Problem A. War Academy

Input file:	`academy.in`
Output file:	`academy.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

The War Academy of Flatland has received the request from the government to prepare n special agents for the secret operation against its neighbor Edgeland. Each agent will have a special mission assigned, the i-th mission requires the *experience* of an agent that will perform this mission to be at least p_i.

The main instructor of the academy has selected n students for the program. The i-th of the selected students will be assigned to the i-th mission. Now the instructor needs to plan the training sessions.

Each student can have individual training, and all students together can have group training. Initially all students have experience equal to 0. The i-th student is characterized by his *intelligence* q_i and his *sociality* s_i. After taking individual training for m hours the experience of the i-th student increases by mq_i. After taking group training for m hours the experience of the i-th student increases by ms_i. Training can continue for any non-negative number of hours (and can be non-integer if needed).

Training is performed by the instructors. The instructor must be paid a Flatland dollars per hour of individual training and b Flatland dollars per hour of group training.

Help the main instructor of the academy to plan the training in order to fulfil the request of the government and spend as little money as possible.

Input

The first line of the input file contains three integer numbers: n ($1 \le n \le 100$), a and b ($1 \le a, b \le 10^6$). The following n lines contain three integer numbers each: p_i, q_i and s_i ($1 \le p_i, q_i, s_i \le 1000$).

Output

The first line of the output file must contain one real number: w — the total cost of preparing students for the operation, in Flatland dollars. The second line must contain one real number g — the length of group training in hours. The third line must contain n real numbers — for each student specify the length of his individual training in hours.

Output as many digits after the decimal point as possible (the more you output, the less is the chance that the verifying program will have precision errors). The verifying program will use threshold of 10^{-6} when making comparisons of real numbers.

Example

academy.in	academy.out

2 50 13
10 10 2
3 5 1
	59.0
	3.0
	0.4 0.0

Problem B. Discount

Input file:	`discount.in`
Output file:	`discount.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

The Megga Mall is running a special discount program for its customers. Each customer who buys something really big in the Mall gets n discounts for his purchase. Each discount is presented to the customer as a gift card. The front side of the contains label "$-a_i$", and the reverse side contains label "$-b_i\%$" (a_i and b_i are integer numbers, they can be different for different cards).

Let the initial price of the purchase be x. To decrease the price of the purchase the customer can use the gift cards. The cards are used one after another in some arbitrary order. Each card can be used in one of two possible ways: if the card has labels "$-a_i$ and "$-b_i\%$" it can either decrease the current price of the purchase by a_i (front use), or multiply its current price by $1 - b_i/100$ (reverse use). All operations with the price of the purchase are performed with the real number. If the price becomes negative, the purchase is free for the customer.

Peter is going to buy Ferrari in the Megga Mall. The price of Ferrari is x. Help Peter to use his cards optimally, so that the final price of Ferrari is minimal possible.

Input

The first line of the input file contains two integer numbers: n ($1 \le n \le 50$) and x ($1 \le x \le 10^9$). The following n lines describe gift cards, each line contains two integer numbers — a_i and b_i ($1 \le a_i \le 10\,000$, $1 \le b_i \le 99$).

Output

Output n lines. Lines must describe the gift cards in order they must be used. Each line must contain the number of the card followed by the word "`front`" if the card must be used with front label ("$-a_i$") or the word "`reverse`", if the card must be used with reverse label ("$-b_i\%$").

The final price calculated with your answer must be accurate within 10^{-9} of the optimal price (absolute or relative).

Example

discount.in	discount.out

3 1000
10 1
20 1
10 2
	3 reverse
	1 front
	2 front

After applying the third card, Peter gets 2% discount which is 20, so the price becomes 980. After that he uses first and second cards in any order, to get a discount of another $10 + 20 = 30$, so the price becomes 950.

Problem C. Dowry

Input file:	`dowry.in`
Output file:	`dowry.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

The daughter of the King of Flatland is going to marry the beautiful prince. The prince is going to give the generous dowry for the King's daughter, but he is unsure which jewels from his collection to give.

There are n jewels in the collection of the prince, each jewel is characterized by its weight w_i and its value v_i. Prince would like to give as valuable dowry to the King as possible. But the King is wise and he would accept the jewels only if their total weight doesn't exceed R. On the other side the prince would consider himself greedy for the rest of his life if he gave the jewels with the total weight less than L.

Help the prince to choose jewels from his collection so that their total weight was between L and R (inclusive), and the total value of the selected jewels was maximal possible.

Input

The first line of the input file contains n ($1 \le n \le 32$), L and R ($0 \le L \le R \le 10^{18}$). The following n lines describe jewels and contain two numbers each — the weight and the value of the corresponding jewel ($1 \le w_i, v_i \le 10^{15}$).

