IDI Open Programming Contest March 19th, 2022

Problem Set

A Another Ancient Cipher
B BBBBBB
C Changing Complexity
D Delicious Diet for Ducks
E Eyeballing Extraterrestials
F Frugal Ferry Fees
G Game Party (Easy)
H Healthy Headgear
K Kattis Completionist (Easy)
L Laser-linked Lighthouses

Jury and Problem Writers

Nils Barlaug
Sondre Sortland
Jean Niklas L'orange (Head Judge)

Rules

TL;DR: Teams of up to three persons try to solve as many problems as possible from a set, without external help.

Before the contest begins, you are allowed to log in on your assigned computer, and log in on the submission system. You may do nothing else with the computer (such as starting to write code). You may not touch the problem set before the contest has started.

Contestants are *only* allowed to communicate with members of their own team, and the organisers of the contest. You may not surf the web (except for allowed content), read e-mail, chat on Slack, or similar things. The only network traffic you may generate is from submitting problem solutions, and access to content specified by the local organisers.

- What you may bring to the contest floor:
  - Any written material (Books, manuals, handwritten notes, printed notes, etc).
  - Pens, pencils, blank paper, stapler and other useful non-electronic office equipment.
  - NO material in electronic form (CDs, USB pen and so on).
  - NO electronic devices (Cellular phone, PDA and so on).
- What you may use during the contest:
  - What you brought to the contest floor (see above).
  - Your assigned (single) computer.
  - The specified system for submitting solutions: `https://idio22.kattis.com`
  - Printers designated by the organiser.
  - The external documentation for your language of choice. This is documented in the tutorial for your language at `https://idio22.kattis.com/help`
  - All compilers and IDEs pre-installed on your assigned computer
  - Non-programmable tools which are a natural part of the working environment (such as `diff` and `less`).

The problem set consists of a number of problems (usually 8-12). The problem set will be in English, and given to the participating teams when the contest begins. For each of these problems, you are to write a program supported by the Kattis system. The jury guarantees that each problem is solvable in C, C++, Java and Python 3. No guarantees for other languages are given due to the large number of allowed languages, however the jury guarantees that for every language there is at least one problem solvable in that language. It has always been the case that several of the problems were solvable in all available languages, but there is no guarantee of this.

Your programs should read from standard in and write to standard out. See `https://idio22.kattis.com/help` for a tutorial and the compiler options for your language of choice.

The team that solves the most problems correctly wins. If two teams solve the same number of problems, the one with the lowest total time wins. If two top teams end up with the same number of problems solved and the same total time, then the team with the lowest time on a single problem is ranked higher. If two teams solve the same number of problems, with the same total time, and the same time on all problems, it is a draw. The time for a given problem is the time from the beginning of the contest to the time when the first correct solution was submitted, plus 20 minutes for each incorrect submission of that problem. The total time is the sum of the times for all solved problems, meaning you will not get extra time for a problem you never submit a correct solution to.

If you think some problem is ambiguous or underspecified, you may ask the judges for a clarification request through the Kattis system. The most likely response is "No comment, read problem statement", indicating that the answer can be deduced by carefully reading the problem statement or by checking the sample test cases given in the problem, or that the answer to the question is simply irrelevant to solving the problem.

Input Validation

In general we are lenient with small formatting errors in the output, in particular whitespace errors within reason. But not printing any spaces at all (e.g. missing the space in the string "`1 2`" so that it becomes "`12`") is typically not accepted. The safest way to get accepted is to follow the output format exactly.

For problems with floating point output, we only require that your output is correct up to some error tolerance. For example, if the problem requires the output to be within either absolute or relative error of 10^{-4}, this means that

- If the correct answer is 0.05, any answer between 0.0499 and .0501 will be accepted.
- If the correct answer is 500, any answer between 499.95 and 500.05 will be accepted.

Any reasonable format for floating point numbers is acceptable. For instance, "`17.000000`", "`0.17e2`", and "`17`" are all acceptable ways of formatting the number 17. For the definition of reasonable, please use your common sense.

