Problem A : Cinema

The main movie theater of the city consists of a single auditorium with rows of comfortable padded seats. Surprisingly, the comfortness of seats are not necessarily equal. Precisely, each seat has its own comfort value which is a non-negative integer number. A seat is more comfortable than another seat if its comfort value is larger. It is only possible to enter a row from the left side of the auditorium. Assume the seats in a row are numbered 1 to m from left to right. When a person enters a row, he/she always sits on the most comfortable seat which is free and accessible to him/her. If he/she sits at seat i, he/she blocks other persons coming later to sit on seats $i+1$ to m. If there are more than one free and accessible seat being the most comfortable, he/she sits on the leftmost one. The owner of the movie theater plans to improve the comfortness of some seats to have more audiences in the auditorium. Improving one unit in the comfortness of a seat costs some fix value. With the budget available, the owner knows the total improvement over all seats must not exceed a value k. Help the owner find the best way to improve the comfortness of seats by at most k units in total to have the maximum number of audiences in the auditorium.

Input

The first line contains three non-negative integers n, m, and k ($1 \leqslant n \cdot m \leqslant 3 \times 10^5, 0 \leqslant k \leqslant 10^{12}$) which are the number of rows, the number of seats in each row, and the total comfortness that can be added to all seats. The next n lines describe the comfort values of seats; each line contains m non-negative integers not more than 10^6 denoting the comfort values of seats from left to right for a row.

Output

Print a single line containing the maximum number of audiences.

Example

Standard Input	Standard Output
2 3 10 10 1 12 8 3 6	5

Standard Input	Standard Output
1 4 6 9 8 10 8	3

Standard Input	Standard Output
1 3 2 10 10 10	2

Problem B : JackRabbit Slim

We all love rabbits, right? Unfortunately, they don't even like us, rather, they love carrots instead!

They love carrots so much that they would do anything to get to a carrot, even running. That might sound easy, but rabbits are lazy, it's the jackrabbits who are the active ones. That's why this problem is about jackrabbits and not rabbits. In fact, it's about a very specific black-tailed jackrabbit, named Slim.

Earlier this morning, before the contest started, a truck loaded with carrots passed through the road by Slim's house. Slim was so lucky that n carrots dropped off on the road. We number them 1 to n from left to right. When Slim left home, he found a carrot. After taking a look around, his plan for the day became clear: Locate the closest carrot, get to it and eat it, then repeat.

The road by Slim's house is a straight road of length 10^9 and the i-th carrot on the road is at coordinate x_i.

Consider Slim starts his day by standing at the position of the j-th carrot and eating it. Then as long as there are more carrots, he follows the following steps:

- Locate the closest remaining carrot. If there are two closest carrots, choose the one to the right.
- Run to it and eat it. Simple!

For each value of j, we call D_j the total distance Slim runs if it starts the day by eating carrot j. For an unknown reason, we are interested in finding the sum of all D_j values, for all values of j ($1 \leqslant j \leqslant n$).

Input

The first line of the input contains a single integer n ($1 \leqslant n \leqslant 10^5$), the number of carrots. The second line of the input contains n space-separated distinct integers $x_1, \ldots, x_n$ ($0 \leqslant x_i \leqslant 10^9$). The coordinates of the carrots are in an increasing order.

Output

Output an integer value, which is the sum of all D_j values (for j from 1 through n).

Example

Standard Input	Standard Output
6 1 14 19 20 21 24	168

Problem C : Relay Race

In a relay running race, n athletes of a team are initially positioned along a road. Specifically, the initial position of athlete i is x_i (in meters from an origin). This athlete can run up to v_i meters per second. Initially, each athlete holds a baton.

The race starts by blowing a whistle and finishes when any athlete of the team holds all of the batons. In any moment during the race, each athlete can run along the road (in any of the two directions), or simply stop. When two athletes meet at the same position, each of them can pass all batons she or he holds to the other athlete.

You are the coach of the team. Your task is to find the shortest possible time the team can finish the race.

Input

In the first line of input, a single integer n ($1 \leqslant n \leqslant 10^5$) is given. In each of the next n lines, two space-separated integers x_i ($0 \leqslant x_i \leqslant 10^6$), and v_i ($1 \leqslant v_i \leqslant 10^6$) are given.