Output

The first line of the output file must contain k — the number of jewels to present to the king. The second line must contain k integer numbers — the numbers of jewels. Jewels are numbered from 1 to n in order they are given in the input file.

If it is impossible to choose the jewels, output 0 at the first line of the output file.

Example

dowry.in	dowry.out

3 6 8
3 10
7 3
8 2
	1
	2

Problem D. Minimal Cut Matrix

Input file:	`matrix.in`
Output file:	`matrix.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Consider a connected weighed undirected graph G. A *minimal edge cut* between vertices u and v is the set $C_{u,v}$ of edges with minimal total weight, such that each path from u to v contains at least one edge from $C_{u,v}$. Let us denote the weight of a minimal edge cut between vertices u and v as $c_{u,v}$.

The matrix $c_{u,v}$ is called the *minimal cut matrix* of graph G. You are given a matrix $c_{u,v}$. Find the graph G such that $c_{u,v}$ is the minimal cut matrix of this graph, or detect that there is no such graph.

Input

The first line of the input file contains n — the size of the matrix ($2 \le n \le 50$). The following n lines contain n integer numbers each — the entries of the matrix. It is guaranteed that the matrix is symmetric and has zeroes at the main diagonal. All other entries are positive and do not exceed 1000.

Output

If there exists a graph G such that the matrix in the input file is its minimal cut matrix, print "`YES`" at the first line of the output file. In this case print m — the number of edges in the graph at the second line, and describe edges in the following m lines. Each edge must be described by the numbers of vertices it connects, and the weight of the corresponding edge. There must be no loops, neither parallel edges. No weight must exceed 1000.

If there is no such graph, print "`NO`" at the first line of the output file.

Example

matrix.in	matrix.out

3
0 2 2
2 0 2
2 2 0
	YES
	3
	1 2 1
	1 3 1
	2 3 1

Problem E. Shortest Path

Input file:	`path.in`
Output file:	`path.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

The Unbalan planet has the form of a union of two spheres of equal radius of r kilometers. Unbalan is covered by a desert, so inhabitants of the planet use special sandmobiles to move from one point to another. Sandmobiles can move along any curve on the planet surface, but cannot fly away from it. It is also allowed to walk on the part of the surface that is inside any of the spheres.

Andy lives on Unbalan. One day he decided to visit his friend Geo. Help Andy to find the length of the shortest path he needs to travel on his sandmobile to get from his house to Geo's one.

Input

The first line of the input file contains two integer numbers r and d ($0 < d < 2r \leq 1000$) — the radii of spheres that form Unbalan, and the distance between their centers. Let the coordinate system be introduced in such way that the coordinates of sphere centers are $(0, 0, 0)$ and $(0, 0, d)$.

The second line contains three real numbers x_a, y_a and z_a — the coordinates of Andy's house. The third line also contains three real numbers: x_g, y_g and z_g — the coordinates of Geo's house. Each of the points is at the distance equal to r from the center of one of the shperes, and at the distance greater than r from the center of another sphere. All equalities are within 10^{-9}.

Output

Output one real number — the length of the shortest path between Andy's and Geo's houses along the surface of the planet. Your answer must be accurate up to 10^{-5}.

Example

path.in	path.out

`5 8`
`0 0 -5`
`0 0 13`
	`24.9809154479650885`

Problem F. Infinite Recursion

Input file:	`recursion.in`
Output file:	`recursion.out`
Time limit:	5 seconds
Memory limit:	256 megabytes

Sam is developing a new IDE for ZH± programming language. Now he is writing a part that is responsible for detecting infinite recursion.

ZH± is a procedure-based language. All variables in ZH± are binary — each variable can take values "0" and "1". Each procedure may have several arguments. There are following statements available in ZH±:

- increase variable value "`x++`" — if x is 0, it becomes 1;
- decrease variable value "`x--`" — if x is 1, it becomes 0;
- conditional statement "`if (x) S1 else S2`" where S1 and S2 are any statements — execute S1 if x is 1, or S2 if x is 0;
- procedure call "`f(x, y)`";
- block statement "`{S1 S2 ...}`" where S1, S2, etc are statements — execute S1, S2, etc one after another.

The procedure itself is described as "`<function name>(<argument list>) S`" where S is a statement.

All arguments are passed by their values (changes inside a procedure do not cause changes to the corresponding variables in the calling procedure).

For example, a program in the first example has procedure "`inc`" that receives a the 3-bit value represented by arguments `x2`, `x1` and `x0` and eventually calls procedure "`done`" with three arguments that represent this value increased by one.

Given a program, find out whether there exists a call to some function with some arguments that leads to infinite recursion.

