Problem A
Baggage
Time Limit: 1 second

An airline has two flights leaving at about the same time from ICPCity, one to city B and one to city A. The airline also has n counters where passengers check their baggage. At each counter there is a pair of identical baggage bins, one for city B and one for city A.

Just before the flights depart, each pair of baggage bins is moved by a motorized cart to a sorting area. The cart always moves two bins at a time, one for city B and one for city A. After all the bins have been moved, they line up in the sorting area like this:

B A B A B A ... B A

That is, there are $2n$ baggage bins in a row, starting with a bin for city B, then one for city A, and so forth. The task now is to reorder them so all the baggage bins for city A precede the baggage bins for city B. Then the bins can be loaded on the appropriate aircraft.

The reordering is done by moving pairs of adjacent baggage bins (not necessarily B then A), again via the motorized cart. For proper balance, the cart must always carry two bins, never just one. A pair of bins must always be moved to an empty space that is at least two bins wide. On the left of the first bin are some empty spaces that can be used as needed during the reordering.

When the reordering process begins, the bin locations are numbered from 1 (initially containing the leftmost B baggage bin) to $2n$ (initially containing the rightmost A baggage bin). There are $2n$ initially empty spaces to the left of the bins, numbered from 0 to $-2n+1$, as shown in Figure A.1 for the case $n = 4$.

								B	A	B	A	B	A	B	A
-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8

Figure A.1: Initial configuration of bins and empty spaces for $n = 4$

Given n, find a shortest sequence of moves that will reorder the bins so that all the A bins are to the left of all the B bins. At the end of the process, it is possible that the leftmost A bin is at some location other than 1, but the bins must be adjacent in a sequence of $2n$ locations.

Input

The input consists of a single test case, which consists of the integer n ($3 \leq n \leq 100$).

Output

Display a shortest sequence of moves that will correctly reorder the bins. Each move is of the form "f `to` t", where f and t are integers representing the movement of the bins in locations f and $f+1$ to locations t and $t+1$. If multiple solutions are possible, display any one of them.

Sample Input 1

```
5
```

Sample Output 1

```
8 to -1
3 to 8
6 to 3
0 to 6
9 to 0
```

Sample Input 2

```
8
```

Sample Output 2

```
10 to -1
3 to 10
14 to 3
7 to 14
0 to 7
11 to 0
4 to 11
15 to 4
```

Problem B
Buffed Buffet
Time Limit: 4 seconds

You are buying lunch at a buffet. A number of different dishes are available, and you can mix and match them to your heart's desire. Some of the dishes, such as dumplings and roasted potatoes, consist of pieces of roughly equal size, and you can pick an integral number of such pieces (no splitting is allowed). Refer to these as "discrete dishes." Other dishes, such as tzatziki or mashed potatoes, are fluid and you can pick an arbitrary real-valued amount of them. Refer to this second type as "continuous dishes."

Of course, you like some of the dishes more than others, but how much you like a dish also depends on how much of it you have already eaten. For instance, even if you generally prefer dumplings to potatoes, you might prefer a potato over a dumpling if you have already eaten ten dumplings. To model this, each dish i has an initial tastiness t_i, and a rate of decay of the tastiness Δt_i. For discrete dishes, the tastiness you experience when eating the n^{th} item of the dish is $t_i - (n-1)\Delta t_i$. For continuous dishes, the tastiness you experience when eating an infinitesimal amount dx grams of the dish after already having eaten x grams is $(t_i - x\Delta t_i)dx$. In other words, the respective total amounts of tastiness you experience when eating N items of a discrete dish or X grams of a continuous dish are as follows:

$$\sum_{n=1}^{N}(t_i - (n-1)\Delta t_i) \qquad \text{and} \qquad \int_0^X (t_i - x\Delta t_i)dx$$

For simplicity, do not take into account that different dishes may or may not go well together, so define the total tastiness that you experience from a meal as the sum of the total tastinesses of the individual dishes in the meal (and the same goes for the weight of a meal – there are no food antiparticles in the buffet!).

