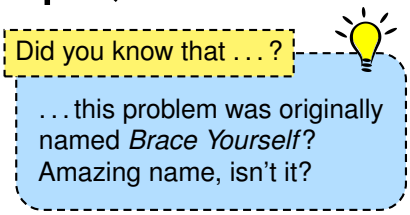
Problem X (don't pick as seminar topic)



When writing text, people sometimes rely on braces to inject additional details to certain statements. Due to human defectiveness, braces (especially closing ones are prone to be missing. A different race – the compilers – heavily relies on braces to be balanced and well nested. To prevent misapprehensions and ensure proper interaction between humans and compilers, all humans' texts need to be verified prior to passing them on to the compilers.

The pairs of characters `( )`, `{ }`, `[ ]`, and `< >` are used as control symbols in source code and each define a *control block*. An opening brace (i.e., one of `({[<`) marks the start of a control block, the corresponding closing brace (i.e., the matching brace `)}]>`) marks the end of the control block. Control blocks can encapsulate other control blocks, but no two control blocks may intersect each other. A text may also contain *string literals*, which are character sequences delimited by a pair of quotation marks `"`. Quotation marks cancel out control characters, which means that after the first quotation mark no character is interpreted as control character until the next quotation mark. Write a source code check that tests if all string literals and control blocks are valid and closed.

Input

The input consists of:

- One or more lines (separated by `\n`) made up of whitespace, numbers, alphabetic characters, and all other printable ASCII characters. More precisely, the set of allowed characters is the ASCII range `0x20-0x7e`, as well as `0x09` (tab) and `0x0a` (newline).

The total number of characters in the input is at most $1\,048\,576$ ($= 1$ MiB).

Output

Output `correct` if all control blocks and strings are valid and closed or `incorrect` if not.

Sample Input 1

```
{ [ < "this is a > string" > ]
this is not a string } ()
```

Sample Output 1

```
correct
```

Sample Input 2

```
{ [ < "this is a > string " ]
this is not a string } ()
```

Sample Output 2

```
incorrect
```

Problem A: Ghost Methods

Once again you lost days refactoring code, which never runs in the first place. Enough is enough – your time is better spent writing a tool that finds unused code!

Your software is divided into packages and executables. A package is a collection of methods. Executables are packages defining among other methods exactly one method with name `PROGRAM`. This method is executed on the start of the corresponding executable. Ordinary packages have no method named `PROGRAM`.

Each method is uniquely identified by the combination of package and method names. E.g. the method with the identifier `SuperGame::PROGRAM` would be the main method of the executable `SuperGame`.

For every method in your software you are given a list of methods directly invoking it. Thus you can easily identify methods, that are never called from any method. However, your task is more challenging: you have to find unused methods. These are methods that are never reached by the control flow of any executable in your software.

Input

The first line of the input contains an integer N, the number of methods in your software ($1 \leq N \leq 400$).

Each method is described by two lines, totaling in $2 \cdot N$ lines. The first line consists of the unique identifier of the method and k_i, the number of methods directly invoking this one ($0 \leq k_i \leq N$). The second line consists of a set of k_i identifiers of these calling methods or is empty if there are no such methods, i.e. $k_i = 0$.

Method identifiers consist of a package name followed by two colons and a method name like `Packagename::Methodname`. Both strings, the package and the method name, each consist of up to 20 lowercase, uppercase characters or digits (`a-z`, `A-Z`, `0-9`).

There will be exactly N different method identifiers mentioned in the input.

Output

A line containing the number of unused methods in your software.

Sample Input 1

```
2
SuperGame::PROGRAM 0

HelpPackage::HelpFunction 2
HelpPackage::HelpFunction SuperGame::PROGRAM
```

Sample Output 1

```
0
```

Sample Input 2

```
2
Loop::CallA 1
Loop::CallB
Loop::CallB 1
Loop::CallA
```

Sample Output 2

```
2
```

Sample Input 3

```
2
SuperGame::PROGRAM 1
SuperServer42::PROGRAM
SuperServer42::PROGRAM 1
SuperServer42::PROGRAM
```

Sample Output 3

```
0
```

Problem B: Let's Call a Taxi

For a long time Tim wanted to visit Greece. He has already purchased his flight to and from Athens. Tim has a list of historical sites he wants to visit, e.g., Olympia and Delphi. However, due to recent political events in Greece, the public transport has gotten a little complicated. To make the Greek happy and content with their new government, many short-range bus and train lines have been created. They shall take the citizens around in their neighborhoods, to work or to their doctor. At the same time, long-range trains that are perfect for tourists have been closed down as they are too expensive. This is bad for people like Tim, who really likes to travel by train. Moreover, he has already purchased the Greece' Card for Public Conveyance (GCPC) making all trains and buses free for him.

