Problem A. Dwarves

Source file name:	dwarves.c, dwarves.cpp, dwarves.java, dwarves.py
Input:	`Standard`
Output:	`Standard`

Once upon a time, there arose a huge discussion among the dwarves in Dwarfland. The government wanted to introduce an identity card for all inhabitants.

Most dwarves accept to be small, but they do not like to be measured. Therefore, the government allowed them to substitute the field "height" in their personal identity card with a field "relative dwarf size". For producing the ID cards, the dwarves were being interviewed about their relative sizes. For some reason, the government suspects that at least one of the interviewed dwarves must have lied.

Can you help find out if the provided information proves the existence of at least one lying dwarf?

Input

The input consists of:

- one line with an integer n ($1 \leq n \leq 10^5$), where n is the number of statements;
- n lines describing the relations between the dwarves. Each relation is described by:
  - one line with "$s_1 < s_2$" or "$s_1 > s_2$", telling whether dwarf s_1 is smaller or taller than dwarf s_2. s_1 and s_2 are two different dwarf names.

A dwarf name consists of at most 20 letters from "A" to "Z" and "a" to "z". A dwarf name does not contain spaces. The number of dwarves does not exceed 10^4.

Output

Output "`impossible`" if the statements are not consistent, otherwise output "`possible`".

Example

Input	Output
3 Dori > Balin Balin > Kili Dori < Kili	impossible
3 Dori > Balin Balin > Kili Dori > Kili	possible

Problem B. Correcting Cheeseburgers

Source file name:	correcting.c, correcting.cpp, correcting.java, correcting.py
Input:	`Standard`
Output:	`Standard`

Cheeseburgers are serious business. They are the most delicious food on earth, but there is a lot of room for error when making a cheeseburger. Even otherwise capable cooks often mess up the order of the assembled ingredients.

The only correct order of ingredients between the buns is, *of course*, as following from top to bottom:

1. Ketchup & Mustard
2. Beef Tomato
3. Pickles
4. Red Onions
5. Cheddar Cheese
6. Garlic
7. Salt & Pepper
8. Beef Patty, medium grilled
9. Corn Salad
10. Mayonnaise

Any deviation from this order is completely unacceptable. Therefore it is sometimes necessary to reassemble a cheeseburger.

Space on an average plate and social norms are rather restrictive when it comes to operating on a cheeseburger. The only feasible operation is the bit-shuffle (burger-ineptly-transformed). The bit-shuffle separates the entire burger into four parts of contiguous ingredients a, b, c and d and arranges them in the new order $c\ a\ d\ b$. The size of each of the four parts is selectable and may be zero.

Since the burger cools rapidly we are interested in the minimum required bit-shuffles to arrive at an acceptable burger.

Each given cheeseburger consists of n unique ingredients labeled from 1 to n. The correct order is always the natural order $1, 2, \ldots, n$.

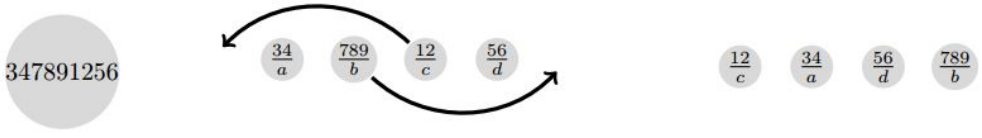

Figure B.1: Illustration of the first sample input.

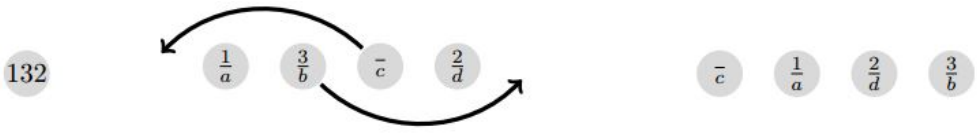

Figure B.2: Illustration of the second sample input.

Input

The input consists of:

- one line with an integer n ($1 \leq n \leq 10$), where n is the number of ingredients used;
- one line with n integers describing the order of the ingredients of the given cheeseburger. The ingredients are numbered from 1 to n.

