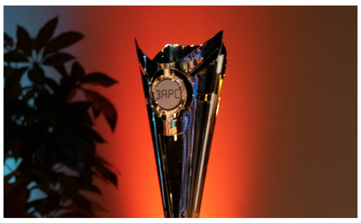
DAPC 2024

Delft Algorithm Programming Contest 2024

Problems

A Awkward Auction

Time limit: 1s



The coveted BAPC cup.
© GEWIS on 2022.bapc.eu,
used with permission

Your local warehouse's marketing department has thought of a new way of extracting money from their consumers: organizing the Battle in Assessing Precisely Costs (BAPC).

A battle is played against an auctioneer. The auctioneer has an unlimited number of identical gems available, which all have the same fixed secret worth. This secret worth is an integer between 1 and n (inclusive, in euros). In each round, you have to bid an integer amount of euros on such a gem, and your goal is to bid exactly the secret worth. If you bid at least the secret worth, you have to buy the gem for the amount of your bid. On the other hand, if you bid less than the secret worth, you do not get the gem and keep your bid. However, if you bid less than the worth, the auctioneer will give an endless speech why the gem is clearly worth more than the bid. To stop this speech and be able to make a new bid, you have to bribe the auctioneer with b euros each time. Finally, if you bid exactly the secret worth, then in addition to buying the gem, you get a nice cup showing that you won the battle, and the battle immediately ends.

Since the BAPC is your favourite competition, you of course want this cup! Therefore, you keep bidding until you bid the right amount and get the cup. You wonder how much this will cost you in the worst case, assuming that you make optimal decisions for the amounts to bid.

Input

The input consists of:

- One line with two integers n and b ($1 \leq n \leq 400$, $1 \leq b \leq 10\,000$), the given maximum worth of the gem and the bribe you need to pay when bidding too low.

Output

Output the maximum cost in euros that you have to pay in the worst case if you bid optimally.

Sample Input 1	Sample Output 1
4 2	6

Sample Input 2	Sample Output 2
8 3	16

B Battle of Nieuwpoort

Time limit: 1s

The battle of Nieuwpoort occurred in the year 1600. This is famously easy to remember, because it ends in two zeros. Alas, not all historical events have been so obliging!

You suspect that the problem is with the fixation of historians on the decimal system. Maybe, given the year of another battle, there exists a small base (at most 16) in which this year would also be easy to remember?

Prince Maurice at the Battle of Nieuwpoort, 2 July 1600, Pauwels van Hillegaert (1596–1640), Oil on panel, circa 1632–1640. Public domain, Rijksmuseum Amsterdam and Wikimedia Commons

Input

The input consists of:

- One line with 4 tokens:
  - One integer y ($1 \leq y \leq 2024$, in base-10), the year of the battle.
  - Three words w ($2 \leq |w| \leq 20$), naming the battle.
    The words only consist of English letters (`A-Z` and `a-z`).

Output

If it is possible to rewrite the year to make it easier to remember, output this base b ($2 \leq b \leq 16$, in base-10) and the year written in base-b. Otherwise, output "`impossible`".

The year in base-b must end with "`00`" and must not start with '`0`'.

Use letters '`a`', '`b`', '`c`', etc. for the digits following '`9`' in bases higher than 10.

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

Sample Input 1	Sample Output 1
`1600 Battle of Nieuwpoort`	`10 1600`

Sample Input 2	Sample Output 2
`625 Battle of Sarus`	`5 10000`

Sample Input 3	Sample Output 3
`1600 Battle of Sekigahara`	`8 3100`

Sample Input 4	Sample Output 4
`1815 Battle of Waterloo`	`11 1400`

Sample Input 5	Sample Output 5
`1859 Battle of Solferino`	`13 b00`

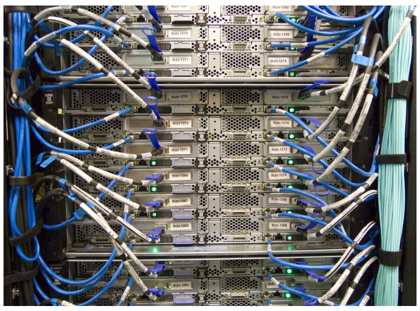
Sample Input 6	Sample Output 6
`1848 Battle of Bov`	`2 11100111000`

Sample Input 7	Sample Output 7
`1453 Fall of Constantinople`	`impossible`

C Chaotic Cables

Time limit: 1s

Your friend Claas is in charge of designing the network for the newly constructed computer lab. Aware of the critical importance of efficiency in network design, Claas opted for the sophisticated Binary Access Point Configuration (BAPC) network topology.