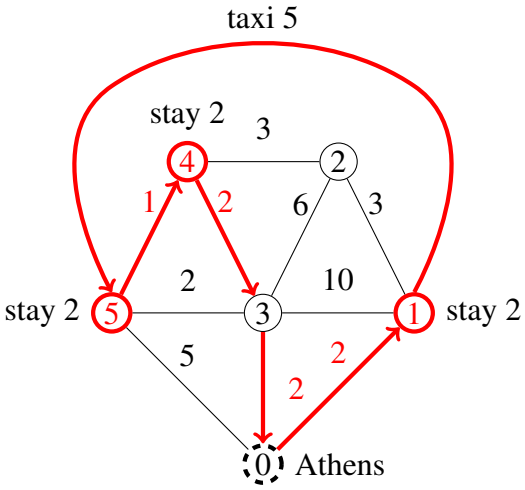


Figure A.1: Visual representation of the Sample Input: Tim's tour has length 18.

Despite his preferred railway lines being closed down, he still wants to make his travel trough Greece. But taking all these local bus and train connections is slower than expected, so he wants to know whether he can still visit all his favorite sites in the timeframe given by his flights. He knows his schedule will be tight, but he has some emergency money to buy a single ticket for a special Greek taxi service. It promises to bring you from any point in Greece to any other in a certain amount of time.
For simplicity we assume, that Tim does never have to wait for the next bus or train at a station. Tell Tim, whether he can still visit all sites and if so, whether he needs to use this taxi ticket.

Input

The first line contains five integers N, P, M, G and T, where N denotes the number of places in Greece, P the number of sites Tim wants to visit, M the number of connections, G the total amount of time Tim can spend in Greece, and T the time the taxi ride takes ($1 \leq N \leq 2 \cdot 10^4$; $1 \leq P \leq 15$; $1 \leq M, G \leq 10^5$; $1 \leq T \leq 500$).
Then follow P lines, each with two integers p_i and t_i, specifying the places Tim wants to visit and the time Tim spends at each site ($0 \leq p_i < N$; $1 \leq t_i \leq 500$). The sites p_i are distinct from each other.
Then follow M lines, each describing one connection by three integers s_i, d_i and t_i, where s_i and d_i specify the start and destination of the connection and t_i the amount of time it takes ($0 \leq s_i, d_i < N$; $1 \leq t_i \leq 500$).
All connections are bi-directional. Tim's journey starts and ends in Athens, which is always the place 0.

Output

Print either "`impossible`", if Tim cannot visit all sites in time, "`possible without taxi`", if he can visit all sites without his taxi ticket, or "`possible with taxi`", if he needs the taxi ticket.

Sample Input 1

```
6 3 10 18 5
1 2
4 2
5 2
0 1 2
1 2 3
2 4 3
1 3 10
2 3 6
0 3 2
3 4 2
4 5 1
3 5 2
0 5 5
```

Sample Output 1

```
possible with taxi
```

Problem C: Intergalactic

Space pirate Captain Krys has recently acquired a map of the artificial and highly secure planet Alpha-Zet which he has been planning to raid for ages. It turns out the whole planet is built on a 2D plane with modules that serve as one room each. There is exactly one module at every pair of integer coordinates and modules are exactly 1×1 units big. Every module is bidirectionally connected to at least one adjacent module. Also, for any two modules there exists exactly one path between them. All in all the modules create a rectangular maze without any loops.