Output

In the only line of the output, print a single number, the minimum amount of time the team can finish the race (in seconds). Your answer is considered to be correct if it has an absolute error of at most 10^{-6}.

Example

Standard Input	Standard Output
3 4 1 0 3 10 1	2.50000

Standard Input	Standard Output
1 100000 1000000	0

Problem D : Cafebazaar's Chess Tournament

Ali hosts a yearly chess tournament for CafeBazaar's Shab-e Yalda festival. In a chess tournament, each pair of participants play a game against each other exactly once. Moreover, players are granted one point for a win, a half point for a draw, and no points for a loss toward their tournament score.

Danial has built a system to predict the result of Ali's tournament. Based on experience, he has assigned an opening skill and an ending skill to each of n participants in the tournament. For the i-th participant, let us denote the former with o_i and the latter with e_i. In a game between the i-th and j-th participants, Danial decides the result of the game according to the following rules:

1. If $o_i > o_j$ and $e_i > e_j$, then the i-th participant wins the game.
2. If $o_j > o_i$ and $e_j > e_i$, then the j-th participant wins the game.
3. Otherwise, the game ends in a draw.

To make the tournament more exciting, Ali wants to invite Danial to join the other n participants in the tournament. Since Danial has no prior experience in chess, he decides to practice for the tournament. Based on the amount of training, Danial can end up with any opening and ending skill. However, Danial has promised Ali that he will train in such a way that his opening skill will be **different** from the opening skill of the other participants. He will also keep his ending skill **different** from the ending skill of the other participants.

For his advertisement campaign, Ali wants to know the number of distinct possible final scores that Danial might get based on Danial's rules mentioned above. For example, Danial can achieve the scores 0, 1.5, 2.5, 3, 4, and 5 in the sample. For instance, the score 3 is obtained by setting the opening and ending skills of Danial to be 1.5. Since Ali and all other CafeBazaar programmers are busy planning the event, he has turned to you for help. Write a program to calculate this value.

Input

The first line of the input contains a single integer n ($1 \leqslant n \leqslant 200\,000$), the number of participants. The i-th line of the next n lines contains two integers o_i and e_i ($1 \leqslant o_i, e_i \leqslant n$), the opening and ending skills of the i-th participant, respectively. Note that the limits for opening and ending skills do not apply to Danial's opening and ending skills. More specifically, Danial's opening and ending skills can be any real numbers.

Output

In the only line of the output, print the number of distinct possible final scores for Danial.

Example

Standard Input	Standard Output
5 1 1 1 2 1 1 2 1 2 2	6

Problem E : The Big Surprise

"A big surprise is coming on the next Thursday!", the young mayor of TetrisCity announced in the social media. TetrisCity is the most populous and modern city in Neverland constructed on a flat area with endless clusters of high-rise buildings packed so closely together that they resemble a game of Tetris. Buildings look like axis-parallel boxes constructed on the ground and they are disjoint (they do not even touch each other).

The big surprise announced by the mayor is going to be a special delivery service using drones. The drones used in this service are a generation of quadcopters which can physically move only in one of x, y, and z directions. So, the distance traveled by a drone is the sum of distances traveled by it in each axis. The young mayor now has ordered to make the drones smart by equipping them with a software that computes the shortest path from any source to any destination avoiding the buildings. Your job is to develop this software.

Input

The first line of the input contains an integer n ($0 \leqslant n \leqslant 100$), specifying the number of buildings in the TetrisCity. Each of the next n lines contains 5 space-separated integers x, y, x', y', and h specifying a building: the coordinates (x, y) and (x', y') respectively specify the west-south corner and the east-north corner of the building, and h determines its height. It is guaranteed that the volume of the building is not zero. The source and destination appear at the end of the input in two separated lines; each containing x, y, and z coordinates. All numbers in the input are non-negative integers being at most 100 000. It is guaranteed that the source and destination are outside the buildings (they can be on the boundary of buildings). The shortest path can touch buildings and it is assumed that a drone looks like a point.