You have spent days of painstaking research determining the numbers t_i and Δt_i for each of the dishes in the buffet. All that remains is to compute the maximum possible total tastiness that can be achieved in a meal of weight w. Better hurry up, lunch is going to be served soon!

Input

The input consists of a single test case. The first line of input consists of two integers d and w ($1 \le d \le 250$ and $1 \le w \le 10\,000$), where d is the number of different dishes at the buffet and w is the desired total weight of your meal in grams.

Then follow d lines, the i^{th} of which describes the i^{th} dish. Each dish description is in one of the following two forms:

- A description of the form "`D` w_i t_i Δt_i" indicates that this is a discrete dish where each item weighs w_i grams, with initial tastiness t_i and decay of tastiness Δt_i.
- A description of the form "`C` t_i Δt_i" indicates that this is a continuous dish with initial tastiness t_i and decay of tastiness Δt_i.

The numbers w_i, t_i, and Δt_i are integers satisfying $1 \le w_i \le 10\,000$ and $0 \le t_i, \Delta t_i \le 10\,000$.

Output

Display the maximum possible total tastiness of a meal of weight w based on the available dishes. Give the answer with a relative or absolute error of at most 10^{-6}. If it is impossible to make a meal of total weight exactly w based on the available dishes, display `impossible`.

Sample Input 1

```
2 15
D 4 10 1
C 6 1
```

Sample Output 1

```
40.500000000
```

Sample Input 2

```
3 15
D 4 10 1
C 6 1
C 9 3
```

Sample Output 2

```
49.000000000
```

Sample Input 3

```
2 19
D 4 5 1
D 6 3 2
```

Sample Output 3

```
impossible
```

Problem C
Crane Balancing
Time Limit: 1 second

Wherever there is large-scale construction, you will find cranes that do the lifting. One hardly ever thinks about what marvelous examples of engineering cranes are: a structure of (relatively) little weight that can lift much heavier loads. But even the best-built cranes may have a limit on how much weight they can lift.

The Association of Crane Manufacturers (ACM) needs a program to compute the range of weights that a crane can lift. Since cranes are symmetric, ACM engineers have decided to consider only a cross section of each crane, which can be viewed as a polygon resting on the x-axis.

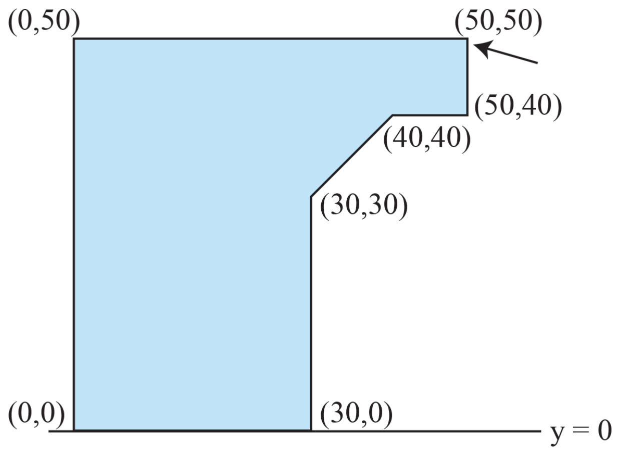


Figure C.1: Crane cross section

Figure C.1 shows a cross section of the crane in the first sample input. Assume that every 1×1 unit of crane cross section weighs 1 kilogram and that the weight to be lifted will be attached at one of the polygon vertices (indicated by the arrow in Figure C.1). Write a program that determines the weight range for which the crane will not topple to the left or to the right.

Input

The input consists of a single test case. The test case starts with a single integer n ($3 \le n \le 100$), the number of points of the polygon used to describe the crane's shape. The following n pairs of integers x_i, y_i ($-2\,000 \le x_i \le 2\,000, 0 \le y_i \le 2\,000$) are the coordinates of the polygon points in order. The weight is attached at the first polygon point and at least two polygon points are lying on the x-axis.

