

Problem 1. ICPC Contest Resolver

Input file: `input.txt`
Output file: `output.txt`
Time limit: 3 seconds
Memory limit: 256 mebibytes

Noname State University (NSU) Team has made it into ACM ICPC programming contest finals. Four hours into the competition, the rating has just been frozen, and our team is struggling. Luckily, Chuck Norris has come to the rescue. With his help, the team can become world champions! Now the team wants to make their victory as spectacular as possible during the ceremony where the results will be revealed.

A few Chuck Norris facts:

- Chuck Norris solved all problems way before their authors.
- Chuck Norris writes code as fast as the RAM allows.
- At any minute Chuck Norris can send any number of submissions for each problem. However, he has agreed to send submissions only for the NSU team.
- Chuck Norris can get any verdict he wishes for any of his submissions, however he has pitied the jury and decided to confine himself to two possible verdicts: "ACCEPTED" and "Wrong Answer".
- Chuck Norris can predict the future, so he knows everything about all submissions sent by other teams during the competition, including those sent after the score has been frozen.

The contest lasts for 300 minutes. There are K problems, numbered from 1 to K . Any team can submit a solution to any problem to the testing system at any moment during a competition. The testing system checks the submissions in order of their arrival and returns verdicts. A submission with the "ACCEPTED" verdict is considered successful, all other submissions are considered unsuccessful. A problem is considered solved by a team when there is at least one successful submission to it by the team. Penalty time for a problem equals the number of full minutes from the beginning of the contest to the first successful submission plus the number of submissions before the first successful submission multiplied by 20. Only submissions for a particular problem are taken into account when calculating penalty time.

The teams in the rating are listed in order of number of solved problems decreasing. Teams with equal numbers of solved problems are ordered by total penalty time increasing. The total penalty time equals the sum of penalty times for all solved problems. If teams have equal numbers of solved problems and equal penalty time, then they are sorted by team number increasing.

During the first 240 minutes of the contest the current standings (i.e. rating) are available. After that the rating is "frozen", i.e. it is no longer updated. During the closing ceremony the results of the submissions sent during the last hour are gradually disclosed using the ICPC Contest Resolver software.

Contest Resolver works in the following way. Initially it displays the rating at the freezing point. It is considered that none of the submissions sent after that moment has been judged yet. The unjudged submissions do not affect the rating, as if they didn't exist at all. Recalculation continues for as long as there are unjudged submissions. Recalculation consists in consecutive execution of the following steps.

- A team with the worst place with at least one unjudged submission is selected from the rating.

- A problem with the lowest number is selected, given that the team has at least one unjudged submission for this problem.
- All unjudged submissions of the team for this problem are judged in chronological order.
- After that the rating is updated based on the judged submissions. As the result the team can jump up, i.e. move to a better place in the rating.

Jump height is the difference between the position before recalculation and the position after recalculation.

Let *spectacularity* of a team's performance equal the sum of squared heights of all jumps of the team during Contest Resolver runtime. Note that a change in the team's position resulting from judging successful solutions of other teams is not considered to be a jump. Find the *optimal behavior* of the team (together with Chuck, of course) after the freeze. The optimal behavior is defined by the following conditions:

1. The team must become champion (i.e. take the first place in the final rating) if that is possible.
2. The spectacularity must be the maximal given that the first condition is satisfied.

If there are several variants of the optimal behavior any of them may be displayed.

Chuck Norris **can't** change the past, so the number of full minutes from the beginning of the contest to the moment of any additional submission must be in range from 240 to 299 inclusive. Assume that it is impossible to send a problem exactly in a whole number of minutes since the beginning of the contest. Thus it is impossible to receive penalty time of 300 for a problem solved from the first attempt.

Input

The first line of the input file contains three integers K , N and M , where K — the number of tasks in the contest, N — the number of participating teams and M — the number of submissions ($1 \leq K \leq 10$, $1 \leq N \leq 10^3$, $0 \leq M \leq 10^5$).

Each of the next M lines contains a description of a single submission. A submission is described in the format $t \ a \ p \ r$, where t — the number of full minutes from the beginning of the contest to the moment of submission, a — team number, p — problem number, and r — check verdict ($0 \leq t \leq 299$, $1 \leq a \leq N$, $1 \leq p \leq K$). The numbers t , a , p are integers. The verdict r is denoted by a single letter: A for a successful submission and W for an unsuccessful submission. Submissions are given in order of their arrival to the testing system.