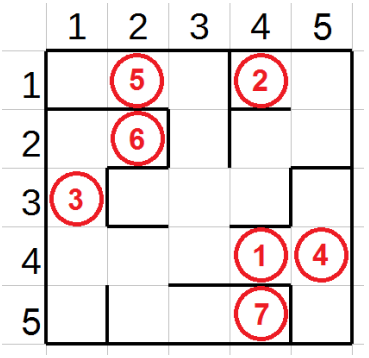


Figure A.1: Illustration of Sample Input 2

On the map Captain Krys has marked several modules he wants to visit in exactly the marked order. What he intends to do there is none of your business, but he promises you a fortune if you determine the number of modules he has to walk through along the route (since there are no loops he will always take the direct route from one marked module to the next). The first marked module indicates where he starts his journey, the last where he wants to finish.

Input

The input consists of:

- one line with two integers h and w ($2 \leq h, w \leq 1\,000$) describing the height and the width of the maze.
- $h + 1$ lines follow, describing the maze in ASCII, each line containing $2 \cdot w + 1$ characters. The description always follows these rules:
  - In every row, columns with odd index (starting at index 1) contain either vertical walls or spaces and columns with even index contain either horizontal walls or spaces.
  - The first row describes the northern wall of the maze (which always consists only of horizontal walls). Every subsequent row describes a row of modules.
  - A module is located at every even column index. Its western and eastern walls are located at the directly neighboring odd column indices respectively, its northern wall is located at the same column index but one row above and its southern wall can be found at its own position. If a wall is missing, the corresponding position contains a space instead.
- After the description of the maze, an integer m ($2 \leq m \leq 10^4$) is given.
- Each of the following m lines describes a marked module with two integer coordinates x and y ($1 \leq x \leq h$; $1 \leq y \leq w$). The first pair of coordinates is the start point of the journey, the last pair the end point. Modules may appear multiple times but never twice or more in a row. $(1, 1)$ is the top left module and (h, w) is the bottom right module.

It is guaranteed that the maze itself is enclosed. Furthermore it is guaranteed that exactly one path exists between any two modules.

Output

Output one integer, the number of modules Captain Krys has to travel through if he follows the route in the exact order given in the input.

Sample Input 1

```
2 6
 _ _ _ _ _ _
|  _ _ _ _ _|
|_ _ _ _ _ _|
5
1 5
1 1
1 6
1 1
1 5
```

Sample Output 1

```
18
```

Sample Input 2

```
5 5
 _ _ _ _ _
|_ _   |_  |
|  _|  |  _|
| |_   _| |
|    _ _  |
|_|_ _ _|_|
7
4 4
1 4
3 1
4 5
1 2
2 2
5 4
```

Sample Output 2

```
43
```

Problem D: unum modo platea

In the country of Via, the cities are connected by roads that can be used in both directions. However, this has been the cause of many accidents since the lanes are not separated: The drivers frequently look at their smartphones while driving, causing them to collide with the oncoming traffic. To alleviate the problem, the politicians of Via came up with the magnificent idea to have one-way roads only, i.e., the existing roads are altered such that each can be only used in one of two possible directions. They call this "one-way-ification".

The mayors do not want too many one-way roads to lead to their cities because this can cause traffic jam within the city: they demand that the smallest integer d be found such that there is a 'one-way-ification' in which for every city, the number of one-way roads leading to it is at most d.

Input

The input consists of:

- one line with an integer n ($1 \leq n \leq 500$), where n is the number of cities labeled from 1 to n;
- one line with an integer m ($0 \leq m \leq 2.5 \cdot 10^3$), where m is the number of (bi-directional) roads;
- m lines describing the roads. Each road is described by:
  - one line with two integers a and b ($1 \leq a, b \leq n$, $a \neq b$) indicating a road between cities a and b.

There is at most one road between two cities.

Output

Output the minimum number d.

Sample Input 1

```
2
1
1 2
```

Sample Output 1

```
1
```

Sample Input 2

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

Sample Output 2

```
2
```

Problem E: Why is this cable not longer?

Adam just moved into his new apartment and simply placed everything into it at random. This means in particular that he did not put any effort into placing his electronics in a way that each one can have its own electric socket.

Since the cables of his devices have limited reach, not every device can be plugged into every socket without moving it first. As he wants to use as many electronic devices as possible right away without moving stuff around, he now tries to figure out which device to plug into which socket. Luckily the previous owner left behind a plugbar which turns one electric socket into 3.

Can you help Adam figure out how many devices he can power in total?