Tips

- Tear the problem set apart and share the problems among you.
- Problems are **not** sorted by difficulty.
- Try solving the easy problems first. Two problems in this set are tagged with "(Easy)" to help point you in the right direction.
- If your solution fails on a problem, you can print your program and debug it on paper while you let someone else work on a different problem on the computer.
- All problems are guaranteed to be solvable in C, C++, Java and Python 3
- Look at the scoreboard if you are unsure which problem to work on next.

Another Ancient Cipher

Problem ID: anotherancientcipher

You have probably heard about the Caesar cipher: It is one of the most well-known ciphers out there. In an alternative timeline, the Roman emperor Caesar Augustus builds upon this cipher to create an encryption even harder to break: The Augustus cipher.

In the Caesar cipher, you shift each letter by a certain number of digits forwards or backwards, and start over at the other side if you roll over. For the English alphabet, the formulas for encrypting and decrypting the Caesar cipher are

$$E_{Caesar}(M_i, n) = (M_i + n) \mod 26$$
$$D_{Caesar}(C_i, n) = (C_i - n) \mod 26$$

Where M_i and C_i is the ordinal number of i^{th} letter in the message and the ciphertext, respectively, and the number n is the integer to shift by. (The ordinal number of a letter is defined as $\texttt{a} \to 0, \texttt{b} \to 1, \ldots, \texttt{z} \to 25$ for the Latin alphabet.)

The Augustus cipher goes above and beyond, and uses a *key* to encrypt messages. The i^{th} letter of the message is defined as

$$E_{Augustus}(M_i, K_i) = \begin{cases} E_{Caesar}(M_i, K_i) & \text{if } M_i \text{ is even} \\ E_{Caesar}(M_i, -K_i) & \text{if } M_i \text{ is odd} \end{cases}$$

If there are more letters in the message than in the key, the key is repeated. For example, the 4^{th} 0-indexed letter of the key `key` is `k`, the 5^{th} is `e` and so on.

Adam has decided to use this cipher for his new social media platform CyberLounge, and has implemented code to encrypt the messages with the Augustus cipher. But Adam hasn't yet found a way to decrypt the messages! Could you help him with the decryption algorithm?

Input

The input consists of two lines. The first line contains the encrypted string C, and the second line contains the key K. Both strings contain only lowercase characters from the Latin alphabet (`a-z`).

Output

Output the decrypted message.

Limits

- $0 < |C|, |K| \leq 200$

Sample Input 1

```
ccvpvygcoyjc
cryptography
```

Sample Output 1

```
attackatonce
```

Sample Input 2

```
tnkkmbwctzhdjesfqugjmgcyzlpxeeyclhg
augustusisthebest
```

Sample Output 2

```
thequickbrownfoxjumpsoverthelazydog
```

Sample Input 3

```
unnmzaxunlzjdzfnnejdx
z
```

Sample Output 3

```
toolazytomakeagoodkey
```

BBBBBB
Problem ID: bbbbbb

Our favourite game developer Saskia has started on a new game, called BBBBBB! In this game, you're Captain Biryan and have to save your crew. They have been scattered around in an alternate dimension named bbbbbb.

The game consists of a number of puzzles in a 2D grid, where you are the `@` symbol and have to get to the `X`. `#` represents immovable blocks, and `^` and `v` are deadly spikes. The player, playing Captain Biryan, can do one of three actions:

- Move one tile right (if the tile does not contain a `#`)
- Move one tile left (if the tile does not contain a `#`)
- Flip gravity: If gravity is pulling down, it is now pulling you up instead (and vice versa)

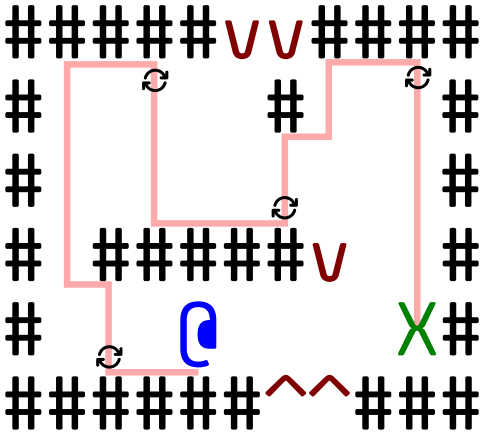

Figure 1: Sample Input 1, and one of the ways to the X with the least number of gravity flips.