Our NSU team has number 1. All given submissions of our team have been completed before the freeze, i.e. if $a = 1$ is true for a submission, then $t \leq 239$ is also true.

It is guaranteed that the total penalty time of any team except the NSU team cannot exceed 2 500 in the final rating.

Output

If the NSU team cannot become champion, the only line of the output file must contain the word **Losers**.

Otherwise the first line must contain the word **Champions**, and the second line must contain a single integer — the maximal possible spectacularity of the NSU team.

After that the optimal behavior must be displayed. The third line must contain an integer Q — the number of submissions after the freeze. Each of the following Q lines must contain information

about the corresponding submission. The submission must be described in the same format as in the input data. All submissions must be displayed in order of their arrival to the testing system.

The total number of submissions Q must not exceed 10^4 . It is guaranteed that there exists an optimal behavior that complies with this restriction if the NSU team can win.

Example

input.txt	output.txt
8 5 16 0 1 1 W 30 1 1 A 30 1 1 W 30 1 1 W 30 1 1 A 40 2 1 A 50 3 1 A 60 4 1 A 120 5 4 W 150 2 2 W 170 2 2 A 239 3 2 A 240 4 3 A 260 5 3 W 270 5 4 A 290 4 4 A	Champions 10 2 250 1 2 A 290 1 3 A

Комментарий

At the freezing moment the standing looked like this:

P	Team	1	2	3	4	5	=	Penalty
1	#2	+	+1	.	.	.	2	230
2	#3	+	+	.	.	.	2	289
3	#1	+1	?	?	.	.	1	50
4	#4	+	.	?	?	.	1	60
5	#5	.	.	?	?1	.	0	0

First, the submissions of team #5 for problem 3 (unsuccessful) and for problem 4 (successful) was judged. Since 5-th team remained on the last place, the places of the teams did not change:

P	Team	1	2	3	4	5	=	Penalty
1	#2	+	+1	.	.	.	2	230
2	#3	+	+	.	.	.	2	289
3	#1	+1	?	?	.	.	1	50
4	#4	+	.	?	?	.	1	60
5	#5	.	.	-1	+1	.	1	290

XV Open Cup named after E.V. Pankratiev
Grand Prix of Siberia, Sunday, November 2, 2014

After that the submission of the team #4 for problem 3 was judged, which turned out to be successful, and the team jumped up by one place:

P	Team	1	2	3	4	5	=	Penalty
1	#2	+	+1	.	.	.	2	230
2	#3	+	+	.	.	.	2	289
3	#4	+	.	+	?	.	2	300
4	#1	+1	?	?	.	.	1	50
5	#5	.	.	-1	+1	.	1	290

Next, the solution of problem 2 by the NSU team was judged as successful. As a result, it jumped from the 4-th place to the 3-rd, increasing its spectacularity by 1.

P	Team	1	2	3	4	5	=	Penalty
1	#2	+	+1	.	.	.	2	230
2	#3	+	+	.	.	.	2	289
3	#1	+1	+	?	.	.	2	300
4	#4	+	.	+	?	.	2	300
5	#5	.	.	-1	+1	.	1	290

As a result of successful submission of problem 4, the 4-th team jumped to the first place:

P	Team	1	2	3	4	5	=	Penalty
1	#4	+	.	+	+	.	3	590
2	#2	+	+1	.	.	.	2	230
3	#3	+	+	.	.	.	2	289
3	#1	+1	+	?	.	.	2	300
5	#5	.	.	-1	+1	.	1	290

Finally, the successful submission of the NSU team for problem 3 was judged, the team jumped from the 4-th place to the 1-st, and its spectacularity increased by 9.

P	Team	1	2	3	4	5	=	Penalty
1	#1	+1	+	+	.	.	3	590
2	#4	+	.	+	+	.	3	590
3	#2	+	+1	.	.	.	2	230
4	#3	+	+	.	.	.	2	289
5	#5	.	.	-1	+1	.	1	290

Problem 2. Horns and Hooves

Input file: `input.txt`
Output file: `output.txt`
Time limit: 2 seconds
Memory limit: 256 mebibytes

The attempted assault on Koreyko, a secret millionaire, was a disastrous failure. The foster brothers Panikovskiy and Balaganov have barricaded themselves in their office. They are playing a strange game. One of them has a room full of horns, and the other has a room full of hooves. They are taking all possible pairs (*horn*, *hoof*), and whoever has more expensive object in the pair wins. They want to know one thing — who will win more often. Their psychological integrity has been somewhat compromised and it might be beyond their powers at the moment to solve this problem by their own efforts. Please help them, and they will thank you... with horns and hooves.