Output

Display the weight range (in kilograms) that can be attached to the crane without the crane toppling over. If the range is $[a, b]$, display $\lfloor a \rfloor$ `..` $\lceil b \rceil$. For example, if the range is $[1.5, 13.3]$, display `1 .. 14`. If the range is $[a, \infty)$, display $\lfloor a \rfloor$ `.. inf`. If the crane cannot carry any weight, display `unstable` instead.

Sample Input 1

```
7
50 50
0 50
0 0
30 0
30 30
40 40
50 40
```

Sample Output 1

```
0 .. 1017
```

Sample Input 2

```
7
50 50
0 50
0 0
10 0
10 30
20 40
50 40
```

Sample Output 2

```
unstable
```

Problem D
Game Strategy
Time Limit: 8 seconds

Alice and Bob are playing a board game. The board is divided into positions labeled $a, b, c, d, \ldots$ and the players use a gamepiece to mark the current position. Each round of the game consists of two steps:

1. Alice makes a choice. Depending on the current position, she has different options, where each option is a set of positions. Alice chooses one set S among the available sets of positions.
2. Bob makes a choice. His choice is one position p from the set S that Alice chose in step 1. Bob moves the gamepiece to position p, which is the position for the start of the next round.

Prior to the first round, each player independently selects one of the positions and reveals it at the start of the game. Bob's position is where the game starts. Alice wins the game if she can force Bob to move the gamepiece to the position she has chosen. To make things interesting, they have decided that Bob will pay Alice a certain amount if he loses, but Alice must pay Bob a certain amount after every round. The game now ends if Alice's position is reached or when Alice runs out of cash.

Both Alice and Bob play optimally: Alice will always choose an option that will lead to her winning the game, if this is possible, and Bob will always try to prevent Alice from winning.

For all possible start and end positions, Alice would like you to determine whether she can win the game and if so, how many rounds it will take.

Input

The input consists of a single test case. The first line contains the number of positions n ($1 \le n \le 25$). The n positions are labeled using the first n letters of the English alphabet in lowercase. The rest of the test case consists of n lines, one for each position p, in alphabetical order. The line for position p contains the options available to Alice in position p. It starts with the number of options m ($1 \le m < 2^n$), which is followed by m distinct strings, one for each option. Each string contains the positions available to Bob if Alice chooses that option. The string has at least 1 character, the characters (which correspond to valid board positions) are in alphabetical order, and no characters are duplicated. The total number of options for the test case is at most 10^6.

Output

For each position p in alphabetical order, display one line. In that line, for each position q in alphabetical order display the minimal number of rounds in which Alice can be guaranteed to arrive at position q when starting the game in position p, or -1 if Alice cannot be guaranteed to reach q from p.

Sample Input 1

```
2
2 ab b
1 b
```

Sample Output 1

```
0 1
-1 0
```

Sample Input 2

```
3
1 b
2 b a
2 ab ac
```

Sample Output 2

```
0 1 -1
1 0 -1
2 2 0
```

Problem E
Maze Reduction
Time Limit: 2 seconds

Jay runs a small carnival that has various rides and attractions. Unfortunately, times are tough. A recent roller coaster accident, flooding in the restrooms, and an unfortunate clown incident have given Jay's carnival a bad reputation with the public. With fewer paying customers and reduced revenue, he will need to cut some costs to stay in business.

One of the biggest carnival attractions is a large, confusing maze. It consists of a variety of circular rooms connected by narrow, twisting corridors. Visitors love getting lost in it and trying to map it out. It has come to Jay's attention that some of the rooms might be effectively identical to each other. If that's the case, he will be able to reduce its size without anyone noticing.

