birthday Jason Li!

Problem D
Fleecing the Raffle
Problem ID: raffle
Time limit: 2 seconds

The Raffle (Raffling for the Goose) by William Sidney Mount, public domain

A tremendously exciting raffle is being held, with some tremendously exciting prizes being given out. All you have to do to have a chance of being a winner is to put a piece of paper with your name on it in the raffle box. The lucky winners of the p prizes are decided by drawing p names from the box. When a piece of paper with a name has been drawn it is not put back into the box – each person can win at most one prize.

Naturally, it is against the raffle rules to put your name in the box more than once. However, it is only cheating if you are actually caught, and since not even the raffle organizers want to spend time checking all the names in the box, the only way you can get caught is if your name ends up being drawn for more than one of the prizes. This means that cheating and placing your name more than once can sometimes increase your chances of winning a prize.

You know the number of names in the raffle box placed by other people, and the number of prizes that will be given out. By carefully choosing how many times to add your own name to the box, how large can you make your chances of winning a prize (i.e., the probability that your name is drawn exactly once)?

Input

The input consists of a single line containing two integers n and p ($2 \leq p \leq n \leq 10^6$), where n is the number of names in the raffle box excluding yours, and p is the number of prizes that will be given away.

Output

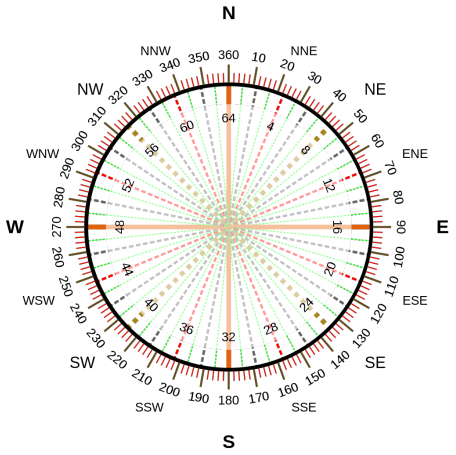
Output a single line containing the maximum possible probability of winning a prize, accurate up to an absolute error of 10^{-6}.

Sample Input 1	Sample Output 1
3 2	0.6

Sample Input 2	Sample Output 2
23 5	0.45049857550

Problem E
Compass Card Sales
Problem ID: compasscard
Time limit: 6 seconds

Katla has recently stopped playing the collectible card game Compass. As you might remember, Compass is a game where each card has a red, a green and a blue angle, each one between 0 and 359, as well as an ID. Since she has stopped playing, Katla has decided to sell all her cards. However, she wants to keep her deck as unique as possible while selling off the cards. Can you help her figure out the order in which she should sell the cards?



Picture via Wikimedia Commons, public domain

To decide how unique a card is in the deck, she proceeds as follows. For each of the three colors she finds the closest other card in both directions, and then computes the angle between these two other cards. For instance if she has three cards with red angles 42, 90 and 110, then the uniqueness values of their red angles are 340, 68, and 312, respectively. If two cards A and B have the same angle, B is considered the closest to A in both directions so that the uniqueness value of A (and B) for that color is 0.

By summing the uniqueness values over the three colours, Katla finds how unique each card is. When selling a card, Katla sells the currently least unique card (smallest uniqueness value). If two cards have the same uniqueness value, she will sell the one with the higher ID first. After each card is sold, the uniqueness values of the remaining cards are updated before selling the next card.

Input

The first line of input contains an integer n, the number of cards ($1 \leq n \leq 10^5$). Then follows n lines. Each of these n lines contains 4 integers r, g, b, id ($0 \leq r, g, b < 360$, $0 \leq \text{id} < 2^{31}$), giving the red, green and blue angles as well as the ID of a card. No two cards have the same ID.

Output

Output n lines, containing the IDs of the cards in the order they are to be sold, from first (least unique) to last (most unique).

