Problem A. Maximum by Inclusion Matching

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

Your task is to find maximum by inclusion matching in the given bipartite graph. Maximum by inclusion matching is a matching for which it is impossible to add an edge from the given graph preserving a matching property. I.e. a maximum by inclusion matching is not included in other matching.

Input

The first line contains two integers n and m ($1 \le n, m \le 100000$) — the number of vertices in the part A and in the part B.

The following n lines contain description of the edges. The i-th vertex from A described in the $(i+1)$-th line. Each line contains the vertices from B connected with the i-th vertex from A. The vertices in both parts are numbered independently. Each line ends with 0.

Output

The first line must contain integer l — number of edges in maximum by inclusion matching. Each of the following l lines must contain two integers u_j, v_j — the edges of maximum matching.

Example

standard input	standard output

2 2
1 2 0
2 0
	2
	1 1
	2 2

Problem B. Minimum Edge Cover

Input file:	*standard input*
Output file:	*standard output*
Time limit:	0.5 seconds
Memory limit:	256 mebibytes

Your task is to find minimum edge cover in the given bipartite graph.

The set of edges R is an edge cover if and only if each vertex of the given graph is incident to at least one edge from R.

Input

The first line contains two integers n and m ($1 \leq n, m \leq 250$) — the number of vertices in the part A and the number of vertices in the part B.

The following n lines contain edges description. The i-th vertex from A is described in $(i + 1)$-th line. Each line contains numbers of the vertices from B connected to the i-th vertex from A. The vertices from parts are numbered independently. Each line ends with 0.

Output

The first line must contain integer l — the number of edges in the minimum edge cover. Each of the following l lines must contain two integers u_j, v_j — edges of the minimum edge cover. If there are multiple solutions, print any of them.

Example

standard input	standard output

2 2
1 2 0
2 0
	2
	1 1
	2 2

Problem C. Cubes

Input file:	*standard input*
Output file:	*standard output*
Time limit:	0.5 seconds
Memory limit:	256 mebibytes

Parents' gift to Peter is a set of kids' cubes. Each cube has six faces and on each face there was written one letter.

Peter wants to show his older sister that he learnt to read. To make it he wants to form her name from those cubes. But it is difficult for Peter because different letters can be on the same cube and in this case Peter can not use both of them to make name of his sister. But it is possible that the same letter is written on different cubes. Your task is to help Peter.

You are given the set of cubes and the name of his sister. You need to check if it is possible to make his sister's name from the cubes given. If it is possible you need to determine the order of cubes in which Peter needs to arrange them.

Input

The first line contains integer n ($1 \le n \le 100$) — the number of the cubes.

The second line contains the name of Peter's sister — the word consisting of capital Latin letters. The length of the name does not exceed 100.

Each of the following n lines contains 6 letters (only capital Latin letters) which are written on the corresponding cube.

Output

Print "YES" in the first line if it is possible to form sister's name from the cubes and "NO" in the other case.

If the answer is "YES" print in the second line m different numbers from the range $1 \ldots n$, where m — the number of letters in sister's name. The i-th number must be equal to the number of cube which is needed to put in the i-th place. The cubes are numbered from 1 in the order they appear in the input. If there are multiple solutions you are allowed to print any of them.

Examples

standard input	standard output
4 ANN ANNNNN BCDEFG HIJKLM NOPQRS	NO
6 HELEN ABCDEF GHIJKL MNOPQL STUVWN EIUOZK EIUOZK	YES 2 5 3 1 4

Problem D. Dominoes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	256 mebibytes

There is a $N \times N$ square chessboard. P squares were removed from the chessboard. Find out if it is possible to cover the remaining part of the chessboard with domino pieces (each piece is 2×1).

Put each domino piece on two neighboring cells exactly. No two pieces can cover the same cell.

Your task is to find the required tiling, if it exists.

Input

The first line contains two integer numbers N ($1 \leq N \leq 40$) and P ($0 \leq P < N^2$) separated by a space.

Each of the following P lines contains a pair of numbers separated by a space — coordinates of the removed cell ($1 \leq X_i, Y_i \leq N$). The bottom left square has the coordinates $(1, 1)$, the bottom right square — $(N, 1)$.