Two rooms A and B are *effectively identical* if, when you are dropped into either room A or B (and you know the map of the maze), you cannot tell whether you began in A or B just by exploring the maze. The corridor exits are evenly spaced around each room, and you cannot mark or leave anything in a room (in particular, you cannot tell whether you have previously visited it). The only identifying feature that rooms have is their number of exits. Corridors are also twisty enough to be indistinguishable from each other, but when you enter a room you know which corridor you came from, so you can navigate a little by using the order they appear around the room.

Jay has appealed to the Association for Carnival Mazery for help. That's you! Write a program to determine all the sets of effectively identical rooms in the maze.

Input

The input consists of a single test case. The first line contains an integer n, the number of rooms in the maze ($1 \le n \le 100$). Rooms are numbered from 1 to n. Following this are n lines, describing each room in order. Each line consists of an integer k, indicating that this room has k corridors ($0 \le k < 100$), and then k distinct integers listing the rooms each corridor connects to (in clockwise order, from an arbitrary starting point). Rooms do not connect to themselves.

Output

Display one line for each maximal set of effectively identical rooms (ignoring sets of size 1) containing the room numbers in the set in increasing order. Order the sets by their smallest room numbers. If there are no such sets, display `none` instead.

Sample Input 1

```
13
2 2 4
3 1 3 5
2 2 4
3 1 3 6
2 2 6
2 4 5
2 8 9
2 7 9
2 7 8
2 11 13
2 10 12
2 11 13
2 10 12
```

Sample Output 1

```
2 4
5 6
7 8 9 10 11 12 13
```

Sample Input 2

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

Sample Output 2

```
none
```

Problem F
Messenger
Time Limit: 4 seconds

Misha needs to send packages to his friend Nadia. Both of them often travel across Russia, which is very large. So they decide to hire a messenger. Since the cost of the messenger service depends on the time it takes to deliver the package, they need your help to optimize a little bit.

Assume Misha and Nadia move on a two-dimensional plane, each visiting a sequence of places and moving along straight line segments from place to place. Your task is to find the shortest possible delivery time given their two paths.

Misha hands the package to the messenger at some point along his path. The messenger moves without delay along a straight line from the pick-up to intercept Nadia, who is traveling along her path. Misha, Nadia and the messenger move with a constant speed of 1 distance unit per time unit. The delivery time is the time between Misha handing over the package and Nadia receiving it.

Input

The input consists of a single test case. The test case contains two path descriptions, the first for Misha and the second for Nadia. Each path description starts with a line containing an integer n, the number of places visited ($2 \leq n \leq 50\,000$). This is followed by n lines, each with two integers x_i and y_i specifying the coordinates of a place ($0 \leq x_i, y_i \leq 30\,000$). Coordinates of the places are listed in the order in which they are to be visited, and successive places do not have the same coordinates.

Misha and Nadia start their journeys at the same time, visiting the places along their paths without stopping. The length of each path is at most 10^6. The package must be picked up at the latest when Misha reaches his final place and it must be delivered at the latest when Nadia reaches her final place.

Output

Display the minimal time needed for delivery. Give the answer with an absolute error of at most 10^{-3} or a relative error of at most 10^{-5}. If the package cannot be delivered, display `impossible` instead.

Sample Input 1

```
2
0 0
0 10
2
4 10
4 0
```

Sample Output 1

```
4.00000
```

Sample Input 2

```
2
0 0
1 0
3
2 0
3 0
3 10
```

Sample Output 2

```
5.00000
```

Problem G
Metal Processing Plant
Time Limit: 4 seconds

Yulia works for a metal processing plant in Ekaterinburg. This plant processes ores mined in the Ural mountains, extracting precious metals such as chalcopyrite, platinum and gold from the ores. Every month the plant receives n shipments of unprocessed ore. Yulia needs to partition these shipments into two groups based on their similarity. Then, each group is sent to one of two ore processing buildings of the plant.

