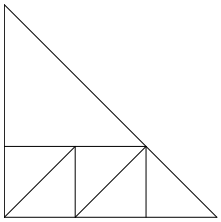
Problem A. All for Love

Input file:	`stdin`
Output file:	`stdout`
Time limit:	10 seconds
Memory limit:	64 megabytes

Masha has already tried all traditional ways to attract attention of Misha. The last chance to win the desired man is to use magic. And the most suitable moment to use magic is Halloween evening. To summon dark forces she bought large magic plate and a set of small plates. Every plate has a form of the right isosceles triangle. To achieve a magic effect, one need to completely cover the large plate with small ones. Moreover, every small plate should be placed in such a way that each of it's cathetuses is parallel to some cathetus of the large plate. To make the process of covering easier, an instruction is included into magic collection. Instruction says: let's introduce coordinate system in such a way that the large plate will become a triangle with vertices in $(0, 0)$, $(0, n)$ and $(n, 0)$. Then the instruction gives the exact position of every small plate in this coordinate system. Masha carefully followed this instruction and constructed the coverage. But she is still in doubt: what if the instruction is wrong and all efforts are useless? This question gives her no sleep, and she asks you to verify the instruction. More formally, let's call a coverage of the large triangle by small triangles correct if all the following conditions hold:

- all triangles are right and isosceles,
- no two triangles cross,
- every triangle is inside the large one,
- every part of the large triangle is covered by some small triangle.



An example of the correct coverage.

Input

Input consits of several test cases. The first line contains one positive integer $t \leq 10$ — the number of test cases. Every case starts with a line with two integers n and m — the length of the large triangle's cathetus and the number of small triangles in the coverage correspondingly ($1 \leq n \leq 25000, 1 \leq m \leq 100000$). The rest of the test case consists of m lines containing 6 non-negative integers each: $x_1, y_1, x_2, y_2, x_3, y_3$, which denote the coordinates of vertices of the small triangles. These numbers don't exceed 25000. Every triangle is guaranteed to be nondegenerate. Total number of triangles in all test cases doesn't exceed 200000.

Output

For every test case output one line with a word «`YES`», if the given coverage is correct, and «`NO`» otherwise.

Examples

stdin	stdout
3 1 1 0 0 0 1 1 0 2 3 0 0 1 1 0 1 0 1 1 1 0 2 1 1 2 0 1 0 3 6 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 2 1 1 0 2 0 2 1 2 0 2 1 3 0 0 1 2 1 0 3	YES NO YES

Problem B. Be a Smart Raftsman

Input file:	`stdin`
Output file:	`stdout`
Time limit:	6 seconds
Memory limit:	64 megabytes

On Halloween, when other people are gathering to tell terrifying stories (see problem H. «Hero of our time»), to torment poor pumpkins (see problem J. «Jealous Cucumber»), to create demonic teams (see problem C. «Coach's Trouble»), in other words, to do all that stuff, you don't want to be ordinary, so now you are planning to try some extreme sports. Rafting would be the best choise, because when water is cold enough (as it is now!) rafting is extremely extreme.

You are members of a rafting crew which consists of n participants. You navigate the river, and your goal is to pass m consecutive riffles and to get from the start to the finish as soon as possible. The i-th riffle is characterized by critical weight c_i, and the j-th participant is characterized by his or her weight w_j. If your raft goes through the i-th riffle with people on board with total weight exceeding c_i, it capsizes. This part of rafting is the most exciting, but it requires additional time to get on the raft after capsize. So sometimes it is better to take a group of people with total weight not exceeding critical weight on the raft, while others pass the distance by foot.

More formally, consider $m + 1$ points $p_0, p_1, \ldots, p_m$, where p_0 is the start and p_m is the finish. Each of the intermediate points p_i ($1 \le i \le m - 1$) is located between the i-th and $(i + 1)$-th riffle. It takes D_i minutes for the raft to get from p_{i-1} to p_i, if it passes the i-th riffle with capsize, and d_i minutes otherwise. The j-th participant can get from p_{i-1} to p_i in t_j minutes by foot. It takes him or her s_j minutes to get on the raft or to get off the raft on the bank. Before each of the riffles your group is divided into two parts. The first part rafts through the riffle (with or without capsize), and the second part goes on the bank to the next intermediate point. At this point they always wait for the raft, or the raft waits for all the walkers, if it arrives sooner. Then some participants who are on the raft can get off, and some participants who are on the bank can get on the raft. They can't start doing it before the raft and all the walkers arrive to the point. The total time required for this action is equal to the sum of s_j for people changing their place. Then you pass the next riffle and so on, until you get to the finish. Note that no one can start moving to the next point until others have changed places.