The game logic is rather short and can be summarised by this pseudocode:

```
gravity_direction = down
while true:
  if player is at the location of the 'X':
    return WIN
  if player is at the location of a '^' or 'v':
    return DEAD
  if player does not touch the ground:
    player_y += 1 * gravity_direction
    continue (restart loop from the top)
  if player touches the ground:
    perform_next_player_action()
```

Touching the ground is defined as being placed exactly one tile above or below a `#`, depending on which way gravity pulls.

To ensure that there's a good variety of puzzles in the game, Saskia wants to implement a shortest path algorithm that finds the minimum number of gravity flips needed to solve a puzzle.

Input

The first line consists of two integers W and H, representing the width and the height of the 2D grid to be parsed. Then follows H lines each consisting of a string of length W. Each string can only contain the characters ' ', '`#`', '`v`', '`^`', '`@`' and '`X`'.

Output

Output the minimum number of gravity flips needed to solve the puzzle, along with the number of user actions needed. If there are multiple solutions with the same number of gravity flips, output the one with the least amount of user actions needed.
If there is no valid path to the exit, output '`impossible!`' (without the quotes).

Limits

- $4 \leq W, H \leq 450$
- The border of the map can only contain '`#`', '`v`' and '`^`' tiles.
- The grid will only contain the characters' ', '`#`', '`v`', '`^`', `@` and `X`, and there will be exactly one `@` and one `X` symbol in the grid.

Sample Input 1

```
11 6
#####vv####
#     #   #
#         #
# #####v  #
#   @    X#
######^^###
```

Sample Output 1

```
4 15
```

Sample Input 2

```
10 4
#^^^^^^^^#
#         #
#@       X#
##vvvvvv##
```

Sample Output 2

```
impossible!
```

Changing Complexity

Problem ID: changingcomplexity

Creating star ship software modules and components isn't easy, especially when the engineering and operations staff on those ships think it's easy to maintain backwards compatibility. "Remodulate the deflector emitters"? "Reverse the polarity of the shield matrix"? The shield matrix was removed from the deflector shields 3 versions ago! We gave them ample time to migrate their procedures and training to the newer quantum computers that now does the graviton flux analysis. We even said we could help out with documentation and resolving common procedures, but it turns out we just can't teach an old dog new tricks...

To maintain the old ways of doing it, we've implemented a shield matrix translation layer so that they can use the same interface as they've always used. However, with the newer quantum computers changing their internal logic, we've been forced to rewrite the translation layer completely!

That being said, not everyone's angry about maintaining a legacy interface. Our newest hire, Glaucia, actually seems to love maintenance and backwards compatibility work. And she's very eager to get this to work.

While the rest of the team has taken a vacation at Sikaris, Glaucia wanted to take a stab at the rewrite project. Fortunately, the team's been good at mapping up all the different tasks required to complete the rewrite, and has assigned all tasks their complexity and dependencies.

None of the rewrite tasks are too challenging for Glaucia. However, some of the tasks are quite boring. To avoid too much boredom at the same time, Glaucia has decided to complete the different tasks in alternating difficulty: They will first pick the least complex task of all the tasks they can perform, then pick the most complex task they can perform. They repeat this pattern until all tasks have been completed.

Glaucia can of course pick any task they want to, as long as it's not been completed already, or some of its requirements have not yet been completed.

Of course, Glaucia managed to complete the entire rewrite project before the rest of the team came back from Sikaris. Unfortunately, the git history is a bit of a mess, and although it is not essential for anything, Glaucia's manager wants to know which order they performed the tasks. Can you help their manager reconstruct which order Glaucia performed the tasks?