Visualization of a possible BAPC network.
CC0 by David Lohner on Flickr

A network is classified as a BAPC network precisely if we can assign a binary address of a fixed length to each device within the network, ensuring that:

1. Devices are connected if and only if their addresses differ in exactly one bit.
2. Each possible address is assigned to exactly one device.

Claas started out wiring devices together, but as the intricate web of connections began to take shape, doubt crept into his mind. Was the network he painstakingly constructed truly a BAPC network?

Help Claas determine if the network is a BAPC network.

Input

The input consists of:

- One line with two integers n and m ($2 \leq n \leq 2 \cdot 10^5$, $1 \leq m \leq 2 \cdot 10^5$) the number of devices and the number of wires in the network.
- m lines with integers a and b ($1 \leq a, b \leq n$, $a \neq b$), indicating that there is a wire between devices a and b.

It is guaranteed that each pair of devices is connected by at most one wire.

Output

Output "`yes`" if the network is a BAPC network. Otherwise, output "`no`".

Sample Input 1	Sample Output 1

4 3
1 2
2 3
1 4
	no

Sample Input 2	**Sample Output 2**

`8 12`
`1 2`
`6 2`
`8 2`
`3 1`
`1 7`
`3 6`
`6 5`
`3 4`
`8 7`
`8 5`
`7 4`
`5 4`
	`yes`

D Dialling Digits

Time limit: 2s

Disaster struck at the Billboards And Phone-numbers Company! Due to a bug in their database system, they lost the corresponding mnemonic phrases for each of the registered phone numbers. These mnemonic phrases are typically used on billboards, to make a phone number for an advertisement easier to remember by people who read them. To dial the phone number from a mnemonic phrase, you simply have to press the number keys corresponding to each letter, as shown in Figure D.1. For example, the phone number "`1-800-BAPC`" would be dialled as "`1-800-2272`".

CC BY 2.0 by Elliot Brown on Flickr, modified

Using this information and a given list of valid words, reconstruct the possible mnemonic phrases for each registered phone number. Each mnemonic phrase consists of exactly one word from the word list. In the input, you will only receive the part of the phone number that should be exactly linked to possible mnemonic phrases, so it does not include the "`1-800-`" prefix (or any other prefix).

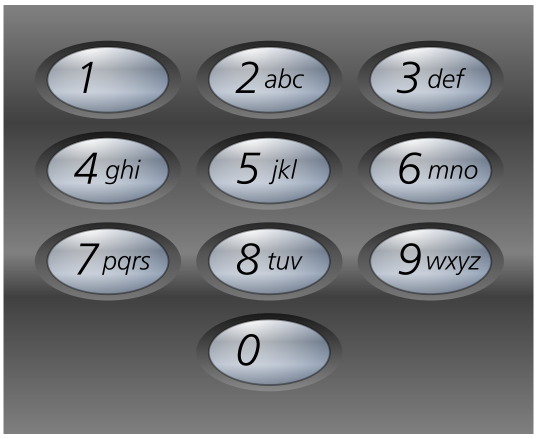


Figure D.1: The keypad of a telephone, where some numbers are assigned several letters.

Public Domain by Sakurambo on Wikimedia Commons, modified

Input

The input consists of:

- One line with two integers n and m ($1 \leq n, m \leq 10^5$, $n \cdot m \leq 10^5$), the number of words and the number of phone numbers.
- n lines, each with a word w ($1 \leq |w| \leq 10$), consisting only of English lowercase letters (`a-z`). The words are unique and given in alphabetical order.
- m lines, each with a phone number p ($1 \leq |p| \leq 10$), consisting only of digits that correspond to letters in a mnemonic phrase (`2-9`).

