Problem A. Kidnapping

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	256 megabytes

Berland's Police has a serious problem. A foreign ambassador arrived to Berland with an important mission, and his daughter was kidnapped just from the Royal Palace! Inspired by adventures of Erast Fandorin, the Police Chief developed the following ingenious plan.

The ambassador agrees to pay ransom, but only if the kidnappers allow his servant to visit the girl and ensure that she is alive. The kidnappers take the blindfolded servant into a coach and transport him to the secret place, where they keep the ambassador's daughter. Certainly, the role of the servant is certainly played by a secret agent of the Police. The Police Chief knows that when the coach is moving, the wheels are creaking once on each full rotation. So, by counting the number of creaks and multiplying it by the length of the rim, one can easily calculate the distance covered by the coach.

In spite of this brilliant idea, the affair turned to be much more difficult than it could be in a detective story. There are n intersections in the city numbered from 1 to n, some pairs of intersections are connected by bidirectional roads. The kidnappers agreed to take the "servant" to the secret place, and the servant is quite sure that this place is located at one of the intersections. Also the agent has calculated the lengths of roads between each pair of consecutive intersections on the route passed by the coach. But during the trip the agent was concentrated on counting creaks, so he could not remember in which directions the coach turned at the intersections.

Now the route probably couldn't be restored uniquely! Moreover, the agent has a suspicion that the kidnappers could intentionally pass the same intersection or even the same road more than once to confuse the Police.

Your task is to determine all possible locations of the secret place, given that the trip starts at the intersection number 1.

Input

The first line of the input contains a single integer n ($2 \leq n \leq 200$). Each of the next n lines contains n integers each. The i-th number in the j-th line l_{ij} is the length of the road between the i-th and the j-th intersections. If $l_{ij} = 0$ then the road doesn't exist.

It is guaranteed that $0 \leq l_{ij} \leq 200$, $l_{ii} = 0$ and $l_{ij} = l_{ji}$. The next line contains one integer k ($1 \leq k \leq 200$) — the number of roads passed by the couch. The following line contains k integers r_1, r_2, …, r_k ($1 \leq r_i \leq 200$) — the lengths of roads between each pair of consecutive intersections on the route passed by the coach from the starting point to the secret place.

Output

To the first line of the output write m — the number of all possible locations of the secret place. The second line should contain the numbers of intersections in increasing order separated by spaces.

If there are no possible locations of the secret place, the output must contain the only integer `0`.

Sample input and output

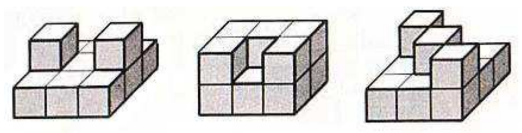
stdin	stdout

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

Problem B. 3D City Model

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	256 megabytes

A city is built on the top of a rectangular $n \times m$ grid where all the grid cells are equal squares. Each of the $n \cdot m$ grid cells can serve as a foundation of a single building in the city. A building is represented as a number of $1 \times 1 \times 1$ cubes stacked on the top of each other. The cube that lays in the foundation of a building entirely occupies a single cell on the grid. It is clear that adjacent buildings can share a wall or a part of it. Typical cities can be seen on the image below.



The King of Berland has a 3D model of the capital city in his office. This model was made on a special 3D-printer out of plastic. It represents a layout of the capital city, but the scale is smaller, so it's very convenient for the King to examine the model without having to visit the city itself. The King is bored though because the model is colorless, so he wants to paint the model. To calculate the exact amount of required paint he should know the total area of the model's surface.

You have to help the King and write a program that will calculate the required surface area of the given model. While calculating the surface area you should count not only the side surfaces, but also the areas of the top and bottom facets.

The model is given to you as $n \times m$ matrix of digits. A digit in the j-th position of the i-th row stands for the height of the building with its foundation in cell (i, j) of the model. If the corresponding digit is equal to "0", it means there is no building built on the top of this cell.

