German Collegiate Programming Contest 2021

June 26th

Problems

This page is intentionally left (almost) blank.

Problem A: Amusement Arcade

Julia wants to celebrate her birthday in a very special way this year – she wants to invite her friends to an amusement arcade and play all the awesome video arcade games from her childhood with them!

For this purpose she rented a venue with n arcades where all machines are arranged in one straight line. Some of her friends are very vivid players who tend to scream whenever they manage to beat the game. As this is very distracting for the others, Julia rented a venue that is so large that in case the very first and very last machine is occupied, everyone else can be seated such that there is always exactly one empty seat between any two neighbouring players.

Her friends, however, arrive at the amusement arcade one by one and do not know how many people will come in total. To sit as comfortable as possible, they always choose an arcade that is maximally far away from any other player. In case there are multiple machines that are equally secluded, they choose one of them uniformly at random. After having taken a seat, they will stay there until the end of the party.

As the host, Julia is the first person to arrive at the venue. Now she is wondering where she should take a seat such that all of her friends find an arcade to play and there is exactly one empty seat between neighbouring players once everyone is seated.

Input

The input consists of:

- One line with an integer n $(1 \leq n \leq 10^{18})$, the number of arcades at the venue. The arcades are labelled from 1 to n.

It is guaranteed that n is an odd number.

Output

If it is impossible to find a valid seating, output `impossible`. Otherwise, output an integer a $(1 \leq a \leq n)$, the label of the arcade at which Julia starts playing. In case there exist multiple valid solutions, any of them will be accepted.

Sample Input 1	Sample Output 1
`7`	`3`

Sample Input 2	Sample Output 2
`15`	`impossible`

Notes

In the first sample there are $n = 7$ seats. There will be 4 players in total, i.e., Julia and three guests. Julia takes a seat on machine 3. Her first guest will choose machine 7 as this machine is maximally far away from Julia. Her second guest will either choose machine 1 or 5 as both are equally secluded from the other players. Regardless of this choice, the last guest will choose the other of these two machines in both cases. In the end, machines 1, 3, 5, and 7 are occupied and there is one empty machine between any two neighbouring players.

This page is intentionally left (almost) blank.

Problem B: Brexiting and Brentering

This is not a problem about Brexit. Or at least not about the social or economic implications of Brexit. Instead, we – the Grammatical Correctness Policing Committee (GCPC) – want to focus exclusively on the linguistic challenges posed by this combination of the words "Britain" and the noun "exit". We are very concerned about the ambiguity of this construction. When using the term Brexit, it is not clear at all whether it is supposed to refer to to Great Britain or Brazil. Or perhaps Bremen leaving the Bundesliga.

And it's not just Brexit, you might also have heard of "Megxit" (having to do something with the British royal family) and similar constructions.

Looking ahead, we want to avoid a similar disaster in the future. For that purpose we, as a European agency with German roots, would like to implement some standardisation. We have focused on the act of entering. If a subject (a person, an organization, a plant, etc.) enters (or possibly reenters) something, our leading linguistic scientists suggest we look for the last vowel in the subjects name (`a`, `e`, `i`, `o` and `u` are considered vowels). We cut off any letters after this last vowel and add the ending `ntry` instead. Here are some examples:

- If Britain were to reenter the European Union, we would call it "Britaintry"[1].
- If Canada were to enter, say, the NWERC region, that would be a "Canadantry".
- And whenever a person named "Paul" enters someplace, we clearly have a "Pauntry".

Given a subject's name, determine how the act of entering should be called for this subject.

Input

The input consists of:

- One line with a string s ($1 \leq |s| \leq 50$), the name of the subject. This name can refer to any person, animal, country, organization, etc.

All characters in s are uppercase letters `A`-`Z` or lowercase letters `a`-`z`. The first letter may be uppercase or lowercase, all other letters are lowercase. s will contain at least one lowercase vowel.

Output

Output one single word, the term for the act of the subject entering.

Sample Input 1	Sample Output 1
`Britain`	`Britaintry`

Sample Input 2	Sample Output 2
`Canada`	`Canadantry`

Sample Input 3	Sample Output 3
`Paul`	`Pauntry`

[1]We are convinced this is going to happen, because Brexit must have been such a joy for the people of Britain (otherwise, their political leaders certainly would not have taken this step). When something brings so much joy to everybody, you want to repeat it. But to repeat Brexit, there has to be Britaintry first!

This page is intentionally left (almost) blank.

Problem C: Card Trading

Recently, I got into playing the trading card game *Wizardry – The Meeting*. And since I really wanted to build an awesome deck, I decided to search online for only the best cards. It turns out most of those cards are quite expensive and can only be acquired by insane luck, when purchasing a random set of cards, or by bidding in online auctions. As auctions are a huge time sink and I really rather wanted to play instead of bidding the whole day, I came up with a different idea: A trading card marketplace.