Input

The input consists of:

- one line containing three integers m, n and k, where
  - m ($1 \leq m \leq 1\,500$) is the number of sockets;
  - n ($1 \leq n \leq 1\,500$) is the number of electronic devices;
  - k ($0 \leq k \leq 75\,000$) is the number of possible connections from devices to sockets.
- k lines each containing two integers x_i and y_i indicating that socket x_i can be used to power device y_i.

Sockets as well as electronic devices are numbered starting from 1.

The plugbar has no cable, i.e. if it is plugged into a socket it simply triples it.

Output

Output one line containing the total number of electrical devices Adam can power.

Sample Input 1

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

Sample Output 1

```
5
```

Sample Input 2

```
4 5 11
1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
4 4
4 5
```

Sample Output 2

```
5
```

Sample Input 3

```
3 5 7
1 1
1 2
2 2
2 3
2 4
3 4
3 5
```

Sample Output 3

```
5
```

Problem F: Those who trespass against us

Somewhere in the great North American plains live the tribes of chiefs *Blue Eagle*, *Red Beaver*, and *Green Serpent*. Their population is scattered over numerous villages all over the land and conflict arises whenever members of different tribes meet while traveling across the plains.

To put an end to the constant animosities the chiefs have decided that the land should be divided between the tribes so that they can avoid each other when moving between villages belonging to the same tribe. More precisely, they want to construct two border fences – thus dividing the land into three regions – such that two villages lie in the same region precisely when they belong to the same tribe.

The villages are represented by points in the Euclidean plane that are colored blue, red or green, depending on the tribe, and the fences should be drawn in the form of two polygons. The polygons may not touch or intersect themselves or each other and none of the points may lie on their boundary. (Make sure to read the constraints in the Output section!)

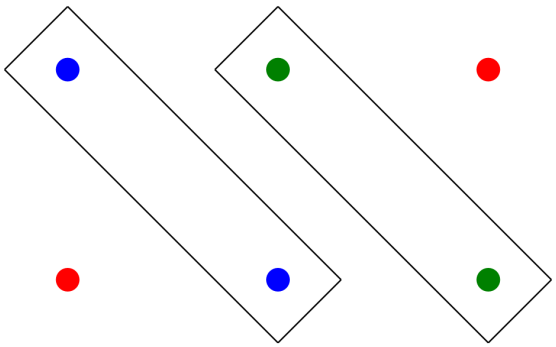


Figure A.1: Illustration of the sample.

Input

The input consists of:

- one line with an integer n ($3 \leq n \leq 100$), the number of villages.
- n lines, each with three integers x, y, c ($-1\,000 \leq x, y \leq 1\,000, 1 \leq c \leq 3$), representing a village at coordinates (x, y) of color c ($1 =$ blue, $2 =$ red, $3 =$ green). All positions are unique. There is at least one village of each color.

Output

If there is no solution, print `impossible`. Otherwise, print the two polygons in the following format:

- one line with an integer m ($3 \leq m \leq 1\,000$), the number of vertices of the polygon.
- m lines, each with two real numbers x, y ($-3\,000 \leq x, y \leq 3\,000$), the vertices of the polygon in either clockwise or counter-clockwise order. The numbers may be given with up to five decimal places (additional places will be rounded off).

Sample Input 1

```
6
0 0 2
0 1 1
1 0 1
1 1 3
2 0 3
2 1 2
```

Sample Output 1

```
4
-0.3 1.0
1.0 -0.3
1.3 0.0
0.0 1.3
4
0.7 1.0
2.0 -0.3
2.3 0.0
1.0 1.3
```

Problem G: Patent Claims

The year is 1902. Albert Einstein is working in the patent office in Bern. Many patent proposals contain egregious errors; some even violate the law of conservation of energy. To make matters worse, the majority of proposals make use of non-standard physical units that are not part of the metric system (or not even documented). All proposals are of the following form:

- Every patent proposal contains n energy converters.
- Every converter has an unknown input energy unit associated with it.
- Some energy converters can be connected: If converter a can be connected to converter b such that one energy unit associated with a is turned into c input units for b, then this is indicated by an arc $a \xrightarrow{c} b$ in the proposal. The output of a can be used as input for b if and only if such an arc from a to b exists.

Einstein would like to dismiss all those proposals out of hand where the energy converters can be chained up in a cycle such that more energy is fed back to a converter than is given to it as input, thereby violating the law of conservation of energy.