Input

The first line of input contains a pair of integers n, m ($1 \le n, m \le 100$), where n — amount of rows in the given grid, m — amount of columns. The following n lines contain the description of the model. These n lines contain m digits each representing heights of the buildings. It's guaranteed that the given matrix contains at least one non-zero digit.

Output

Output the only positive integer — surface area of the model.

Sample input and output

stdin	stdout

`3 3`
`111`
`212`
`111`
	`38`

`3 4`
`1000`
`0010`
`0000`
	`12`

Note

The first sample test corresponds to the leftmost picture from the problem statement.

Problem C. Fire in the Country

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	256 megabytes

This summer's heat wave and drought unleashed devastating wildfires all across the Earth. Of course, a tiny country on the island "Yars and Eva" is also affected by this ecological disaster. Thanks to the well-organized actions of rescuers, all the citizens were evacuated to the nearby planets on a spaceship.

To save the country, a small fire robot was left on its territory. He managed to extinguish fire in all cities except the capital before running out of liquid. The robot can't extinguish fire anymore, so the country is still in danger at the moment.

There are n cities in the country connected by m two-way roads. Each road connects a pair of cities. There is at most one road between any pair of cities. The cities are numbered from 1 to n, with capital having the number 1.

The fire spreads very quickly. On the very first day only the capital is on fire. But with every subsequent day, the fire devours all the cities connected by a road with the cities that are already on fire. Once the fire gets to a certain city, this city will continue to stay on fire till the very end.

The robot can't extinguish the fire anymore and there are no other means of firefighting left in the country, so obviously the country is going to be burned down to the ground. And you don't have to be a hero and save it. The key thing is that the robot is going to be destroyed by fire as well, and you need to figure out who will actually pay for the loss of government property.

Two pilots, Nikolay and Vladimir, are on Earth's natural satellite. They alternately take turns controlling the robot. The pilots alternate each day. Robot's speed is equal to the speed of fire, so the robot can get to the neighboring city in a day. Each pilot does not want the robot to be destroyed on his turn. For such a valuable loss they will have to pay a huge fee to the government.

On the first day the robot is located in the capital. Nikolay controls the robot on the first day. Thus, Nikolay controls the robot on the days with odd numbers, and Vladimir controls it on the days with even numbers. Taking turn, a pilot has to move the robot from the current city to any city connected by a road with the current one. If a pilot moves the robot to a city which is on fire, the robot is destroyed.

You task is to figure out who will pay the fine for the destroyed robot, assuming both pilots act optimally.

Input

The first line of input contains the amount of cities n and the amount of roads m in the country ($2 \le n \le 1000, n - 1 \le m \le 1000$). The following m lines contain description of the roads: a, b — indices of the cities connected by roads ($1 \le a \le n$, $1 \le b \le n$, $a \ne b$). The roads are bidirectional. No pair of cities will be connected by more than one road. There will be a path between any two cities.

Output

Output the name of the pilot who will pay the fine, assuming both pilots act optimally ("`Nikolay`" — if it is Nikolay, "`Vladimir`" — if it is Vladimir).

Sample input and output

stdin	stdout
`4 3`	`Vladimir`
`1 2`	
`1 3`	
`2 4`	
`4 4`	`Nikolay`
`1 2`	
`1 3`	
`2 4`	
`3 4`	
`4 5`	`Nikolay`
`1 2`	
`1 3`	
`2 4`	
`3 4`	
`2 3`	

Note

In the first sample test, an optimal strategy for Nicolay is to send the robot to the city 3 on the first day. Vladimir then will be forced to send the robot back to the capital, so the robot will be destroyed and Vladimir will have to pay.

Problem D. “North-East”

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	256 megabytes

The popular music band of international fame “North-East” is coming to Berland! This news has spread all over the country, so numerous fans are now ready to rush and buy all the tickets!

At present the fans still don’t know in which cities the band plans to give concerts. The only thing is known at the moment is that the band will visit several cities, and as their name says, they will strictly move north and east when going to the next city. In other words when the band moves from city i to city j, city j is always located northward and eastward of the city i.