Each card type is produced in bulk, so a buyer does not really care from which seller they buy a specific card. Therefore, the idea is to create one web page for each card type and users can set buy and sell offers. Take the card "Green Mana" for instance. If you wanted to buy one, you could create a *buy offer*, e.g. for 10.00€. This offer means that you are willing to buy the card for 10.00€ or less (if there is a seller for less). On the other hand, if you wanted to sell one "Green Mana" card, you could create a *sell offer*, e.g. for 12.01€. This offer means you are willing to sell your card for 12.01€ or more (if there is a buyer for more).

Now, every couple of seconds, the website automatically calculates a card price based on both types of offers. It then considers only those offers that are compatible with this price (as described above) and satisfies as many of those as possible.

As an aspiring entrepreneur, I decided that I deserve a cut of every sale happening on the website. But I have a little trouble to come up with an algorithm that determines the price such that the *turnover*, i.e. the price times the number of successful sales, is as high as possible (which would mean my cut being as high as possible).

Input

The input consists of:

- One line with one integer n ($1 \leq n \leq 10^5$), the number of different prices at which offers exist.
- n lines, each containing one real number p and two integers b and s ($0 < p \leq 10^4, 0 \leq b, s \leq 10^6$), the price of the offers with exactly two decimal places, the number of buy offers at this price and the number of sell offers at this price.

It is guaranteed that each price in the input has at least one buy or sell offer and that no price appears more than once.

Output

If no price exists, such that at least one sale occurs, output "`impossible`". Otherwise, output the price resulting in the highest turnover and that turnover itself. If multiple such prices exist, output any. Output both numbers to *exactly* two decimal places.

Sample Input 1

```
5
12.00 0 3
11.99 2 0
11.98 5 0
10.00 1 0
12.01 0 6
```

Sample Output 1

```
impossible
```

Sample Input 2

```
6
2.85 14 0
4.50 0 1
5.26 3 3
6.17 1 0
14.78 0 2
21.04 1 0
```

Sample Output 2

```
5.26 21.04
```

Sample Input 3

```
6
2.85 14 0
4.50 0 1
5.26 2 3
14.78 0 2
1.83 0 1
21.04 1 0
```

Sample Output 3

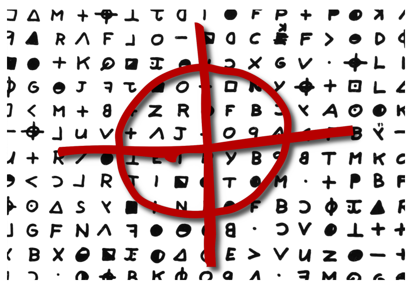
```
21.04 21.04
```

Notes

In the second sample case, the optimal card price is 5.26€, as it results in the highest possible turnover of 21.04€, with four sales happening. In total, there are five buyers willing to pay at least 5.26€: Three are willing to pay exactly 5.26€, one is willing to pay 6.17€ and one is even willing to pay 21.04€. On the other hand, there are just four sellers willing to part with their card at 5.26€: Three at exactly this price and one would already be happy with 4.50€.

Note that there is an alternative solution: at a card price of 21.04€, there will be exactly one sale, resulting in the same optimal turnover.

Problem D: Decrypting Zodiac


Original message from Zodiac

In the late 1960s, a serial killer committed his monstrous deeds. He was neither caught nor was he identified and due to a series of cryptic letters he sent to the press he was called *Zodiac*. It was assumed that those letters contain the killer's real name, but even to this day not all of them have been decrypted. One of the reasons for this is that the encrypted messages contain mistakes. It is not known if Zodiac made those mistakes on purpose to make the decryption harder.

For one of his first letters he used the following two step encryption scheme.[2] First, he applied a *Caesar cipher* which means that he replaced each letter with the one that comes k steps later in the alphabet, where k is a fixed number between 0 and 25 inclusive. Note that for this step it is assumed that after `z` the alphabet starts again with `a`. In the second step, he cut the message into two parts at an arbitrary position and swapped the parts. It is allowed for one of the two parts to be empty, in which case the message did not change during this second step.

Normally, a simple brute force search could be used to decrypt the message. However, to do this, one needs to automatically check if a message makes sense. Since Zodiac might have made some mistakes during the first step of the encryption, this is not easy to decide.

For this reason, you decided to try another approach. You want to try meaningful candidate sentences and encrypt them and then count how many mistakes would be required to make them match with Zodiac's encrypted message.

Input

The input consists of:

- One line with a single integer n ($1 \leq n \leq 1.5 \cdot 10^5$), the length of the messages.
- Two lines each with one string of length n. The first string is the encrypted message and the second string is your guess for the decrypted message.

Both strings consist of lowercase letters `a-z` only.

Output

Output a single integer, the minimal number of mistakes Zodiac must have made during the encryption, assuming you correctly guessed the decrypted message.

Sample Input 1	Sample Output 1
6 drhmex zodiac	2

[2]He did in fact not.