Remember, that you start at the point p_0 having all your group on the bank (not on the raft), and you must finish at the point p_m also with all participants on the bank and the raft with you. You are not allowed to leave the raft at the start or at some intermediate point, and walk to the finish without it. You can take all the group to the raft to pass a riffle, but you can't leave the raft empty during the trip. Your task is to calculate the minimal time required to reach the finish.

Input

The first line of the input contains two numbers n and m ($1 \le n \le 10, 1 \le m \le 1000$). Following n lines describe participants. Each of them consists of three integers w_j, t_j, s_j. Following m lines also contains three integers each — c_i, D_i and d_i for the i-th riffle. All numbers in the input are positive integers not exceeding 10000.

Output

Output the minimal time.

Examples

stdin	stdout
2 3 50 5 1 70 20 1 30 15 10 60 100 10 70 100 10	51

Note

Consider an optimal strategy for the example:

1. both participants get on the raft (2 minutes),
2. they pass the first riffle with capsize (15 minutes),
3. second participant gets off (1 minute),
4. first participant raft through the riffle, while his mate walks ($\max(10, 20) = 20$ minutes),
5. they swap places (2 minutes),
6. they get to the finish — first one by foot, second one by raft ($\max(10, 5) = 10$),
7. second participant gets off (1 minute).

Problem C. Coach's Trouble

Input file:	stdin
Output file:	stdout
Time limit:	2 seconds
Memory limit:	64 megabytes

Berland is a modern country, and one could hardly find a man, who would believe evil spirits. Everybody knows that celebrating Halloween is just tribute to old tradition. However, a coach of Berland University student programming teams wants to divide students ($3N$ in total) into teams of three to have as many demonic teams as possible at the moment of future Halloween contest. But he knows for some triples of students that they can't make a demonic team, as they haven't performed well in action in previous contests. He has a list of these K forbidden triples. The coach supposes that any three students can make a demonic team unless they do not form a triple contained in the list. And now he wants to know the number of such partitions that all teams are demonic.

Input

The first line of the input contains two integers N and K, separated by one space ($1 \le N \le 1000, 0 \le K \le 20$). Next K lines contain three integers each a_i, b_i, c_i ($1 \le i \le K, 1 \le a_i, b_i, c_i \le 3N$). All triples are unique, that is they all are diffent as sets, and $a_i \ne b_i, a_i \ne c_i, b_i \ne c_i$.

Output

The output should contain the only number without leading zeroes — the answer to the task.

Examples

stdin	stdout
2 0	10
2 3 1 2 3 4 5 6 1 4 5	8

Problem D. Doors

Input file:	`stdin`
Output file:	`stdout`
Time limit:	6 seconds
Memory limit:	64 megabytes

It seems that the Museum of the Modern Technology is the only place where they don't celebrate Halloween! How can they be so far from all the mistical and mysterious? You can hardly believe that two weeks ago the Museum staff have destroyed the last detail that could bring with its ominous creak a bit of Halloween atmosphere to the realm of technology. They have replaced the old wooden doors with new automatic ones, and now they are scratching their heads over how to configure the doors.

By the order of the Director, two automatic doors were purchased. An automatic door is characterized by parameter t, called the *waiting period*, which can be set to an integer value from 1 to 10^9 during the door installation. Then the door functions on the following principle. If a person passes through the door at time p, the door opens at time $p-t$ and closes at time $p+t$. There is an exceptional case when several people go in a row with a time interval not exceeding $2t$ between any two consecutive people. In this case the door opens only once, t seconds before the first person in the row, and it closes t seconds after the last person in the row has passed through it.

It is very important to set the optimal values of the door parameters. On the one hand, if the doors open and close too often, it will annoy visitors. On the other hand, if both doors stay opened for a long time, visitors can get cold.

More formally, two lists of time moments are given. At the moments $p_1 < p_2 < \cdots < p_n$ people have passed through the first door, and at the moments $q_1 < q_2 < \cdots < q_m$ people have passed through the second one. The task is to use the given statistics to find the optimal *waiting periods* for the doors — t_1 for the first door and t_2 for the second one that satisfy the following conditions:

1. The total number of openings of the doors must be minimal possible.
2. There is no continuous interval of time that both doors are opened during this interval and its length exceeds the given value d.

