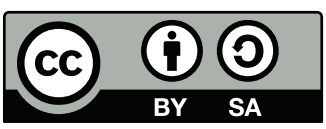
BAPC 2026

The 2026 Benelux Algorithm Programming Contest



Problems

A	Alternating Branch Cutting
B	Balloons and Dubloons
C	Circadian Rhythm
D	Discception
E	Endurtaka Mig
F	Flippin' Cards
G	Graph Theft Auto
H	Historical Hierarchy
I	Irritating Interruption II
J	Juxtaposort
K	Kaktus Kourse
L	Letter Lottery
M	Memory Management



Copyright © 2026 by The BAPC 2026 Jury. This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.
https://creativecommons.org/licenses/by-sa/4.0/

A Alternating Branch Cutting

Time limit: 1s

As you might know, gardening trends are constantly subject to change. Not too long ago, everyone wanted to decorate their garden with an immense, lush ternary tree. Nowadays, it is however more common for homeowners to want only a simple, slender unary tree. Alice and Emmy saw a clear business opportunity here, so they founded the Botanical Alternative Pruning Cooperative. For a fair price, they will turn any ternary tree into a unary tree in no time!

Currently, they are travelling to a customer who has a complete ternary tree of height n: there is a root at the bottom layer, and each vertex has 3 neighbouring children in the next layer, continuing until layer n, which consists of 3^n leaves. The leaves come in two varieties: some of the leaves are green, while others are teal. From experience, Alice and Emmy know that every leaf is green with probability $0.5 + p$ and teal with probability $0.5 - p$. The colours of different leaves are independent from each other.

To prune the tree, they will start at the root vertex. Alice will cut off one of the three branches extending from this vertex, then Emmy will cut one of the two other branches. Then they both climb up along the remaining branch to the next branching vertex and repeat this process n times, until they reach a leaf. Each time, Alice gets to cut the first branch, and Emmy cuts the second branch.

However, there is some disagreement on which branches to cut: Alice believes that they should do the cutting in such a way that the remaining leaf is green, while Emmy thinks that the remaining leaf should be teal. Thus, they will both cut branches in such a way to try to get the final remaining leaf to be of their preferred colour.

Assuming Alice and Emmy make optimal cuts, what is the probability that the final leaf will be green? Note that both Alice and Emmy will be able to see all the colours of the leaves before they start cutting the tree.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 10^6$), the height of the tree.
- One line with a floating point number p ($-0.5 < p < 0.5$, $p \neq 0$), where the probability of any leaf being green is $0.5 + p$. p is given in scientific notation, consisting of the following three parts (which are not separated by any whitespace):
  - a floating point number m ($1 \leq m < 10$ or $-10 < m \leq -1$) given with at most 10 digits after the decimal point, the mantissa,
  - the character 'e', which is shorthand for "$\cdot 10\hat{}$" (times ten to the power of),
  - an integer e ($-10^7 \leq e \leq -1$), the exponent.

Output

Output the probability that the final remaining leaf is green.

Your answer should have an *absolute* error of at most 10^{-6}.

Sample Input 1	Sample Output 1
1 -4e-1	0.028

Sample Input 2	Sample Output 2
10 -2.10e-2	1.5245881e-3

Sample Input 3	Sample Output 3
256 1.2345e-46	0.6448034868

Sample Input 4	Sample Output 4
2 7.7777777777e-10000000	0.5

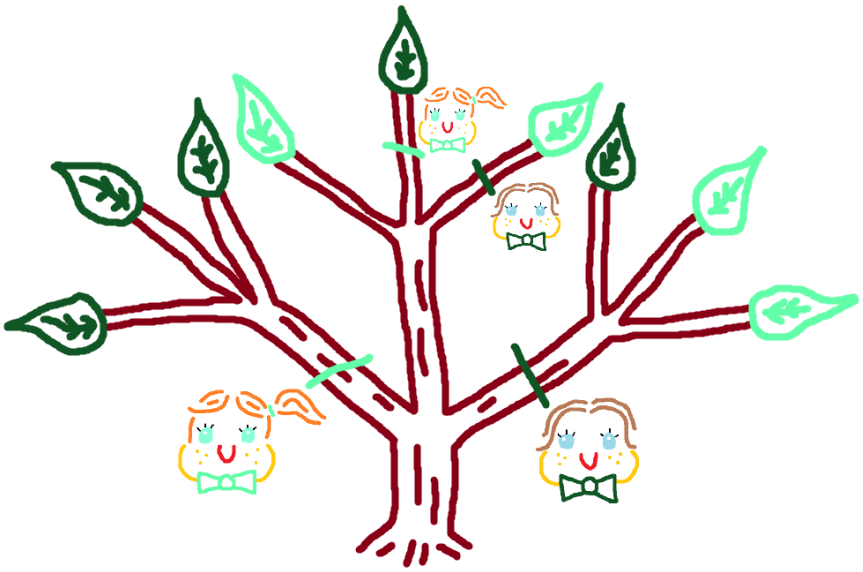


Figure A.1: An example of a tree of height 2, like in Sample Input 4. In this instance of a tree, the final leaf is teal.