Output

For each phone number, output the number of words that are a valid mnemonic phrase for this phone number, followed by these words in alphabetical order.

Sample Input 1

```
5 3
algorithm
bapc
benelux
contest
progaming
2272
424242
2363589
```

Sample Output 1

```
1 bapc
0
1 benelux
```

Sample Input 2

```
3 1
peer
reds
refs
7337
```

Sample Output 2

```
3 peer reds refs
```

Sample Input 3

```
7 3
black
judge
my
of
quartz
sphinx
vow
25225
782789
774466
```

Sample Output 3

```
1 black
1 quartz
0
```

Sample Input 4

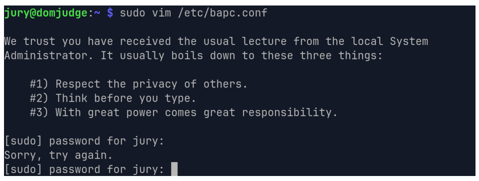
```
5 1
and
bland
e
land
of
63
```

Sample Output 4

```
1 of
```

E Expected Error

Time limit: 1s



```
jury@domjudge:~ $ sudo vim /etc/bapc.conf

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for jury:
Sorry, try again.
[sudo] password for jury:
```

You are typing your password at a `sudo` prompt, but suddenly, one of your fingers slipped onto a wrong key. Because the terminal hides the characters that you type, you are uncertain whether you have typed an extra character. To finish typing your password, you consider three possible strategies.

1. *Continue* to type the rest of your password.
2. Press *backspace* to delete the last character and type the rest of your password.
3. *Restart* typing your password from scratch.

To determine the optimal strategy, make use of the following typing speed assumptions.

- Typing any character of your password takes 0.1 seconds.
- Pressing backspace or submitting your password also takes 0.1 seconds.
- To restart typing your password, you delete all characters, which takes 0.3 seconds.
- If you submit the wrong password, it takes an additional 0.3 seconds to realize and start typing your password at a new, empty prompt.

You are given the number of characters in your password n, the number of correctly typed characters k before your finger slipped, and a probability of $p\%$ indicating the likelihood that you pressed a wrong key and ended up with $k + 1$ characters. Assuming you make no further errors, determine which strategy yields the lowest expected time to finish typing your password.

Input

The input consists of:

- One line with three integers n, k, and p ($1 \leq n \leq 1000$, $1 \leq k \leq n$, $0 \leq p \leq 100$), the number of characters in your password, the number of characters you correctly typed before your finger slipped, and the probability percentage that you pressed a wrong key.

Output

Output one of the strings "`continue`", "`backspace`", or "`restart`" indicating the optimal strategy. It is guaranteed that the expected time of the optimal strategy is at least one millisecond shorter than the other strategies.

Sample Input 1
```
10 8 20
```
Sample Output 1
```
continue
```

Sample Input 2
```
10 8 80
```
Sample Output 2
```
backspace
```

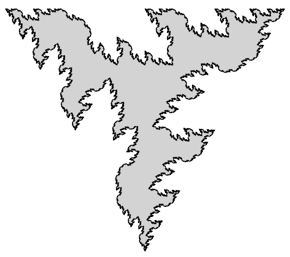
Sample Input 3
```
10 2 50
```
Sample Output 3
```
restart
```

Sample Input 4
```
10 4 55
```
Sample Output 4
```
restart
```

F Fractal Area

Time limit: 1s

The director of the local Mathematical Institute has decided to brighten up the walls by adding some pictures of Bounded Auto-similar Periodic Curves, which are geometric structures usually known as fractals. The director has some great ideas for beautiful fractals, but they are not sure whether these will fit on the walls of the institute.


The fractal from the second sample.

Since these fractals will be painted all over the walls of the institute, the director has asked you to determine the area of these fractals, so they know exactly how much paint they will need to use for this.

The fractals are constructed from a polyline[1] between $(0, 0)$ and $(1, 0)$. Starting with an equilateral triangle with side length 1, each segment of the boundary is recursively replaced by a scaled and rotated version of the original polyline, so that the endpoints and orientation match.