Sample Input 2	**Sample Output 2**

`8`
`dicepara`
`paradise`
	`1`

Sample Input 3	**Sample Output 3**

`13`
`lvlvdvdqsonwk`
`thisisasample`
	`2`

Notes

In the first sample the message can be encrypted by Caesar shifting each letter by four, resulting in `dshmeg`. After this, all letters match except for the second and sixth.

In the second example we can Caesar shift by zero, then split the message in the middle and swap both halves. After this, there is only a single mismatch: `s` $\leftrightarrow$ `c`.

In the third example the message can be encrypted by Caesar shifting by three in the first step, resulting in the message `wklvlvdvdpsoh`. Then, the first two letters can be cut off and swapped with the rest of the string to create `lvlvdvdpsohwk`. After this, only two letters will differ: `p` $\leftrightarrow$ `q` and `n` $\leftrightarrow$ `h`.

Problem E: Excursion to Porvoo

It is a lovely summer day, and Alice wants to do a day trip. She lives in Tampere, and wants to travel to Porvoo to enjoy the Old Town and the surrounding nature. Alice does not only love travelling, but also planning.

She has created a map of the most beautiful paths to Porvoo. On her trip she needs to visit n cities in order, where Tampere is the first city and Porvoo is the last city. The cities are connected by roads, with each road connecting two consecutive cities, and there is always at least one road between each pair of consecutive cities.

When driving from one city to the next, Alice needs to choose which road to take. Some of these roads have a tarmac surface, while others are just gravel roads and some roads have bridges which will not support vehicles that are too heavy. For each road it is known how long it takes to traverse it and what is the maximal weight of vehicles that can safely drive on it.

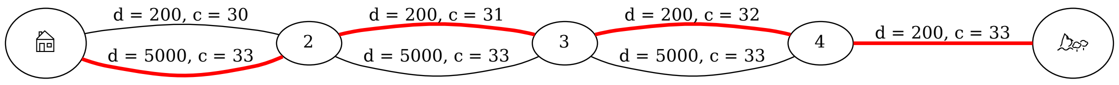


Figure E.1: Illustration of the second sample input. The red path from Tampere to Porvoo is the optimal choice for a car of weight 31.

Alice collects many different cars of different weights, but she is not sure yet which car she will use for the day trip. As she wants to enjoy as much time in Porvoo as possible, she wants you to help her find the minimal travel time for each car.

Input

The input consists of:

- Two integers n and m $(2 \leq n \leq 10^5, n-1 \leq m \leq 10^5)$, the number of cities and the number of connections, respectively. The cities are numbered from 1 to n, Tampere is city 1, and Porvoo is city n.
- m lines, each containing three integers i, d and c $(1 \leq i < n, 1 \leq d \leq 10^4, 1 \leq c \leq 10^6)$, which each describe a connection between city i and city $i+1$ which takes d minutes to traverse and can can be used by vehicles of weight c kilograms or less.
- One integer q $(1 \leq q \leq 10^5)$, the number of cars that Alice has collected.
- q lines, where the ith line contains one integer w_i $(1 \leq w_i \leq 10^6)$, the weight of the ith car in kilograms.

There is at least one connection from city i to city $i+1$ for each i $(1 \leq i < n)$.

Output

Output q lines, where the ith line describes the shortest time in minutes Alice needs to drive to get from Tampere to Porvoo with the ith car. If there is no feasible path for the ith car, output `impossible`.

Sample Input 1

```
2 2
1 100 300
1 1 30
5
400
500
300
20
1
```

Sample Output 1

```
impossible
impossible
100
1
1
```

Sample Input 2

```
5 7
1 200 30
2 200 31
3 200 32
4 200 33
1 5000 33
2 5000 33
3 5000 33
3
30
31
33
```

Sample Output 2

```
800
5600
15200
```

Sample Input 3

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

Sample Output 3

```
2
3
3
```

Problem F: Flappy Bird

Help the bird Faby to navigate through a sequence of n pairs of pipes, by finding the shortest line he can fly on to reach his destination. For simplicity, we represent Faby as a single point in the plane and assume every pipe has width zero. This way, the gap between every pair of pipes can be represented as an interval on the y axis. The bird starts out at $s = (x_s, y_s)$ and his goal is to reach $t = (x_t, y_t)$. Find the shortest line from s to t, passing through all intervals in between in increasing order of their x coordinates.