Sample Input 1

```
3
42 1 1 1
90 1 1 2
110 1 1 3
```

Sample Output 1

```
2
3
1
```

Sample Input 2

```
4
0 0 0 0
120 120 120 120
240 240 240 240
0 120 240 2017
```

Sample Output 2

```
2017
240
120
0
```

Problem F
Highest Tower
Problem ID: tower
Time limit: 7 seconds

Photo by Matt Schilder on flickr, cc by-sa

Oni loved to build tall towers of blocks. Her parents were not as amused though. They were on the verge of going crazy over that annoying loud noise whenever a tower fell to the ground, not to mention having to pick up blocks from the floor all the time. Oni's mother one day had an idea. Instead of building the tower out of physical blocks, why couldn't Oni construct a picture of a tower using two-dimensional rectangles that she montaged on a board on the wall? Oni's mother cut out rectangles of various sizes and colors, drew a horizontal line representing the ground at the bottom of the board, and explained the rules of the game to Oni: every rectangle must be placed immediately above another rectangle or the ground line. For every rectangle you can choose which of its two orientations to use. I.e., if a rectangle has sides of length s and t, you can either have a side of length s horizontally or a side of length t horizontally. You may place exactly one rectangle immediately above another one if its horizontal side is *strictly* smaller than the horizontal side of the rectangle beneath. Exactly one rectangle must be placed on the ground line. Now try to build as tall a tower as possible!

Oni's mother took extra care to make sure that it was indeed possible to use all rectangles in a tower in order not to discourage Oni. But of course Oni quickly lost interest anyway and returned to her physical blocks. After all, what is the point of building a tower if you cannot feel the suspense before the inevitable collapse? Her father on the other hand got interested by his wife's puzzle as he realized this is not a kids' game.

Input

The first line of input contains an integer n ($1 \leq n \leq 250\,000$), the number of rectangles. Then follow n lines, each containing two integers s and t ($1 \leq s \leq t \leq 10^9$ nm), the dimensions of a rectangle.

You may safely assume that there is a way to build a tower using all n rectangles.

Output

Output a single line containing the height in nm of the tallest possible tower using all the rectangles while having the horizontal side lengths strictly decreasing from bottom to top.

Sample Input 1	Sample Output 1
3 50000 160000 50000 100000 50000 100000	200000

Problem G
Exponial
Problem ID: exponial
Time limit: 1 second

Everybody loves big numbers (if you do not, you might want to stop reading at this point). There are many ways of constructing really big numbers known to humankind, for instance:

- Exponentiation: $42^{2016} = \underbrace{42 \cdot 42 \cdot \ldots \cdot 42}_{2016 \text{ times}}$.
- Factorials: $2016! = 2016 \cdot 2015 \cdot \ldots \cdot 2 \cdot 1$.

Illustration of $\mathrm{exponial}(3)$ (not to scale), Picture by C.M. de Talleyrand-Périgord via Wikimedia Commons

In this problem we look at their lesser-known love-child the *exponial*, which is an operation defined for all positive integers n as

$$\mathrm{exponial}(n) = n^{(n-1)^{(n-2)^{\cdots^{2^1}}}}.$$

For example, $\mathrm{exponial}(1) = 1$ and $\mathrm{exponial}(5) = 5^{4^{3^{2^1}}} \approx 6.206 \cdot 10^{183230}$ which is already pretty big. Note that exponentiation is right-associative: $a^{b^c} = a^{(b^c)}$.

Since the exponials are really big, they can be a bit unwieldy to work with. Therefore we would like you to write a program which computes $\mathrm{exponial}(n) \bmod m$ (the remainder of $\mathrm{exponial}(n)$ when dividing by m).

Input

The input consists of two integers n ($1 \leq n \leq 10^9$) and m ($1 \leq m \leq 10^9$).

Output

Output a single integer, the value of $\mathrm{exponial}(n) \bmod m$.

Sample Input 1	Sample Output 1
2 42	2

Sample Input 2	Sample Output 2
5 123456789	16317634

Sample Input 3	Sample Output 3
94 265	39