It’s also known that the tour is planned in such a way that the maximum possible number of cities will be visited. The musicians refuse to reveal other details. As you know, fans always get ready for the arrival of their idols, so they would appreciate any single detail about possible movements of their favorite musicians.

Your task is to help the fans and find two lists of cities — A and B. The first list A should contain the cities, which the band might visit during the tour. The second list B should contain the cities, which the band will have to visit for sure during the tour.

Input

The first line of input contains a single integer n ($1 \le n \le 10^5$) — amount of cities in the country. The following n lines contain coordinates of the cities. Each line contains a pair of integers x_i, y_i ($-10^6 \le x_i, y_i \le 10^6$) — the coordinates of the i-th city. Ox axis is directed west-to-east, and Oy axis — south-to-north. No two given cities will be located at the same point.

Output

Print the required list A to the first line of output and B to the second line. Each list should start with the amount of cities in it, followed by the indices of cities in increasing order. Cities are numbered from 1 to n.

Sample input and output

stdin	stdout

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

5
1 1
10 10
5 6
10 1
6 5
	4 1 2 3 5
	2 1 2

Problem E. Oil Wells

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	256 megabytes

These are tough times for Berland nowadays. State property is being sold piece by piece, and even state oil company "BerOil" hasn't escaped the common lot.

A well-known entrepreneur Tapochkin decided to buy a piece of "BerOil"'s land. Tapochkin is very smart, so he wants to buy a piece of land that will contain all of the company's oil wells within its area or on its border. He has already bought a fence building machine that will move along the border of Tapochkin's new land and build a high fence! There is only one problem — the machine is defective.

Normally, the machine can move in any of the four directions: "north", "east", "south" and "west". Now, when it's broken, the situation is a bit different. Before the engine is turned on, the driver divides these four directions into two groups — two perpendicular directions in each group (e.g. "north" and "east" in one group, "south" and "west" in the other group). Then for some time the machine moves using only directions from the first group, after that — only directions from the second group. It's the driver who determines the moment of switching from one group to the other one.

The company's territory is split into equal squares by a system of roads. Half of the roads go in "north-south" direction, the other half — in "east-west" direction. The company has a vast territory that can be regarded as infinite. The roads form a square grid, each square of the grid has a side length equal to 1 km. Oil wells are located on the crossroads and their size is negligible, so they can be considered points.

Cartesian coordinate system can be applied to the company's territory, so that Ox axis is directed west-to-east, and Oy axis — south-to-north. Unit of length — 1 km. Initial position of the machine and coordinates of the oil wells are known. Find the area of the smallest possible piece of land that the machine can fence around, leaving all the oil wells inside the fenced area or on its border. Also, find the sequence of machine's movements. The border of Tapochkin's piece of land should be a closed non-degenerate polyline that doesn't cross or touch itself.

Input

The first line of input contains three integers n, x_0 and y_0 ($1 \le n \le 400, -400 \le x_0, y_0 \le 400$), where n — amount of oil wells, (x_0, y_0) — initial position of the fence building machine. The following n lines contain coordinates of the wells x_i, y_i ($-400 \le x_i, y_i \le 400$). No two wells have identical coordinates.

Output

Print the only number `-1` to the output, if there is no solution. Otherwise print the area of Tapochkin's land in square kilometers to the first line, and the path of the fence building machine — to the second. The path of the machine is a sequence of characters "`W`", "`N`", "`E`" and "`S`", standing for movements one kilometer "west", "north", "east" or "south" correspondingly. You are allowed to print the path in any direction of driving — clockwise or counterclockwise, as it doesn't really change the closed polyline which the path represents. It there is more than one answer, you may print any of them.