Input

The first line contains two integers, V and E, the number of tasks and the number of different requirements, respectively. The complexity of a task is given by its ID: Task 1 has 1 complexity, task 2 has 2 complexity, and so on.

Then follow E lines, each representing a requirement. Each line contains two integers, u_i and v_i, indicating that task v_i can only be completed if u_i has been completed.

As the input may be very large, you should consider buffering it.

Output

Output

$$\sum_{i=1}^{V} i \times O_i$$

where O_i is the complexity of the i^{th} task that was completed by Glaucia.

Limits

- $1 \leq V \leq 200\,000$
- $0 \leq E \leq \min(400\,000, \frac{V(V-1)}{2})$
- $0 < u_i, v_i \leq V$
- For all $i \neq j$: If $u_i = u_j$, then $v_i \neq v_j$
- The tasks form a forest of DAGs.

Sample Input 1

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

Sample Output 1

```
114
```

Sample Input 2

```
6 4
1 5
1 6
3 4
6 2
```

Sample Output 2

```
78
```

Sample Input 3

```
200 0
```

Sample Output 3

```
2025050
```

Delicious Diet for Ducks

Problem ID: deliciousdietforducks

Photo by Michael Anfang

Duckies! Valarie loves them so much that she got her own pet duck, which she has called Quackie Chan. Of course, as a mathematician, she wants to optimise Quackie's happiness: everything from the duck pond, playtime and food choices. But she hasn't picked out the optimal duck treat yet.

Valarie has done some due diligence, however: She has found 3 shops selling duck treats nearby, and all of them provide a wide variety of treats. In fact, they are so unique that all the different treats a shop sells aren't provided by any other shop.

To figure out which of the treats are the best, Valarie is going to let Quackie decide between two different treats every day. The one Quackie prefers is the winner and will be compared with another the next day, until all have been evaluated.

Valarie has given every treat a score from 1 to 100, as part of the due diligence. A score of 1 means she thinks Quackie won't like it at all, and a score of 100 means she thinks Quackie will love it.

Because she doesn't want Quackie to evaluate the treats by ascending or descending score, she has planned to do the following semi-random method of picking the new treat to evaluate against:

She first picks the lowest scoring treat that Quackie hasn't yet evaluated from each shop. Then she will pick the treat from store S_x with probability

$$\frac{S_x}{S_A + S_B + S_C}$$

where S_A is the score for the treat from the first shop, S_B for the second shop, and S_C for the last shop.

Of course, this method will break down if all treats from one shop have been evaluated! Valarie knows how to handle that, but she is curious which of the shops will have all their treats evaluated first. Assuming she already has a treat to compare with, can you help her?

Input

The first line contains an integer A, the number of treats in the first shop. Then follows a single line with A integers a_i, representing the score of all the unique treats in that shop. This pattern repeats for the remaining two shops, where B/b_j is the second shop, and C/c_k is the last shop.

Output

Output three numbers on three lines: The probability that the first shop will have their treats evaluated first, the probability that the second shop will have their treats evaluated first, and the probability that the last shop will have their treats evaluated first.

All numbers must have a relative or absolute error of at most 10^{-6}.

Limits

- $0 < A, B, C \leq 250$
- $0 < a_i, b_j, c_k \leq 100$, for $0 \leq i < A$, $0 \leq j < B$, $0 \leq k < C$

Sample Input 1	Sample Output 1
1 1 1 2 1 3	0.16666666666666666 0.33333333333333333 0.5

Sample Input 2	Sample Output 2
2 1 1 4 1 5 5 5 3 1 2 2	0.31676683165958763 0.42124927454905214 0.26198389379136006

Eyeballing Extraterrestials

Problem ID: eyeballingextraterrestials

Nevada has been working hard to incentivise companies to move their offices there, both by active forest restoration and tax incentives. And it has worked! Companies moved their main offices there, and people soon followed suit.