B Balloons and Dubloons

Time limit: 2s

Every year, the Be A Pirate Competition is held. This year, Captain Redbeard has become part of the jury. While he does enjoy being a part of the jury, it also means he cannot participate, and thus cannot win the coffer of gold dubloons reserved for the winner.



Blackbeard, Brownbeard, and Redbeard as team *Segfault go YARRRR*, before Redbeard joined the jury.

Now he could, like many of his fellow jury members, pocket a few gold coins every once in a while. But Captain Redbeard has a better plan: his friends (Captain Brownbeard and Captain Blackbeard) are still participating this year, and they agreed to share some of the bounty if he helped them win.

The next task in the competition is finding a treasure on an island. Participants are given a map on which the treasure is marked with a big '`X`'. However, to make things more interesting, there are a number of false '`X`'s to lead pirates astray. Captain Redbeard would like to somehow indicate which treasure marker leads to the actual treasure, but he has no opportunity to directly tell his friends.

But he has a plan! Each jury member will add a single false '`X`' to the map, and he made sure to be last in line. By cleverly placing this final '`X`', he can communicate the location of the real treasure to his friends through a previously established protocol.

If you help Captain Redbeard design and implement such an algorithm, he will award you with one large gold ~~dubloon~~ balloon.

Input

This is a *multi-pass* problem. Your program will be invoked two times.

For each pass, the input consists of:

- One line with a single string, indicating which role your program takes:
  - "`captain`" if you are Captain Redbeard and need to place an additional '`X`'.
  - "`matey`" if you are one of Captain Redbeard's accomplices and need to determine the location of the actual treasure.
- Regardless of the role, one line with two integers h and w ($1 \leq h, w \leq 1000$), the height and width of the treasure map.
- Next follow h lines, each with a string consisting of w characters, describing the treasure map:
  - A character '`~`' indicates water.
  - A character '`.`' indicates unmarked land.
  - A character '`X`' indicates a treasure marker.
  - A character '`#`' indicates a treasure marker pointing to the actual treasure.

If your role is "`captain`", the treasure map contains precisely one '`#`', and it is guaranteed that $n > k^2$ where n is the number of unmarked land tiles, and k is the total number of treasure markers, including the one pointing to the actual treasure.

If your role is "`matey`", the treasure map is the same map that you received as "`captain`", but with the '`#`' replaced by '`X`' and an additional '`X`' placed at the coordinates output by your program as "`captain`".

A testing tool is provided to help you develop your solution.

Output

If your role is "`captain`", output the treasure map from the input, with exactly one piece of unmarked land ('`.`') replaced by an additional treasure marker ('`X`').

If your role is "`matey`", output the treasure map from the input, with exactly one treasure marker ('`X`') replaced by the character '`#`', pointing to the actual treasure.

Sample Case 1

Input	**Pass 1**	**Output**
captain 2 3 ~~. ~.#	~~. ~X#	

Input	**Pass 2**	**Output**
matey 2 3 ~~. ~XX	~~. ~X#	

Sample Case 2

Input	**Pass 1**	**Output**
captain 4 5 ~~.X~ .X..~ ..#.. ~..~~	~~.X~ .XX.~ ..#.. ~..~~	

Input	**Pass 2**	**Output**
matey 4 5 ~~.X~ .XX.~ ..X.. ~..~~	~~.X~ .XX.~ ..#.. ~..~~	

C Circadian Rhythm

Time limit: 5s

You would like to make a schedule for your weekend. The weekend consists of two days, Saturday and Sunday, of $24 \cdot 60 \cdot 60 \cdot 1000 = 86\,400\,000$ milliseconds each. You have a list of n tasks for Saturday and m tasks for Sunday, and your primary objective is to complete all tasks.

Every task has a start time s, a deadline e, and a duration d, each given in milliseconds. At any moment in time, you are only able to work on a single task. However, you may distribute the work of any task over several periods of time, so long as the total time you work on any task is equal to its duration.

At any moment you are not working on any task, you will sleep. For reasons having to do with your circadian rhythm, you would like to synchronize your sleep schedules on Saturday and Sunday as much as possible. Your secondary objective is thus to maximize the total time overlap of scheduled sleep between Saturday and Sunday.

Input

The input consists of:

- One line with two integers n and m ($n, m \geq 1$, $n + m \leq 10^5$), the number of tasks you need to complete on Saturday and Sunday respectively.
- $n + m$ lines, each with three integers s, e, and d ($0 \leq s < e \leq 86\,400\,000$, $1 \leq d \leq e - s$), the start time, deadline, and duration respectively of each task in milliseconds. The first n of these lines describe the tasks for Saturday, and the remaining m lines describe the tasks for Sunday.

Output

If it is impossible to complete all the tasks, output `impossible`.

Otherwise, output the maximum number of milliseconds of overlap of scheduled sleep between Saturday and Sunday.