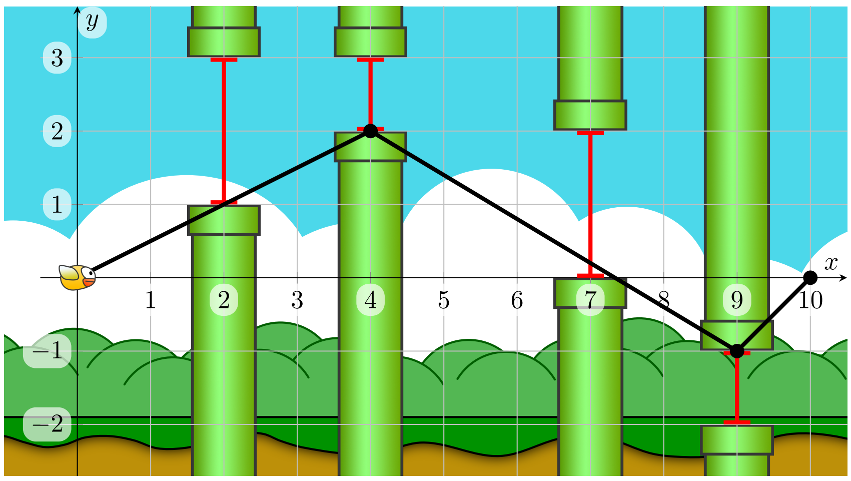


Figure F.1: Visualisation of the second sample input. The red lines represent the intervals and the black line the shortest possible path. Faby and the black dots are the points in the output. Note that $(2, 1)$ can optionally be included in the output too.

Input

The input consists of:

- One line with four integers x_s, y_s, x_t and y_t ($-10^9 \leq x_s, y_s, x_t, y_t \leq 10^9$), the start and end points.
- One line with an integer n ($0 \leq n \leq 10^6$), the number of intervals.
- n lines, the ith of which contains three integer $x_i, y_{i,1}$ and $y_{i,2}$ ($-10^9 \leq x_i, y_{i,1}, y_{i,2} \leq 10^9$, $y_{i,1} < y_{i,2}$), the intervals.

It can be assumed that $x_s < x_1 < \cdots < x_n < x_t$.

Output

Output a sequence of k ($2 \leq k \leq n + 2$) points $p_1, \ldots, p_k$, one per line, such that:

- All points have integer coordinates.
- $p_1 = s$ and $p_k = t$.
- Let P be the path obtained by connecting $\overline{p_i p_{i+1}}$ for all $1 \leq i < k$. Then:
  - P passes through all intervals in increasing order of their x coordinates.
  - The length of P is minimal.

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

Sample Input 1

```
0 0 10 0
1
5 -10 10
```

Sample Output 1

```
0 0
10 0
```

Sample Input 2

```
0 0 10 0
4
2 1 3
4 2 3
7 0 2
9 -2 -1
```

Sample Output 2

```
0 0
4 2
9 -1
10 0
```

Problem G: Grid Delivery

Your friend Ellie owns a local parcel delivery business called Grid City Parcel Courier (GCPC) which operates in Grid City, a town where all houses are aligned on a rectangular grid of streets. Each house is placed at the intersection of two streets, one running in north-south direction (vertically) and one running in east-west direction (horizontally). There are w vertical streets and h horizontal streets, resulting in a $h \times w$ grid of houses.

To grow her business, Ellie wants to start offering parcel pickup too. However, the mayor of Grid City recently decided that all streets will be one-way streets during the day to combat traffic jams. During this time, the streets of Grid City can only be passed from north to south or west to east, respectively.

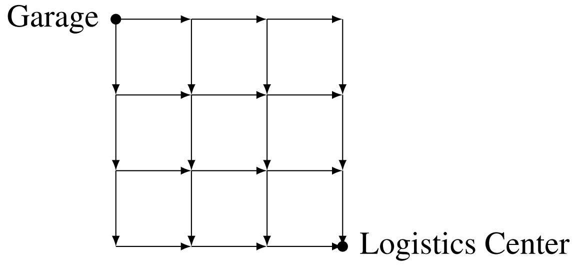


Figure G.1: Visualization of the grid of one-way streets given in the first sample input.

Ellie already rented a large garage located at the city's northwesternmost intersection, from which her drivers will start their journeys to collect parcels. She asked you to help her figure out how many drivers she needs to hire to collect all parcels during the day and bring them to her logistics center located at the city's southeasternmost intersection.

Input

The input consists of:

- One line with two integers h and w ($1 \leq h, w \leq 2\,000$), the height and width of the grid.
- h lines, each with w characters which are either `C`, indicating the house of a customer where a parcel has to be collected, or `_`, indicating a house where nothing has to be collected.

Output

Output the minimal number of drivers required to collect all parcels while all streets are one-way streets.

Sample Input 1	Sample Output 1
4 4 __C_ C_C_ _C_C _CCC	2

Sample Input 2

```
4 6
CC____
_CCC__
___C_C
C__CCC
```

Sample Output 2

```
2
```

Sample Input 3

```
3 5
CC__C
_C_CC
CCCCC
```

Sample Output 3

```
3
```

Problem H: Hectic Harbour II

An upcycled shipping container makes a good site to open a pop-up store in a trendy part of town. Such a business comes with its own risks – for example, this morning a local freight company mistook your premises for one of their crates and sent it to the shipyard for loading.

Your crate is now sitting in the shipyard in one of two stacks ready for loading onto the ship. Each crate except yours has its own tracking number.