Output

If the required covering exists, output "`Yes`" in the first line and "`No`" in the opposite case.

If the first answer is positive, then output in the second line integer number N_h — the number of horizontally placed pieces. Each of the following N_h lines should contain two integers — the coordinates of the left cell covered by a corresponding piece.

Output in the next line N_v — the number of vertically placed pieces. And the following N_v lines should contain the coordinates of the bottom cell covered by a corresponding piece.

Example

standard input	standard output

4 10
1 3
1 2
1 1
2 1
3 1
4 1
3 2
4 2
3 3
4 3
	Yes
	2
	1 4
	3 4
	1
	2 2

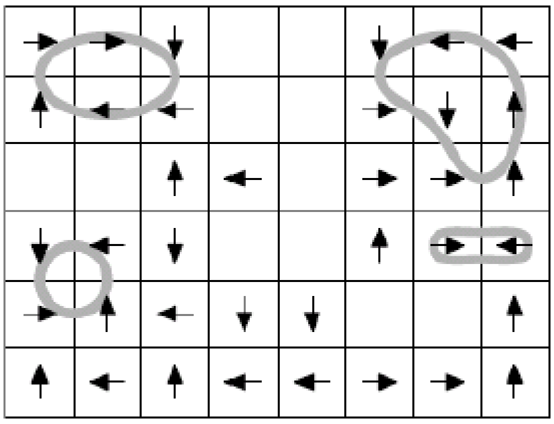
Problem E. Arrows

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	256 megabytes

Kostya found a puzzle game which his grandfather played being a child — a board containing $n \times m$ cells. Rotating arrows were attached to some of the cells (Kostya thinks that many years ago the arrows were on each cell). Unfortunately, Kostya's grandfather did not remember what had to be done in this puzzle, so Kostya invents rules by himself.

He noted the following feature. Let's direct each arrow to one of the side neighboring cells, which also has an arrow (we consider that each arrow on the board has the neighboring cell with an arrow). Now there are cycles on the board and we move on these cycles in directions of the arrows.

For example, the picture shows 4 cycles.



Kostya invented two tasks for this puzzle: "rotate the arrows so that the number of cycles becomes minimum possible" and "rotate the arrows so that the number of cycles becomes maximum possible". Your task is to find these two numbers.

Input

The first line contains two integers n and m $(1 \leq n, m \leq 100)$.

Each of the following n lines contains m symbols — the description of the board. The symbol "`?`" corresponds to the cell with an arrow and the symbol "`.`" — to the empty cell.

Output

Print the board which contains minimum possible number of cycles and print the board which contains maximum possible number of cycles. Separate both boards by one empty line.

To print the directions of the arrows change symbols "`?`" to letters which correspond to directions. Use letters "`R`", "`L`", "`U`" and "`D`" to right, left, up and down directions respectively.

Example

standard input	standard output
3 4 ???? ???? .??.	RDLD ULUL .RU. DRLD UDDU .UU.

Problem F. The Greatest Dominance

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	256 megabytes

You are given a bipartite graph. The first part contains N vertices and the second part contains M vertices.

Let's choose some subset of the vertices from the first part and let's call them X. Let Y be such a subset of the vertices from the second part that are connected to at least one vertex from X (i.e. Y is a subset of the neighboring vertices of X).

Your task is to find such subset X making the expression $|X| - |Y|$ as maximal as possible.

Input

The first line contains two integers N and M ($1 \le M, N \le 300$).

The following N lines contain description of the vertices from the first part. Each description is given in the format: the i-th line begins with the number D_i equals to the numbers of neighbors of vertex i from the first part. After it D_i integers are given — the numbers of neighbors in the second part of the vertex i from the first part.

Output

Print in the first line the number of vertices in X.

Print in the second line the vertex numbers from X.

It is possible that X is empty. In this case the expected expression value equals to 0.

Examples

standard input	standard output

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

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

Problem G. Voting Block

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	256 megabytes

The members of a committee vote "`Yes`" or "`No`" on various issues. However, some pairs of committee members have formed alliances, promising never to cast opposite votes. No one is willing to vote contrary to her own opinion, so some of the members may need to abstain from voting in order to avoid conflict between allies.