Sample Input 1	**Sample Output 1**

`1 1`
`40000000 60000000 15000000`
`50000000 70000000 15000000`
	`66400000`

Sample Input 2	**Sample Output 2**

`2 1`
`40000000 60000000 15000000`
`40000000 60000000 15000000`
`50000000 70000000 15000000`
	`impossible`

Sample Input 3	Sample Output 3
3 2 25000000 55000000 10000000 35000000 45000000 5000000 60000000 75000000 10000000 30000000 40000000 10000000 45000000 65000000 15000000	56400000



A schedule for the weekend with perfectly synchronized 8 hours of sleep on each day.

D Discception

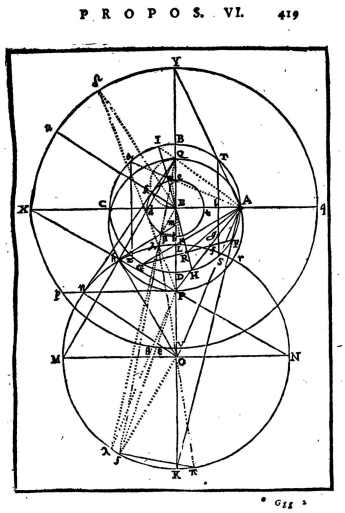
Time limit: 2s

After Ferdinand Magellan's expedition, it became clear to everyone in Europe what scholars and the ancient Greeks had known for a long time: the Earth was, indeed, not a disc.

Less well known is that, around the same time, European scholars were working on a much more ambitious theory of the motion of the celestial bodies. Astronomers often described planetary motion using deferents and epicycles, circles whose centres themselves moved along other circles. Some scholars took this idea further and hypothesized that a system of circular orbits was stable precisely when the centre of every orbit lay on another orbit in the same system.

Today we know that celestial mechanics are somewhat more complicated than that.

From their observations, the scholars calculated the radii of the circles needed for their model, but not their positions. Therefore, they wanted to know whether these circles could be arranged in a stable manner, that is, so that the centre of every circle lies on the circumference of another circle.



Christoph Clavius, *Astrolabium* (1593), Book II, Proposition VI, p. 419. Public domain, via Archive.org, scan by University of Michigan

Checking all possible arrangements by hand was rather inconvenient, but fortunately, we now have computers to determine whether such an arrangement is possible.

Input

The input consists of:

- One line with an integer n ($2 \leq n \leq 2 \cdot 10^5$), the number of radii.
- One line with n integers r ($1 \leq r \leq 10^9$), the radius of each circle.

Output

If it is possible to arrange the circles in a stable manner, output "`possible`". Otherwise, output "`impossible`".

Sample Input 1	Sample Output 1
3 1 1 1	possible

Sample Input 2	Sample Output 2
2 1 2	impossible

Sample Input 3	**Sample Output 3**
6 7 2 1 4 3 5	possible

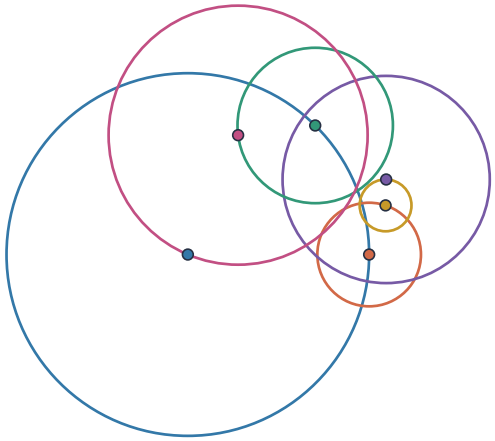


Figure D.1: A stable arrangement for the third sample. Each dot marks the centre of the circle with the corresponding colour.

E Endurtaka Mig

Time limit: 3s

```python
while True:
    print("repetitionlegitimizes", end="")
```

Generator for a test case of this problem.

A certain Icelandic singer absolutely hates repeating himself. It seems tedious and not very interesting to him. This means each word he says is not equal to a string concatenated 2 or more times. Formally, he can say a word w, if w cannot be written as $s + s + \cdots + s$ (k times, with $k > 1$). So he cannot say "`hahaha`" (can be written as "`ha`" + "`ha`" + "`ha`"), but he can say "`endurtakamig`".[1] You are analyzing a sentence spoken by the singer. You only forgot to add spaces between the words, so you are looking for an optimal way to do so. What is the minimum number of words to split the string into, such that each word could have been spoken by this singer?

Input

The input consists of:

- One line with a string s ($1 \leq |s| \leq 3 \cdot 10^5$), the sentence spoken.

The sentence only consists of English lowercase letters (`a`–`z`), it does not contain spaces.

Output

Output the minimum number of words k in the sentence, followed by k strings which form a possible subdivision of the sentence into words.

If there are multiple valid solutions, you may output any one of them.

Sample Input 1	Sample Output 1
`hahaha`	`2` `hah aha`

Sample Input 2	Sample Output 2
`endurtakamig`	`1` `endurtakamig`

[1] The phrase "Endurtaka mig" is Icelandic for "Repeat myself".