Input

The first line of the input file contains two integers n and m , where n is the number of horn types and m is the number of hoof types ($1 \leq n, m \leq 10^5$).

The following n lines contain descriptions of horns, one type per line. Each type is defined by two positive integers; the first integer is the cost of one horn of the given type and the second integer is the supply of horns of this type (the number of individual units).

The following m lines contain descriptions of hoof types in the same format.

The cost of any horn or hoof does not exceed 10^9 .

The number of horns and hooves of each type is not greater than 10^4 .

Output

The output file must contain three numbers — the number of pairs where the price of horn is greater, equal or less than the price of hoof, respectively.

Example

input.txt	output.txt
3 2 1 2 2 3 4 1 2 3 3 1	4 9 11

Problem 3. String

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

The Research Institute of the Given Strings (RIGS) is investigating a new method of infinite string construction.

Initially there is an empty string $S_0^{(k)} = \varepsilon$. Each next version of the string is created in the following way. The current version of the string is repeated k times, and an arbitrary symbol is inserted between every two consecutive occurrences. The same number k is used for the construction of all versions, however, the inserted symbols may differ. This construction ultimately results in the infinite string $S_\infty^{(k)}$.

To illustrate, let's consider the series of strings ($k = 3$):

$$\begin{aligned} S_0^{(3)} &= \varepsilon(\text{empty}) \\ S_1^{(3)} &= \mathbf{rt} & (S_1^{(3)} = \boxed{\varepsilon} \mathbf{r} \boxed{\varepsilon} \mathbf{t} \boxed{\varepsilon}) \\ S_2^{(3)} &= \boxed{rt} \mathbf{x} \boxed{rt} \mathbf{r} \boxed{rt} \\ S_3^{(3)} &= \boxed{rtxttrrt} \mathbf{a} \boxed{rtxttrrt} \mathbf{r} \boxed{rtxttrrt} \\ &\dots \\ S_\infty^{(3)} &= rtxtrrtartxttrrttrrtxttrrtzrtxttrrtartxt \dots \end{aligned}$$

Given a string A of length n , which is the prefix of $S_\infty^{(k)}$, find the minimal k , for which this is possible.

In other words, your task is to find the minimal k , for which it is possible to construct a string $S_\infty^{(k)}$ in a way described above, so that it will have A as a prefix.

Input

Each input file contains several tests. The first line of the file contains a single integer T — the number of tests. Then T lines follow; each line describes a single test. The tests are nonempty and contain only lowercase Latin letters.

The number of tests does not exceed 10^5 . The sum of lengths of all the tests in the file does not exceed 10^6 .

Output

The output file must contain exactly T lines. Each line must contain the minimal value of k for the corresponding test.

Example

input.txt	output.txt
2	2
abacabab	3
rtxrtrrtartxttrrttrrtxt	

Problem 4. Books

Input file: `input.txt`
Output file: `output.txt`
Time limit: 6 seconds
Memory limit: 256 mebibytes

The Noname State University (NSU) is preparing for a new type of programming contest. Each university in this contest can be represented only by a single team of two students. There is also the official list of N books which the contestants must read to perform successfully. There's little time left before the contest, and not every student is able to read all books in time.

Every student knows for each set of books whether she will be able to read it in time. There are M students willing to participate in the contest in the NSU. The coach prepared a list of K candidate teams. To choose a single team for the contest, she wants to know each team's reading capability.

Assume a team has read a book if at least one member of the team has read it. Your task is to find for each candidate team and for each set of books whether the team members can plan their training in such a way that the team will have read all the books of this set in time.

Input

The first line of the input file contains three integers: N — number of books, M — number of students, and K — number of candidate teams ($1 \leq N \leq 21$, $2 \leq M \leq 6$, $1 \leq K \leq 6$). Zero-based numbering is used everywhere in this problem.

The i -th of the next M lines defines the capabilities of the i -th student. The student's capabilities are described with the string S_i of the length 2^N containing 0's and 1's. The k -th position of the string contains information on whether the student is able to read the set of books with a bitmask of k in time. This rule is described in detail below.