Sample input and output

stdin	stdout
3 4 2	13
5 6	NENNNEEEEESSWWSSWWW
7 2	
9 4	
2 1 -2	5
-1 2	NWNNNWSSSSEE
1 -2	
1 1 2	1
1 2	ESWN

Problem F. Elevator

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	256 megabytes

The Berland State Building is the highest building in the capital of Berland. Curious Polikarp was studying the principle of operation of an elevator in the Berland State Building for a quite a while. Recently he has finally understood the algorithm behind its operation, in case a person enters the elevator on the floor f and presses the floor buttons $e_1, e_2, \ldots, e_n$ one by one. The buttons are pressed sequentially but very quickly while the elevator is still located on the floor f. All the pressed buttons are distinct and differ from the floor f. No other button pressings are considered in this problem.

After the buttons $e_1, e_2, \ldots, e_n$ have been pressed, all of them become highlighted and the elevator starts moving according the following rules:

- The elevator starts moving towards the floor, the button of which is highlighted and pressed first among all highlighted buttons. Say, it's floor/button a.
- If on its way to a the elevator passes the floor b, the button of which is highlighted, it stops there, the light goes out for the button b unhighlighting it, and the floor b is considered visited. Then the elevator continues moving towards the floor a. It is possible that there will be more than one floor such as b on the way to floor a — all these floors will be passed one by one according to the described algorithm.
- Having reached the floor a, the elevator stops there, the light goes out for the button a unhighlighting it, and the floor a is considered visited. Then the elevator starts to move towards the floor, the button of which has been pressed the earliest among the currently highlighted buttons. That floor becomes a new value of a. The elevator continues moving according to the rules described in the previous paragraph. If it's impossible to find a new value for a because there are no highlighted floor buttons, it means that all floors have been visited and the elevator stops.

Now, when the principle of the elevator's operation is clear, Polikarp wants to experiment with the elevator's movements without the elevator itself. He wants to write a program that simulates elevator's operation. Unfortunately, he didn't attend any programming lessons and it's a challenge for him. Can you please help Polikarp and write a program which will simulate movements of the elevator?

Input

The first line of input contains a pair of integers n, f $(1 \le n, f \le 100)$, where n — amount of pressings made, f — index of the current floor where all these pressings were made. The second line contains distinct integers $e_1, e_2, \ldots, e_n$ $(1 \le e_i \le 100, e_i \ne f)$ — buttons indices in the order they were pressed.

Output

Output all the floors where the elevator stops, in a chronological order of the stops.

Sample input and output

stdin	stdout

`4 5`
`10 9 2 1`
	`9 10 2 1`

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

Problem G. Buoys

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	256 megabytes

The swimming area of Berhattan's city beach is marked out with n buoys. The buoys form a straight line. When the buoys were being put into the water, nobody cared to observe the same distance between each pair of adjacent buoys.

Now the beach keeper wants the distance between any two adjacent buoys to be the same. He plans to shift some or all of the buoys without changing their respective order. To facilitate the task, he wants the total length of all shifts to be as small as possible.

Given coordinates of the buoys, you should find the minimum possible length of all shifts, as well as new coordinates of the buoys.

Input

The first line of input contains a single integer n ($2 \le n \le 400$), n — the number of buoys. The second line contains buoys' integer coordinates $x_1, x_2, \ldots, x_n$ ($-10000 \le x_i \le 10000$). No two given buoys will share the same place. The coordinates are given in strictly increasing order.

Output

To the first line print a real number t — the minimum possible total length of required shifts. Output this value with at least 4 digits after the decimal point.

To the second line print n numbers — new coordinates of the buoys. The new coordinates should be printed in strictly increasing order with at least 7 digits after the decimal point. If there are several optimal ways to shift the buoys, you may output any of them.

Sample input and output

stdin

```
4
-2 2 6 9
```

stdout

```
1.0000
-2.0000000000 1.6666666667 5.3333333333 9.0000000000
```

Note

All buoys are located on the Ox axis. You may move buoys only along the Ox axis.

Problem H. Revolutionary Roads

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	256 megabytes

Governments of different countries like to boast about their achievements. For instance, the President of Flatland has announced that his country has the most advanced road system. He said the degree of a country road system development is equal to the amount of cities in the largest *connected* subset of cities. A subset of cities is called *connected* if it is possible to get from any city of the subset to all other cities of the subset.