Most families decided to buy houses along the Nevada state route 375 (SR 375), which for our purposes can be considered a long non-branching road. This previously desolate highway is now the location of N houses, some possibly vacant, all on the same side of the road. All the F families that live here have a lot of money. So much money, in fact, that they all have a vacation house as well. Because they all now love Nevada so much, they have obviously decided to have their vacation house along the SR 375 as well. The home of family i is located at $loc_{H,i}$ and their vacation house is located at $loc_{V,i}$.

Lately, a lot of explosions and alien sounds have come from Area 51. Although this family neighbourhood isn't too religious about the entire alien thing, they are all a bit cautious. What if there actually are aliens escaping, and what if they manage to kidnap and steal the identities of one of the families? Or maybe they will just occupy a vacant house without us noticing?

"The Surveillance Camera Company" (SCC) has for that reason seen great potential for selling a lot of cameras to this recently rich suburban area. They have managed to encourage all the families to "observe" and "take care of" their neighbours by watching them through surveillance cameras.

More specifically, they've gone to family $i = 1$ and asked what kind of camera package they want: They can either pick one that will cover the nearest H_i houses near their home (excluding their own), or one that covers the nearest V_i houses near their vacation home (again excluding their own). Every family always picks one package, the one for the house with the least cameras already placed near their neighbour's houses *on average*. If there is a tie, they prefer to have the cameras near their homes.

Mathematically speaking, if $h_{i,j}$ is the jth closest house to family i's home, and $v_{i,j}$ is the jth closest house to their vacation house, then they will pick the home package if and only if

$$\frac{\sum_{j=1}^{H_i} h_{i,j}}{H_i} \leq \frac{\sum_{j=1}^{V_i} v_{i,j}}{V_i}$$

Of course, all M_i family members want their own surveillance camera to watch on, so they buy and place M_i cameras for each neighbouring house they want to "observe". Then, the surveillance camera seller continues on to the next family $(i + 1)$ until all the families have bought a camera set.

A vague, yet menacing government agency has heard about the sales of these cameras. They wonder if the security's tight enough that they need to add in additional measures, or if the neighbourhood "watch" is sufficient enough. For that reason, they need to detect whether there are enough cameras near a range of continuous houses, and they need your help.

Input

The first line contains two integers, N and F: The number of houses and the number of families.

Then follow F lines, one for each family. Each line contains 5 integers: M_i, $loc_{H,i}$, H_i, $loc_{V,i}$ and V_i, as described in the problem statement above.

Finally, a line with an integer Q follows, representing the number of queries the vague, yet menacing government agency wants to perform. Then Q lines follow, each containing two integers $Q_{start,i}$ and $Q_{end,i}$, representing the query "how many cameras are there in the house range $[Q_{start,i}, Q_{end,i}]$?".

Output

For each query, print out the total number of cameras in that particular house range after all cameras have been installed.

Limits

- $2 < N \leq 200\,000$
- $0 < F < 100\,000$

- $0 < loc_{H,i}, loc_{V,i} \leq N$
- $0 < H_i, V_i < N$, and H_i and V_i are divisible by 2
- $0 < M_i < 10$
- $0 < Q < 20\,000$
- $0 < Q_{start,i} \leq Q_{end,i} \leq N$
- Families can live with other families in the same house. They can even live at someone's vacation house, and can share a vacation house with another family.