Input

Input file contains a program in ZH± language. It has at most 20 procedures. Each procedure has at most 15 parameters. The total length of the input file is at most 1000 bytes. The names of procedures and parameters consist of letters and digits and are case sensitive. No name is longer than 40 characters. No procedure nor parameter is named "`if`" nor "`else`".

Output

Print "`YES`" at the first line of the output file if it is possible to make a procedure call that would lead to infinite recursion. In the other case print "`NO`".

Example

recursion.in	recursion.out

`inc(x2, x1, x0) {`
`  if (x0) {`
`    x0--`
`    inc1(x2, x1, x0)`
`  } else {`
`    x0++`
`    done(x2, x1, x0)`
`  }`
`}`
 	
`inc1(x2, x1, x0) {`
`  if (x1) {`
`    x1--`
`    inc2(x2, x1, x0)`
`  } else {`
`    x1++`
`    done(x2, x1, x0)`
`  }`
`}`
 	
`inc2(x2, x1, x0) {`
`  if (x2) {`
`    x2--`
`    done(x2, x1, x0)`
`  } else {`
`    x2++`
`    done(x2, x1, x0)`
`   }`
`}`
 	
`done(x2, x1, x0) {`
`}`
	`NO`

`a(x0) {`
`  if (x0) {`
`  } else {`
`    a(x0)`
`  }`
`}`
	`YES`

Problem G. Yet Another Rooks Problem

Input file:	`rooks.in`
Output file:	`rooks.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

You have k rooks and an $m \times n$ board. The placement of rooks on the board is called *correct* if no rook is between two other rooks, horizontally or vertically.

Given m, n and k find the number of correct placements of rooks on the board. Since this number may be quite large, find it modulo 10003.

Input

Input file contains m, n and k ($1 \le m, n \le 50$, $1 \le k \le mn$).

Output

Output one number — the number of correct placements of k rooks on the $m \times n$ board modulo 10003.

Example

`rooks.in`	`rooks.out`
`3 2 3`	`18`

Problem H. School of Magic

Input file:	`school.in`
Output file:	`school.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Harry Nomoretter is studying in School of Magic. Today he is planning to start preparing for his first exam. There are four skills that will be tested on the exam: alchemy, basic magic, chronology of magic and demonology.

Harry's experience in each skill can be characterized by an integer number, ranging from 0 to a for alchemy, from 0 to b for basic magic, from 0 to c for chronology of magic, and from 0 to d for demonology. To pass the exam Harry must have each of his skills maximized, that is, his alchemy skill must be equal to a, basic magic skill equal to b, chronology of magic skill equal to c and demonology skill equal to d. However Harry used not to visit lectures, so now his skills are all equal to 0.

To increase his skills Harry can read textbooks. He's got four textbooks: "Applied Alchemy", "Basics of Basic Magic", "Can Chronology Be As Easy?" and "Demonology for Dummies". Reading a book for an hour increases his alchemy, basic magic, chronology of magic, or demonology skill, respectively, by one.

However, when reading books Harry abstracts from other topics, so his other skills can also change — he can forget something he has read, so the skill decreases, or he can recall something he discussed with his friends before, so the skill increases. Formally, if Harry reads "Applied Alchemy" for one hour, his alchemy skill increases by one, and each of his basic magic, chronology of magic and demonology skills independently uniformly randomly either increases by one, or doesn't change, or decreases by one, with the exception that the skill never decreases below 0, and never increases above its maximum (should the corresponding skill be 0 or maximal, respectively, the uniform choice is made from two variants). Similarly, "Basics of Basic Magic" concerns Harry's basic magic skill, etc.

Now Harry wonders what is the expected time he needs to reach the maximum in each of the skills if he acts optimally. Help him to find that out.

Input

Input file contains four integer numbers: a, b, c and d ($1 \le a, b, c, d \le 4$).

Output

Output one real number — the expected number of hours Harry needs. Your answer must be accurate up to 10^{-6}.

Example

`school.in`	`school.out`
`1 1 1 1`	`8.0`

Problem I. Shepherd's Problem

Input file:	`shepherd.in`
Output file:	`shepherd.out`
Time limit:	7 seconds
Memory limit:	256 megabytes

Anthony is a shepherd. He has n sheep that usually graze calmly on the pasture. While the sheep are grazing, Anthony is sitting on a hill, playing the pipe, and enjoys life.

However, sometimes sheep might be in danger. So Anthony always watches thoroughly that all his sheep are safe. In order to notice the problem and deal with it swiftly, he must keep an eye on each of his sheep. Therefore he would like the angle of view to his sheep be minimal possible. Also the view angle must not exceed 180 degrees because Anthony has got no eyes on the back of his head.

The angle of view to the sheep is the smallest angle such that all sheep are within it.

But Anthony's eyesight is already quite poor, so no sheep must be at a distance greater than D from Anthony, in other case he would not be able to see whether it is safe. On the other side Anthony doesn't want his sheep to worry, so the distance to the closest sheep must be at least d.