This page is intentionally left blank.

F Flippin' Cards

Time limit: 6s

You are playing the card game *Building A Perfect Combo*, which combines elements from *Risk*, *The Crew*, *Catan*, and *Monopoly*. This game consists of multiple playing rounds in which you must find the best combination. Before each round, the cards are shuffled and each player is dealt n cards. Just like in any card game, the players start by orienting the cards right side up before playing the game. Unfortunately, this can be an annoying time-consuming task, as many of those flippin' cards are upside down! To prevent losing unnecessary time on this task, you want to determine the optimal orienting strategy.



The perfect combo (if you turn the cannon, chance and some of *The Crew* cards right side up). In this combo, you must defeat the army of blue balloons. You will recruit these 8 knights led by Sir Regendruppel de Wraakzuchtige as crew for your holy quest. IJzeren Hein will train them in the closed defense to make them unstoppable. Then, they will use his cannon to defeat the blue balloon army completely. As this might not be entirely legal, you will use your get-out-of-jail-free-card to evade prison.

If each card orientation was independently uniformly right side up or upside down, then you would look at the cards one by one and flip them if they are upside down. However, often, many cards have the same orientation, due to the way the cards are collected for shuffling between rounds. Thus, after you go through half of your hand, and you had to flip almost every card, it would make more sense to flip the other cards (that you have not yet seen) all at once before moving on.

To model this, assume that there exists some real number $p \in [0, 1]$ chosen uniformly at random, such that each of the n cards is upside down with probability p, independent of the others. The value of p is fixed beforehand, but unknown to you. To orient your cards right side up, you look at each card one by one. For each card i, if it is right side up, it takes x milliseconds to register this and move on to card $i + 1$. Otherwise, you can either flip card i and move on, or flip cards $i, i + 1, \ldots, n$ all at once (while keeping their order the same) and move on. Flipping one card takes y milliseconds, and flipping all your remaining cards takes z milliseconds.

Determine the expected time to turn all cards the right side up using an optimal strategy.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 1000$), the number of cards.
- One line with three integers x, y, and z ($1 \leq x < y < z \leq 10^9$), the time in milliseconds for each of the three card moves, respectively: moving on from a card that is right side up, flipping the current card, and flipping all your remaining cards.

Output

Output the minimum possible expected time in milliseconds to turn all cards the right side up.

Your answer should have an absolute or relative error of at most 10^{-9}.

Explanation of samples

In the first sample, the optimal strategy is to never flip all your remaining cards. Then the expected time per card is 300 milliseconds, so the total expected time is 3 seconds.

In the second sample, the optimal strategy is to flip both cards if the first is upside down.

Sample Input 1	**Sample Output 1**
10 200 400 2000	3000

Sample Input 2	**Sample Output 2**
2 200 800 1000	1000

Sample Input 3	**Sample Output 3**
10 250 500 1000	3580.952381

G Graph Theft Auto

Time limit: 4s



The members of the LIMO-gang, who specialize in removing unwanted elements from society. From right to left: Brother Ed the Oak Processionary (because of his sophisticated methods of torture), Brother Sleep (because he only dreams about the best strategies to kill people), Brother Lampie (for his great technical skills in designing weapons), and Brother Beast (no comment).

Brother Duck is the leader of the Brotherhood of Assiduously Plundering Con-men (he is called Duck because you better duck if you ask him stupid questions like this one). Today, he has an important mission for you. He has learned that the dreaded Gangsterhood of Continuously Programming Criminals has come to Leiden! These gangsters construct brilliant master plans with sophisticated optimization algorithms, which they execute with great problem-solving skill. This time, they are after the original problem set of the very first BAPC in 1991, which is kept in Rijksmuseum Boerhaave in Leiden. Of course, brother Duck is not one of those politically correct soreheads that oppose fair robbery, but he cannot tolerate an important theft by a rival gang in *his* city. Hence, he has decided to initiate operation LIMO (Liquidate Interfering Mafia Overlords) for the sixth time: in this operation, the specialized unit of the brotherhood known as the LIMO-gang will give the gangsters an offer they cannot survive (i.e. kill them all). The only problem is to determine their secret hideout. That is where you come in.

Leiden can be represented as a connected undirected graph, where the museum is located at vertex 1 and the gangsters have their hideout on some different vertex. On the day of the crime, the gangsters will leave their hideout at midnight (as is their custom) and head to the museum following the shortest path (a habit of their optimizing nature). Each edge takes the same time to traverse, and from each vertex, there is a unique shortest path to the museum.

Brother Duck can install CCTV cameras at some vertices. Each camera registers whether the gangsters walk past its vertex and the time at which this happens (specified as the number of edges traversed by the gangsters from their hideout to this vertex). In particular, if a camera is installed at the gangsters' hideout or the museum, they will be registered as well. Using the cameras, Brother Duck wants to be able to deduce the location of the gangster's hideout. In particular, he wants the gangsters to be spotted by at least one camera to make sure that they have performed the crime: otherwise, the LIMO-gang might eliminate some random unrelated household, and hushing this up requires a significant amount of paperwork. Determine the minimum number of CCTV cameras needed to achieve this goal, or you may never see all your balloons again!