We have devised a method to determine who should abstain. Before each issue is voted on, we randomly assign each committee member an identifying number 1, 2, $\cdots$, n. The member will then indicate her opinion on the issue. Then we will calculate the smallest collection of abstentions that will avoid conflict.

You are given the descriptions of the voters, calculate the smallest collection of abstentions that will avoid conflict.

Input

The first input line contains the number n ($1 \leq n \leq 50$) – the number of voters. The i-th following line contains the description of the i-th voter. The description of the i-th voter contains its member's opinion, either "`Y`" or "`N`", followed by the identifying number of each higher-numbered member who has formed an alliance with its member (or nothing if there is no such member).

Output

In the first line print the minimal amount of abstentions. In the second line print their indexes in ascending order, if there is more than one answers, print the one that is earliest lexicographically.

Examples

standard input	standard output

`2`
`Y 2`
`N`
	`1`
	`1`

`3`
`N 2`
`N 3`
`Y`
	`1`
	`2`

Problem H. Kind Vitya

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	256 megabytes

In the summer camp teachers are able to hold N different lectures. Unfortunately the camp will be conducted during only M days.

For humanitarian reasons at most one lecture can be hold in one day. Teachers will be in the camp not for all the time, that's why some lectures can be held only on certain days only. It was decided to hold the maximum number of lectures, because the national team must prepare for the international competition as good as possible.

Kind Vitya gave to the participants the information about days when each lecture can be held. Participants do not know which of the camp schedule will be chosen, but they are interested in lectures which will be held in any schedule.

Input

The first line contains two integers — N and M ($1 \le N, M \le 500$).

Each of the following lines contains description of the lectures — one per line. For each lecture the numbers of days within this lecture can be held is specified. The numbers in line are separated by one or more spaces. The days and the lectures are numbered from one.

Output

Print the lecture numbers which will be held in any camp schedule in ascending order.

Example

standard input	standard output

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

Problem I. Edges Belonging to Some Maximum Matching

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are given a bipartite graph. Your task is to find edges which belong to at least one maximum matching in this graph.

Input

The first line contains three integers n, m, k ($1 \le n, m \le 500$, $0 \le k \le min(n \cdot m, 10^5)$), where n, m are the sizes of the parts and k is the number of the edges in this graph.

Each of the following k lines contains two integers a_i, b_i: descriptions of the endpoints of an edge. The value a_i ($1 \le a_i \le n$) equals to the vertex number from the first part and b_i ($1 \le b_i \le m$) equals to the vertex number from the second part.

Output

Print in the first line the required number of required edges.

Print in the second line the indexes of all edges which belong to at least one maximum matching. All indexes must be printed in ascending order. Edges numbered from one, in the same order that they appear in the input.

Example

standard input	standard output

3 3 6
1 2
1 3
2 1
2 3
3 3
3 2
	5
	1 2 3 5 6

Problem J. Removing Magical Tiles

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

Ayimok is a wizard.

His daily task is to arrange magical tiles in a line, and then cast a magic spell.

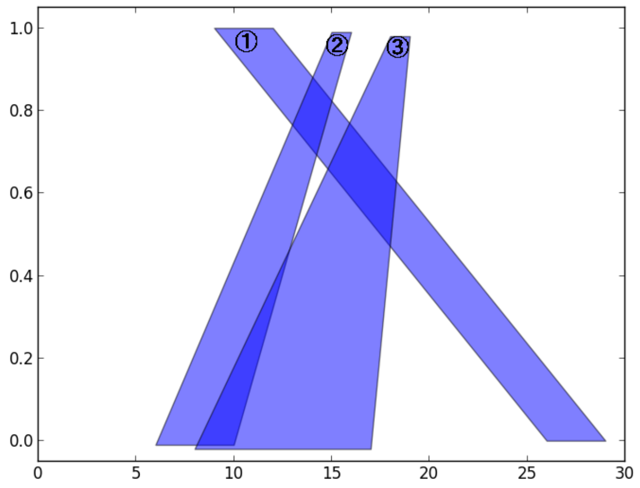
Magical tiles and their arrangement follow the rules below:

- Each magical tile has the shape of a trapezoid with a height of 1.
- Some magical tiles may overlap with other magical tiles.
- Magical tiles are arranged on the 2D coordinate system.
- Magical tiles' upper edge lay on a horizontal line, where its y coordinate is 1.
- Magical tiles' lower edge lay on a horizontal line, where its y coordinate is 0.

