Problem A. Party Planning

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds (*10 seconds for Java*)
Memory limit:	256 mebibytes

After the contest, Misof wants to make a party for some of the participants. And everybody knows that the secret to the perfect party is in inviting just the right set of participants.

Based on knowing his friends, Misof has established a set of very simple rules he has to follow, such as "if I invite Constantine, I should not invite Michael" or "if I invite mihnevsev, I should invite Volodymyr too".

But there is one more problem. Some of Misof's friends don't really care about invitations: if they feel like it, they will come to the party even if they are not invited, and if they don't feel like it, they will not come even if they got an invitation. Luckily, at least Misof is able to find out their decisions in advance. Once he knows which of these friends will and which ones won't attend the party, he can send out invitations to some of the other people.

You are (implicitly) given a set of people Misof knows. Each of them is either careless about invitations, or respects the invitations. First, each of the careless people will decide whether or not they want to come to the party. Afterwards, Misof may send out invitations to some of the people who respect the invitations. Finally, the people who come to the party are precisely all the careless people who decided to come to the party and all the respectful people who were invited. The party will be a success if and only if the set of invited people follows *all* of Misof's rules. Each of the rules has the form of an implication: "if event A happens, then event B must also happen".

Depending on what the careless people decide, Misof might be able or unable to organize the perfect party. Find out whether Misof is able to organize the perfect party for every possible decision of the careless people, or at least for one such decision, or never at all.

Input

The first line of input contains the number of rules r. This number is between 0 and 10^5, inclusive.

Each of the next r lines describes one rule "if event A happens, then event B must also happen". The line has the form "eventA eventB", where each event is either "`+Someone`" for the event "Someone comes", or "`-Someone`" for the event "Someone does not come". Each person is uniquely identified by a case-sensitive string of 1 to 10 alphanumeric English characters. The names are case-sensitive, that is, "`AK47`" and "`ak47`" are two distinct people.

The next line of input contains the number s of people who don't care about invitations. This number is between 0 and 10^5, inclusive. Each of the next s lines contains the name of one of these people. All of these names are pairwise distinct.

The set of people who might come to the party is the set of people who are mentioned in the input at least once.

Output

Output a single line with a single string. If it is always possible for Misof to organize the perfect party, output the string "`YES`". If it is never possible, output "`NO`". Otherwise, output "`MAYBE`" (in this last case, the possibility to organize a perfect party depends upon the decisions of people who cannot be influenced by Misof).

Examples

standard input	standard output

`1`
`+Usamec +Maru`
`0`
	`YES`

`6`
`+a +b`
`+b +c`
`+c -a`
`-c +a`
`+c -b`
`-b +a`
`0`
	`NO`

`1`
`+Hermi +Tanicka`
`1`
`Hermi`
	`YES`

`1`
`+JohnNy64 -Katka`
`3`
`JohnNy64`
`Katka`
`Hermi`
	`MAYBE`

Explanations

In first example, there are three good options: either Misof invites nobody, or he invites only Maru, or he invites both of them.

In the second example, the set of rules is contradictory.

In third example, if Hermi comes to the party, Misof will invite Tanicka and the party will be perfect. And if Hermi does not come, Misof may or may not invite Tanicka, the party will be perfect anyway.

In fourth example, if nobody comes to the party, it will be perfect. But if both JohnNy64 and Katka come, the party will not be good, as the only rule will be violated. Mišof has no control over this, so he only has to hope that things will turn out well.

Problem B. Flights

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Petya is the lead engineer of Air Bubundia. Currently he works on a new aircraft model and needs to decide on the fuel tank size.

Vasya is the lead cartographer of Air Bubundia. He made a precise map of all Bubundian cities and calculated the amount of fuel necessary for a direct flight between each pair of cities.

New aircraft must be able to get from any city to any another city in Bubundia (maybe with intermediate stops). Petya wants to make as small as possible. How small the tank can be if the given requirement must be satisfied?