Output

Output the minimum number of bit-shuffles to correct the given cheeseburger.

Example

Input	Output
9 3 4 7 8 9 1 2 5 6	1
3 1 3 2	1

Problem C. Knapsack in a Globalized World

Source file name:	knapsack.c, knapsack.cpp, knapsack.java, knapsack.py
Input:	`Standard`
Output:	`Standard`

Globalization stops at nothing, not even at the good old honest profession of a burglar. Nowadays it is not enough to break in somewhere, take everything you can carry and dart off. No! You have to be competitive, optimize your profit and utilize synergies.

So, the new game rules are:

- break only into huge stores, so there is practically endless supply of any kind of items;
- your knapsack should be huge;
- your knapsack should be full (there should be no empty space left).

Damn you, globalization, these rules are not easy to follow! Luckily, you can write a program, which will help you decide whether you should loot a store or not.

Input

The input consists of:

- one line with two integers n ($1 \leq n \leq 20$) and k ($1 \leq k \leq 10^{18}$), where n is the number of different item types and k is the size of your knapsack;
- one line with n integers $g_1, \ldots, g_n$ ($1 \leq g_i \leq 10^3$ for all $1 \leq i \leq n$), where $g_1, \ldots, g_n$ are the sizes of the n item types.

Output

Output "`possible`" if it is possible to fill your knapsack with items from the store (you may assume that there are enough items of any type), otherwise output "`impossible`".

Example

Input	Output
2 10000000000 3 6	impossible
2 10000000000 4 6	possible

Problem D. Matrix Cypher

Source file name:	matrixcypher.c, matrixcypher.cpp, matrixcypher.java, matrixcypher.py
Input:	`Standard`
Output:	`Standard`

Alice and Bob communicate via a matrix channel. Alice wants to send a message to Bob. She has a bitstring representing her message and performs a bitwise encoding algorithm: She starts with the identity matrix

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

and then reads the bitstring starting from the left-most bit. For each 0-bit she multiplies the matrix A from the right with

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad i.e. \quad A \leftarrow A \cdot \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

For each 1-bit she multiplies the matrix A from the right with

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad i.e. \quad A \leftarrow A \cdot \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

Then the result is transmitted.

Now Bob accidentally deleted the software to decrypt a message from Alice. Can you help him to rewrite it?

Input

The input consists of:

- two lines, the i-th of them with two integers a_{i1} and a_{i2} ($0 \leq a_{i1},\ a_{i2} \leq 2^{128} - 1$ for all $1 \leq i \leq 2$), where

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

  is the matrix containing the encoded message.

The bitstring representing the message consists of at most 120 characters.

Output

Output the decoded bitstring.

Example

Input	Output
2 1 3 2	010
18 29 13 21	10010101

Problem E. Model Railroad

Source file name:	railroad.c, railroad.cpp, railroad.java, railroad.py
Input:	`Standard`
Output:	`Standard`

Since childhood you have been fascinated by model railroads. Designing your own tracks, complicated intersections, train stations with miniatures of travellers, train operators, luggage is so much fun! However, it also needs a lot of space. Since your house is more than full by now, you decide to move to the garden.

You have already moved all your completed tracks outside when you notice an important flaw: Since different tracks were in different rooms before, there are stations which cannot be reached from each other. That has to change!

Since you have already fixed the exact positions of the stations, you know the lengths for all possible connections you can build and also which stations are connected already. All connections can be used in both directions. You can decide to remove some existing connections and instead build new ones of at most the same total length. Is it possible to rearrange the railroads so that every station is reachable from all other stations?

Input

The input consists of:

- one line with three integers n ($2 \leq n \leq 5 \cdot 10^4$), m ($0 \leq m \leq 2.5 \cdot 10^5$) and l ($0 \leq l \leq m$), where n is the number of stations, m is the number of possible connections and l is the number of connections already built;
- m lines describing the connections. Each connection is described by:
  - one line with three integers a, b ($1 \leq a,\ b \leq n$), and c ($0 \leq c \leq 5 \cdot 10^3$) describing that there is a connection from station a to station b of length c.