Not to lag behind the neighbors Berland's President decided to undertake a reform and modernize roads in his country. All the roads in Berland are one-way, each of them connects a pair of cities in one direction. There is at most one road in each direction between any two given cities.

Since there is little money in the budget, President's plans aren't very ambitious. He can turn at most one of all given one-way roads into a two-way road. And he wants to do it in such a way that the resulting road system degree of development in Berland becomes as high as possible. Let's say the maximum degree of development, which can be achieved by this action, is equal to w.

A road is called *revolutionary* if, after it is changed from one-way to two-way, the degree of road system development becomes equal to w. Your task is to find all *revolutionary* roads.

Input

The first line of input contains a pair of numbers n, m ($1 \le n \le 1000, 0 \le m \le 20000$), where n — the number cities, m — the number of roads. The following m lines contain descriptions of the roads. Each line contains a pair of integers a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$), representing a one-way road from city a_i to city b_i. Cities are numbered from 1 to n.

Output

Write w to the first line of output. To the second line write t — number of roads in the required subset. To the third line write indices of the roads in this subset. Roads are numbered from 1 to m according to their order in the input file.

Sample input and output

stdin	stdout

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

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

Problem I. Running Hero

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	256 megabytes

Aladdin is running through a long cave. He is very fast and can move with a speed of at most v meters per second, but falling stones prevent him from running always at maximum speed.

Now imagine a 2-dimensional world with Aladdin being a point moving in the positive or negative direction of the X-axis, and stones, falling down on the Al's head, being segments parallel to the X-axis. Another point in this world is princess Jasmin. She is waiting for Al at the point (G, 0). Aladdin starts his way at the point (0, 0). A stone kills Aladdin if he is located strictly between the endpoints of the stone when the stone reaches the ground. As Jasmin is in a fortified trench, the stones can't hurt her even if they fall down on her head.

You are given Aladdin's maximum speed, Jasmin's position and information about times and places of falling stones. Your task is to find the whether it is possible for Aladdin to to reach Jasmin. If it's possible, you also have to find the sequence of actions which gets Aladdin to Jasmin in the minimum amount of time possible.

Input

The first line of the input contains three integers v, G and n ($1 \le v \le 10^5$, $0 < |G| \le 10^5$, $0 \le n \le 3000$) — Aladdin's maximum speed, Jasmin's x-coordinate and the number of stones. Each of the following n lines contains three integers $x_{1,i}$, $x_{2,i}$ and t_i ($-10^5 \le x_{1,i} \le x_{2,i} \le 10^5$, $1 \le t_i \le 10^5$) — left and right endpoints of the i-th stone, and time when the i-th stone reaches the ground.

Output

If there is no solution, the output should contain a single integer -1.

Otherwise the first line of the output should contain one integer k — the number of instructions for Aladdin. Each of the following k lines should contain a pair of space-separated real numbers — speed w and time t. A pair (w, t) means that Aladdin should run t seconds at speed w. Speed w should be negative, if Aladdin should move in the negative direction of the X-axis and non-negative otherwise. Value t should always be positive. The value $w = 0$ means that Aladdin doesn't move for time t.

If there are many solutions, you may output any of them. All real values should be printed with at least 9 digits after the decimal point. The number of instructions k in your output must not exceed 10000.

Sample input and output

stdin

```
5 35 2
-5 6 1
5 35 9
```

stdout

```
2
-5.000000000000000000 1.000000000000000000
5.000000000000000000 8.000000000000000000
```

Note

Aladdin can change his speed instantly. If some stone falls down on Al's head at the moment of or after his meeting with Jasmin, it can't prevent the two from kissing each other and can't change anything.

Problem J. Explode 'Em All

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	512 megabytes

The prime minister of Berland decided to build a new city in the country. It's hard to describe the excitement of all Berland citizens, but indeed this is great news from the economic, social and cultural standpoints.