Let's number all books from 0 to $N-1$. Let X be a set of books. Define an array $a_0, a_1, a_2, \dots, a_{N-1}$ such that $a_j = 1$, if j -th book belongs to X , and $a_j = 0$ otherwise. Let's call the number

$$k = \sum_{j=0}^{N-1} a_j 2^j$$

the bitmask of the set X . Then the i -th student is able to read the set of books X in time if and only if the k -th symbol of the string S_i equals one.

t -th of the next K lines describes t -th candidate team. The team is defined by two integers a_t and b_t — the numbers of the students in the team. ($0 \leq a_t, b_t < M$, $a_t \neq b_t$, $t = 0 \dots K-1$).

Regarding the capabilities of each student the following is guaranteed. First, if a student can read a subset of books, she can read any of its subsets. Second, a student can always read an empty set of books, i.e. the starting symbol of her capability string always equals one.

Output

The output file must contain K lines consisting of 0's and 1's, each containing 2^N symbols. The t -th line must contain the capabilities of the t -th team in the same format in which the students' capabilities are given in the input. I. e. the k -th symbol of the t -th line must equal 1 if and only if the t -th team is able to read the set X of books with a bitmask k in time.

Example

input.txt	output.txt
3 5 3 11111010 11000000 11001000 11101000 11101000 0 1 1 2 3 4	11111111 11001100 11111110

Комментарий

The 0-th student can read one of the following sets: $\emptyset$, $\{0\}$, $\{1\}$, $\{2\}$, $\{0,1\}$, $\{1,2\}$. The first student can only read the 0-th book (or nothing). The second student can read either the 0-th book or the book numbered 2 (or nothing). The third and the fourth students can read any single book (or nothing).

The 0-th team together is able to read all the books if the 0-th student reads the books $\{1,2\}$, and the first student reads the book 0. It's easy to check that the team can also read any other set of books.

The first team can read the books 0 and 2, if its members read different books. The team can also read any subsets of the set $\{0,2\}$.

In the second team any member can read not more than one book. Hence, the team together can read any set of two or less books.

Problem 5. Big Bed

Input file: `input.txt`
Output file: `output.txt`
Time limit: 1 second
Memory limit: 256 mebibytes

Somewhere in the galaxy far, far away, on the planet Nibiru, Basilius, one of the most promising students, has received a special scholarship in the prestigious Nibirian State University (NSU). He's decided to buy a bed for his dorm room.

He wants a bed with the maximum possible volume, but, on the other hand, the size of his future bed is limited — Basilius wants to save some space in his room for his future shopping trophies.

Any bed can be crafted in the local store, provided it's a rectangular parallelepiped. But not any bed can be delivered from the store to the dorm room. It will have to be carried through portals, which resemble our common rectangular doors. The portals are perpendicular to the floor. Passing through a portal brings you either to the vicinity of some other portals or to a room. Help Basilius calculate the optimal bed size for his room. The distance between portals is significantly greater than the size of the dorm room.

It's assumed that the bed is oriented so that a fixed face will always be parallel to the floor and another fixed face will be parallel to the plane containing current portal.

The bed will not be rotated once it's lifted to be carried through portals to the room.

Input

The first line of the input file contains three integers a , b and c limiting the bed dimensions. One of the dimensions (length, width, and height) must not exceed a , another dimension must not exceed b , and the remaining dimension does not exceed c ($1 \leq a, b, c \leq 500$).

The next line contains a single integer n — the number of portals on the planet ($2 \leq n \leq 500$). The portals are numbered from 1 to n . The portal in the shop has number 1, and the portal leading to Basilius' room has number n .

The next n lines of the input file describe the Nibiru portals, one per line. i -th of these lines contains a description of the i -th portal and begins with three real numbers separated by spaces — w_i , h_i and k_i , the first two defining the width and height of the portal and the third one denoting the number of portals which can be reached directly by passing through the described portal ($1 \leq w_i, h_i \leq 300, 0 \leq k_i < n$). Then k_i pairwise distinct integers p_{ij} follow: they describe numbers of portals directly accessible from the i -th portal ($1 \leq p_{ij} \leq n$, $p_{ij} \neq i$).

Output

The output file must contain three positive integers x , y , z in any order — the dimensions of the bed with the maximum possible volume that Basilius can bring to his dorm room from the store. If there are several possible variants with the largest volume, print any of them. It is guaranteed that there exists a solution to the problem.