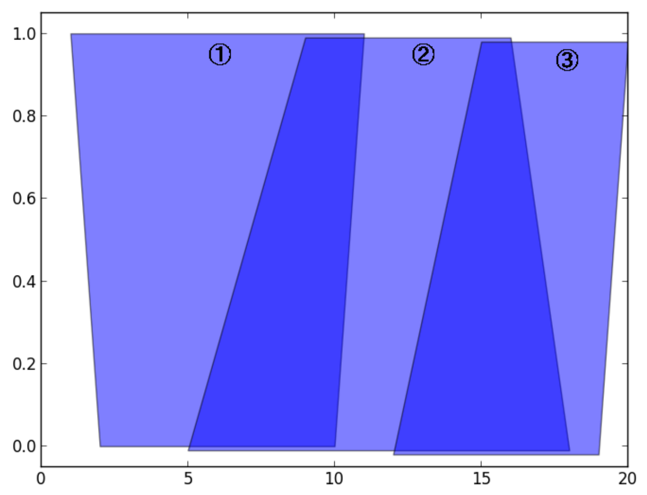
He has to remove these magical tiles away after he finishes casting a magic spell. One day, he found that removing a number of magical tiles is so tiresome, so he decided to simplify its process.

Now, he has learned "Magnetization Spell" to remove magical tiles efficiently. The spell removes a set of magical tiles at once. Any pair of two magical tiles in the set must overlap with each other.

For example, in figure below, he can cast Magnetization Spell on all of three magical tiles to remove them away at once, since they all overlap with each other. (Note that the heights of magical tiles are intentionally changed for easier understandings.)

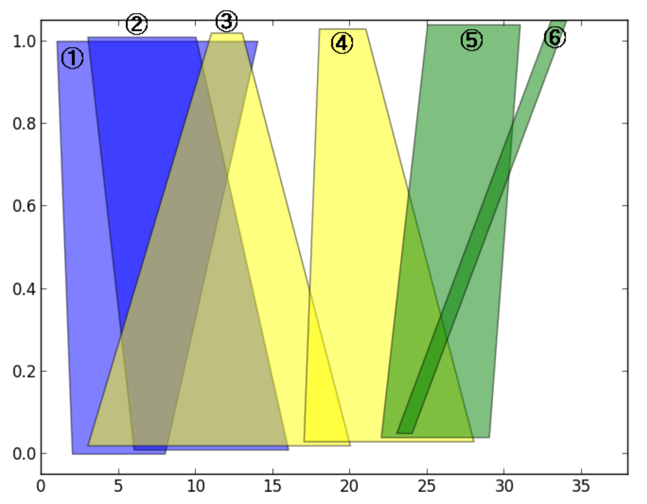


However, in second figure, he can not cast Magnetization Spell on all of three magical tiles at once, since tile 1 and tile 3 are not overlapped.

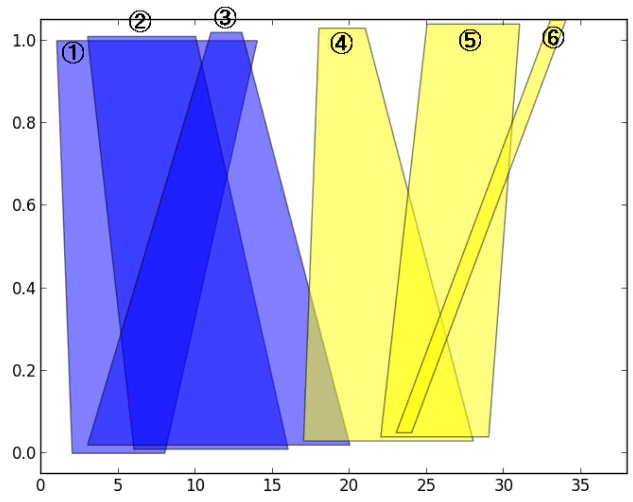


He wants to remove all the magical tiles with the minimum number of Magnetization Spells, because casting the spell requires magic power exhaustively.