Picture from Wikimedia Commons

To perform this partitioning, Yulia first calculates a numeric distance $d(i, j)$ for each pair of shipments $1 \le i \le n$ and $1 \le j \le n$, where the smaller the distance, the more similar the shipments i and j are. For a subset $S \subseteq \{1, \ldots, n\}$ of shipments, she then defines the *disparity* D of S as the maximum distance between a pair of shipments in the subset, that is,

$$D(S) = \max_{i,j \in S} d(i, j).$$

Yulia then partitions the shipments into two subsets A and B in such a way that the sum of their disparities $D(A) + D(B)$ is minimized. Your task is to help her find this partitioning.

Input

The input consists of a single test case. The first line contains an integer n ($1 \le n \le 200$) indicating the number of shipments. The following $n - 1$ lines contain the distances $d(i, j)$. The i^{th} of these lines contains $n - i$ integers and the j^{th} integer of that line gives the value of $d(i, i + j)$. The distances are symmetric, so $d(j, i) = d(i, j)$, and the distance of a shipment to itself is 0. All distances are integers between 0 and 10^9 (inclusive).

Output

Display the minimum possible sum of disparities for partitioning the shipments into two groups.

Sample Input 1	Sample Output 1

5
4 5 0 2
1 3 7
2 0
4
	4

Sample Input 2	Sample Output 2

7
1 10 5 5 5 5
5 10 5 5 5
100 100 5 5
10 5 5
98 99
3
	15

Problem H
Pachinko
Time Limit: 6 seconds

You have been hired by Addictive Coin Machines to help design the next hit in their line of eye-catching, coin-guzzling, just-one-more-try Pachinko machines for casinos around the world.

Playing a Pachinko machine involves launching balls into a rectangular grid filled with pegs, obstacles, and targets. The ball bounces around the grid until it eventually hits one of the targets. The player earns a certain number of points depending on which target is hit.

The grid pattern for the next Pachinko machine has already been designed, but point values for the targets have not been assigned. These must be set so that like all casino machines, the machine is profitable but not *too* profitable. Thus it is important to figure out the probability of a ball hitting any particular target. That's your job!

For simplicity, the grid is modeled as a tall rectangle filled with mostly-open spaces (each represented by '`.`'), impassable obstacles (each represented by '`X`'), and targets (each represented by '`T`').

A ball is launched randomly with uniform probability into one of the mostly-open spaces on the top row of the grid. From that point on, collisions with pegs cause the ball to randomly bounce up, down, left, or right, with various given probabilities. For simplicity, assume these probabilities are the same for every space in the grid. If the ball bounces into an obstacle or attempts to move off the grid, it won't actually move from its current space. When the ball moves into a target it is removed from play.

You can safely assume that the average number of spaces visited by a ball before hitting a target will not exceed 10^9. It would not make for a very enjoyable game if the ball just bounces forever!

For each target, calculate the probability that it is the one hit by a launched ball.

Input

The input consists of a single test case. The first line contains integers w and h, which are the width and height of the Pachinko grid ($1 \leq w \leq 20$ and $2 \leq h \leq 10\,000$). The next line contains four non-negative integers u, d, l, and r, which sum to 100 and are the percentage probabilities of the ball bouncing up, down, left, or right from any open space.

Each of the next h lines contains w characters, each of which is '`.`', '`X`', or '`T`'. These lines describe the Pachinko grid. The first line, which describes the top row of the grid, contains at least one '`.`' and no '`T`'s.

Output

Display one line for each '`T`' in the grid, in order from top to bottom, breaking ties left to right. For each target, display the probability that a launched ball will hit it. Give the answer with an absolute error of at most 10^{-6}.