Einstein's assistants know that he is born for higher things than weeding out faulty patent proposals. Hence, they take care of the most difficult cases, while the proposals given to Einstein are of a rather restricted form: Every *admissible* patent proposal given to Einstein does not allow for a cycle where the total product of arc weights exceeds 0.9. By contrast, every *inadmissible* patent proposal given to Einstein contains a cycle where the the number of arcs constituting the cycle does not exceed the number of converters defined in the proposal, and the total product of arc weights is greater or equal to 1.1.

Could you help Einstein identify the inadmissible proposals?

Input

The input consists of:

- one line with two integers n and m, where
  - n ($2 \leq n \leq 800$) is the number of energy converters;
  - m ($0 \leq m \leq 4000$) is the number of arcs.
- m lines each containing three numbers a_i, b_i, and c_i, where
  - a_i and b_i ($1 \leq a_i, b_i \leq n$) are integers identifying energy converters;
  - c_i ($0 < c_i \leq 5.0$) is a decimal number

  indicating that the converter a_i can be connected to the converter b_i such that one input unit associated with a_i is converted to c_i units associated with b_i. The number c_i may have up to 4 decimal places.

Output

Output a single line containing `inadmissible` if the proposal given to Einstein is inadmissible, `admissible` otherwise.

Sample Input 1

```
2 2
1 2 0.5
2 1 2.3
```

Sample Output 1

```
inadmissible
```

Sample Input 2

```
2 2
1 2 0.5
2 1 0.7
```

Sample Output 2

```
admissible
```

Problem H: Keychain Shuffle

When Julia was a child, she just had one key for her savings box. But with greater age comes greater responsibility. Today, she owns n distinct keys: one for her apartment, her mail box, her car, her secret box filled with chocolate bars, and so on. It got to a point where it sometimes becomes a hassle to find the right key whenever she needs to open a lock. Luckily, Julia owns a metallic ring which serves as her key chain. In order to find her keys quicker, she always keeps her n distinct keys in the exact same order on the key chain. Last night, however, Julia's sister Anna was bored and started playing around with Julia's key chain. Unfortunately, the keys were not in the original order when she handed the key chain back to Julia. Annoyed by this childish behaviour, Julia started detaching and reattaching keys from her key chain that seemed to be in a wrong position. Now she wonders what the minimum number of keys is that she needs to detach and reattach from her key chain in order to reestablish the correct order. Note that it does not matter to Julia whether she reestablishes the correct order in clockwise or counterclockwise order as she can always flip around her key chain.

Did you know that . . . ?

. . . shortly before the maiden voyage of the RMS Titanic, the seaman who kept the keys to the storage locker containing the binoculars intended for use by the crow's nest lookout was removed from its command roster? Due to his hasty departure, he accidentally kept the key, and the absence of any binoculars within the crow's nest is believed to be one of the main contributory factors in the Titanic's ultimate demise.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 1\,000$), the number of keys on Julia's key chain. All keys are distinct and have a unique label ranging from 1 to n.
- One line with n integers $k_1, \ldots, k_n$ ($1 \leq k_i \leq n$ for all i), the original order of Julia's keys on her key chain that she wants to reestablish. All keys in the original order are distinct, i.e., $k_1, \ldots, k_n$ form a permutation of $[n]$.
- One line with n integers $\ell_1, \ldots, \ell_n$ ($1 \leq \ell_i \leq n$ for all i), the order of Julia's keys on her key chain after Anna has played with them. All keys in the modified order are distinct, i.e., $\ell_1, \ldots, \ell_n$ form a permutation of $[n]$.

Output

Output a single integer z where z is the minimum number of detach/reattach operations necessary to reestablish the correct order.

Sample Input 1	Sample Output 1
3 1 3 2 2 3 1	0

Problem K: Daily Dividend

David is a young boy and he loves numbers. Recently he learned how to divide two numbers. David divides the whole day. He is happy if the result of the division is an integer, but he is not very amused if this is not the case. After quite a while he decided to use only a single dividend each day.

The parents of David are very careful and they would like to ensure that David experiences enough happiness. Therefore they decide which number David will use as the dividend for this day.

There is still a problem: The parents are not very good at math and don't know how to calculate the number of positive integral divisors for a given dividend N, which lead to an integral result.