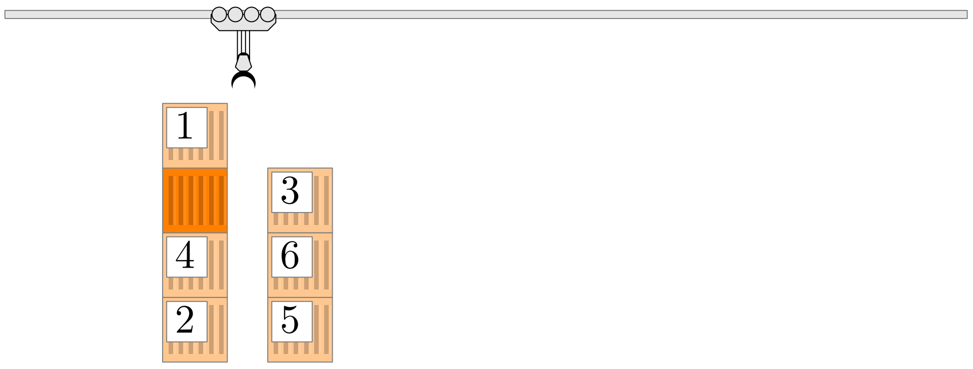


Figure H.1: Illustration of Sample Input 2. Your business is in the unmarked crate.

The system for loading crates is automated and proceeds in a preset order. First, the crate with the next tracking number is uncovered by picking up all of the crates on top, one-by-one, and moving every single one across to the other stack individually. Then the crate is taken to the ship. Since your crate is not part of this order, it is generally ignored and will not be loaded.

After loading a crate, some time is spent securing the whole cargo on board. This is your chance to recover your container – if it is on top of one of the stacks, you will have just enough time to slide it off and get it back.

How many such opportunities will you have in total?

Input

The input consists of:

- One line with three integers n, s_1 and s_2 ($2 \leq s_1, s_2 \leq 2 \cdot 10^5, s_1 + s_2 = n + 1$), the number of crates with a tracking number, the number of crates on the first stack, and the number of crates on the second stack respectively.
- One line containing s_1 integers, the tracking numbers of the crates on the first stack, in order from bottom to top.
- One line containing s_2 integers, the tracking numbers of the crates on the second stack, in order from bottom to top.

The crates with tracking number are numbered from 1 to n and are removed from the stacks in that order. Your crate has tracking number 0 and will never be on top of one of the stacks initially.

Output

Output the number of occasions at which your crate is on top of one of the stacks and the crane is busy loading a crate.

Sample Input 1

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

Sample Output 1

```
3
```

Sample Input 2

```
6 4 3
2 4 0 1
6 3 5
```

Sample Output 2

```
4
```

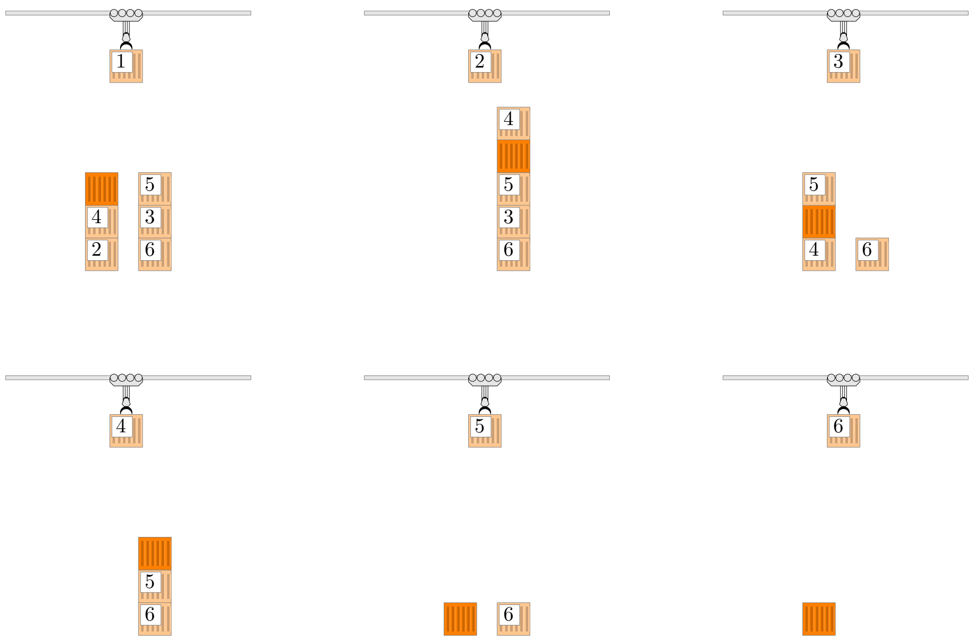


Figure H.2: Step by step illustration of Sample Input 2. There are 4 occasions at which your crate is on top of one of the stacks, while any of crates 1, 4, 5 or 6 is loaded.