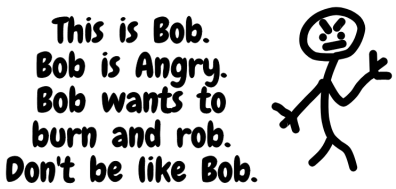
Output

In the output, print the length of the shortest path from the source to the destination avoiding the buildings.

Example

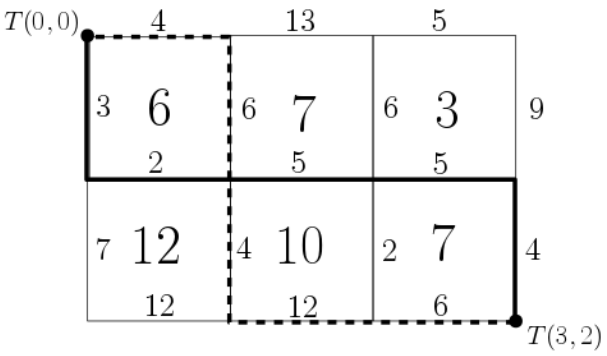
Standard Input	Standard Output
1 1 1 11 21 40 5 0 5 6 23 8	35

Problem F : Lets Burn and Rob Manhootan



There are two types of angry people in this world, those who burn and those who rob. But we programmers know that there is a third type; those who counter their anger, by both burning and robbing.

Bob lives in Manhootan. The city of Manhootan is like a grid of n rows and m columns, containing $n \times m$ blocks. The rows are numbered from 0 to $n-1$ from north to south and the columns are numbered from 0 to $m-1$ from west to east. The j-th block on i-th row is worth A_{ij}. Before the first row, between every two consecutive rows, and after the last row, there is a west-east street. The $n+1$ west-east streets are numbered from 0 to n from north to south. Similarly, before the first column, between every two consecutive columns, and after the last column, there is a north-south street. The $m+1$ north-south streets are numbered from 0 to m from west to east. The part of a street that is between two adjacent blocks is called a *street segment*. Each west-east street contains m street segments, numbered from 0 to $m-1$ from west to east. Similarly, each north-south street contains n street segments, numbered from 0 to $n-1$ from north to south. Since Manhootan is an expensive city, passing through street segments costs money. Passing through the j-th segment of the i-th west-east street costs H_{ij} and passing through the j-th segment of the i-th north-south street costs V_{ij}.



After a recent crisis in Manhootan, Bob got angry. He pierced his car's fuel tank to make it leak on the streets he passes. Let's call the intersection of i-th west-east street and j-th north-south street, $T(i,j)$. At first, Bob is at $T(0,0)$. He is planning to drive to $T(n,m)$ only going east and south, then returning to $T(0,0)$ only going west and north. Then, he is going to light the leaked fuels and put the streets on fire. After that, Bob will rob all the blocks that are caught inside the fire, i.e., any block that can not reach outside of Manhootan without crossing a burning street, will be robbed by Bob. The figure shows one possible plan for Rob in the sample.

Now, you can't be like Bob, but you can help him find the most profitable burn-and-rob plan. In other words, maximize the total value of the robbed blocks minus the total cost of the passed street segments. A street segment may be passed twice, which should be paid for each separately.

Input

The first line of input contains two integers n and m ($1 \leqslant n, m \leqslant 200$), the number of rows and columns, respectively. The next n lines describe the value of blocks; each containing m numbers, where the j-th number of the i-th line denotes A_{ij} ($1 \leqslant A_{ij} \leqslant 100$). The next $n+1$ lines describe the cost of west-east street segments. Each line contain m numbers, where the j-th number of the i-th line denotes H_{ij} ($1 \leqslant H_{ij} \leqslant 1000$). Finally, the next $m+1$ lines describe the cost of north-south street segments. Each line contains n numbers, where the j-th number of the i-th line denotes V_{ij} ($1 \leqslant V_{ij} \leqslant 1000$).