The land in Berland is occupied almost entirely and it's very hard to find free space for construction, so it was decided to build the city on a stony terrain. The map of this terrain is represented as an $n \times m$ grid, where each cell of the grid is either an empty space or a rock.

Of course, before construction is started, the given terrain must be completely cleared from rocks. As you may guess, you were hired to complete this mission. Your goal is to destroy all rocks by dropping bombs from a plane. A bomb can be dropped on any cell of the map, and you are free to select where you want to drop each bomb. When a bomb targeted for cell $(i,\ j)$ reaches the ground, it destroys all rocks in row i and also all rocks in column j of the grid. If cell $(i,\ j)$ contains a rock, this rock is also destroyed.

Please help the prime minister of Berland to find the minimum number of bombs required to completely clear the given terrain from rocks.

Input

The first line of input contains two integers n and m $(1 \le n, m \le 25)$ — the number of rows and columns correspondingly. Each of the next n lines contains m characters describing the terrain. An empty space is denoted by "`.`", while a rock is denoted by "`*`".

Output

Write a single integer to the output — the minimum numbers of bombs required for destroying all rocks on the terrain.

Sample input and output

stdin	stdout

`8 10`
`..........`
`..***..*.*`
`.*.......*`
`.*.......*`
`.*.......*`
`.....*****`
`..........`
`.........*`
	`2`

`3 4`
`....`
`....`
`....`
	`0`

Note

In the first sample test it's only required to drop 2 bombs from a plane: one bomb to cell (2,2) and another bomb to cell (6, 10). Row and column indices in this explanation are 1-based.

Problem K. Bencoding

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	256 megabytes

Bencoding is a modern technique which is used for representing data structures as sequences of characters. It it capable of encoding strings, integers, lists and dictionaries as specified below:

- every string s is encoded as “¡k¿:¡s¿”, where “¡s¿” is the string itself and “¡k¿” is its length written with no leading zeros (with the exception of integer zero, which is always represented as “0”). Bencoding is capable of encoding empty strings, so s is allowed to be empty.

  For example, “`4:spam`” represents the string “spam”, “`0:`” represents an empty string.

- every integer n is encoded as “i¡n¿e”, where “¡n¿” is the number itself written with no leading zeros (with the exception of integer zero, which is always represented as “0”). Bencoding is capable of encoding very large integers, so n doesn’t necessarily fit into any of the standard integer data types provided by programming languages. Only non-negative integers can be encoded using bencoding.

  For example, “`i1024e`” represents the number 1024.

- every list *items* containing n elements $item_0, item_1, \ldots, item_{n-1}$ is encoded as “l¡$item_0$¿¡$item_1$¿ $\ldots$ ¡$item_{n-1}$¿e”. Each item of the list $item_i$ is a supported data structure encoded using bencoding technique. Lists are allowed to be empty.

  For example, “`li101el4:spami1024eee`” represents the list “[101, [“spam”, 1024]]”.

- every dictionary *dict* consisting of n keys $key_0, key_1, \ldots, key_{n-1}$ mapped into n values $value_0, value_1, \ldots, value_{n-1}$ correspondingly is encoded as “d¡key_0¿¡$value_0$¿¡key_1¿¡$value_1$¿ $\ldots$ ¡key_{n-1}¿¡$value_{n-1}$¿e”. All keys and values are supported data structures encoded using bencoding technique. A dictionary may have duplicate keys and/or values. Dictionaries are allowed to be empty.

  For example, “`d1:a0:1:pl1:b2:cdee`” represents the dictionary with string key “a” mapped into an empty string “”, and key “p” mapped into the list “[“b”, “cd”]”.

A character sequence c is called a *valid bencoded object* if the following two conditions are met:

- c is a correct bencoded representation of a single string, integer, list or dictionary;
- the number of characters in c doesn’t exceed a given number m.

For example, when $m = 3$, the sequence $c =$“`2:bc`” is not considered a valid bencoded object even though it represents a correctly encoded string “bc”.