Sample Explanation

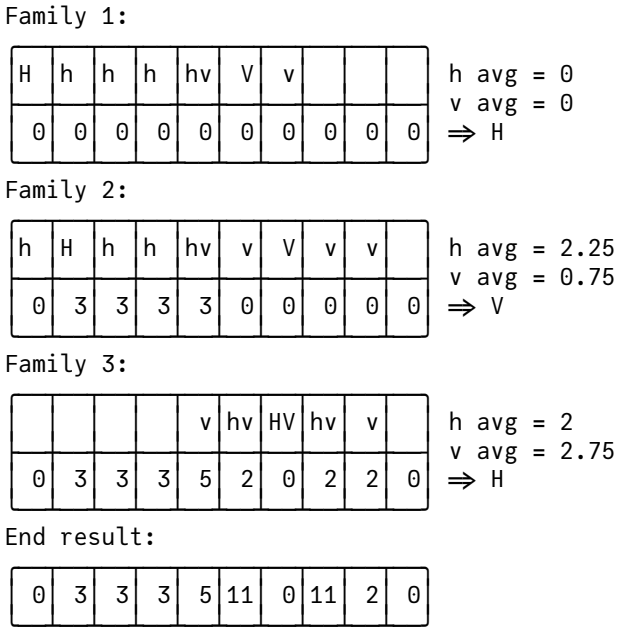

Figure 1: Each step of the camera installation in Sample Input 1. `H` is the family's home, `h` is their closest neighbours, `V` is their vacation home, and `v` is their closest vacation home neighbours

Sample Input 1	**Sample Output 1**
10 3 3 1 4 6 2 2 2 4 7 4 9 7 2 7 4 4 1 3 4 6 7 7 6 10	6 19 0 24

Frugal Ferry Fees

Problem ID: frugalferryfees

Here at Mercury Merchants, we do not only sell grain, leather and high-quality wood, we also deliver it straight to our customer's door! And if we don't have it in stock, we simply send it by horse and cart from the nearest town.

Of course, getting the goods delivered over water is an entirely different matter, as we don't own any ships ourselves. In order to save costs for ferry fees, we happily formed an agreement with Charon Corporation some time ago. The prices we get for ferry trips are amazing, although our couriers are a bit creeped out by the ferrymen: They are all slender, tall men in black hoods, and all of them are named Charon. Oh well – some sleepless nights are warranted in the pursuit of profit and increased margins.

Figure 1: Map over Sample Input 1

However, Charon Corporation has had to raise their prices quite significantly since the black plague due to "abnormal demand for other services we provide", according to their sales representative (also named Charon). While the prices are still much better than Charon Corporation's competitors, our bottom line will turn red if we don't optimise the ferry fees.

The solution to our problems is quite simple: While our couriers in the past took the shortest path from A to B, we'll now order them to take the route which minimises ferry costs instead.

Input

The first line contains two integers, V and E, the number of cities and the number of cart routes between them. Then follow E lines, describing the cart routes. Each line contains two integers $town1_i$ and $town2_i$, describing an undirected cart route between the two towns.

Next follows a single line containing an integer F, representing the number of ferry lines Charon Corporation has set up. F lines follow, all with three integers each: $town1_i$, $town2_i$ and $cost_i$, where the two first integers denote the towns the ferry go via, and $cost_i$, which is the number of oblates you have to pay to Charon Corporation for the trip.

Finally a line with an integer Q follows, representing the number of trips you have to perform. Q lines follow, and each contains two integers $town1_i$ and $town2_i$, describing that you have to travel from town 1 to town 2.

As the input may be very large, you should consider buffering it.

Output

Output the minimum number of oblates needed to perform all travels.

Limits

- $1 < V < 200\,000$, $0 \leq E < 500\,000$
- $0 < F < 300$
- $0 < Q < 500\,000$
- $0 \leq town1_i, town2_i < V$, $town1_i \neq town2_i$
- $0 < cost_i < 1\,000$
- There exists at least one path from any town to any other town.

Sample Input 1

```
12 12
0 2
0 3
2 3
3 4
3 5
6 7
6 8
6 10
7 8
7 10
8 9
8 10
6
0 1 7
0 10 5
1 2 4
5 6 1
5 11 9
9 11 2
7
1 11
5 8
4 1
4 10
9 11
2 4
3 8
```

Sample Output 1

```
16
```

Game Party

Problem ID: gameparty

It's game night and you're inviting all your friends over. To play the game you have to divide yourself and all your friends into teams of three. You're nervous that it won't be possible to get everyone a full team, so you count everyone at your door as they arrive in groups of different sizes. Find out how many will have to sit and watch because they can't form a full team.

Input

The first line consists of one integer G, the number of groups arriving. Next follows one line with G integers S_i, each denoting the size of the arriving group.

Output

Output the number of people that are not able to join a full team.