Sample Input 1

```
3 2
20 20 20 40
X.X
T.T
```

Sample Output 1

```
0.333333333
0.666666667
```

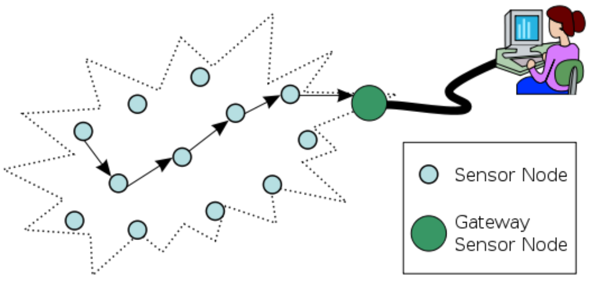
Sample Input 2

```
4 5
12 33 28 27
....
.XX.
....
T..T
XTTX
```

Sample Output 2

```
0.435853889
0.403753221
0.081202502
0.079190387
```

Problem I
Sensor Network
Time Limit: 2 seconds



Picture from Wikimedia Commons

A wireless sensor network consists of autonomous sensors scattered in an environment where they monitor conditions such as temperature, sound, and pressure.

Samantha is a researcher working on the Amazon Carbon-dioxide Measurement (ACM) project. In this project, a wireless sensor network in the Amazon rainforest gathers environmental information. The Amazon rainforest stores an amount of carbon equivalent to a decade of global fossil fuel emissions, and it plays a crucial role in the world's oxygen-transfer processes. Because of the huge size of this forest, changes in the forest affect not only the local environment but also global climate by altering wind and ocean current patterns. The goal of the ACM project is to help scientists better understand earth's complex ecosystems and the impact of human activities.

Samantha has an important hypothesis and to test her hypothesis, she needs to find a subset of sensors in which each pair of sensors can communicate directly with each other. A sensor can communicate directly with any other sensor having distance at most d from it. In order for her experiments to be as accurate as possible, Samantha wants to choose as many sensors as possible.

As one does not simply walk into the Amazon, Samantha cannot add new sensors or move those that are currently in place. So given the current locations of the sensors, she needs your help to find the largest subset satisfying her criteria. For simplicity, represent the location of each sensor as a point in a two-dimensional plane with the distance between two points being the usual Euclidean distance.

Input

The input consists of a single test case. The first line contains two integers n and d ($1 \leq n \leq 100$ and $1 \leq d \leq 10\,000$), where n is the number of sensors available and d is the maximum distance between sensors that can communicate directly. Sensors are numbered 1 to n. Each of the next n lines contains two integers x and y ($-10\,000 \leq x, y \leq 10\,000$) indicating the sensor coordinates, starting with the first sensor.

Output

Display a maximum subset of sensors in which each pair of sensors can communicate directly. The first line of output should be the size of the subset. The second line of output should be the (one-based) indices of the sensors in the subset. If there are multiple such subsets, any one of them will be accepted.

Sample Input 1

```
4 1
0 0
0 1
1 0
1 1
```

Sample Output 1

```
2
1 2
```

Sample Input 2

```
5 20
0 0
0 2
100 100
100 110
100 120
```

Sample Output 2

```
3
4 3 5
```

Problem J
Skiing
Time Limit: 2 seconds

As you know, the ACM ICPC is not the only major sporting event taking place in Russia this year. Several months ago, the 2014 Winter Olympics were held in Sochi, which is about 3 000 km from Ekaterinburg.

In an increasing number of sports, it is not only the ability of the athletes that determines who wins a competition but also their equipment. For example in downhill skiing, having the latest ski technology enables athletes to increase their speeds and improve their turning ability.

You have been hired to determine the effect of the latest ski technology on the ability of skiers to navigate a downhill course. The course contains several target locations, and the skier wants to pass over as many of them as possible. Naturally, the better the ski technology, the easier it will be to do this.

For simplicity, use a two-dimensional coordinate system where the skier starts at position (0,0) and where "downhill" corresponds to the direction of the positive y-axis.

Assume the y-component of the athlete's velocity is a constant v_y. The athlete can change speed laterally (in the x-direction), but the skiing equipment limits this to a maximal lateral acceleration a_{max}. The skier starts with a lateral velocity of 0.