Input

The first line contains an integer n ($1 \leq n \leq 1\,000$), the number of cities in Bubundia.

n lines of n integers follow. j-th number in i-th line is the amount of fuel needed for a direct flight from city i to city j (or zero if $i = j$). All amounts are nonnegative and less than 10^9.

Output

A single number — the minimum possible tank size.

Example

standard input	standard output

4
0 10 12 16
11 0 8 9
10 13 0 22
13 10 17 0
	10

Problem C. Recoloring

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

There is an undirected graph on n vertices and m edges. Initially each vertex is of red, green, or blue color.

We call the coloring of the graph *valid* if endpoints of any edge are of different colors. Is is possible to change the color of every vertex such that the new coloring becomes valid?

Note that you have to change color of every vertex, it is not allowed to keep the same color anywhere. Also note that initial coloring is not necessary valid.

Input

In the first line there are two numbers n, m ($1 \le n \le 1000$, $1 \le m \le 20\,000$).

Next line contains n characters, each of them being `R`, `G` or `B`, i-th of them denoting the color of i-th vertex.

Next m lines contain two integers each, denoting edges of the graph.

The graph contains no loops or multiple edges.

Output

Print n characters `R`, `G` or `B` — new colors of each vertex. If it is not possible to make a valid coloring after recoloring all vertices, print a single word “`Impossible`” (without quotes).

Examples

standard input	standard output
4 5 RRRG 1 3 1 4 3 4 2 4 2 3	GGBR
4 5 RGRR 1 3 1 4 3 4 2 4 2 3	Impossible

Problem D. Bridges

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Given undirected graph. Find all bridges in it.

Input

First line of the input consists of two positive integers n and m ($1 \le n \le 2 \cdot 10^4$, $1 \le m \le 2 \cdot 10^5$) — number of vertices of edges in the graph, respectively. Each of next m lines contains description of one edge, given with two integers — numbers of starting and ending points, respectively.

Output

In first line print one integer b — number of bridges in given graph. Next line must contain b integer — indices of bridge edges in increasing order. Edges are numbered with 1 in order given in the input.

Example

standard input	standard output

6 7
1 2
2 3
3 4
1 3
4 5
4 6
5 6
	1
	3

Problem E. Cut Vertices

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	64 mebibytes

Given undirected graph. Find all cut vertices in it. The vertice is called *cut vertice*, if after its deletion number of connected components increased.

Input

First line of the input contains two integers n and m — number of vertices and edges in the given graph, respectively ($1 \le n \le 2 \cdot 10^4$, $1 \le m \le 2 \cdot 10^5$).

Each of next m lines contains description of one edge. Edge i is described by two integers b_i and e_i — indices of vertices, connected by this edge ($1 \le b_i, e_i \le n$).

Output

In first line print one integer b — number of the cut vertices in the given graph. On the next line print b integers — indices of cut vertices in the increasing order.

Example

standard input	standard output

9 12
1 2
2 3
4 5
2 6
2 7
8 9
1 3
1 4
1 5
6 7
3 8
3 9
	3
	1
	2
	3

Problem F. Triplets

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	64 mebibytes

Lets define *triplet* as an triangle with vertices in the selected points; two points are connected directly if they are vertices of one triplet and indirectly if there exists a sequence of triplets t_1, t_2, …, t_n such as first point is vertice of first triplet, last point is vertice of last triplet and each two neighboring triplets in the sequence share a vertice.

Given n point on the plane and several tripets connecting them such as each two points are connected directly or indirectly. Set of triplets may contain the same triplets twice. Find all *important* triplets, i.e. triplets with next property: after removal of this triplet at least two points are not connected by the system of remaining triplets.

Input

First line of the input contains two integers n and m — number of points and number of triplets, respectively ($3 \le n \le 10^5$, $1 \le m \le 10^5$). Then m lines follow, describing the triplets. Each line contains three pairwise different integers between 1 and n — numbers of vertices, forming this triplet. It is guaranteed that in the given system each two of n points are connected directly or indirectly.

Output

In first line you must print number of important triples. In the next line print indices of important triples in ascending order. Triplets are sorted in the order given in the input, numeration of triplets starts with 1.