Problem I: Index Case

The epidemiologist W. Andy wants to find the index case of an ongoing crisis. To do this, he modelled the city of the outbreak and its n residents with a *cellular automaton*. The city is represented by n cells numbered from 1 to n and each cell has two neighbouring cells, one to its left and one to its right. The left neighbour of cell i is cell $i - 1$ and the right neighbour is cell $i + 1$. Additionally, the left neighbour of cell 1 is cell n and the right neighbour of cell n is cell 1. Thus, the city and the corresponding automaton form a simple cycle.

Each cell contains an integer between 1 and m which represents how likely it is that this person is infected. Since the virus can only be transmitted by personal contact, the value in the ith cell on day d only depends on the values of its neighbours and itself on the previous day. If we denote this value by $s_d[i]$, then the outbreak can be simulated by a function f using the formula:

$$s_d[i] = f\big(s_{d-1}[i-1], s_{d-1}[i], s_{d-1}[i+1]\big).$$

Note that as the city is cyclic both $i + 1$ and $i - 1$ are calculated modulo n.

Andy wants to find the index case, so he first has to find s_0, the state of the city on day zero. This poses a problem, however, as it is not known on which day the crisis started. Right now, Andy believes that he accomplished the task and found the state s_0, but you are not convinced. Therefore, you want to check if there may be a state previous to the initial state proposed by Andy, i.e. whether there exists any state s_{-1} that gets transformed into s_0 by applying f.

Input

The input consists of:

- One line with two integers n and m ($3 \le n \le 200, 2 \le m \le 10$), the number of cells and the number of states.
- m^3 lines describing the values $f(x, y, z)$ ($1 \le f(x, y, z) \le m$ for each $1 \le x, y, z \le m$) of the function f modelling the automaton. The values are given in lexicographic order of the arguments: The first value is $f(1, 1, 1)$, the next is $f(1, 1, 2)$, and so on until $f(1, 1, m)$, followed by $f(1, 2, 1)$ and so forth. The last value is $f(m, m, m)$.
- One line with n integers $s_0[1], \dots, s_0[n]$ ($1 \le s_0[i] \le m$ for each i), the initial state that has been proposed by Andy.

Output

Output `yes` if there exists at least one possible previous state and `no` otherwise.

Sample Input 1

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

Sample Output 1

```
yes
```

Sample Input 2

```
6 2
1
2
1
2
2
1
2
1
1 2 1 2 1 2
```

Sample Output 2

```
no
```

Sample Input 3

```
10 2
1
2
1
1
2
2
2
2
1 2 2 2 1 2 1 2 1 2
```

Sample Output 3

```
yes
```

Problem J: Joined Sessions

Lucy is very lazy. Her boss asked her to go to a conference and of course she wants her to go to as many meetings as possible. But Lucy is lazy and so she decides to choose her meetings such that all other meetings overlap with at least one of the meetings she is attending. This way, her boss cannot complain as there is no way for her to attend additional meetings.

While reading the schedule of the conference, Lucy finds out that even with this approach of choosing the meetings there are quite many meetings she has to attend. Luckily, her good friend Max is one of the organisers of the conference and in particular responsible for the timetable. Max is unfortunately not able to cancel meetings or to reschedule them, but he can help Lucy in another way.

Since the meetings are normally boring, nobody will pay too much attention if the topic changes. Therefore, whenever there are two meetings that overlap, he can combine them into a single meeting instead. Two meetings a and b overlap if start$(a) \leq$ start$(b) \leq$ end(a) or vice versa, and when they are combined the new meeting will start at time $\min(\text{start}(a), \text{start}(b))$ and end at time $\max(\text{end}(a), \text{end}(b))$. Max can then repeat this merging process and it is even possible to further combine these combined meetings with other meetings. Non-overlapping meetings cannot be combined as then people would notice that someone is tampering with the schedule.

Lucy now wonders if she can reduce the number of meetings she has to attend with this method. If it is possible, how many such merges are necessary to reduce this number by at least one?

Input

The input consists of:

- One line with an integer n ($2 \leq n \leq 10^6$), the number of meetings.
- n lines describing the meetings, each with two integers a and b ($0 \leq a \leq b \leq 10^9$), where a is the start time and b the end time of one of the given meetings.

Output

If it is possible to reduce the number of meetings Lucy has to attend, then output the minimum number of merging operations needed to do so. Otherwise output `impossible`.

Sample Input 1

```
4
1 3
2 5
4 7
6 9
```

Sample Output 1

```
1
```

Sample Input 2

```
5
1 3
4 7
8 10
2 5
6 9
```

Sample Output 2

```
2
```

Sample Input 3

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

Sample Output 3

```
impossible
```

Problem K: Killjoys' Conference

The General Counsel for Peaceful Congregations (GCPC) has a very, very stressful job. Almost every day they are approached by someone who has to organise a meeting whose attendees do not really get along. More specifically, in any group of attendees there may be several pairs of people who are known to dislike each other. Nevertheless people sometimes need to meet, so the general strategy of the GCPC is to split up the meeting attendees into two groups. These groups will then meet in different rooms and GCPC employees will deliver messages back and forth between the two rooms. Let's call these two rooms the East and the West room – for no particular reason. To ensure peaceful and productive meetings, the GCPC assigns people to the East and West room such that no two people in each room dislike each other.