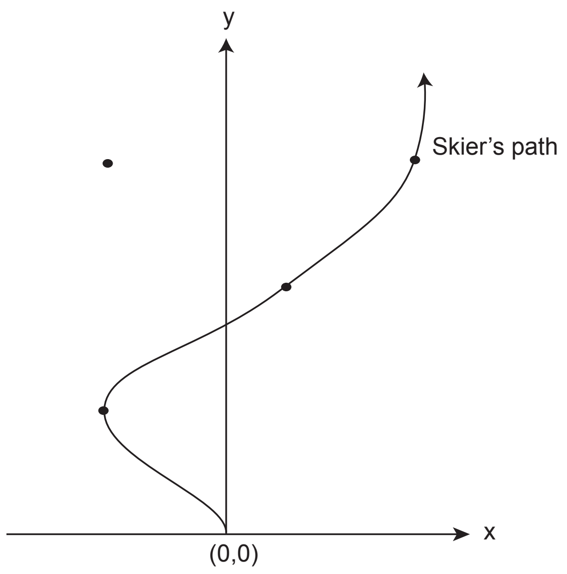


Figure J.1: Downhill ski path passing over three targets

In Figure J.1 (which corresponds to the first sample input), the optimal path passes over three out of four possible targets. If a_{max} were smaller, then the skier might be able to pass over only two or fewer of the targets.

Input

The input contains a single test case. The first line contains three integers n, v_y, and a_{max} ($0 \leq n \leq 250$, $0 \leq v_y \leq 10^5$ and $0 \leq a_{max} \leq 10^7$), where n is the number of targets, v_y is the y-component of the skier's velocity, and a_{max} is the maximum lateral acceleration. Here v_y is given in meters per hour and a_{max} in meters per hour squared.

Following this are n lines, each containing two integers x_i and y_i ($-10^5 \leq x_i, y_i \leq 10^5$). These give the coordinates of each target to be visited on the course. All coordinates are given in meters. Targets are numbered 1, 2, ..., n in the order they are given.

Output

Display the maximal-length sequence of targets that the athlete could pass over on the course in a single run. Display the targets in the order they are visited. If there are multiple maximal-length sequences, display only the lexicographically first one. (So the sequence `2 15` would come before the sequence `10 15`.) If the athlete cannot pass over any targets, print `Cannot visit any targets` instead.

To ensure floating-point stability, you may assume the answer will not change if a_{max} is perturbed by up to 0.1.

Sample Input 1	Sample Output 1

`4 100 400`
`-100 100`
`50 200`
`-100 300`
`150 300`
	`1 2 4`

Sample Input 2	Sample Output 2

`1 100 100`
`1000 10`
	`Cannot visit any targets`

Problem K
Surveillance
Time Limit: 4 seconds

The International Corporation for Protection and Control (ICPC) develops efficient technology for, well, protection and control. Naturally, they are keen to have their own headquarters protected and controlled. Viewed from above, the headquarters building has the shape of a convex polygon. There are several suitable places around it where cameras can be installed to monitor the building. Each camera covers a certain range of the polygon sides (building walls), depending on its position. ICPC wants to minimize the number of cameras needed to cover the whole building.

Input

The input consists of a single test case. Its first line contains two integers n and k ($3 \leq n \leq 10^6$ and $1 \leq k \leq 10^6$), where n is the number of walls and k is the number of possible places for installing cameras. Each of the remaining k lines contains two integers a_i and b_i ($1 \leq a_i, b_i \leq n$). These integers specify which walls a camera at the i^{th} place would cover. If $a_i \leq b_i$ then the camera covers each wall j such that $a_i \leq j \leq b_i$. If $a_i > b_i$ then the camera covers each wall j such that $a_i \leq j \leq n$ or $1 \leq j \leq b_i$.

Output

Display the minimal number of cameras that suffice to cover each wall of the building. The ranges covered by two cameras may overlap. If the building cannot be covered, display `impossible` instead.

Sample Input 1	Sample Output 1

100 7
1 50
50 70
70 90
90 40
20 60
60 80
80 20
	3

Sample Input 2	Sample Output 2

8 2
8 3
5 7
	impossible

Sample Input 3	Sample Output 3

8 2
8 4
5 7
	2

This page is intentionally left blank.