The first l of those connections already exist.

Output

Output "`possible`" if it is possible to construct a connected network as described above, otherwise output "`impossible`".

Example

Input	Output
4 6 3 1 2 1 2 1 2 3 4 2 1 3 2 1 4 3 2 4 2	possible
3 3 2 1 2 1 1 2 2 3 2 3	impossible

Explanation

Figure E.1 depicts the first sample case. It is possible to connect all stations by removing the connections between stations 1 and 2 of length 2 and instead building the connection between stations 2 and 4. The curvature of the rails does not matter because you have a hammer.

In the second case, depicted in Figure E.2, it is not possible to connect all three stations.

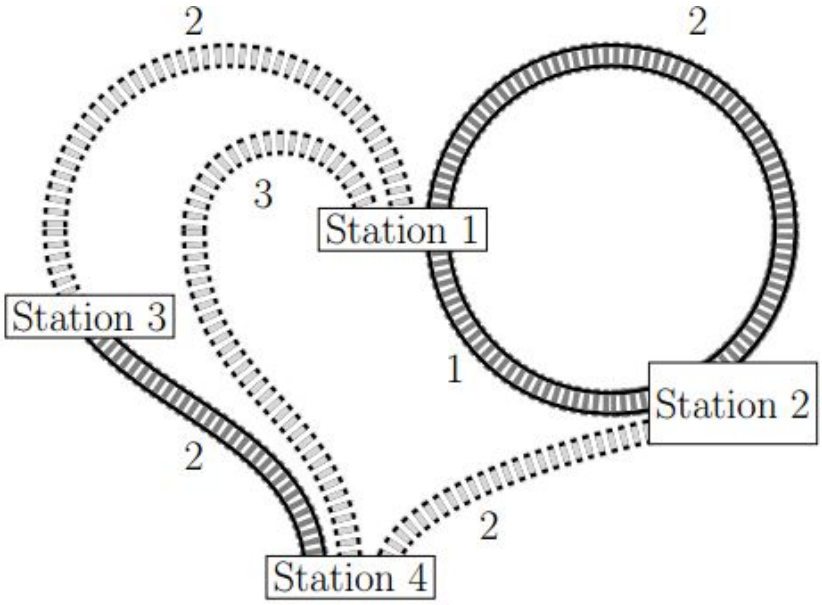

Figure E.1: Illustration of the first sample input.

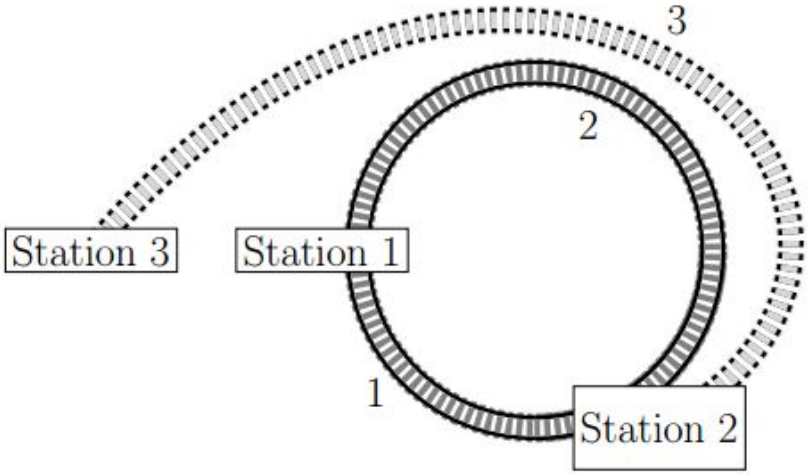

Figure E.2: Illustration of the second sample input.

Problem F. One-Way Roads

Source file name:	onewayroads.c, onewayroads.cpp, onewayroads.java, onewayroads.py
Input:	`Standard`
Output:	`Standard`

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.