Input

The input consists of:

- One line with two integers n and m ($2 \leq n \leq 5 \cdot 10^5$, $1 \leq m \leq 5 \cdot 10^5$), the number of vertices of the graph and the number of edges of the graph.

- m lines, each with two integers a and b ($1 \leq a < b \leq n$), indicating that there is an edge between vertices a and b. No pair (a, b) is listed more than once.

The graph is guaranteed to be connected and each vertex has a unique shortest path to vertex 1.

Output

Output the minimum number of CCTV cameras required to catch the gang and locate their hideout.

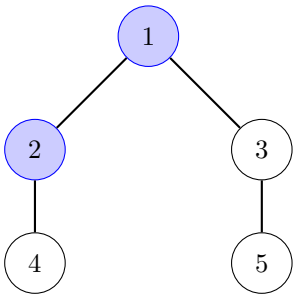


Figure G.1: An optimal solution for the second sample. Brother Duck can install CCTV cameras at the blue vertices 1 and 2, with which he can uniquely determine the hideout of the gang. It can be proven that Brother Duck needs at least 2 cameras.

Sample Input 1

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

Sample Output 1

```
1
```

Sample Input 2

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

Sample Output 2

```
2
```

Sample Input 3

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

Sample Output 3

```
3
```

H Historical Hierarchy

Time limit: 2s



Do you know the correct ordering of: the first Dutch Programming Championship, the first ICPC Programming Contest ever held, and the first International Olympiad in Informatics?

While taking an online history quiz, you encounter a question that has never been your strong suit: ordering a list of n historic events from different years in chronological order. Upon seeing that you are allowed to make $4n^2$ attempts at submitting an answer, you decide to open the browser console and write a script to automatically submit answers for you.

From the browser console, you find you can only perform one kind of operation. By providing two indices i and j ($1 \leq i, j \leq n$), the following two things will happen:

- The ith element is removed from the list. This will shift all elements to the right of position i one position to the left.
- The deleted element is inserted back at position j. This will shift all elements at position j or to the right of it one position to the right.

After such an operation, the current order is submitted and your answer receives a score, equal to the current length of a Longest Increasing Subsequence.[2]

Interaction

This is an interactive problem. Your submission will be run against an *interactor*, which reads from the standard output of your submission and writes to the standard input of your submission. This interaction needs to follow a specific protocol:

The interactor first sends one line with an integer n ($2 \leq n \leq 100$), the number of historic events.

Then, your program should send at most $4n^2$ commands, to reorder the events in chronological order. Each command is sent by printing one line of the form "i j" ($1 \leq i, j \leq n$), which takes out the element at position i and reinserts it at position j.

The interactor will respond with one line with an integer k ($1 \leq k \leq n$), the current length of a Longest Increasing Subsequence.

When the interactor responds with $k = n$, all events are in order and the interaction will stop. Sending any additional commands after this point or using more than $4n^2$ commands will result in a wrong answer.

The interactor is not adaptive.

Make sure you flush the buffer after each write.

A testing tool is provided to help you develop your solution.

[2] A subsequence is obtained from a sequence $a_1, a_2, \ldots, a_n$ by deleting zero or more elements, without changing the order of the remaining elements.

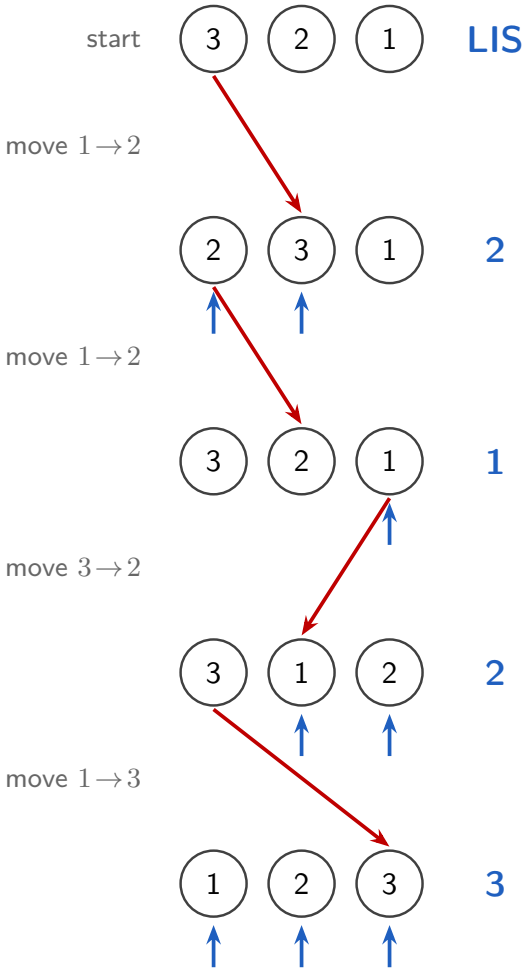


Figure H.1: Visualization of the interaction in the second example. At the start all the historic events are in reverse order. The red arrows show the operations, and the blue arrows show a possible Longest Increasing Subsequence (it may not be unique).