Input

The first line of the input contains three integers n, m and d ($1 \le n, m \le 5000$, $1 \le d \le 10^9$). The second line contains numbers p_i, and the third line contains numbers q_i, given in the ascending order ($1 \le p_i, q_i \le 10^9$).

Output

Output two integers t_1 and t_2, separated by a single space. If there are multiple solutions, output any. If there is no solution, output «`No solution`».

Examples

stdin	stdout
3 2 4 1 6 13 7 11	3 2

Problem E. Excursion

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	64 megabytes

One day a group of students, consisting of boys (a heads) and girls (b heads), got to an idea to make an excursion led by their school teacher over the city, they lived in. At the very start of the party schoolboys went away walking separately from the teacher, rather than obedient school girls, who stayed with the teacher. Anyhow, when approaching any boutique during the excursion, some girls can leave the group to go for a fitting their lovely dresses, and they will never come back again. On the contrary, ruddy cheeked and cheered up boys can go back to the teacher and stay with him with great enjoy until the end of excursion. At some points of time scrupulous teacher recalculates the number of students in the group and writes it down in his notebook.

Now the teacher wants to evaluate the effectiveness of extracurricular work, so he askes you to find any feasible student joining/disjoining schedule.

Input

The first line of input contains two integers a and b ($1 \le a, b \le 100$) — the number of boys and girls respectively. The second line has the only integer n ($1 \le n \le 100$) — the number of notes in teacher's book. The third one contains exactly n non-negative integers not exceeding 200, denoting numbers the teacher wrote down, in the order of their appearance in the notebook.

Output

If there are any mistakes in teacher's notes, that is no feasible schedule exists, print «ERROR». Otherwise, print to output n lines. i-th line should contain two non-negative integers, denoting the number of schoolboys joined and the number of schoolgirls separated from the teacher respectively exactly before he recalculated students at i-th time. If there are multiple solutions, output any.

Examples

stdin	stdout
3 3 2 2 3	1 2 2 1
3 3 3 1 2 5	ERROR
2 2 1 2	0 0

Problem F. Funny Feature

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	64 megabytes

In the process of preparation to Halloween it was decided to plant some pumpkins on the $n \times m$ meters rectangular platform. The platform is divided to $n \times m$ identical square cells with 1 meter length sides.

The arrangement of pumpkins was carefully prepared, and you are given the resulting plan. For each cell of this platform it is given how many pumpkins should be planted in it. All these quantities appear to be from 1 to 5, inclusive.

Special pumpkin-landing machine was bought to plant Halloween symbols. It can perform a simple operation: plant pumpkin to the cell, specified by its coordinates (the first coordinate ranges 1 to n, and the second one ranges 1 to m).

At the last moment an unpleasant feature of the machine was discovered. Every time this machine plants a pumpkin to the specified cell, one more pumpkin is also planted to all cells, which are adjacent to the specified one and already have at least one pumpkin in it. One cell is called adjacent to another one if they share a side.

Besides for technical reasons you cannot specify the same cell twice.

Now Halloween celebration is under the threat of failure. You are asked to write the program which finds the sequence of $n \times m$ operations leading to demanded landing of pumpkins, or informs that it is impossible.

Input

The first line of input contains two integers n and m — sizes of the field ($1 \le n, m \le 200$). Next n lines contain m integers from 1 to 5 — how many pumpkins should be planted in each cell of the platform. Numbers in lines are separated by single spaces.

Output

If the solution exists, print $n \times m$ lines with two numbers in each: a line and a column numbers, where the next pumpkin should be landed. If there are multiple solutions, print any of them.

If the solution does not exist, print a single line «`No solution`» (quotes for clarity).

Examples

stdin	stdout
1 2 1 2	1 2 1 1
1 3 1 3 1	1 2 1 3 1 1
3 3 2 4 2 2 1 2 3 2 3	1 2 3 1 3 3 1 1 1 3 3 2 2 1 2 3 2 2
2 1 1 1	No solution

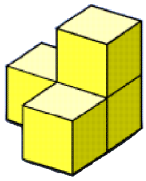
Problem G. Gena's Soul Cakes

Input file:	`stdin`
Output file:	`stdout`
Time limit:	6 seconds
Memory limit:	64 megabytes

This story is about a boy, called Gena. He is quite a usual boy who goes to school, celebrates All Hallows Eve, also known as Halloween Day, and besides likes programming contests.