Given the locations of the sheep and the distances D and d, find the location for Anthony so that the view angle to the sheep is minimal possible (and doesn't exceed 180), the distance to the furthest sheep is at most D, and the distance to the closest sheep is at least d.

Input

The first line of the input file contains three integer numbers: n, d and D ($3 \le n \le 10$, $1 \le d \le D \le 10^3$).

The following n lines contain two integer numbers each — the coordinates of the sheep. Coordinates do not exceed 10^3 by their absolute values. There are three sheep not on the same line.

Output

Output two real numbers — the coordinates of the point where Anthony should sit in order to minimize his view angle. If there are several solutions, output any one. If there is no solution, output "`impossible`" (without quotes).

Verifying program will use precision of 10^{-5} when making comparisons of floating point numbers.

Example

shepherd.in	shepherd.out

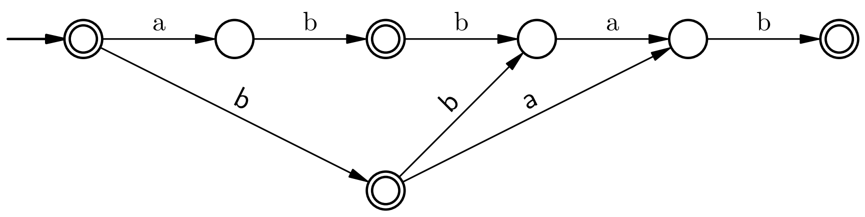
4 1 5
0 0
1 0
0 1
1 1
	4.9749371855331 0.5

Problem J. Suffix Automaton

Input file:	`suffix.in`
Output file:	`suffix.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Suffix automaton for the word w is the deterministic finite automaton A that accepts the language $Suff(w)$ — the set of suffices of w. For example, the suffix automaton for the word *abbab* must accept exactly the following words: $\{abbab, bbab, bab, ab, b, \varepsilon\}$. We will also require that suffix automaton had no unreachable states and no states such that no terminal state was reachable from it. We will not require other constraints such as minimality.

The suffix automaton for the word *abbab* is shown on the figure below.



You are given the *skeleton* of the suffix automaton of some word. That is — you are given the states, the transitions, the starting state and the terminal states. But the labels at the transitions are deleted.

You have to put labels at the transitions of the given automaton skeleton, so that it became the suffix automaton of some word w, and restore this word as well. For simplicity we will assume that the alphabet can have any size, that is, you can use positive integer numbers from 1 to k (k to be chosen by you) as characters for the word w.

Input

The first line of the input file contains three integer numbers: n, m and t — the number of states, the number of transitions, and the number of terminal states, respectively ($2 \le n \le 200$, $1 \le m \le 1000$, $1 \le t \le n$). The second line contains t integer numbers — the numbers of terminal states. States are numbered from 1 to n. The starting state is the state number 1.

The following m lines describe transitions: each line contains two integer numbers s_i and t_i and describes the transition from s_i to t_i.

Output

The first line of the output file must contain two integer numbers: l and k — the length of the word w and the size of the alphabet. Use numbers $\{1, \ldots, k\}$ as elements of the alphabet. k must not exceed m.

The second line of the output file must contain l integer numbers — the word w.

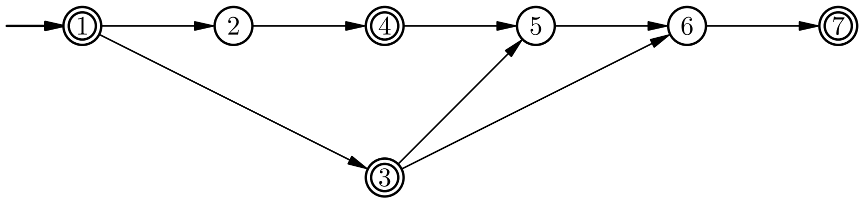
Finally, the third line must contain m integer numbers — labels of the transitions of the skeleton in the input file that turn it to the suffix automaton for w.

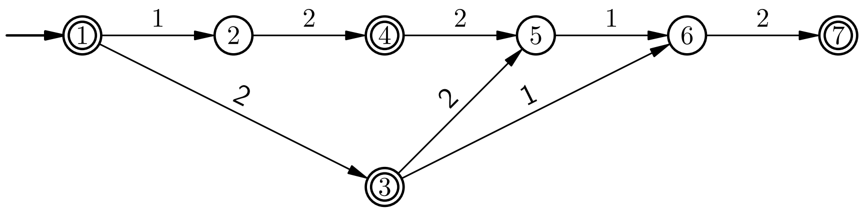
It is guaranteed that the answer always exists.

Example

suffix.in	suffix.out

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




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