Now it's up to you to help David's parents.

Input

The single input line contains the single integer N, where N is chosen as a dividend ($1 \leq N \leq 10^{18}$).

Output

Print the number of positive integral divisors of N that lead to an integral result of the division.

Sample Input 1	Sample Output 1
12	6
Sample Input 2	**Sample Output 2**
999999999999999989	2
Sample Input 3	**Sample Output 3**
100000007700000049	4

Problem L: Hiking Preparations

The Chilean Andes have become increasingly popular as a destination for backpacking and hiking. Many parts of the Andes are quite remote and thus dangerous. Because of this, the Ministry of Tourism wants to help travelers plan their trips. In particular, the travelers need to know how high they will have to climb during their journey, as this information will help them decide which equipment they need to bring. The Ministry has tasked you to provide the aspiring mountaineers with this data.

You are given a topographic map of a part of the Andes, represented as a two-dimensional grid of height values, as well as the list of origins and destinations. Mountaineers can move from each grid cell to any of the four adjacent cells. For each mountaineer find the minimal height that they must be able to reach in order to complete their journey.

Input

The input consists of:

- one line with three integers m, n and q ($1 \leq m, n \leq 500, 1 \leq q \leq 10^5$), where m is the number of rows, n is the number of columns, and q is the number of mountaineers;
- m lines, each with n integers $h_1, \ldots, h_n$ ($1 \leq h_i \leq 10^6$), the height values in the map;
- q lines, each with four integers x_1, y_1, x_2, y_2 ($1 \leq x_1, x_2 \leq m, 1 \leq y_1, y_2 \leq n$), describing a mountaineer who wants to trek from (x_1, y_1) to (x_2, y_2).

The top left cell of the grid has coordinates $(1, 1)$ and the bottom right cell has coordinates (m, n).

Output

Output q integers, the minimal height for each mountaineer, in the same order as in the input.

Sample Input 1

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

Sample Output 1

```
2
4
3
```

Problem M: Train Reversal

The German train services have a problem. Due to construction errors, they are not able to reverse the order of a train in place once it is on the railway. This means that currently trains enter stations either forwards or backwards at random. This frequently leads to chaos, e.g. when people with seat reservations try to find their seat. But isn't there a better solution?

Did you know that . . . ?

. . . to the rest of the world, Germans have a reputation for being punctual? Yet when it comes to the country's train system, passengers experience shocking delays.

Assume a train is currently at a certain train station with the coaches in some order. What is the shortest way to route this train through the railway system in such a manner that it arrives back at that same station but with the reverse order of coaches? Each station has two opposite sides at which tracks lead to other stations. Trains can drive in either direction. When a train leaves a station at the same side it arrived, the direction of driving is reversed. When it leaves at the opposite side, the direction of driving stays the same.

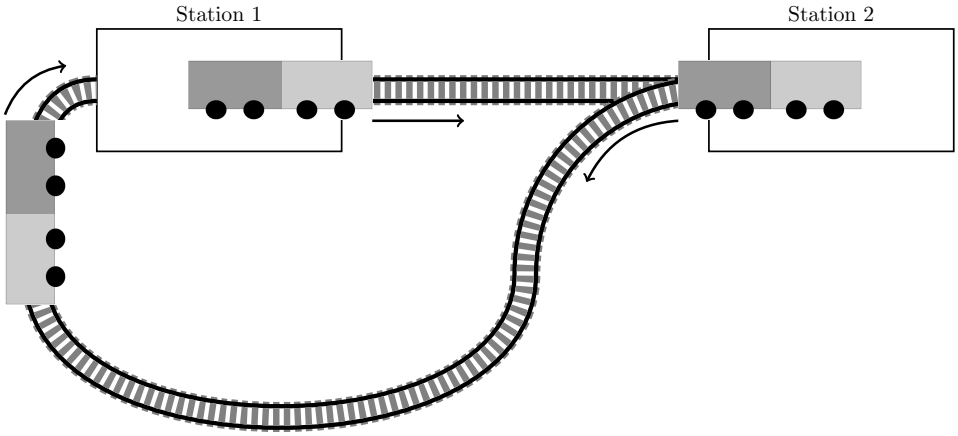


Figure A.1: Illustration of Sample Input 1.