Limits

- $0 \leq G \leq 100\,000$
- $0 < S_i \leq 1\,000$

Sample Input 1

```
3
2 3 3
```

Sample Output 1

```
0
```

Sample Input 2

```
2
1 3
```

Sample Output 2

```
2
```

Healthy Headgear

Problem ID: healthyheadgear

You have decided to start your own face mask factory to produce face masks with custom prints. The orders are in already even though you haven't set up the factory for production yet. The machines needed for the production are expensive and you want to buy as few as possible while still being able to fulfill all orders within the agreed delivery deadline. Modern manufacturing technology has come a long way and the production machines can operate fully automatic all day long without any human interaction or maintenance.

All orders ask for a certain number of face masks with the same unique print. The machines have a start-up time for adjusting to a new print and make 10 at the same time. So the time it takes to produce 1 face mask is the same as 10 (a machine can't produce different prints at the same time). One can formulate the time in seconds it takes to produce one order of n face masks as:

$$t(n) = 20 + 10 * \left\lceil \frac{n}{10} \right\rceil \tag{1}$$

To be fair to your customers, you want to start production of the orders in the same order as you got them. Luckily, all customers have agreed to the same delivery deadline in their contract.

You have also decided that the orders won't be split up and ran on multiple machines, as you would have to repackage the output from the machines. Right now, there is no machine that can do such a job, and it's just too time consuming to do it with humans.

What is the smallest number of machines that would make it possible to finish all orders in time?

Input

The first line consists of two integers T and N, the total time you have to produce all the face masks, and the number of orders of face masks respectively. Next follows one line with N integers n_i, each denoting the size of a batch.

Output

Output the smallest number of machines you have to invest in in order to complete all batches within the total time T. If it is not possible to complete all orders within the deadline output "impossible".

Limits

- $0 < T \leq 1\,000\,000$
- $0 \leq N \leq 1\,000\,000$
- $0 < n_i \leq 10\,000$

Sample Input 1

```
100 3
1 1 1
```

Sample Output 1

```
1
```

Sample Input 2

```
60 3
3 13 11
```

Sample Output 2

```
3
```

Kattis Completionist

Problem ID: kattiscompletionist

Katryna has one goal in her life, and that is to solve all the problems published on Kattis. Of course, she can't do them all at once, as she still has to work on her computer science degree. Also, attempting to solve the most difficult problems early on could hurt her motivation and persistence.

For that reason, she has concocted a plan. Every Kattis problem is assigned a difficulty score s_i. That score can go up and down over time, depending on how difficult Kattis thinks the problem is. Katryna wants to solve so many problems that her score increases by at least G every day, but she wants to solve the easiest problems first.

So Katryna will, at the start of every day, pick the unsolved problem with the lowest difficulty score and solve it (if there are multiple with the lowest difficulty score, she will pick one of them at random). If her score has increased by at least G today, she stops. But if not, she will again pick and solve the unsolved problem with the lowest difficulty score, until either there are no more problems left to solve, or until her score has increased by at least G points today. She repeats this every day until she has solved all the problems.

Assuming all the problems' difficulty scores does not change, how many days would it take Katryna to solve every Kattis problem?

Input

The first line consists of the integer N and the real number G, representing the total number of unsolved Kattis problems, and the lowest score increase Katryna is happy with in a single day.

Then follows a single line with N real numbers, each separated by a space, denoting all the difficulty scores s_i of the Kattis problems.

Output

Output the number of days Katryna will use to solve all her currently unsolved Kattis problems.

Limits

- $0 < N \leq 100\,000$
- $1.0 \leq G \leq 99.9$
- $1.3 \leq s_i \leq 9.6$
- All real numbers will have exactly one digit after the decimal point.