Output

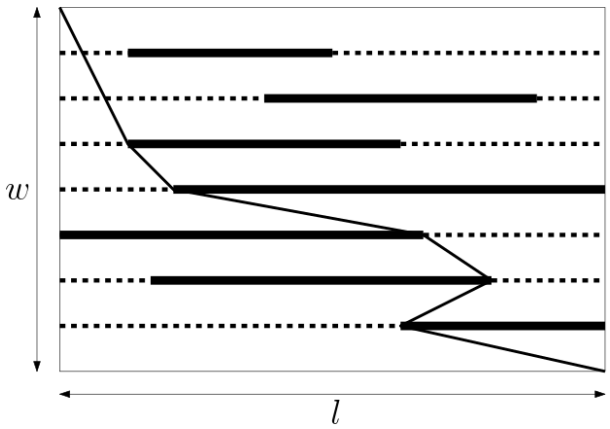
Print the profit of the most profitable plan. Note that the answer can be negative, zero, or positive.

Example

Standard Input	Standard Output
2 3 6 7 3 12 10 7 4 13 5 2 5 5 12 12 6 3 7 6 4 6 2 9 4	-28

Problem G : Gift Puzzle

Peyman's birthday party is being held. Keivan, his old friend, has bought a puzzle as a birthday gift. The puzzle consists of a flat rectangular board with a length of l and a width of w, and a thread. The board has n horizontal rails of length l placed at different distances from the top horizontal side. On each rail, there is one obstacle which can *slide freely* on the rail. An example of the board is depicted in the figure. The rails are illustrated by dotted lines, and the obstacles are illustrated by thick segments.



In order to solve the puzzle, one must connect the top-left corner of the board to the bottom-right corner using the supplied thread. The thread must be inside the board and cannot pass through obstacles. In the figure, one possible way to do the puzzle is shown. Since Keivan believes in Peyman's ability to solve hard puzzles, he wants to give Peyman the shortest thread while it is still possible to connect the two corners. So, kindly help Keivan to find the desired length of the thread.

Input

The first line of the input contains three integers l, w ($2 \leqslant l, w \leqslant 10^9$), the length and the width of the board, and n ($1 \leqslant n \leqslant \min(100\,000, w-1)$), the number of the rails. Each of the next n lines contains two integers y_i ($1 \leqslant y_i \leqslant w-1$), indicating the distance between the i-th rail and the top horizontal side, and l_i ($1 \leqslant l_i \leqslant l-1$), length of the obstacle on the i-th rail. Note that all y_i's are distinct. You may assume that all obstacles and the thread have a width of zero.

Output

In the only line of the output, print the minimum t for which it is possible to configure obstacles such that the top-left corner can be connected to the bottom-left corner using a thread of length t while avoiding obstacles. Your answer is considered to be correct if it has a relative error of at most 10^{-9}.

Example

Standard Input	Standard Output
5 6 5 1 1 2 3 3 3 4 1 5 4	7.848191962583

Problem H : Passport Control Gates

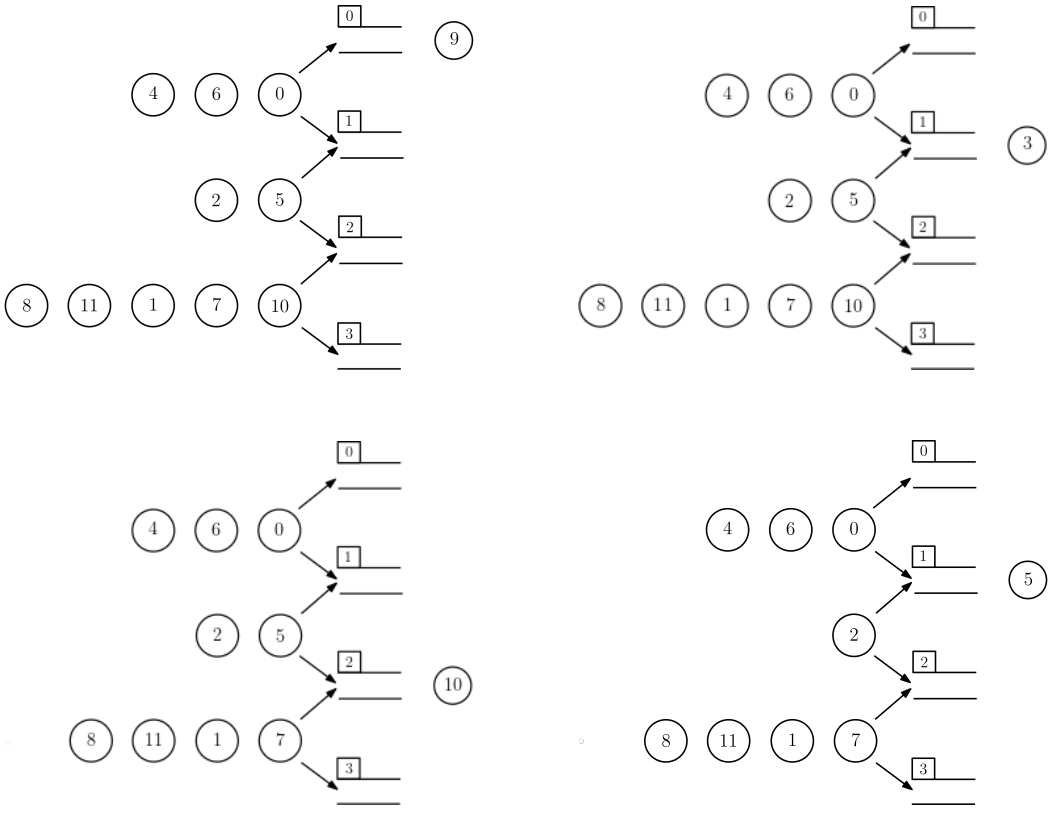
Everland and Neverland are two neighboring countries, and a huge number of Everlandian tourists visit Neverland every year. The governments want to analyze the passport control process in the border of Neverland and Everland, and need your help!