As an example, consider the first sample case, visualized in Figure F.1. The resulting fractal in this case is called the *Koch Snowflake*.

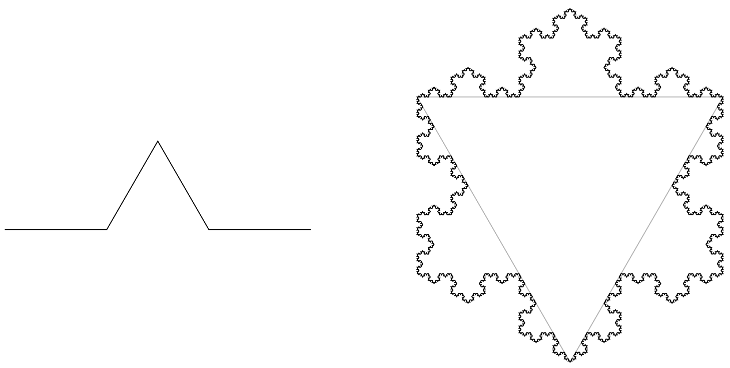


Figure F.1: Visualization of the first sample case, with the given polyline on the left and the resulting fractal on the right.

Input

The input consists of:

- One line with an integer n ($2 \leq n \leq 1000$), the number of points defining the polyline.
- n lines with two floating point numbers x and y ($0 \leq x \leq 1$, $|y| < 0.5$), the coordinates of a point defining the polyline.

All floating point numbers consist of exactly 6 digits behind the decimal point. It is guaranteed that the first point is $(0, 0)$ and the last point is $(1, 0)$. The resulting fractal converges and does not overlap with itself.

[1]A polyline is a shape made by connecting a series of straight line segments at their endpoints.

Output

Output the area of the resulting fractal.

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

Sample Input 1

```
5
0.000000 0.000000
0.333333 0.000000
0.500000 0.288675
0.666667 0.000000
1.000000 0.000000
```

Sample Output 1

```
0.692820545
```

Sample Input 2

```
5
0.000000 0.000000
0.200000 0.000000
0.600000 -0.200000
0.500000 0.000000
1.000000 0.000000
```

Sample Output 2

```
0.237360528
```

G Giganotosaurus Game

Time limit: 2s

Suffering from a poor internet connection, you are playing a casual game in your web browser to pass the time. You, the player, control a Giganotosaurus that is running through a linear world with obstacles (cactuses). You win the game if you reach the end of the world without hitting any cactuses.

The world consists of n cells, which can either be empty or contain a cactus. You start at the leftmost cell (which is always empty) and the goal is to get past the rightmost cell. At each cell, the Giganotosaurus can either move one position to the right, or jump over some fixed number of cells. For the first jump, you skip one cell, but with each subsequent jump, you skip one additional cell compared to the previous jump. That is, the kth jump skips exactly k cells.

You quickly master this simple game, so you pose a more interesting challenge: count how many ways there are to win the game. As an example, consider the second sample case, visualized in Figure G.1.

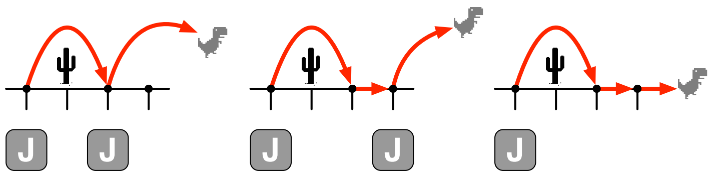


Figure G.1: Visualization of the second sample input, for which there are three ways to win the game.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 10^5$), the length of the world.
- One line with n characters, each character being either '#' or '.', indicating a cactus or an empty cell, respectively.

Output

Output the number of ways to win the game, modulo $10^9 + 7$.

Sample Input 1	Sample Output 1
4	8

Sample Input 2	Sample Output 2
4 .#..	3

Sample Input 3

```
7
.#...##
```

Sample Output 3

```
1
```

Sample Input 4

```
7
..#.#.#
```

Sample Output 4

```
0
```

H Human Pyramid

Time limit: 1s

As chairman of the Building A Pyramid Committee, you are specialized in breaking one specific world record: building the highest human pyramid. Unfortunately, you only know a limited number of people who are willing to be in the pyramid. After all, building human pyramids does not make much money, so most people are volunteers.