Example

Input	Output
2 1 1 2	1
4 5 1 2 1 3 2 3 2 4 3 4	2

Problem G. Formula

Source file name:	formula.c, formula.cpp, formula.java, formula.py
Input:	`Standard`
Output:	`Standard`

Tim is quite a bookworm. Each Saturday he goes to the local library and spends the whole day reading old books. He is mostly interested in ancient history, but from time to time he also reads scientific books. Last weekend he found a series of books called "The Elements" written by some old Greek called Euclid. Tim had never heard of him. All 12 books were filled with definitions, propositions and proofs concerning elementary geometry.

Tim read all books very carefully, but one handwritten note just stuck in his mind. Someone wrote just after Book 4, Proposition 4:

> Using Heron's formula, one can easily derive that the following formula holds for a
>
> triangle with incircle radius r, area A, and lengths a, b, c of the three sides.

$$A = r \cdot \frac{a+b+c}{2}$$

Since Tim is not a very trusting person, he doesn't believe that this formula is correct. However, Tim remembers from school that Heron's formula states a relationship between the area of a triangle and the lengths of its three sides. And of course he trusts this formula. After all, we can trust teachers, right?

$$A = \frac{1}{4} \cdot \sqrt{4a^2b^2 - (a^2+b^2-c^2)^2}$$

To check the formula he found in the book, he has constructed a lot of triangles on paper, inscribed their incircles and measured their radius. Now he wants to check the formula against his measurements, but he is already quite tired from constructing all those triangles on paper. He asks you to write a program that, given the coordinates of the triangle's vertices, computes the incircle radius according to the formula he found in the book and outputs the difference to the value he measured in percent.

Input

The input consists of:

- three lines, the i-th of them with two integers x_i and y_i ($-10^3 \leq x_i,\ y_i \leq 10^3$ for all $1 \leq i \leq 3$), where $(x_i,\ y_i)$ are the coordinates of one of the triangle's vertices;
- one line with one real number r ($0.1 \leq r \leq 10^6$), where r is the incircle radius that Tim measured when he constructed the triangle on paper.

The three coordinates will never be collinear. The triangle's area will always be greater or equal to 0.1.

Output

Output the difference between the measured incircle radius and the radius computed according to the formula as a percentage value, where the measured radius corresponds to 100%. The output is positive if the computed radius is larger than the measured one, 0 if they are equal, or negative otherwise. The output must be accurate to an absolute or relative error of at most 10^{-3}.

Example

Input	Output
0 0 10 0 0 10 5.0	-41.421

Problem H. Celestial Map

Source file name:	celestial.c, celestial.cpp, celestial.java, celestial.py
Input:	`Standard`
Output:	`Standard`

In the distant future humanity has developed into a highly evolved race with vast knowledge of the universe. In particular, they now have an exhaustive map of all stars in the universe and know their trajectory as well as their speed.

Bob works at the center for interplanetary communication. This is usually the most boring job on the planet, since you sit around all day waiting for other yet unknown races to send messages. Today however, as Bob comes back from his daily ping-pong session, he sees a red light blinking away in his control station. As the light is very annoying and keeps him from napping, he decides to look up its meaning in his manual. It says:

Congratulations! You have just made contact with another race. Please submit the coordinates of the newly discovered race to the administration office for proper filing. They can be found on panel 42.

"Neat!" Bob thinks and takes a look at panel 42. Unfortunately, something seems to be broken since instead of showing the coordinates it shows two vectors, a number and "Warning 54816". Another look at the manual reveals:

Warning 54816: Something went wrong while calculating the exact coordinates of the message's origin. Instead a plane from which the message must have come from as well as the origin's distance were computed.

After a quick check, Bob finds out that there are quite a few stars whose trajectories intersect with the plane at the given distance, but none that intersect at a slightly different distance (up to 0.1 lightyears). Also, stars which have not been in the plane at the time of the signal transmission were at a distance of at least 0.1 lightyears to the plane. Furthermore, the trajectories of all stars seem to intersect the given plane at an angle between 10 and 90 degrees.

Since Bob wants to plan his afternoon, he first wants to know how many stars he needs to check individually to find out from which one the message originated.

Can you help Bob figure out how many stars fit the description?