Problem L
Wire Crossing
Time Limit: 2 seconds

Moore's Law states that the number of transistors on a chip will double every two years. Amazingly, this law has held true for over half a century. Whenever current technology no longer allowed more growth, researchers have come up with new manufacturing technologies to pack circuits even denser. In the near future, this might mean that chips are constructed in three dimensions instead two. But for this problem, two dimensions will be enough.

A problem common to all two-dimensional hardware design (for example chips, graphics cards, motherboards, and so on) is wire placement. Whenever wires are routed on the hardware, it is problematic if they have to cross each other. When a crossing occurs special gadgets have to be used to allow two electrical wires to pass over each other, and this makes manufacturing more expensive.

Our problem is the following: you are given a hardware design with several wires already in place (all of them straight line segments). You are also given the start and end points for a new wire connection to be added. You will have to determine the minimum number of existing wires that have to be crossed in order to connect the start and end points. This connection need not be a straight line. The only requirement is that it cannot cross at a point where two or more wires already meet or intersect.

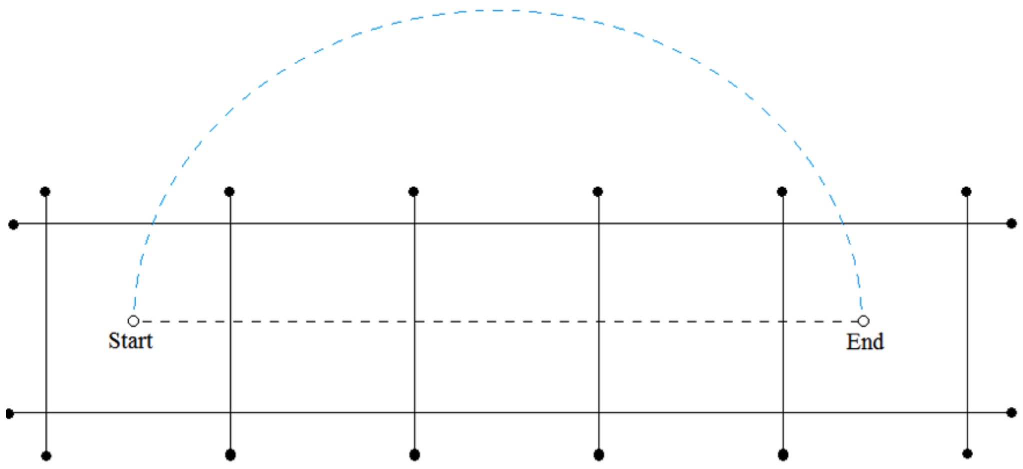


Figure L.1: First sample input

Figure L.1 shows the first sample input. Eight existing wires form five squares. The start and end points of the new connection are in the leftmost and rightmost squares, respectively. The black dashed line shows that a direct connection would cross four wires, whereas the optimal solution crosses only two wires (the curved blue line).

Input

The input consists of a single test case. The first line contains five integers m, x_0, y_0, x_1, y_1, which are the number of pre-existing wires ($m \leq 100$) and the start and end points that need to be connected. This is followed by m lines, each containing four integers x_a, y_a, x_b, y_b describing an existing wire of non-zero length from (x_a, y_a) to (x_b, y_b). The absolute value of each input coordinate is less than 10^5. Each pair of wires has at most one point in common, that is, wires do not overlap. The start and end points for the new wire do not lie on a pre-existing wire.

Output

Display the minimum number of wires that have to be crossed to connect the start and end points.

Sample Input 1

```
8 3 3 19 3
0 1 22 1
0 5 22 5
1 0 1 6
5 0 5 6
9 0 9 6
13 0 13 6
17 0 17 6
21 0 21 6
```

Sample Output 1

```
2
```

Sample Input 2

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

Sample Output 2

```
0
```