Input

The input consists of:

- One line with two integers n and m ($2 \leq n \leq 2 \cdot 10^5$; $1 \leq m \leq 2 \cdot 10^5$) where n is the number of stations and m the number of tracks connecting the stations. The stations are numbered from 1 to n and each station has two sides 1 and 2.
- m lines, each with five integers a_s, a_t, b_s, b_t and ℓ ($1 \leq a_s, b_s \leq n$; $1 \leq a_t, b_t \leq 2$; $1 \leq \ell \leq 10^9$). This means that stations a_s and b_s are connected by a track which connects to side a_t of station a_s and to side b_t of station b_s. The length of the track is ℓ kilometres.
- One line with an integer t ($1 \leq t \leq n$) giving the station the train is in at the moment.

Railway connections are bidirectional and every existing connection is listed exactly once in the input. No station has a direct connection to itself ($a_s \neq b_s$). The railway network is connected, which means that there is a path between every pair of stations.

Output

In case there is no route which reverses the train, output `impossible`. Otherwise, output the minimum possible length of such a route in kilometres.

Sample Input 1

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

Sample Output 1

```
5
```

Sample Input 2

```
2 1
1 2 2 1 1
1
```

Sample Output 2

```
impossible
```

Sample Input 3

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

Sample Output 3

```
6
```

Problem N: Is this a detour?

Every day you drive to work using the same roads as it is the shortest way. This is efficient, but over time you have grown increasingly bored of seeing the same buildings and junctions every day. So you decide to look for different routes. Of course you do not want to sacrifice time, so the new way should be as short as the old one. Is there another way that differs from the old one in at least one street?

Input

The first line of the input starts with three integers N M and K, where N is the number of junctions and M is the number of streets in your city, and K is the number of junctions you pass every day ($1 \leq K \leq N \leq 10\,000,\ 0 \leq M \leq 1\,000\,000$).
The next line contains K integers, the (1-based) indices of the junctions you pass every day. The first integer in this line will always be 1, the last integer will always be N. There is a shortest path from 1 to N along the K junctions given.
M lines follow. The i-th of those lines contains three integers a_i b_i c_i and describes a street from junction a_i to junction b_i of length c_i ($1 \leq a_i, b_i \leq N,\ 1 \leq c_i \leq 10\,000$). Streets are always undirected.
Note that there may be multiple streets connecting the same pair of junctions. The shortest path given uses for every pair of successive junctions a and b a street of minimal length between a and b.

Output

Print one line of output containing "`yes`" if there is another way you can take without losing time, "`no`" otherwise.

Sample Input 1

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

Sample Output 1

```
yes
```

Sample Input 2

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

Sample Output 2

```
no
```

Problem 0: Professor Toving Liles

The computer science Professor Toving Liles loves the floor tiles in his office so much that he wants to protect them from damage by careless students. Therefore, he would like to buy cheap small rectangular carpets from the supermarket and cover the floor such that:

1. The entire floor is covered.
2. The carpets do not overlap.
3. The carpets are rotated arbitrarily.
4. No carpet is cut into pieces.

But when checking the supermarket's stock he begins to wonder whether he can accomplish his plan at all. Can you help him?

Input

The first line contains two integers W and H describing the size of his room ($1 \leq W, H \leq 100$).
The second line contains an integer c, denoting the number of different carpet colors the supermarket has in stock ($1 \leq c \leq 7$).
Each of the following c lines consists of three integers a_i, w_i, and h_i, which means: the supermarket's stock contains a_i carpets of size w_i, h_i and color i ($1 \leq a_i \leq 7$; $1 \leq w_i \leq 100$; $1 \leq h_i \leq 100$).
The supermarket has at most 7 carpets, i.e. $\sum_i a_i \leq 7$.

Output

For the given room dimensions and the supermarket's stock of carpets, print "`yes`" if it is possible to cover the room with carpets as specified above and "`no`" otherwise.