Read	Sample Interaction 1	Write
2		
	1 1	
1		
	1 2	
2		

Read	Sample Interaction 2	Write
3		
	1 2	
2		
	1 2	
1		
	3 2	
2		
	1 3	
3		

I Irritating Interruption II

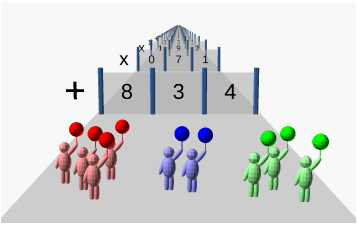
Time limit: 1s

Tonias has successfully determined that the last mobile game ad he looked at was a total scam. He has managed to get back on track with his thesis, and is determined to not get distracted again.

He will just take a 5-minute break.

...

Do you know those mobile game ads where you control an army that must repeatedly choose between different gates, each applying a mathematical operation to the army's strength (such as $\times 3$, $+10$ or -4)?



The high-quality multi-army version of the game, sporting many polygons and at least one texture that is definitely not a rendering artefact.

Tonias is now stuck playing the multi-army version: He starts with k armies, each with a strength of 1. In each round, a mathematical operation ($+$ or $\times$) is given, along with k gates displaying their corresponding integers.

Each of the armies must go through a *different* gate. When an army goes through a gate with value a, the army's strength is either increased or multiplied by a (depending on the operation of that round). Your objective is to help Tonias maximize the *sum of strengths* of all his armies after all rounds, so he can get back to his thesis.

Input

The input consists of:

- One line with two integers n and k ($1 \leq n \cdot k \leq 10^4$), the number of rounds and the number of armies.
- n lines, each containing an operator (either '+' or '$\star$') and k integers a ($0 \leq a \leq 10^{12}$), the value of each gate in this round.

It is guaranteed that, for any given input, no army strength can ever exceed 10^{12}.

Output

Output the maximum achievable sum of strengths of the armies.

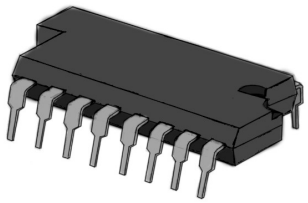
Sample Input 1	Sample Output 1
3 1 + 3 * 10 + 2	42

Sample Input 2	**Sample Output 2**
3 3 + 1 2 3 * 3 2 2 + 7 6 7	42

J Juxtaposort

Time limit: 3s

While working on a sorting algorithm, you discover a big problem! The computer chip you are using has a rare but infamous bug, called Broken Arithmetic Primary Comparison: the chip compares floating-point numbers with an offset. To be precise, given two numbers x and y, the comparison checks whether $x - 1 < y$. You want to know whether you can still sort a list of numbers.



The bugged computer chip.

Hidden from you is a sequence of n *distinct* floating-point values $a_1, a_2, \ldots, a_n$. You are given a list of all m pairs (i, j) with $i \neq j$ for which $a_i - 1 < a_j$. Based on this information, can the sequence be unambiguously sorted?

The hidden sequences that the samples are based on, respectively, are $a = [0.9, 1.1]$, $a = [4, 2]$, and $a = [1.7, 1, 2.5]$.

Input

The input consists of:

- One line with two integers n and m ($1 \leq n \leq 10^3, 0 \leq m \leq n(n-1)$), the number of values in the sequence, and the total number of pairs (i, j) for which $a_i - 1 < a_j$.
- m lines, each with two integers i and j ($1 \leq i, j \leq n, i \neq j$), indicating that $a_i - 1 < a_j$. No pair (i, j) is listed more than once.

The m pairs (i, j) are guaranteed to correspond to a sequence of floating-point values.

Output

If the order of the sequence is unambiguously determined, output the indices such that the values are in increasing order. Otherwise, output "`ambiguous`".

Sample Input 1	**Sample Output 1**
2 2 1 2 2 1	ambiguous

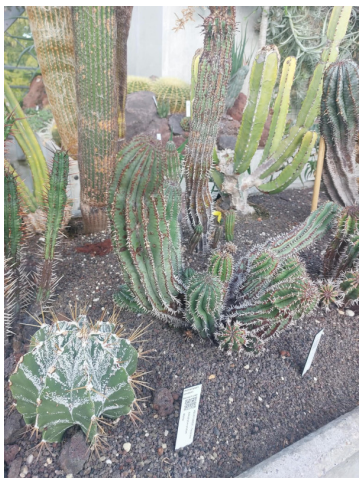
Sample Input 2	**Sample Output 2**
2 1 2 1	2 1

Sample Input 3	**Sample Output 3**
3 5 3 1 1 2 2 3 2 1 1 3	2 1 3

K Kaktus Kourse

Time limit: 10s

Before entering the contest site, you decide to wander around in the Hortus botanicus in Leiden. You come across some huge cacti. After walking some more, you decide to take a look at a map of the Hortus botanicus, and figure that all the walking routes can be modelled as a graph. Each route connects two different intersections and can be represented by an edge with some length. After investigating the map, you come to a delightful conclusion: the graph of the garden is a cactus graph! From a graph theory course you once took, you remembered the definition:



Huge cacti in the Hortus botanicus in Leiden.