In the border, tourists stand in q queues for their passport check, and there are $q + 1$ passport control gates. If we number the gates from 0 to q and number the queues from 0 to $q - 1$, the tourists standing in queue i can only pass through gates i or $i + 1$.

Whenever gate i opens, if one of the queues i and $i - 1$ is empty or non-existent, the tourist at the front of the other queue passes through the gate. If both queues i and $i - 1$ are non-empty, the older tourist between the ones at the front of two queues passes through the gate. It is assumed that no two gates open at the same time.

We have a picture of n tourists standing in queues; waiting for the gates to open. Also, we have another picture that has been taken a while later, that some of the tourists from the first picture have passed through the gates. The tourists in the pictures are numbered from 0 to $n-1$, in the order of their ages such that the person number 0 is the youngest and the person number $n-1$ is the oldest. The picture below shows the first four configurations of the tourists in the first sample.

You are asked to find any valid sequence of gates' opening that might have happened between the times the two pictures were taken, or claim that it is impossible. A gate can be opened multiple times in the sequence.



Input

The first line of the input contains two integers q ($1 \leqslant q \leqslant 100\,000$), the number of queues, and n ($0 \leqslant n \leqslant 100\,000$), the number of tourists in the first picture. The i-th line of the next following q lines describes queue $i-1$ in the first picture. Each queue description starts with a number k ($0 \leqslant k \leqslant n$) that shows the size of the queue, followed by k integers that indicate the tourist numbers in that queue, from the back to the front. The tourist numbers are unique and non-negative integers less than n. In the next following q lines the description of the second picture appears in the same format.

Output

If there is no valid sequence, print *Impossible*. If there are valid sequences, output any of them in the following format. Print the length of the sequence in the first line and the sequence itself in the second line. In your sequence, every time any gate opens, there must be at least one tourist waiting for it.

Example

Standard Input	Standard Output
3 12 4 4 6 0 9 3 2 5 3 5 8 11 1 7 10 3 4 6 0 1 2 2 8 11	6 0 1 2 1 2 3

Standard Input	Standard Output
3 3 1 2 1 0 1 1 1 2 0 1 1	Impossible

Standard Input	Standard Output
1 2 2 0 1 2 1 0	Impossible

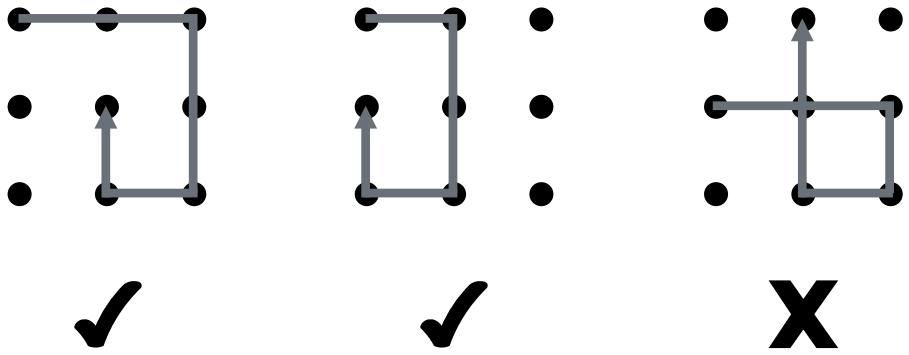
Standard Input	Standard Output
1 2 2 0 1 2 0 1	0