One Halloween Day he went from house to house begging for "soul cakes". He happened to collect many "soul cakes"of various tastes that day, and his happiness seemed have no limits. But his elder brother Roma, who was lazy and rude boy, demanded to give him ten cakes. Gena didn't loose control over the situation and gave his brother ten cakes of the same taste in order not to show that he had many tastes, because Roma could demand more cakes. Since that time Gena was thinking about that situation. And after Gena had learnt on Geometry classes about three-dimentional space figures, he invented the following problem.

Consider non-convex figure denoted as "corner"in three-dimentional space. It consists of four equal cubes. One of the cubes has a common face with each of the others. There is the only one point belonging to all four cubes, and this point coinsides one vertex of any cubes (see picture below). Every "corner"is charaterized by two values: color and an integer length of an edge of any cubes (denoted as "size"). "Size"is always equal to some power of two, so only index is given. The problem is to construct a big "corner"comprising given figures under the restiction of using as little number of different colors as possible given "size"of this big "corner".



Input

The first line of input contains two space separated integers: c — the number of colors, and k — the index of power of two which is equal to "size"of big "corner"($1 \le c, k \le 1000$). Next c lines contain k integers a_{ij} each. a_{ij} equals the number of "corners"with color i and "size"2^j. ($0 \le a_{ij} \le 1000$ for $1 \le i \le c, 0 \le j < k$).

Output

The first line of output should contain integer n denoting the number of figure types used in constructing big "corner". Next n lines should contain a_i, b_i, c_i denoting color, "size"and number of "corner"of such type respectively. Please, separate these numbers by single spaces. If there are multiple solutions, output any of them. Every type of "corner"should appear in output no more than once. If it is impossible to construct big "corner"even using all figures, print «NO SOLUTION» to output.

Examples

stdin	stdout
1 1 8	1 1 0 8
2 2 1 7 7 0	3 1 1 7 1 0 1 2 0 7

Problem H. Hero of Our Time

Input file:	stdin
Output file:	stdout
Time limit:	4 seconds
Memory limit:	64 megabytes

Saratov ACM ICPC teams have a tradition to come together on Halloween and recollect terrifying stories. And the most popular story among the newcomers is the story about the «Mescher Tree». A long time ago, when the famous Dmitry Mescheryakov aka Mescher was very young and he even didn't know how to write Dijkstra algorithm, he faced a difficult problem with a tree. Input file contained n — the number of vertices, and pairs of vertices, connected with an edge. Without thinking a lot (honestly, the exact reason of that mistake is unknown), he wrote the following code:

```
read(n);
for i := 1 to n do begin
read(u, v);
g[u, v] := true;
g[v, u] := true;
end;
```

Mescher successfully compiled his code, got WA on sample test and started long debugging... This story has become a true legend. So it's no surprise that Saratov ACM ICPC teams use the following definition: connected undirected graph with n vertices and n edges is called *Mescheryakov Tree* or, less formally, *Mescher Tree*. The area of application of *Mescher trees* is not well-studied, so we suggest you to solve one of the problems connected with such trees: given n, find the number of distinct *Mescher trees* with n vertices. Trees are labeled, i.e. two trees are considered distinct if and only if their adjacency matrices differ.

Input

Input contains single integer number n ($3 \le n \le 5000$).

Output

Output the number of *Mescher trees* with n vertices without leading zeroes.

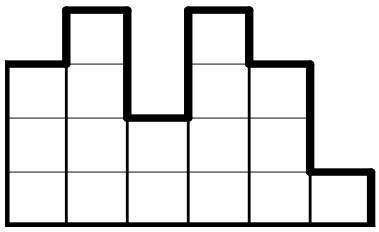
Examples

stdin	stdout
3	1

Problem I. Impudent Thief

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	64 megabytes

For most people Halloween evening is a time of having fun. But Mr. X chose the night after Halloween to commit a crime. He wants to get some boards to build a shed. And he decided to stole it from the fence of the neighbouring factory. But he wants to do it in such a way that nobody will notice boards loss. The fence consists of several boards with width, equal to 1, and integer heights (see picture). Mr. X is going to take some boards from the fence and then put the remaining boards together without changing their order to form a new fence. To be sure that noboby will notice the change, the perimeter of resulting fence should not be less than a half of the perimeter of initial fence. See picure description to understand the way of calculating fence's perimeter. With such constraint, Mr. X wants to maximize total height of extracted boards.



Perimeter of the fence is a perimeter of the figure, which is made by joining the rectangles corresponding to boards. For example, perimeter of the fence in the picture is marked bold and it's length is equal to 24.