Sample Input 1

```
5 3.0
1.3 1.7 1.7 1.7 3.1
```

Sample Output 1

```
3
```

Sample Input 2

```
3 2.1
2.0 2.0 2.0
```

Sample Output 2

```
2
```

Sample Input 3

```
3 3.6
1.3 2.3 2.3
```

Sample Output 3

```
2
```

Laser-Linked Lighthouses

Problem ID: laserlinkedlighthouses

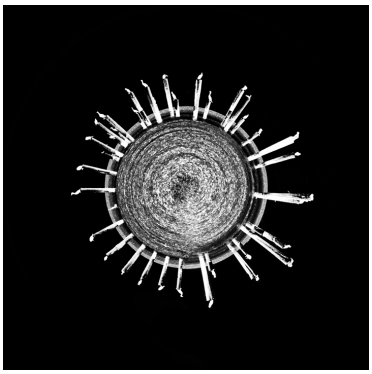

Photo by Bruce Berrien

After centuries of trying, humankind has finally managed to establish N colonies on Europa. Since they are so far away from the other humans in the Solar System, a bunch of satellite repeaters propagate communications to and from the other colonies.

However, critical nodes in the satellite repeater system and the backup solution has had several hiccups, causing the Europa colonies to be without connection to the outside world for several weeks. Although it now has been fixed, the Europa council has decided that it's necessary to have a failsafe, and want to construct a bigger and more sensitive satellite dish. However, they can only afford one big satellite dish, so it has to be put near one of the colonies.

That poses another problem: The colonies on Europa currently communicate with each other via the satellite repeater system (yes, the ping is horrendous), and when the system is down, they have to send vehicles instead. To relay the information from the colony with the big satellite dish and in general, they have to set up a communications system between all the colonies. The plan is to set up communication cables eventually, but in the meantime, they have decided to set up a quick and dirty solution: laser lighthouses.

Laser lighthouses are simple things: they can be considered cylinders with an infinitesimally small radius and some height h, standing perpendicular to the surface of a planet or moon. As long as the top of a lighthouse has a clear line of sight to the top of another lighthouse, they can directly exchange information. On Europa, that means that the two lighthouses must be high enough that the curvature of Europa's surface doesn't interfere with them:

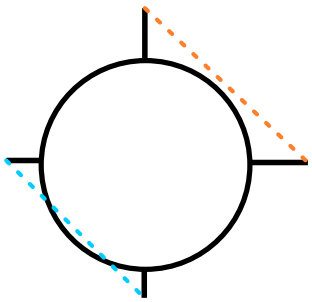

Figure 1: A cut of Europa, where all 4 lighthouses lie exactly on the cut. The lighthouses in the upper right quadrant are able to communicate because they have a line of sight, whereas the lighthouses in the bottom right quadrant do not.

Fortunately, Europa can be considered a perfect sphere, and its gravity is so weak that spacetime is not warped nearby.

The lighthouses can also communicate indirectly: If lighthouse i wants to send information to lighthouse j, it can transmit that information via a sequence of lighthouses with a clear line of sight to each other.

The council has already decided where all the lighthouses' should be placed, but are unsure how high they have to be. To save costs, they want to prefabricate all lighthouses with the same size, and they want to use the minimum height possible. Can you help them?

Input

The first line contains the integer N, the number of laser lighthouses. Then follows N lines, each with two integers lat_i and lon_i, representing the coordinates of the ith lighthouse in degrees.

Output

Output the minimum height h the laser lighthouse needs to be, so that any lighthouse a can communicate with any other lighthouse b directly or indirectly. Output the answer in multiples of Europa's radius.

The output must have a relative or absolute error of at most 10^{-8}.

Limits

- $2 < N < 1000$
- $-90 \leq \text{lat}_i \leq 90$ and $-180 < \text{lon}_i \leq 180$
- $(\text{lat}_i, \text{lon}_i) \neq (\text{lat}_j, \text{lon}_j)$ for all $i \neq j$
- If $\text{lat}_i = \pm 90$, then $\text{lon}_i = 0$

Sample Input 1

```
3
90 0
0 -90
0 0
```

Sample Output 1

```
0.41421356237309515
```

Sample Input 2

```
5
60 11
-33 18
38 -122
-38 145
40 116
```

Sample Output 2

```
0.45514138533258275
```