Examples

input.txt	output.txt
300 300 300 4 300 300 1 2 100 200 1 4 300 200 1 1 300 300 2 3 1	100 200 300
300 300 300 4 300 300 2 2 3 100 200 1 4 300 200 1 4 300 300 1 1	200 300 300

Problem 6. Antichamber

Input file: `input.txt`
Output file: `output.txt`
Time limit: 5 seconds
Memory limit: 256 mebibytes

In the Antichamber video game the player has a special tool called the Brick Tool. You must model its work in a simple two-dimensional variant.

The game takes place in an infinite grid. Each cell is either white or black. Two cells are considered *adjacent* if they share a side. A maximum connected set of black cells is called a *component*. The size of a component is the number of cells in it.

The tool allows the player to repaint any cell to the opposite color. Straight afterwards checks are performed which can change the state of the grid.

If a cell has been repainted to white and the number of components has increased, then some component has split into several *subcomponents*. In this case the newly formed subcomponents are removed except for, possibly, the largest of them. The largest subcomponent is not removed only in the case when its size is strictly greater than the sum of the sizes of all the other subcomponents. Removing a component means repainting all its cells to white.

Next regardless of the color of the repainted cell, «holes» in components are removed: If any white cell is located inside any cycle of black cells, it is repainted to black. Every two neighboring cells in a cycle must be adjacent in the sense defined above.

Initially all cells are white. A sequence of operations is given. Each operation is either repainting of a cell or a query. All the operations must be executed in the given order.

There can be two types of queries. A query of type **Comp**: is it true that the given two cells are black and belong to the same component? A query of type **Size**: find the size of the component that contains the given cell. If this cell is white, the answer must be zero.

Input

The first line contains a single integer N — the number of repaintings and queries ($1 \leq N \leq 10^5$). Each of the following N lines contains a description of a single operation.

The operation formats are presented below:

- “**Draw** x y ” — repaint a cell with the coordinates $(x; y)$ to the opposite color;
- “**Size** x y ” — find the size of the component containing the cell with the coordinates $(x; y)$;
- “**Comp** x_1 y_1 x_2 y_2 ” — determine whether the cells with the coordinates $(x_1; y_1)$ and $(x_2; y_2)$ belong to the same component.

All coordinates are positive integers not exceeding 10^6 .

Output

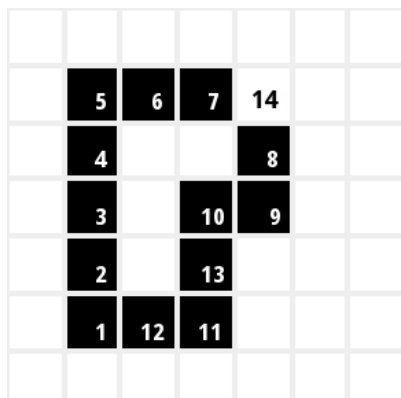
Print answers to the queries in order of their appearance in the input file, one answer per line. An answer to a query of the type **Size** must be an integer. An answer to a query of the type **Comp** must be either YES or NO.

Examples

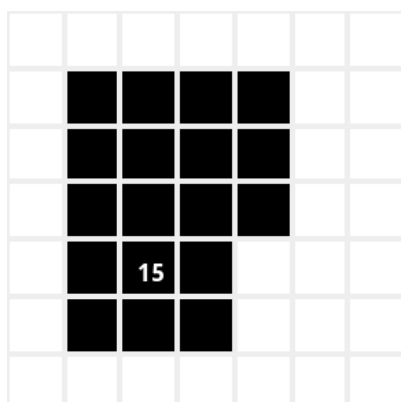
input.txt	output.txt
38	1
Draw 2 2	7
Draw 2 3	3
Draw 2 4	0
Draw 2 5	YES
Draw 2 6	NO
Draw 3 6	NO
Draw 4 6	13
Draw 5 5	18
Draw 5 4	18
Draw 4 4	YES
Draw 4 2	0
Size 4 2	9
Size 2 4	9
Size 4 4	0
Size 3 3	0
Comp 2 2 2 4	0
Comp 2 2 4 4	
Comp 3 3 4 4	
Draw 3 2	
Draw 4 3	
Size 2 2	
Draw 5 6	
Size 2 2	
Draw 3 3	
Size 3 3	
Comp 3 3 4 4	
Draw 2 4	
Draw 3 4	
Draw 4 4	
Size 3 2	
Size 3 6	
Draw 4 7	
Draw 3 5	
Size 2 5	
Draw 4 6	
Size 2 5	
Size 4 5	
Size 4 7	

Комментарий

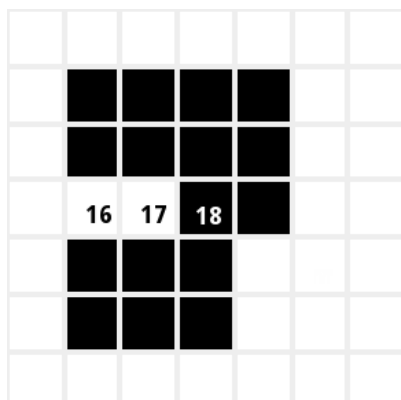
Numbers on the pictures below denote number of Brick tool usage for the corresponding number.