Over time, the process of assigning people to the East and West rooms has become a bit tedious, so you decided to undertake a little experiment. Some of the GCPC's clients schedule the same meeting with the exact same people every year. To keep things interesting, you want to use a new assignment of people to the East and West rooms for each meeting. If the editions of the meeting are numbered starting from 1, what is the number of the first meeting where you are forced to reuse an assignment of people that you have already used before? Note that simply swapping the rooms, i.e. assigning the people from the East room to the West room and vice versa, is not considered a different assignment – after all, the same people will meet. Since your investigation will almost certainly only be of an academic nature, you are not interested in the exact value. It will suffice to find the remainder when dividing the result by a given odd prime number p.

Input

The input consists of:

- One line with three integers n, m and p ($1 \leq n \leq 10^6$, $0 \leq m \leq 10^6$, $3 \leq p \leq 10^9$), where n is the number of people attending the meeting, m is the number of known dislikes between them and p is an odd prime number. The attendees of the meeting are numbered from 1 to n, inclusive.
- m lines, each with two integers a and b ($1 \leq a, b \leq n, a \neq b$), specifying that attendees a and b dislike each other.

Output

Output one integer, the number of the first edition where an assignment must re-occur. Output this number modulo p. If it is impossible to assign the people to the East and West rooms such that no two people disliking each other are placed in the same room, output `impossible`.

Sample Input 1

```
4 2 11
1 2
3 4
```

Sample Output 1

```
3
```

Sample Input 2

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

Sample Output 2

```
2
```

Sample Input 3	Sample Output 3

```
3 3 11
1 2
2 3
3 1
```
	`impossible`

Sample Input 3
```
3 3 11
1 2
2 3
3 1
```

Sample Output 3
```
impossible
```

Sample Input 4
```
100 0 13
```

Sample Output 4
```
9
```

Notes

In the first sample, you could use the following room assignments:

	East	West
Year 1	1,3	2,4
Year 2	1,4	2,3
Year 3	2,4	1,3

In the third year the groups are the same as in the first year, and there is no set of assignments that avoids repetitions this for longer than that.

In the second sample, an optimal set of assignments is given as follows:

	East	West
Year 1	1,3,5	2,4
Year 2	1,4,5	2,3
Year 3	2,4,5	1,3
Year 4	2,3,5	1,4
Year 5	1,3,5	2,4

Problem L: Looking for Waldo

You may know the game *Where is Waldo?*. In this game you need to find a person named Waldo in a crowd of people. This problem is kind of similar. You need to find an axis-aligned rectangle of minimal area which contains the letters `W`, `A`, `L`, `D` and `O` and those letters are hidden in a crowd of other letters.

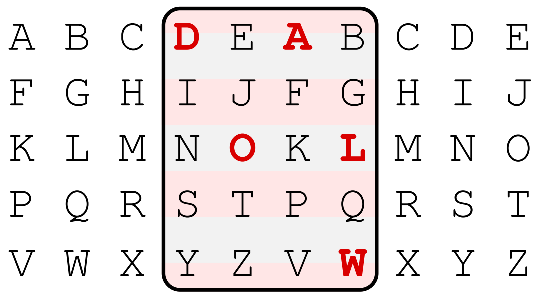


Figure L.1: Illustration of the second sample case.

Input

The input consists of:

- One line with two integers h and w ($1 \leq h, w \leq 10^5$, $h \cdot w \leq 10^5$), the height and width of the grid of letters.
- h lines, each with w upper case letters `A`-`Z`, the grid of letters.

Output

Output the area of the smallest axis-aligned rectangle which contains at least one of each of the letters `W`, `A`, `L`, `D` and `O`. If there is no rectangle containing those letters, output `impossible`.

Sample Input 1

```
5 5
ABCDE
FGHIJ
KLMNO
PQRST
VWXYZ
```

Sample Output 1

```
25
```

Sample Input 2

```
5 10
ABCDEABCDE
FGHIJFGHIJ
KLMNOKLMNO
PQRSTPQRST
VWXYZVWXYZ
```

Sample Output 2

```
20
```

Sample Input 3

```
5 10
WAALDLODOW
AWWLAOODOW
LOLADOWALO
ADALLLWWOL
WWOOAAAALO
```

Sample Output 3

```
5
```

Sample Input 4

```
2 3
WAL
TER
```

Sample Output 4

```
impossible
```

Problem M: Monty's Hall

You have explored the deep catacombs under a long lost city for the past couple of hours and finally you have reached their end: The hall of the undead wizard Monty. His restless spirit materialises in front of you and you prepare for battle.