Definition 1 (Cactus graph) *A connected, simple, undirected graph is a* cactus *graph if every edge lies on at most one cycle.*

Having fallen for the Hortus botanicus, you want each of your k friends to enjoy it too, so you will plan a personal *Kaktus Kourse* for every one of them.

You start a Kourse at one intersection of the garden and walk along the routes to a *different* intersection, never visiting the same intersection twice (repetition would bore your friends). Its length is the total length of the routes traversed. A Kourse is described by the sequence of intersections visited, in order: two Kourses are different if their sequences differ. In particular, walking a Kourse backwards gives a different Kourse.

Every friend must receive a different Kourse, and you want even the shortest one handed out to be as long as possible. What is the largest possible length of the shortest Kourse you give out? If the garden has fewer than k distinct Kourses, output "`impossible`".

Input

The input consists of:

- One line with three integers n, m, and k ($2 \leq n \leq 100$, $n - 1 \leq m \leq 100$, $1 \leq k \leq 10^{18}$), the number of intersections, number of routes in the map of the Hortus botanicus, and the number of friends.
- m lines, each with three integers u, v, and w ($1 \leq u < v \leq n$, $1 \leq w \leq 10^9$), indicating there is a route between intersection u and intersection v of length w.

It is guaranteed that the graph in the input is a cactus graph. Therefore, it is connected, contains no multi-edges (no pair (u, v) is listed more than once), and no two different cycles share a route.

Output

If the garden has fewer than k distinct Kourses, output "`impossible`". Otherwise, output the length of the shortest Kourse handed out to one of your friends.

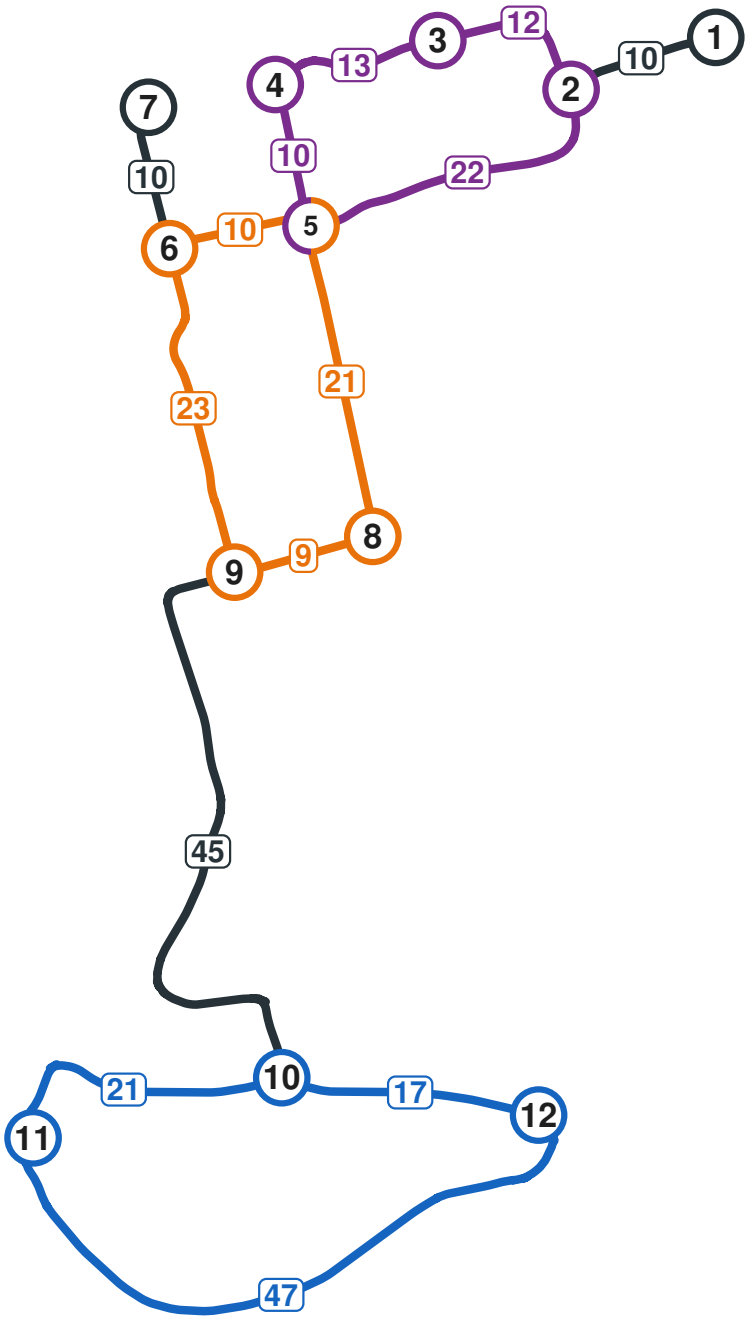


Figure K.1: An illustration of sample 1. The cactus graph mostly matches the actual map of the Hortus botanicus Leiden, but with some minor routes left out. All the cycles of the cactus graph are drawn in different colours. Two cycles can share an intersection, but never a route.