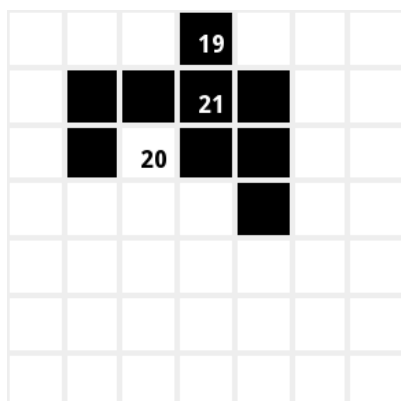
Playfield after 14'th usage of the Brick tool



15'th usage of the Brick tool does not cause changes in the playfield.



Playfield after 18'th usage of Brick tool.



After 21'th usage of Brick tool all the cells in the playfield will become white again.

Problem 7. The weasel in the hen coop

Input file: `input.txt`
Output file: `output.txt`
Time limit: 2 second
Memory limit: 256 mebibytes

During an expedition to one of the remote stars the Group of Free Search made yet another discovery, apparently linked to the Wanderer civilization. It was a device of unknown purpose, and its main control element was a $M \times N$ rectangular field; some cells of the field were removed, some other were colored in some of 26 different colors. A box was found nearby with a large number of 2×1 rectangular plates reminiscent of the terrestrial dominoes. The cell size of the plates corresponded exactly to the cell size on the control panel.

A renowned specialist in solving Wanderers' intentions, Rudolph Sikorski, supposed that the mechanism can be activated by laying out the plates on the field in the following way. None of the removed cells must be covered, only one of the cells of each particular color must be covered, non-colored and non-removed cells must be covered completely.

Leo Abalkin, a former progressor, managed to steal the box with plates. Now he must place the plates on the field correctly to activate the mechanism. Help him do that until Rudolph Sikorski discovers the loss.

Input

The first line of the input file contains two space-separated integers M and N — the sizes of the field ($1 \leq M, N \leq 100$).

Then the field itself is defined: M lines, containing N symbols each — the cell descriptions.

A removed cell is denoted by the symbol '.' (ASCII code 46). A cell of one of the 26 colors is denoted by a capital Latin letter from 'A' to 'Z', with differently colored cells corresponding to different letters and equally colored cells corresponding to the same letter. Other cells are denoted by the symbol '*' (ASCII 46).

The colored cells may appear in the input file in any combination, but it is guaranteed that the equally colored cells have same parity. Parity of the cell is a parity of the sum of its coordinates.

Also it is guaranteed that there exists at least one non-removed cell, either colored or not.

Output

The first line of the output file must contain the word "No" if there is no solution.

Otherwise the line must contain the word "Yes". The next lines must contain the tile layout: M lines containing N symbols each.

The symbol '.' (ASCII 46) denotes a removed cell.

Each plate takes up two cells. Consequently, each covered cell is covered with a half of a plate: right, left, top, or bottom.

The upper part of a plate is denoted by the symbol '^' (ASCII 94).

The lower part of a plate is denoted by the symbol 'v' (ASCII 118).

The left part of a plate is denoted by the symbol '<' (ASCII 60).

The right part of a plate is denoted by the symbol '>' (ASCII 62).

Cells which were left uncovered are marked with the symbol '@' (ASCII 64).

The notions “up”, “down”, “left” and “right” refer to the presentation of the output file on the screen and not to the field itself.

Example

input.txt	output.txt
4 8 ...***.. ..A.A*.. .B***.*. *****B.	Yes ...<>^.. ..^.@v.. .@v<>.^.. <><><>v.

Problem 8. Colonization

Input file: `input.txt`
Output file: `output.txt`
Time limit: 4 seconds
Memory limit: 512 mebibytes

The yoghurt bottle is populated with N bacteria. But bacteria want company, so they form colonies, and this happens in the following way. Initially each bacterium is a colony consisting of itself. While there are at least two colonies in the yoghurt, two nearest colonies are chosen at each step and merged together.