Lets consider next case (third picture). He can remove all of magical tiles by casting Magnetization Spell three times; first spell for tile 1 and 2, second spell for tile 3 and 4, and third spell for tile 5 and 6.



Actually, he can remove all of the magical tiles with casting Magnetization Spell just twice; first spell for tile 1, 2 and 3, second spell for tile 4, 5 and 6. And it is the minimum number of required spells for removing all magical tiles (look at picture below):



Given a set of magical tiles, your task is to create a program which outputs the minimum number of casting Magnetization Spell to remove all of these magical tiles away.

There might be a magical tile which does not overlap with other tiles, however, he still needs to cast Magnetization Spell to remove it.

Input

The first line contains an integer N, which means the number of magical tiles.

Then, N lines which contain the information of magical tiles follow.

The (i+1)-th line includes four integers $xLower_{i1}$, $xLower_{i2}$, $xUpper_{i1}$, and $xUpper_{i2}$, which means i-th magical tile's corner points are located at $(xLower_{i1}, 0)$, $(xLower_{i2}, 0)$, $(xUpper_{i1}, 1)$, $(xUpper_{i2}, 1)$.

You can assume that no two magical tiles will share their corner points. That is, all $xLower_{ij}$ are distinct, and all $xUpper_{ij}$ are also distinct. The input follows the following constraints: $1 \leq N \leq 10^5$, $1 \leq xLower_{i1} < xLower_{i2} \leq 10^9$, $1 \leq xUpper_{i1} < xUpper_{i2} \leq 10^9$. All $xLower_{ij}$ are distinct. All $xUpper_{ij}$ are distinct.

Output

Your program must output the required minimum number of casting Magnetization Spell to remove all the magical tiles.

Examples

standard input	standard output
3	1
26 29 9 12	
6 10 15 16	
8 17 18 19	
3	2
2 10 1 11	
5 18 9 16	
12 19 15 20	
6	2
2 8 1 14	
6 16 3 10	
3 20 11 13	
17 28 18 21	
22 29 25 31	
23 24 33 34	

Problem K. Angel and Demon

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

Once again, we are in the dance class of professor Padegras to solve the problems of the great maestro. This time, at first glance, he faces a classic matching problem: there are N boys and N girls in his class, and for each possible pair of a boy and a girl, Padegras knows if they have a mutual sympathy to each other (and so, they can dance together). Looks like the only thing he needs is to know if there exists a perfect match.

The good news for the professor are that there is an Angel watching over him. The Angel enjoys making new sympathies in the professor's class.

The bad news for the professor are that there is also a Demon who dislikes the Angel. The Demon spoils existing sympathies.

More formally, there are K days left before the graduation ball in Mvodsk State University. On each day, the Angel makes exactly one new sympathy, and the Demon destroys exactly one existing sympathy. At every point of time, all sympathies are mutual. Note that the Angel can not make a sympathy that already exists at that moment.

The Angel and the Demon are actually playing a game. The Angel is said to be the winner if, after both players perform all their K moves, there is a perfect match in the sympathy graph. If there is no perfect match, the Demon wins.

The professor knows that one of the players always performs his move in the morning and another one in the evening, but he does not know who moves when. In both possible cases, determine the result of the game assuming that both players play optimally.

Input

The first line of input contains a single positive integer number T denoting the number of test cases. It is followed by T test case descriptions.

The first line of each description contains two integer numbers N and K ($0 \le K \le 10$).

The next N lines describe the relations in the class at the beginning of the game, each line contains exactly N characters which are 1 or 0. The j-th character of the i-th of these lines is 1 if the boy number i and the girl number j have a sympathy to each other, and is 0 otherwise. It is guaranteed that there will be at least one 1 and at least one 0, so both the Angel and the Demon are always able to make a move.

The sum of the values of N^2 for all test cases will not exceed 100 000.

Output

For each test case, print exactly one line containing two words separated by a space. First, print the result of the game where the Angel moves first. After that, print the result of the game where the Demon moves first. The result of the game is the name of the winner. Each name must be either "`Angel`" or "`Demon`", without quotes.

Example

standard input	standard output

2
3 2
111
101
111
3 1
100
010
001
	Angel Angel
	Demon Angel