Sample Input 1

```
12 14 5
2 3 12
2 5 22
3 4 13
4 5 10
5 6 10
5 8 21
6 9 23
8 9 9
10 11 21
10 12 17
11 12 47
1 2 10
6 7 10
9 10 45
```

Sample Output 1

```
190
```

Sample Input 2

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

Sample Output 2

```
5
```

Sample Input 3

```
4 3 1
3 4 1000000000
1 2 1000000000
2 3 1000000000
```

Sample Output 3

```
3000000000
```

Sample Input 4

```
4 3 12
1 2 1000000000
3 4 1000000000
2 3 1000000000
```

Sample Output 4

```
1000000000
```

Sample Input 5

```
4 3 13
1 2 1000000000
2 3 1000000000
3 4 1000000000
```

Sample Output 5

```
impossible
```

This page is intentionally left blank.

L Letter Lottery

Time limit: 1s

Robots have been serving humanity well for quite some time, and in the last few years, LLMs also became quite useful. What could possibly go wrong when combining them? Well... the worst science fiction scenarios almost became reality. Fortunately, the world's brightest minds managed to find a set of n codes that got the robots under control.



One of the codes keeping a robot under control at the NWERC.

To celebrate the survival of humanity, a Near-World-Ending Robotics Carnival (NWERC) is organized. While most people would not dare to go near the machines that nearly destroyed humanity, Lara is an artificial intelligence enthusiast and would love to attend the carnival. However, normal tickets are already sold out, and her last chance is to get a ticket by participating in the Balancing Access to Phenomenal Carnivals (BAPC) raffle.

To enter the BAPC raffle, Lara has picked a single lowercase letter from the English alphabet. To determine which letter wins the raffle and hence who get a ticket for NWERC, the organizers first draw one of the n codes (consisting of English lowercase letters) uniformly at random, and then independently draw one of the letters from the selected code uniformly at random. Fortunately, the codes have been published, making it possible to calculate which letter is most likely to be chosen in this two-stage process. You know that Lara has picked a letter that maximizes her chances of winning. Given this, what is the probability that Lara will get a ticket for NWERC?

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 1000$), the number of codes.
- n lines, each with a string b ($1 \leq |b| \leq 1000$), the letters in each code.

The tickets only consist of English lowercase letters (`a-z`).

Output

Output the probability of the letter that maximizes Lara's chances of winning.

Your answer should have an *absolute* error of at most 10^{-6}.

Sample Input 1	Sample Output 1
5 abcdef aaabbbccc bbccdd bdffhhii xyz	0.191666667

Sample Input 2	**Sample Output 2**
1 abcdefghijklmnopqrstuvwxyz	0.03846154

Sample Input 3	**Sample Output 3**
1 a	1

M Memory Management

Time limit: 1s

You are working on your new problem for the Benelux Algorithm Programming Contest (BAPC): FizzBuzz. Given two natural numbers a and b, the contestant's program should print the natural numbers in ascending order, starting at 1, with three exceptions:

- If k is a multiple of a, the program should print "`Fizz`".
- If k is a multiple of b, it should print "`Buzz`".
- If both hold, it should print "`FizzBuzz`".



The floppy disk used for the BAPC. CC0 1.0 by Peter Astbury on Wikimedia Commons

Unfortunately, this year, the budget for the BAPC is rather minimal. Therefore, the output of the contestants is stored on floppy disks which can only hold 1440 KiB (1 474 560 bytes). It is your job to make sure that the correct output of a solution fits on such a floppy disk.

Given such a, b, and a natural number n, up to which integer k does the FizzBuzz sequence of k integers fit in n characters? Whitespace such as newlines or spaces should not be counted as characters.

An example of a FizzBuzz sequence of 11 integers for $a = 2$ and $b = 3$ is given by this string of length 37: "`1FizzBuzzFizz5FizzBuzz7FizzBuzzFizz11`".

Input

The input consists of:

- One line with two integers a and b ($2 \leq a, b \leq 1\,474\,560$).
- One line with an integer n ($1 \leq n \leq 1\,474\,560$), the number of bytes of storage available.

Output

Output the largest integer k, such that the FizzBuzz sequence from 1 up to and including k requires at most n bytes of storage.

Sample Input 1	Sample Output 1
2 3 10	3

Sample Input 2	Sample Output 2
2 4 15	5