Initially for each pair of bacteria u and v the distance $d_{u,v}$ between them is known. The distance for the colonies G and H is calculated as the average distance between all pairs of bacteria:

$$\frac{1}{|G| \cdot |H|} \sum_{u \in G} \sum_{v \in H} d_{u,v}$$

Bacteria aren't very particular when it comes to accuracy — you cannot expect too much from single-cell organisms, anyway. So when there are several pairs of colonies with the distance within 10^{-6} of the minimal, then any of these pairs can be selected for merging at this step.

You are to model the colony merging process and find one of the possible scenarios.

Input

The first line of the input file contains an integer N — the number of bacteria ($2 \leq N \leq 2014$).

The following N lines contain the description of the distance matrix $(d_{i,j})$. Each line contains N characters. The character $d_{i,j}$ in the j -th position of the i -th line specifies the distance between the i -th and j -th bacteria.

It is guaranteed that $d_{i,i} = 0$, $d_{i,j} = d_{j,i}$, $0 \leq d_{i,j} \leq 9$ for all $i, j = 1, \dots, N$.

Output

Assign numbers to each initial colony, from 1 to N . The colony obtained on the i -th step, $i = 1, \dots, N - 1$, will have the number $N + i$.

The input file must contain $N - 1$ lines. The i -th line must contain three numbers: the numbers of colonies merged on the i -th step, and the distance between them with an absolute or relative error not greater than 10^{-6} ($1 \leq i \leq N - 1$).

Examples

<code>input.txt</code>	<code>output.txt</code>
4	1 2 1.000000
0146	3 4 3.000000
1025	5 6 4.250000
4203	
6530	

Problem 9. Triangles

Input file: `input.txt`
Output file: `output.txt`
Time limit: 2 seconds
Memory limit: 256 mebibytes

Peter is a freshman in the University. His biggest problem is analytical geometry, because Peter's mind simply resists everything that is new.

Peter is bored at analytical geometry lectures. Nevertheless he attends these lectures for some reason, and at the moment he is sitting at such a lecture. He is desperate to find something to entertain himself. Peter loves triangles, because triangles are simple and he knows them well. He has got a sheet of paper and a blue ballpen. He put n dots on the sheet, and decided that he'll connect these dots with lines and fill triangles by blue color.

At this very instant the bell rings, and everyone leaves the room for a break. Upon his return from the break, he's vexed to see that someone drew a green triangle on his paper. He is not that fond of green. He wants to draw his own blue triangle with three marked dots as vertices so that it would cover the green triangle. This way the nasty green triangle will be completely covered by blue ink, and Peter will be happy. The green triangle must be strictly within the blue one, in particular the boundaries of the two triangles must have no common points. Peter hasn't been listening to his professor, and he cannot solve the problem on his own. He is asking you for help.

Input

The first three lines of the input file contain two integers separated by spaces each — the coordinates of the green triangle vertices, given in counterclockwise order. It is guaranteed that the triangle is not degenerate.

The fourth line contains an integer n — the number of marked dots ($1 \leq n \leq 10^5$). Each of the next n lines contains two integers separated by spaces — the coordinates of the corresponding marked dot. It is guaranteed that all the n dots are pairwise distinct.

All coordinates in the input file do not exceed 10^6 by absolute value.

Output

If the desired blue triangle which strictly contains the green triangle does not exist, the output file must contain the word "NO".

Otherwise the first line must contain the word "YES", and the second line must contain the numbers of dots from the input set which are the vertices of the desired triangle. The vertices of the blue triangle must be printed in the counterclockwise order. The dots are numbered from 1 in the order of appearance in the input file. If there are several solutions, output any of them.

Examples

input.txt	output.txt
0 0 1 0 0 1 3 -1 -1 2 0 -1 2	YES 1 2 3
0 0 1 0 0 1 3 0 -2 -2 1 3 1	NO

Problem 10. Swimming

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

Trying to reach Skumbriewitz, who was floundering around the buoy, Ostap Bender swam on his side, keeping an eye on Berlaga, who was sitting on the shore. Ostap's head was turned by a constant angle for the whole time. For that reason he swam along a curve rather than along a straight line, so the angle between the direction of his swimming and the direction to Berlaga was constant.

Your task is to find the length of Ostap Bender's trajectory.