Input

The input consists of:

- one line with an integer n ($1 \leq n \leq 10^4$) and a real number d ($1.0 \leq d \leq 10^5$), where n is the number of stars in the universe and d is the distance from Bob to the message's origin;
- two lines each with three real numbers p_x, p_y, and p_z ($0.0 \leq p_x,\ p_y,\ p_z \leq 10.0$, $|(p_x,\ p_y,\ p_z)| \geq 0.1$) describing a vector (p_x, p_y, p_z), where the two vectors span a plane from Bob's location in which the message's origin must have been in when the message was sent;
- $2n$ lines describing the stars. Each star is described by:
  - one line with three real numbers $s_x,\ s_y$, and s_z ($-1 \cdot 10^6 \leq s_x,\ s_y,\ s_z \leq 10^6$) where (s_x, s_y, s_z) describes the current location of the star;

– one line with three real numbers t_x, t_y, and t_z ($0.1 \leq |(t_x, t_y, t_z)| \leq 0.95$) where ($t_x$, t_y, t_z) describes the trajectory of the star. $|(t_x, t_y, t_z)|$ gives the speed at which the star is traveling.

You can assume the following:

- The other civilization's message travels at the speed of light.
- All speeds are given in lightyears per year, and distances are in lightyears.
- Bob's location is fixed at $(0, 0, 0)$.
- Stars never collide.
- There are no intersection points of the given plane and any trajectory at distance $d \pm 0.1$ lightyears except for intersection points at distance exactly d lightyears.
- Stars which have not been in the plane at the time of the signal transmission were at a distance of at least 0.1 lightyears to the plane.
- All trajectories intersect the given plane at an angle between 10 and 90 degrees.

Output

Output the number of stars that fit the description.

Example

Input	Output
2 2.0 1.0 0.0 0.0 0.0 1.0 0.0 2.0 0.0 1.0 0.0 0.0 0.5 0.0 2.0 1.0 0.0 0.0 0.7	1
1 2.0 1.0 0.0 0.0 0.0 1.0 0.0 2.0 0.0 2.0 0.0 0.0 0.5	0

Explanation

In the first sample we need to check 2 stars. The star positions in the present as well as the given plane and the trajectories of the stars are shown in figure 1. The plane from which the message originated is given by the vectors (1.0, 0.0, 0.0) and (0.0, 1.0, 0.0), the distance is 2.0. The first star is currently at position (2.0, 0.0, 1.0) and moves in direction (0.0, 0.0, 1.0) with speed 0.5. The second star is currently at position (0.0, 2.0, 1.0) and moves in direction (0.0, 0.0, 1.0) with speed 0.7. After moving the stars back by 2 years the stars' positions are as shown in Figure H.2. As we can see, star 1 is in the plane and has the correct distance to Bob, therefore it fits the description. Star 2 is not in the plane and therefore does not fit the description. Hence the correct answer for the first sample is 1.