Given m and c you have to write a program which should determine whether c is a *valid bencoded object*. If c is not a *valid bencoded object*, it also has to find the longest prefix of c which could be a prefix of some *valid bencoded object*. Formally, you should find a maximal position j within the given character sequence c, such that a prefix of c up to, but not including, character at position j could be a prefix of some *valid bencoded object*. If the given character sequence c is not a *valid bencoded object*, but the entire sequence c is a prefix of some *valid bencoded object*, then j is considered equal to the length of c.

Input

The first line of the input contains one integer m ($2 \le m \le 10^9$) — the maximum possible length of a valid bencoded object. The second line contains a character sequence which you are to process. The sequence will only contain characters with ASCII codes from 33 to 127, inclusive. Its length will be between 1 and 10^6 characters.

Output

Print a single line containing word "`ok`" (without quotes) if the given input character sequence is a valid bencoded object. Otherwise, print message "`Error at position` j`!`". The first character of the input sequence is considered to have position "`0`".

Sample input and output

stdin	stdout

`14`
`li10e11:abcdefghijke`
	`Error at position 6!`

`10`
`i-1e`
	`Error at position 1!`

`3`
`i2`
	`Error at position 2!`

`18`
`dli1eei1ei1eli1eee`
	`ok`

Note

In the first sample test the given sequence is not a valid bencoded object. But its prefix "`li10e1`" can be extended to a valid bencoded object while not exceeding 14 characters in length (for example, "`li10e1:xe`"). It's not the case with longer prefixes of length 7 and more, so $j = 6$ in this case.

Problem L. It's Time to Repair the Roads

Input file:	`stdin`
Output file:	`stdout`
Time limit:	8 seconds
Memory limit:	256 megabytes

Everybody knows about the problems with roads in Berland. The government has been trying to undertake major repairs for many years, but the roads have never been repaired due to the lack of money in the budget.

There are n cities and m roads in Berland. The cities are numbered from 1 to n. The roads are numbered from 1 to m. Each road connects a pair of different cities, all the roads are two-way. There is at most one road between any pair of cities. The cost of repairing is known for each road.

Clearly, repairing all roads in Berland is an unaffordable luxury, so the government decided to repair only such set of the roads, that it's possible to get from any city to any other city by the roads from this repaired set, and the total cost of these road works is minimal.

In the circumstances of the global economic crisis and global warming, road repair costs change every day. Berland's scientists managed to predict these changes, concluding that the cost of road works will change for only one road each day. They created a full list of expected changes for the coming t days — for each day they came up a road and its new repair cost.

The government of Berland would like to know when it would be better to repair the roads, so they need to figure out the cost of road works for every of the coming t days before making a final decision. Your task is to help them and figure out the total repair cost of Berland's road system at the end of each these t days. As repair costs change over time, the set of selected roads can change on a daily basis as well.

Input

The first line contains a pair of integers n, m ($2 \le n \le 40000$, $n - 1 \le m \le 40000$), where n — the amount of cities, m — the amount of roads. Each of the following m lines contains a road description: three integer numbers x_i, y_i and p_i ($1 \le x_i, y_i \le n, x_i \ne y_i, 1 \le p_i \le 40000$), where x_i and y_i are indices of the cities connected by the given road, and p_i — initial cost of repairing it.

Then there follows a line with the only number t ($1 \le t \le 40000$), t — amount of days. The following t lines contain the scientists' predictions for the coming t days. Each of t lines contains a pair of integer numbers e_i, c_i ($1 \le e_i \le m, 1 \le c_i \le 40000$), where c_i — is the new repair cost for the road e_i.

It's possible to get from any city to any other city by the roads. The cost of repair for a single road can be changed more than once over time.

Output

Output t lines, each of them should contain the road system's total repair cost at the end of each day.

Sample input and output

stdin	stdout

4 6
1 2 10
2 3 20
2 4 30
1 3 40
3 4 50
4 1 60
3
4 22
5 17
4 14
	60
	47
	41

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