An example of a non-world record breaking pyramid. CC BY-SA 3.0 by Amotoki on Wikimedia Commons.

A full human pyramid of height h consists of h layers of people. As seen from below, it has h people on the first layer, $h-1$ on the second, $h-2$ on the third, and so forth until eventually the final layer has just a single person. To determine whether you can break the world record, you need to know how high a pyramid you can build. Given how many people are available, how tall is the highest possible human pyramid that these people can make?

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 10^{12}$), the number of people available to build the pyramid.

Output

Output the height of the highest possible pyramid you can build with n people.

Sample Input 1	Sample Output 1
3	2

Sample Input 2	Sample Output 2
12	4

I Investment Investigation

Time limit: 4s

To make some extra money on the side, you have recently started running your own cryptocurrency exchange, where people can trade their Budget Amplifying Profit Coin (BAPC). It is quickly gaining popularity, however, this has also resulted in government regulators asking some questions... As part of their investigation, they have asked for a list of all transactions that have been made via your exchange. You have never bothered to keep track of this, but luckily, you still have the list of all orders that were made since the start of the exchange.

Contentedly looking at the value of your BAPC going through the roof.
Internet meme, fair use

The exchange operates by keeping a list of outstanding buy and sell orders, each with a price and an amount. Whenever a *normal* order comes in, it is checked whether the new lowest sell price is less than or equal to the highest buy price. If this is the case, a transaction is made between the sell order with the lowest price and the buy order with the highest price, such that at least one of these orders is completely fulfilled. In case of a tie in price, older orders are fulfilled first. This is repeated until the lowest sell price is strictly larger than the highest buy price.

If instead a *Fill-or-Kill* (FoK) buy order comes in, there must currently be enough outstanding sell orders with a price of at most the offered price to completely fulfil this order. If there are, the order will be fulfilled in the same way as a normal order. Otherwise, the order is completely cancelled, without any transaction taking place. Note that multiple orders may be used to complete a FoK order, as long as it happens immediately.

FoK sell orders are processed in a similar way, but then there should be sufficient outstanding buy orders with a price of at least the asked price.

As an example, consider the first sample case. The six orders are handled as follows:

1. The first order is added to the list of outstanding orders.
2. The second order is partially fulfilled by selling 10 BAPC to the first order. This removes the first order from the list of outstanding orders, and adds the remainder of the second order (consisting of 10 BAPC) to this list.
3. The third order is added to the list of outstanding orders.
4. The fourth order is a FoK buy order that cannot be immediately fulfilled, so it is ignored. It is not added to the list of outstanding orders.
5. The fifth order can be immediately fulfilled by first buying 10 BAPC from order 2 and then buying 50 BAPC from order 3. The resulting list of outstanding orders only consists of the remaining 8 BAPC of order 3.
6. The sixth order is added to the list of outstanding orders.

Given a list of all orders in the order that they have been made, create a list of all transactions that have been performed by your exchange.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 10^5$), the number of orders.
- n lines, each describing an order:
  - A string s, either "`buy`" or "`sell`", the side of the order.
  - A string t, either "`normal`" or "`fok`", the type of the order.
  - An integer p ($1 \leq p \leq 10^9$), the offered or asked price per BAPC.
  - An integer a ($1 \leq a \leq 10^9$), the amount of BAPC being asked or offered.

Output

The output consists of the number of performed transactions, and then for each transaction, in the order that they have been performed:

- The index of the corresponding "`sell`" order.
- The index of the corresponding "`buy`" order.
- The amount of BAPC being traded.

Here, the index of an order is its position in the input, where the first order has index 1.