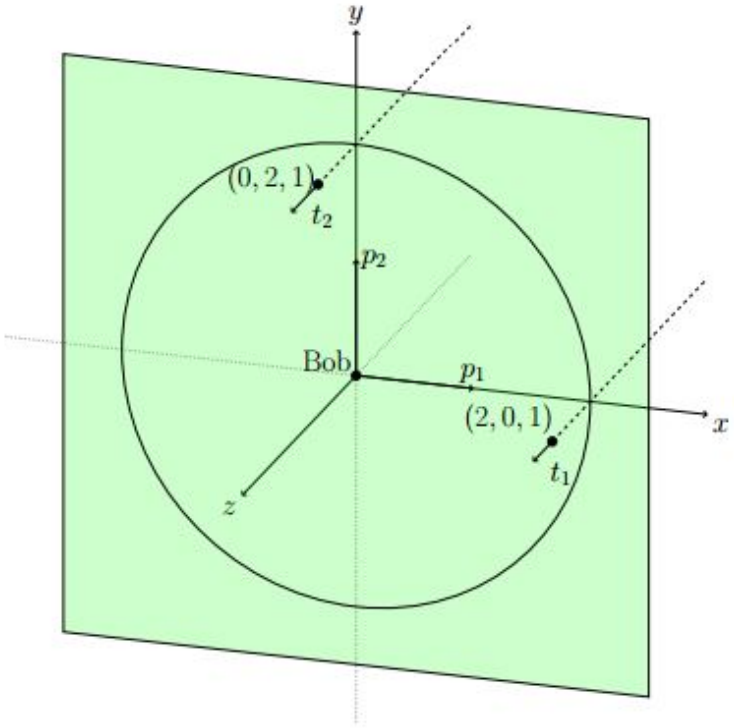

Figure H.1: Star positions in the present for the first sample input.

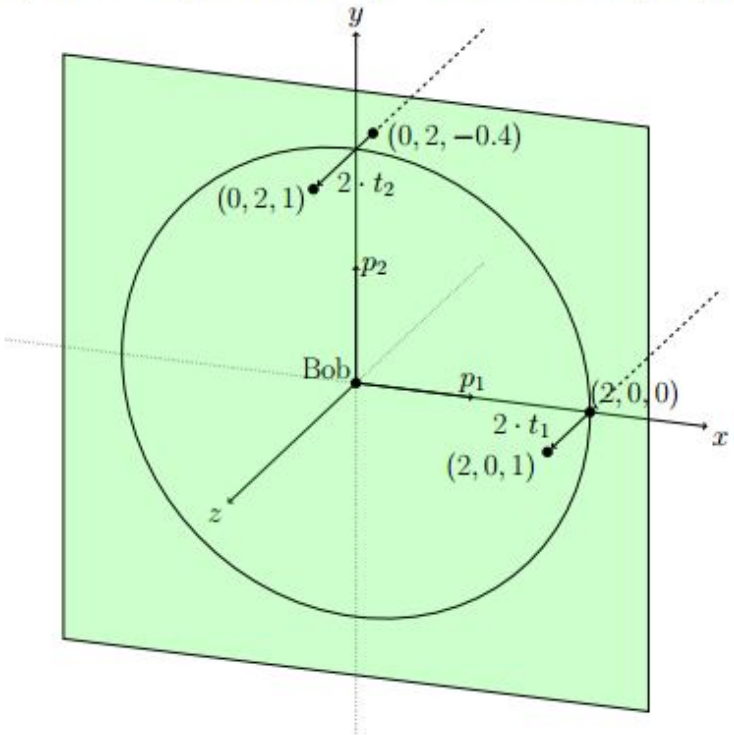

Figure H.2: Star positions two years ago for the first sample input.

Problem I. Common Knowledge

Source file name:	commonkn.c, commonkn.cpp, commonkn.java, commonkn.py
Input:	`Standard`
Output:	`Standard`

Alice and Bob play some game in which they score points. Each of the two has an n-digit scoreboard which depicts numbers in base 10 (with leading zeroes). The digits 0 to 9 are displayed on a seven-segment display in the following fashion:

For some odd reason, the two players cannot see the scoreboards entirely. Alice can only see the lower half of her own scoreboard and the upper half of Bob's scoreboard. Bob can only see the upper half of his scoreboard and the upper half of Alice's scoreboard. Here, 'half' is meant to include the horizontal segments in the digits' centers: they can be seen by both players at all times. For example, if one sees the upper half of an eight, one can conclude that the digit is not a zero.

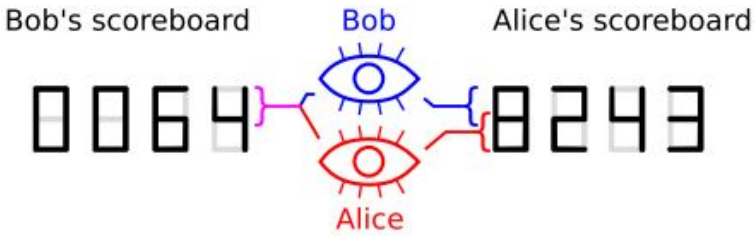

Figure I.1: An example situation for $n = 4$.