Problem I : Password

After working for several months at Cafebazaar, Farhad became rich enough to buy a house in the valley of the rich. There he met Shirin several times. Now, he is considering proposing to her whether she would marry him. To surprise her, he wants to install an application on her phone that pops up at the exact right time and asks if she would marry him.

However, to install the application secretly, he needs her password which he unfortunately does not have. He knows her password is a poly-line consisting of vertical or horizontal line segments. Each line segment connects the center of two cells in a 3×3 grid. Looking at her hand while she unlocked her phone, Farhad learned the direction of each line segment. However, he was too distracted to also learn the length of each segment. He also knows that her phone's operating system does not allow the poly-line to intersect with itself even in one point.

Farhad wants to distract Shirin long enough to try all possible patterns given what he already knows. Unfortunately, he has no idea how long that will take. That is why, he has now turned to you for help. Help him by writing a program that calculates the total number of possible password patterns given the direction of the line segments. The following figure depicts two valid and one invalid patterns given the line segments were directed towards right, down, left, and up in order.



Input

In the only line of the input, a single string is given consisting of characters `R`, `U`, `L`, and `D` which represent a line segment toward right, up, left, and down, respectively. The length of this string is at most 10. Every two consecutive characters is guaranteed to be different.

Output

In the only line of the output, print the number of patterns satisfying Farhad's knowledge of the password. Note that this number might be zero.

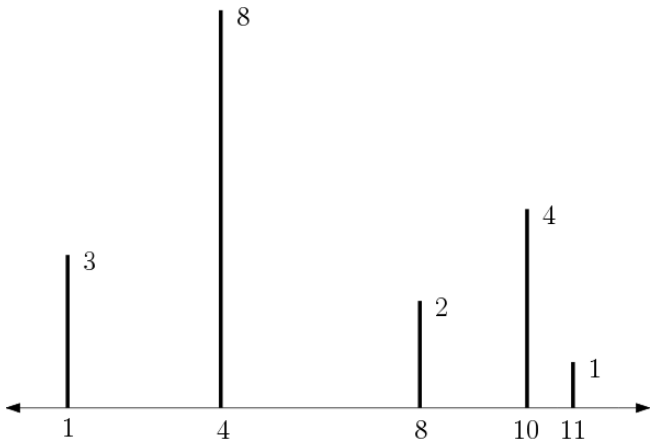
Example

Standard Input	Standard Output
DRU	15

Standard Input	Standard Output
R	9

Problem J : Greedy Termite

There are n wooden rods vertically placed over a horizontal line. The rods are numbered 1 through n from left to right. Each rod i ($1 \leqslant i \leqslant n$) is placed at position x_i and has a height h_i.



A termite wants to eat all the rods one by one. It starts eating from an arbitrary rod s ($1 \leqslant s \leqslant n$). Then, after eating a rod i, the termite selects the next rod to eat based on the following method. Among the remaining rods j, the one with maximum $h_j - |x_i - x_j|$ is selected. If there are ties, the one with minimum $|x_i - x_j|$ is selected. If there are still ties, the left-most rod is selected.

Your task is to calculate the total (horizontal) distance traveled by the termite to eat all the rods.

Input

The first line of the input contains two space-separated integers n, the number of rods, and s, the starting rod number ($1 \leqslant s \leqslant n \leqslant 100\,000$). The rods are described in the next n lines. On the line $1 + i$ ($1 \leqslant i \leqslant n$), the i-th rod is specified with two space-separated integers x_i ($|x_i| \leqslant 10^9$) and h_i ($1 \leqslant h_i \leqslant 10^9$). Additionally, for each i ($1 \leqslant i \leqslant n - 1$), $x_i < x_{i+1}$.