Sample Input 1	Sample Output 1

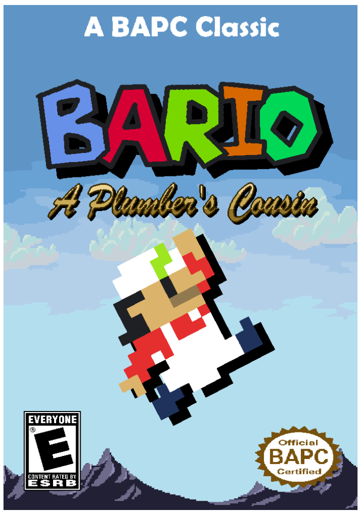
6
buy normal 700 10
sell normal 500 20
sell normal 800 58
buy fok 600 30
buy fok 900 60
sell normal 300 42
	3
	2 1 10
	2 5 10
	3 5 50

Sample Input 2	Sample Output 2

3
buy normal 19 10
buy normal 19 20
sell fok 19 17
	2
	3 1 10
	3 2 7

J Joppiesaus Jailbreak

Time limit: 2s



You have recently decided to pick up speedrunning the video game *Bario: A Plumber's Cousin*. In this 2D platforming console classic, you play as Bario, an Italian electrician travelling the world to find his long lost cousin. The game consists of a number of side-scrolling levels with a bus at the end that takes Bario to the next level. Unfortunately, years of optimizations have led to a world record that is currently tied between hundreds of speedrunners and you feel like matching the world record at this point is no longer that big of an achievement. Instead, you try to beat the tied world record by any means necessary.

At first, this seems impossible: Bario has a maximum right speed of 1000 pixels per second, and the current strategies already hold this speed through the entire level. However, completing a level always takes an integer number of frames. If Bario reaches the bus halfway through a frame, the game still has to wait for the frame to complete before starting the next level. Normally, this does not influence speedrunning, as each console runs the game at the same, constant frame rate f. That is, unless you apply a specific condiment mix to the game disk. You would prefer not to go into detail as to how you know this, but applying a specific mix of mayonnaise and curry spices (more commonly known as the Dutch specialty Joppiesaus) to the game disk allows you to set the frame rate of the game to any positive real number. This new frame rate cannot exceed the original frame rate f and remains constant for the entire game. Using your new strategy, what is the fastest time in which you can finish the game? The timing stops when the final frame ends.

For example, consider the third sample input. By modifying the game to run at $\frac{3000}{1249}$ frames per second, both levels complete in 15 frames, or 6.245 seconds. The total time of 12.49 seconds beats the current world record of 12.6 seconds at the original 10 frames per second.

Input

The input consists of:

- One line with two integers n and f ($1 \leq n \leq 10^5$, $1 \leq f \leq 10^3$), the number of levels and the original frame rate of the game in frames per second.
- One line with n integers ℓ ($1 \leq \ell \leq 10^6$), the length of each level in pixels. The total length of all levels does not exceed 10^6 pixels.

Output

Output the fastest time, in seconds, in which you can finish the game.

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

Sample Input 1

```
1 10
1234
```

Sample Output 1

```
1.234
```

Sample Input 2

```
1 10
12
```

Sample Output 2

```
0.1
```

Sample Input 3

```
2 10
6245 6212
```

Sample Output 3

```
12.49
```

Sample Input 4

```
2 20
6245 6212
```

Sample Output 4

```
12.47409677
```

Sample Input 5

```
3 50
7146 2657 8164
```

Sample Output 5

```
17.96910941
```

K Kitchens of Königsberg

Time limit: 8s

The year is 1764 (or 900 in base-14). During the past few decades, the bridges of Königsberg have become a major attraction for combinatorics tourism from all over the world. In a recent travel brochure, your predecessor on the Königsberg Board of Tourism has promised the existence of k Bridges Alongside Palatable Cuisine, where hungry graph theorists can combine their intellectual and culinary pursuits by finishing their pilgrimage with a delicious bowl of traditional meatballs from a charming street kitchen. Alas, none of these kitchens have actually been built, so this will be your first task!

Naturally, you begin by modelling Königsberg as an undirected multigraph. Rivers divide the city into areas, which you model as vertices, and the bridges become the edges. With this abstraction, you start to investigate whether it is possible to select areas to place kitchens in, so that exactly k bridges end in an area with a kitchen. As an example, consider the first sample case, shown in Figure K.1.