A pair of n-digit scores is called *fully known* if both players know both scores (i.e. all $2n$ digits) by looking at the displays with their restricted vision. The players cannot communicate.

Input

The input consists of:

- one line with an integer n ($1 \leq n \leq 20$), where n is the number of digits.

Output

Output the number of score pairs that can be displayed on two n-digit scoreboards and are fully known by both players.

Example

Input	Output
10	1073741824
13	549755813888

Problem J. Selling CPUs

Source file name:	sellingcpus.c, sellingcpus.cpp, sellingcpus.java, sellingcpus.py
Input:	`Standard`
Output:	`Standard`

You are very happy, that you got a job at ACME Corporation's CPU factory. After a hard month of labor, your boss has given you c identical CPUs as payment. Apparently, ACME Corporation is low on cash at the moment.

Since you can't live on CPUs alone, you want to sell them on the market and buy living essentials from the money you make. Unfortunately, the market is very restrictive in the way you are allowed to conduct business. You are only allowed to enter the market once, you can only trade with each merchant once, and you have to visit the traders in a specific order. The market organizers have marked each merchant with a number from 1 to m (the number of merchants) and you must visit them in this order. Each trader has his own price for every amount of CPUs to buy.

Input

The input consists of:

- one line with two integers c and m ($1 \leq c,\ m \leq 100$), where c is the number of CPUs and m is the number of merchants;
- m lines describing the merchants. Each merchant is described by:
  - one line with c integers $p_1, \ldots, p_c$ ($1 \leq p_i \leq 10^5$ for all $1 \leq i \leq c$), where p_i is the amount of money the merchant will give you for i CPUs.

Output

Output the maximum amount of money you can make by selling the CPUs you have.

Note that you don't have to sell all CPUs.

Example

Input	Output
5 3 1 4 10 1 1 1 1 8 1 1 1 1 9 1 1	13

Problem K. Routing

Source file name:	routing.c, routing.cpp, routing.java, routing.py
Input:	`Standard`
Output:	`Standard`

As part of his new job at the IT security department of his company, Bob got the task to track how fast messages between the companies' different offices are transmitted through the internet. Since some offices are currently rebuilt and not finished yet he is not able to simply measure the transmission speed but needs to compute it somehow. Therefore, Bob created a map of all servers that may be involved in routing the packages through the internet. He also gathered the times each server needs to process a message. The total processing time of a message is the sum of the processing times of the sender, all servers along the path, and the receiver. Furthermore, Bob read that messages are sent through the network of servers along a path such that the total processing time of all servers on the path is minimal.

Bob thought that it might be an easy problem to compute the total transmission time between two offices, but he forgot the intelligence agencies! Each server on the internet is controlled by some agency that can decide which packages are routed and which of them are not. All servers are configured in a way that they read all incoming data, since gathering all kind of information is exactly what the intelligence agencies want to do, but not all data is forwarded to other servers.

Each server has a list of pairs of other servers such that messages from the first of them are not sent to the second one. Can you still help Bob to compute how fast his messages will be transmitted?

Input

The input consists of:

- one line with an integer n ($2 \leq n \leq 100$), where n is the number of servers labeled from 1 to n;
- n blocks describing the servers. Each server is described by:
  - one line with two integers m ($0 \leq m \leq n-1$) and t ($0 \leq t \leq 1000$), where m is the number of outgoing connections from this server and t is the processing time of this server;
  - m lines with two integers s ($0 \leq s \leq n-1$) and x ($1 \leq x \leq n$) and s more distinct integers $a_1, \ldots, a_s$ ($1 \leq a_j \leq n, a_j \neq i$ for all $1 \leq j \leq s$) indicating that server i sends messages to server x, but only if it was not directly transmitted from one of the servers $a_1, \ldots, a_s$ to server i.

Bob's messages start at server 1 and should go to server n.

Output

Output the sum of the processing times of all servers on the shortest path for Bob's message including the first and last one, or "`impossible`" if there is no such path.