Input

The first line contains integer number n ($1 \le n \le 50$) — number of boards in the fence. The second line contains n integer numbers h_i — heights of the boards ($1 \le h_i \le 100$). Boards are given from the leftmost one to the rightmost one.

Output

In the first line output s — maximal total height of some subset of the boards, which can be taken without violating the described rule. In the second line output k — number of boards in such subset. In the third line output k numbers of the boards which should be stolen. Boards are numbered starting from 1 as they appear in the input. Print numbers in any order. If there are multiple solutions, output any.

Examples

stdin	stdout
6 3 4 2 4 3 1	12 4 1 3 4 5

Problem J. Jealous Cucumber

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	64 megabytes

Mr. Cucumber was annoyed by pumpkins' dominance in Vegetable Land every Halloween and today he decided to restore justice. He took N peaceful pumpkins hostages and brought them to the hatch of his hutch. Then he numbered the hostages from 1 to N and started throwing them into the hatch one by one (from the 1-st to the N-th), waiting every time until previous hostage falls down. Cucumber's hutch is both-side infinite horizontally, and it has height N. The hutch was empty when Cucumber came. When a pumpkin is thrown into the hatch it has coordinates $(0, N)$. Then it falls by the following rules:

let (x, y) denote current pumpkin's coordinates.

1. if position $(x, y-1)$ is empty, pumpkin moves there,
2. else if position $(x-1, y)$ is empty and position $(x-1, y-1)$ is empty too, pumpkin moves to position $(x-1, y-1)$,
3. else if position $(x+1, y)$ is empty and position $(x+1, y-1)$ is empty too, pumpkin moves to position $(x+1, y-1)$

Pumpkin moves until it reaches the bottom of the hutch (i.e. its y-coordinate equals to 0) or it can't move anymore.

After that crafty villain demanded Vegetable Land's government to appoint him symbol of Halloween in an hour, otherwise he was going to shoot pumpkins in the hutch in such a way that every pumpkin which y-coordinate equals to H dies.

Some of the hostages taken by Mr. Cucumber are Very Important Pumpkins (VIP), so the government wants to know the exact numbers of pumpkins who are going to be killed to make its decision.

Input

Input contains two integers separated by a single space: N and H $(1 \le N \le 10^9)$, $(0 \le H \le 10^9)$ — the number of Mr. Cucumber's hostages and y-position of endangered pumpkins.

Output

Print numbers of pumpkins who are going to be killed from the leftmost to the rightmost one, separating them by spaces. It is guaranteed that output will contain at least one number.

Examples

stdin	stdout
5 0	5 2 1 3
6 1	6 4

Problem K. Kola

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	64 megabytes

One day Vasya decided to buy a bottle of his favourite Kola in the old vending machine. But he forgot that there was a Halloween that day, so evil forces had made an inner structure of the vending machine quite complicated. Inside the machine is represented by a table of size $n \times m$. Some of the cells are empty, and some contains obstacles of two types: '/' and '\'. One of the cell initially contains a bottle of Kola. After the purchasing it starts to fall vertically down by the following rules:

- If a cell immediately below the bottle is empty, the bottle falls down.
- If the bottle reaches an empty cell in the lowest row, it falls to the tray and Vasya can take it.
- Reaching and obstacle '/' ('\') the bottle moves left (right) without changing it's row and tries to continue to fall down if it is possible.
- The bottle stops to move when there is a wall in the current moving direction.
- The bottle stops to move when it moves from the cell with an obstacle of one type to the cell with an obstacle of another type.
- But if the bottle moves to the cell with the same type of obstacle as in the current cell, it continues moving.

Help Vasya to find out whether the bottle will reach the tray. In case of a positive answer, determine the number of column where it will happen.

Input

The first line of the input contains two integer numbers n and m ($1 \le n, m \le 100$). Then the description of the vending machine follows. It consists of n lines of m characters each: '.' means empty cell, 'P' means initial position of the bottle, '/' and '\' — mean obstacles of the corresponding type. It is guaranteed that the 'P' character appears exactly once.

Output

Print to the output -1 if the bottle doesn't reach the tray. Otherwise print the number of the column where the bottle will leave the vending machine. Columns are numbered starting from 1 from the leftmost one.

Examples

stdin	stdout
2 3 ./P ../	2
2 2 .P \/	-1
5 4 .P.. .\.. .//. ./.. /...	-1

16