Output

You should print a single integer denoting the total distance traveled by the termite.

Example

Standard Input	Standard Output
5 3 1 3 4 8 8 2 10 4 11 1	17

Problem K : Plan B

In CrisisLand, every now and then there is a crisis that seriously threatens the security of the country. In the recent crisis, a series of civil protests occurred in multiple cities across the country. The protest started at a city and quickly spread out to other cities. To prevent this from happening again, the government as Plan B –after shutting down the Internet across the country– has decided to quickly send his forces to surround the city where the protest starts from. A city is surrounded if there are forces in each of its neighboring cities. The government has military bases in b different cities, each having many forces to be sent to all cities. The government knows his forces can not pass through the city from which the protest starts as they may be killed. Knowing this, it may be a case that some cities are not possible to be surrounded by forces. These cities are called *critical*. It is assumed that if there is a military base in a city, that city is not critical. Now, the government is eager to know whether there are critical cities in the country or not. As a legionnaire geek, help the government find his answer.

Oh, we forgot to explain the structure of CrisisLand! To resolve this *crisis*, we should mention that CrisisLand consists of n cities numbered from 1 to n. The cities are connected via m roads which can be used in both directions. Two cities are neighbors if there is a road between them. It is guaranteed that the road network of CrisisLand is connected.

Input

The first line of the input contains three positive integers n, m, and b denoting the number of cities, roads, and military bases, respectively ($1 \leqslant b \leqslant n \leqslant 100\,000$, $1 \leqslant m \leqslant 200\,000$). Each of the next m lines contains two numbers v_i and u_i denoting a road between cities v_i and u_i. The last line consists of b integers, the cities having a military base.

Output

The output consists of two lines. The first line contains the number of critical cities. The second line contains the critical cities in ascending order.

Example

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

Problem L : Python Classes

Mr. Programmer has been busy working on his awesome startup project lately. The project was coded in Python. In the past few days, he was hugely refactoring the source code without any testing. After all changes have been made, he tried to run the project, but he figured out that the code is not executing properly anymore due to some cyclic dependencies created in his source code. He struggled a little bit moving things around only to know he is just messing up things even further. Maybe if I move all the classes to a single file I'll be able to resolve the problem, he thought. But that plan didn't work either. With a great frustration and feeling pressure to launch his project as soon as possible, he has finally decided to outsource the task of putting things in the correct order to algorithmic specialists.

The source code is a single Python file. It consists of a bunch of classes. Mr. Programmer doesn't like multiple inheritances much, so each class in his code has at most one super class. As he does not want to leak the content of the classes, he has nulled out the contents and effectively replaced them with the "pass" command in Python. Therefore, when a class does not have a superclass, it looks like below:

```
class <CLASS_NAME>:
    pass
```

When a class has a superclass, it looks like below:

```
class <CLASS_NAME>(<SUPER_CLASS_NAME>):
    pass
```

In the above codes, *CLASS_NAME* and *SUPER_CLASS_NAME* are arbitrary Python identifiers. For further clarification look at the sample input.

We know that the file is executable if for each class having a super class, its superclass appears somewhere before the class in the file. Mr. Programmer doesn't like too many changes to his file, so he wants to put the file in the correct order with the minimum number of changes. Each change is counted as cutting a single class (along with its content) and pasting it in some other position in the file. Help Mr. Programmer find the minimum number of changes to file so that the file becomes executable.

Input

The input is a python file. It consists of some python classes separated by a single blank line. The classes in the input are exactly in the format described in the problem statement. Class identifiers only consists of English upper or lower case letters, and their length does not exceed 10 characters (for the sake of this problem you should not make any other assumptions about the identifiers). There are exactly 4 space characters before each "pass" command. There are at most $2,000$ classes in the file. It is guaranteed that all super classes are defined somewhere in the file, and no class is defined multiple times. Furthermore, a class is not its own superclass.

Output

Print a single integer as the minimum number of changes that have to be made so that the file becomes executable, or `-1` if it is not possible to do so.

Example

Standard Input	Standard Output

`class B(A):`
`  pass`
 	
`class C(A):`
`  pass`
 	
`class A:`
`  pass`
	`1`