Example

Input	Output
4 2 10 0 2 0 3 1 1 1 4 1 2 10 1 2 1 0 4 0 10	30
3 1 10 0 2 1 10 1 3 1 0 10	impossible

Explanation

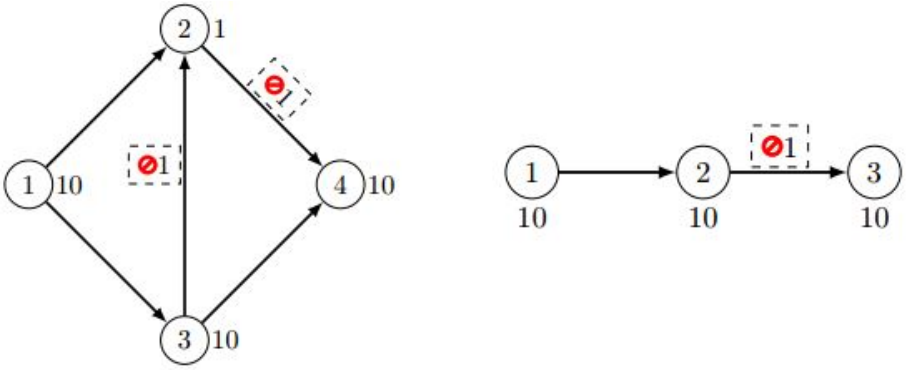

Figure K.1: Illustration of the sample inputs.

Problem L. Maze

Source file name:	maze.c, maze.cpp, maze.java, maze.py
Input:	`Standard`
Output:	`Standard`

Bob is the Captain of the USS Spacey McSpaceface, the pride of the space fleet of the Human Empire. Unfortunately, Bob is also the last one alive on his ship.

Because of an unrecognized virus, the ship's AI "Alpha 5" went rampant and killed everyone by funelling neurotoxin into the crew's quarters. Bob was the only one on the bridge at that time, where he could grab a handy detoxication kit and a gas mask. But alas, Alpha 5 also blocked the ships' controls, so Bob was now drifting in space, alone, caged with a rampant AI that just initiated the self-destruction mechanism.

Bob remembered that some engineer once told him there was an override switch somewhere near the fusion reactor that powered the ship, so he started to make his way down into the engine quarters. Soon he noticed that the virus had also taken over the automatic door system — all doors seemed to open and close seemingly at random.

At least, Bob does know three things:

- every door is labeled with a single big white letter;
- there is a huge display in every room which explains which doors will open and which will close (for example, if the displays show a big "A", all doors labeled with "A" will open next, and all other doors will slam shut);
- every door will stay open for about half a second, just enough for Bob to dash into the next room.

Bob wants to stay on the move (neurotoxin and all), so he will use a door whenever possible. However, Bob does not know the exact way to the core, so if there are multiple possible doors he could take, he just chooses one at random. Given the sequence of letters until the explosion, how high is the probability that Bob can reach the reactor core and shut it all down?

Input

The input consists of:

- one line with two integers n ($1 \leq n \leq 10^3$) and m ($1 \leq m \leq 5 \cdot 10^3$), where n is the amount of rooms in the engine quarters and m is the amount of doors between them;
- m lines describing the doors. Each door is described by:
  - one line with two integers a and b ($1 \leq a,\ b \leq n,\ a \neq b$) and a letter l between "A" and "Z" indicating a door from room a to room b labeled with l (on both sides);

  There may be several doors between the same pair of rooms, which may or may not have the same letter written on them;
- one line with the sequence the doors will open in. The sequence consists of at most 200 letters from "A" to "Z".

After the last letter, all doors slam shut and the ship will explode if Bob has not reached the switch yet. Bob starts on the bridge, room 1, and the override switch is in room n.

Output

Output the percentage that Bob will reach the override switch in time.

The answer should be correct up to an absolute or relative error of at most 10^{-3}.

Example

Input	Output
5 4 1 2 A 2 3 B 1 4 A 4 5 B AB	50.0
3 2 1 2 A 2 3 B AB	100.0
3 2 1 2 A 2 3 B ACDEFGHIJKL	0.0
3 3 1 2 A 2 3 B 1 3 C ACB	100.0