However, it turns out that you are the first explorer to find him in over a hundred years, so he has grown incredibly bored. Instead of a fight, he offers to play a game for his artefacts. The hall has d closed doors, but only one of them leads to the artefacts (Monty knows which one it is, of course). The procedure is as follows:

This could be you if you choose poorly.
Photo by Armin Kübelbeck, cc-by-sa, Wikimedia Commons

1. You choose s closed doors.
2. Monty opens e doors that were not selected by you and lead to empty rooms.
3. Among the remaining closed doors, you may change your selection of s doors however you want (you can even stay with your current selection if you wish to).
4. Monty reveals which door leads to the room with his artefacts.

If the door with the artefacts is among your selected doors, you win and can take them with you unscathed. If not, Monty will transform you into a goat. So you better hope your luck is on point today.

Input

The input consists of:

- One line with three integers d, s and e ($1 \leq d, s, e \leq 10^6$, $s + e < d$), the number of doors in Monty's hall, the number of doors you are allowed to select and the number of doors Monty opens in step 2.

Output

Output your chance to win at Monty's game when playing optimally. Your answer should have an absolute or relative error of at most 10^{-6}.

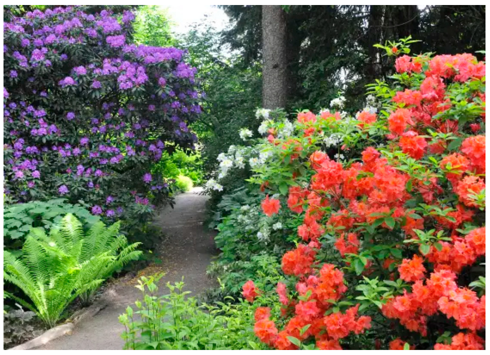
Sample Input 1	Sample Output 1
`3 1 1`	`0.666667`

Sample Input 2	Sample Output 2
`8 4 2`	`0.75`

Sample Input 3	Sample Output 3
`15 4 2`	`0.32592593`

This page is intentionally left (almost) blank.

Problem N: Natural Navigation


This majestic footpath has the colours red, white, purple and green.
SOURCE: Botanical Garden München-Nymphenburg.

You want to build an app to help people navigate through a large botanic garden. This is difficult, because there are many winding footpaths and intersections offering many choices, making traditional directions such as "turn right" or "move further north" unsuitable. Instead, the app should rely on the garden's greatest resource: the numerous exotic plants and their diverse colours. Whenever a user is at an intersection, the app will know where they are and will display one particular colour accordingly. The user will then follow a footpath where this colour is visible. If the colour can be spotted along multiple footpaths originating from the intersection, the user is free to choose any of these footpaths.

You have been given a perfect model of the botanic garden, consisting of n intersections (numbered from 1 to n) and m footpaths going between those. To keep order, each footpath can only be used in the given direction. Currently, the plants are exhibiting k different colours (numbered from 1 to k) and for each footpath, you are given a list of all the colours that are visible along it when viewed from the intersection where it starts. A user is currently at intersection 1 and wants to navigate to intersection n. You can assume that the user will follow the app's directions perfectly, but whenever faced with multiple options (because the given colour is visible along multiple footpaths), you have to assume they will make the worst possible choice. How long will it take to reach the target when your app gives the best possible instructions?

Input

The input consists of:

- A line containing the number of intersections n ($1 \leq n \leq 5 \cdot 10^5$), the number of footpaths m ($1 \leq m \leq 5 \cdot 10^5$) and the number of distinct colours k ($1 \leq k \leq 1\,000$).
- m pairs of lines describing the directed footpaths, each formatted as follows:
  - One line with three integers u, v and t ($1 \leq u, v \leq n, 1 \leq t \leq 10^6$), meaning that the footpath leads from intersection u to intersection v and it takes t seconds to walk along this footpath.
  - One line with an integer ℓ ($1 \leq \ell \leq k$), followed by ℓ distinct integers $c_1, \ldots, c_\ell$ ($1 \leq c_i \leq k$ for each i), listing the colours that appear along this footpath.

The sum of ℓ over all footpaths does not exceed $5 \cdot 10^5$. Note that, as you would imagine in a botanic garden, a footpath can lead back to the intersection it started from and multiple footpaths can exist between a pair of intersections. Moreover, it is not guaranteed that each intersection can be reached via the footpaths.

Output

If it is impossible to lead the user to intersection n, output `impossible`. Otherwise output a single integer, the time it will take to reach the target in seconds. We are only considering the time spent walking along the footpaths.

Sample Input 1

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

Sample Output 1

```
14
```

Sample Input 2

```
3 4 3
1 2 300
2 1 2
2 1 2000
2 3 1
1 3 80
2 2 1
2 2 42
1 2
```

Sample Output 2

```
impossible
```