Sample Input 1

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

Sample Output 1

```
yes
```

Sample Input 2

```
100 100
3
4 42 42
1 100 16
1 32 42
```

Sample Output 2

```
no
```

Problem P: Assistant to the Regional Manager

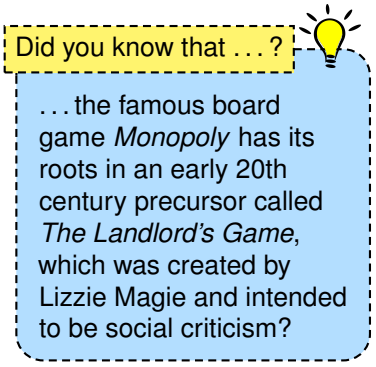
In One Road City the grocery store chain *Great Buy* has a monopoly. As the city's name implies, it conveniently consists of only Smith Street, a single street forming a straight line. There are k houses along the street with given coordinates. There are also n grocery stores belonging to chain *Great Buy*.

The grocery chain *Bestworld* wants to break the monopoly and you, as assistant to the regional manager, are sanctioned to open up to m stores in the city.

Studies show that shoppers are lazy and will always go to the store closest to their home. If the distance is equal, they will go to the store they are more familiar with (in our case *Great Buy*).

With an optimal placement of stores, how many households can you win as new customers?

You may place stores anywhere, including already occupied or non-integer coordinates.



Input

The input consists of:

- One line with three integers k, n and m ($1 \leq k, n, m \leq 10^6$ and $k+n+m \leq 10^6$), where k is the number of houses, n is the number of stores belonging to *Great Buy*, and m is the number of stores chain *Bestworld* can open.
- One line with k integers $h_1, \ldots, h_k$ ($0 \leq h_i \leq 10^6$ for each i), the positions of the houses.
- One line with n integers $b_1, \ldots, b_n$ ($0 \leq b_i \leq 10^6$ for each i), the positions of *Great Buy* stores.

No two entities in the input (stores or houses) are at the same position.

Output

Output the number of houses that are closer to a *Bestworld* store than to a *Great Buy* store if you place the stores optimally.

Sample Input 1

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

Sample Output 1

```
7
```

Sample Input 2

```
10 2 2
1 2 3 4 6 7 8 9 21 22
5 10
```

Sample Output 2

```
7
```

Sample Input 3

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

Sample Output 3

```
2
```

Problem Q: A Stingy Park Visit

It is another wonderful sunny day in July – and you decided to spend your day together with your little daughter Joy. Since she really likes the fairy-park in the next town, you decided to go there for the day. Your wife (unfortunately she has to work) agreed to drive you to the park and pick you up again. Alas, she is very picky about being on time, so she told you exactly when she will be at the park's front entrance to pick you up and you have to be there at exactly that time. You clearly also don't want to wait outside, since this would make your little daughter sad – she could have spent more time in the park!

Now you have to plan your stay at the park. You know when you will arrive and when you will have to depart. The park consists of several rides, interconnected by small pavements. The entry into the park is free, but you have to pay for every use of every ride in the park. Since it is Joy's favorite park, you already know how long using each ride takes and how much each ride costs. When walking through the park, you obviously must not skip a ride when walking along it (even if Joy has already used it), or else Joy would be very sad. Since Joy likes the park very much, she will gladly use rides more than once. Walking between two rides takes a given amount of time.

Since you are a provident parent you want to spend as little as possible when being at the park. Can you compute how much is absolutely necessary?

Input

The input consists of:

- one line with an integer x ($1 \leq x \leq 1\,000$) denoting the time between your arrival and the time you will be picked up (in minutes);
- one line with with three integers n, m, and t, where
  - n ($1 \leq n \leq 1\,000$) is the number of rides in the park;
  - m ($1 \leq m \leq 1\,000$) is the number of pavements;
  - t ($1 \leq t \leq 1\,000$) is the number of minutes needed to pass over a pavement from one ride to another.
- m lines each containing two integers a and b ($1 \leq a, b \leq n$) stating that there is a pavement between the rides a and b.
- n lines each containing two integers t and p ($1 \leq t, p \leq 10^6$) stating that the corresponding ride takes t minutes and costs p Euro.

You always start at ride 1 and have to return to ride 1 at the end of your stay, since the entry is located there. This means that you have to use the ride 1 at least twice (once upon entry and once upon exit). You can take a ride more than once, if you have arrived at it.

Output

Output one line containing either a single integer, the minimum amount necessary to stay x minutes in the park, or `It is a trap.` (including the period) if it is not possible to stay exactly x minutes.

Sample Input 1

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

Sample Output 1

```
8
```

Sample Input 2

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

Sample Output 2

```
5
```