Examples

standard input	standard output
3 1 1 2 3	1 1

3 2
1 2 3
3 2 1
	0

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

Problem G. Eulerian Path

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Given an undirected connected graph such as no more than three vertices have odd degree.

Check if exists the path that visits every edge of the graph exactly once

If such a path exists, then print it.

Input

First line of the input contains one integer n — number of vertices in the graph ($1 \le n \le 100\,000$). Then n lines follow, defining the edges. i-th of those lines starts from integer m_i — number of edges that are incident to i then m_i positive integers follow — indices of vertices connected with i with a single edge.

The graph may contain multiple edges, but it cannot contain self-loops.

The graph contains no more than 300 000 edges.

Output

If solution exists, in the first line print one integer k – number of the edges in the path, and in second print $k+1$ integers — indices of vertices in order of traversal. If there are more than one solution, print any of them.

If there are no solutions, print one integer -1 instead.

Examples

standard input	standard output

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

Problem H. Even and Odd

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

There is a connected undirected graph with n vertices and m edges. There are no self-loops or multiple edges.

An unordered pair of distinct vertices $\{u, v\}$ is called *even* if every simple path (path visits no vertex twice) between u and v contains an even number of edges. Similarly, an unordered pair of distinct vertices is called *odd* if every simple path connecting them consists of an odd number of edges.

Find the number of odd and even vertex pairs in the given graph.

Input

The first line of the input contains two integers n and m ($1 \le n \le 200\,000$, $0 \le m \le 200\,000$) — the number of vertices and edges respectively. Each of the next m lines contains two endpoints of the corresponding edge a_i and b_i ($1 \le a_i, b_i \le n$). It is guaranteed that the graph is connected and contains no self-loops and multiple edges.

Output

Print two integers — the number of even and odd pairs respectively.

Examples

standard input	standard output

4 3
1 2
1 3
1 4
	3 3

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

Note

In the first sample, there are three even pairs $\{2, 3\}$, $\{2, 4\}$, $\{3, 4\}$, and three odd pairs $\{1, 2\}$, $\{1, 3\}$, $\{1, 4\}$.

In the second sample, there is a single odd pair $\{1, 2\}$. One can verify that for any other pair of vertices it is possible to find both an even path and an odd path connecting them.

Problem I. Kingdom and Reforms

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

In the kingdom of Berland there are n cities and m bidirectional roads connecting some pairs of cities. The government did not care for the road system for the long time, hence there was no rhyme or reason to how the roads were constructed. It is possible, for instance, for multiple roads to connect the same pair of cities, or even for a city to have an adjacent road leading to the same city (or multiple such roads even). Furthermore, it may be possible that the country is not connected at all, that is, for a pair of cities there may not be a way to reach one from the other using the roads. Needless to say, all roads are in a very sorry state and desperately need repairs.

A new transportation minister was assigned recently, tasked with renovation of the road system. The minister does not really want to create new roads nor to destroy any of the old ones, since the citizens grew accustomed to the existing road system, however poor it may be. The renovation the minister has in mind will consist of converting each existing road into either a modern automobile highway or a high-speed train track. To avoid anti-monopoly investigations, the minister has to ensure that each city has at least one adjacent highway and at least one adjacent train track. Help the minister choose the new type for each road so that this condition is satisfied, or determine that doing this is impossible.

Input

The first line contains two integers n and m ($1 \le n \le 300\,000$, $0 \le m \le 300\,000$), the number of cities and roads respectively.

The following m lines describe the roads. The i-th of these lines contains two integers u_i, v_i ($1 \le u_i, v_i \le n$), indices of the cities connected by the i-th road.

Note that multiple roads and roads connecting a city to itself are allowed, and it is not guaranteed that all the cities are connected by the road network.

Output

If there is a suitable assignment of road types, print a string of m characters. The i-th of these characters should be “`H`” if the i-th road should be converted into a highway, or “`T`” if it should be made into a train track. If there are many possible assignments, print any of them.

If there is no suitable assignment, print the only string “`Impossible`”.

Examples

standard input	standard output

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

`3 1`
`1 2`
	`Impossible`

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