Input

The input file consists of three lines. Each of these lines contains two integers not exceeding 10^5 by absolute value. The first line contains the initial coordinates of Ostap Bender, the second line contains the coordinates of his destination – Skumbriewitz, and the third line – the coordinates of Berlaga. All defined points are pairwise distinct. It is guaranteed that Berlaga and Skumbriewitz didn't move while Ostap was swimming.

Output

The input file must contain a single real number – the length of the shortest trajectory from the initial position of Ostap to Skumbriewitz's buoy, given that the angle between the tangent to the trajectory and the direction to Berlaga was constant. Print the answer with a relative or absolute error not greater than 10^{-8} .

Example

<code>input.txt</code>	<code>output.txt</code>
10 0 -10 0 0 0	31.4159265359

Problem 11. Faster Than Light

Input file: `input.txt`
Output file: `output.txt`
Time limit: 6 seconds
Memory limit: 256 megabytes

In the **Faster Than Light** video game each spaceship can be represented on a flat grid. All cells of the grid are unit squares. Some cells represent ship sections and they can be fired at. Other sections don't belong to the ship.

There is a beam weapon in the game which shoots in the following way. When fired the weapon draws a line segment of the fixed length L with the beam over the attacked ship. The position of the segment can be chosen arbitrarily given that one of its ends is positioned inside or on the boundary of one of the sections of the attacked ship. The other end of the segment can be anywhere, including outside the grid. The damage to the ship and its crew depends on the set of sections damaged by the beam. A ship section is considered damaged if the line segment and section cell (including its boundary) have at least one common point.

You are invited to develop a targeting program which should work in the following way. For each spaceship a positive number of points for hitting each of its sections is given. Your program must find the position of the segment which yields the maximum sum of points for all damaged sections.

Input

The first line of the input file contains two integers N and M – the numbers of rows and columns in the grid, respectively ($1 \leq N, M \leq 30$).

The second line contains the length of the segment — a real number L given with two or less digits after decimal point ($0.1 \leq L \leq 50$).

The next N lines describe the grid. Each of them contains M integers. Let the j th number in the i th of these lines equal a_{ij} ($0 \leq a_{ij} \leq 10^7$, $i = 1 \dots N$, $j = 1 \dots M$). If $a_{ij} = 0$, then the corresponding cell is empty. If $a_{ij} > 0$, then the cell contains a spaceship section, which yields a_{ij} points when hit.

It is guaranteed that at least one cell containing a spaceship section is present in the grid. Different spaceship sections may be disconnected from each other.

Output

The output file must contain a single integer – the maximum possible score for a single shot of the beam.

Example

input.txt	output.txt
5 4 2.5 1 0 5 1 3 0 3 5 0 0 0 0 1 8 1 3 1 3 1 2	23

Комментарий

The sample test allows to fire the weapon in such a way that it hits two 5-point cells, 1- and 3-point cells, as well as 8- and 1-point cells.

1		5	1
3		3	5
			
1	8	1	3
1	3	1	2

Problem 12. Construction of Chand Baori

Input file: `input.txt`
Output file: `output.txt`
Time limit: 1 second
Memory limit: 256 mebibytes

The Indian stepped well Chand Baori is, with some simplifications, a pyramid tapering down. Faces of the pyramid are made up of rows of trapezoid stairs which can be used to descend to reach the water. The first (lowest) level is a single stairway with two flights of stairs on the left and on the right. The second level has two such stairways, etc.

To visit the “Harshat Mata” shrine dedicated to the goddess of happiness and joy pilgrims need to cleanse themselves in the waters of the well, that is, to descend to the very bottom. Pilgrims on their way to the water can only descend or walk horizontally along the current level, but not climb up.



The builders of the well want to construct it in such a way that the number of possible ways of descend would be no less than the number of pilgrims. Two paths are considered different if there exists a level with a different stairway used for descending, or different flight of the same stairway used.

Input

The first line of the input file contains two integers N and M – the number of levels in the well and the number of pilgrims descending along the single face, respectively ($1 \leq N \leq 20$, $0 \leq M \leq 1.5 \cdot 10^{18}$).

Output

The output file must contain “Harshat Mata”, if the number of possible ways to descend along one face of the well is less than the given number of pilgrims (also for one face), otherwise it must contain “Nope”.

Examples

<code>input.txt</code>	<code>output.txt</code>
1 2	Harshat Mata
1 3	Nope
2 9	Nope
2 8	Harshat Mata

Комментарий

For a two-level well there exist eight different ways of descent shown in the picture (parts of the same route are marked with the same digit).