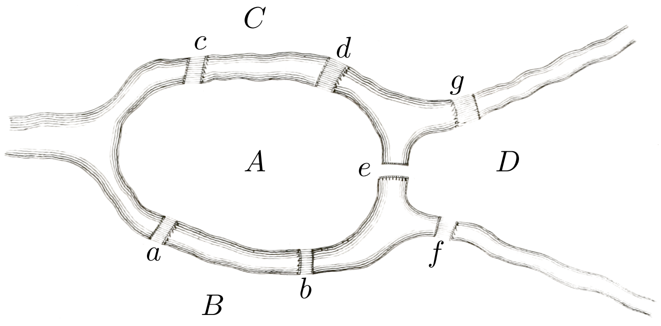


Figure K.1: Visualization of the first sample case. If kitchens are placed at areas A and B, then the 6 bridges a, b, c, d, e, and f are serviced. Another solution would be to place kitchens at B and C.

Modified from *Solutio problematis ad geometriam situs pertinentis* by Leonhard Euler

Input

The input consists of:

- One line with three integers n, m, and k ($1 \leq n \leq 5000$, $0 \leq m \leq 50\,000$, $1 \leq k \leq 6$), the number of areas, the number of bridges, and the number of bridges that must end in an area with a kitchen.
- m lines, each with two integers a and b ($1 \leq a < b \leq n$), indicating a bridge between areas a and b. Note that there can be multiple bridges between the same pair of areas.

Output

If there is a subset of areas in which kitchens can be placed, so that exactly k bridges end in an area with a kitchen, output the number of areas in this subset, followed by these areas. Otherwise, output "`impossible`".

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

Sample Input 1

```
4 7 6
1 2
1 2
1 3
1 3
1 4
2 4
3 4
```

Sample Output 1

```
2
1 2
```

Sample Input 2

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

Sample Output 2

```
3
4 2 3
```

Sample Input 3

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

Sample Output 3

```
6
8 7 6 4 3 2
```

Sample Input 4

```
5000 0 1
```

Sample Output 4

```
impossible
```

Sample Input 5

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

Sample Output 5

```
impossible
```

L Lawful Limits

Time limit: 2s

One late afternoon you are driving to get home in your Big And Pricey Car. You have had a long day and are eager to get home as soon as possible. Your country's road network has many roads with varying speed limits, and has one strange quirk: at some time t, the maximum speed on each road is raised. Because you want to get home as soon as possible, you instantly increase your speed to the new maximum speed of the road you are on at time t.

In this road network, the speed limit increases at 19:00.

You start driving at time 0 at junction 1 and are going to n. What is the earliest time you can reach your destination? As an example, consider the first sample case, visualized in Figure L.1.

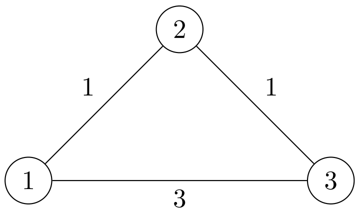


Figure L.1: Visualization of the first sample input. The edges are marked with their lengths. On all roads, the maximum speed is 1 before time t and 2 from time t onwards.

Input

The input consists of:

- One line with three integers n, m, and t ($2 \leq n \leq 10^5$, $1 \leq m \leq 10^5$, $0 \leq t \leq 10^9$), the number of junctions, the number of roads, and the time the speed limit increases.
- m lines, each with five integers x, y, ℓ, v, and w ($1 \leq x, y \leq n$, $1 \leq \ell \leq 10^9$, $1 \leq v < w \leq 10^9$), the start and end junction of a road, length of this road, and the speed limits on this road before time t and from time t onwards.

There is at most one road between any two junctions, and one can travel in both directions on any road. No road leads from one junction to that same junction. It is guaranteed that there is always a path between any two junctions.

Output

Output the minimum amount of time it takes to get from the start to your destination.

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

Sample Input 1

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

Sample Output 1

```
1.5
```

Sample Input 2

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

Sample Output 2

```
2.0
```

Sample Input 3

```
4 4 6
1 2 30 4 6
1 3 12 6 8
2 4 16 4 8
3 4 30 5 10
```

Sample Output 3

```
7.0
```