We can all benefit by doing occasional "toy" programs, when artificial restrictions are set up, so that we are forced to push our abilities to the limit. … The art of tackling miniproblems with all our energy will sharpen our talents for the real problems.
Donald E. Knuth

Quarterfinal

Central region of Russia

Rybinsk, October 15-16-2014

Input file name INPUT.TXT
Output file name: OUTPUT.TXT

A. "Fish" (64 Mb, 1 sec / test)

We all know the old game Dominoes. Let us briefly remind the rules of this game. The traditional domino set consists of 28 tiles. Each domino is a rectangular tile with a length twice longer then a width. Its front side (face) is divided by lines into two square ends. Each end has from zero to six points. The number of tiles in a domino equal to the number of combinations with repetitions of size 2 from the set of 7 digits {0, 1, 2, ..., 6}, and is calculated by the formula: $(n+1)\times(n+2)/2$, where n - the maximum number of points (spots). For example, the traditional set includes $(6+1)\times(6+2)/2 = 28$ tiles.

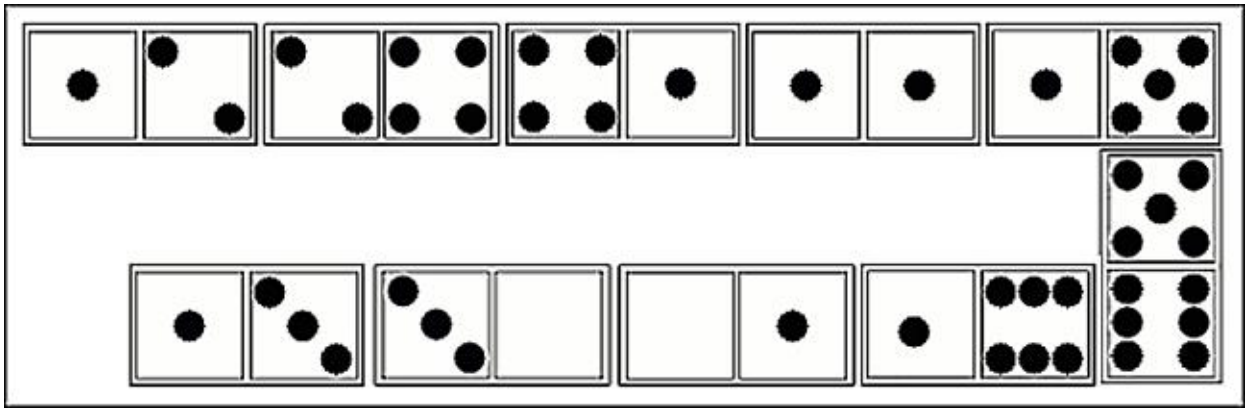
Usually each player takes seven tiles. The rest of the dominoes are placed face down in reserve ("stock"). During the game, the dominoes laid out in a line. After the first player started the game with a tile, the following players add matching tiles with the number of points equal to the number of points on one end of the line. If there are no appropriate tiles, then you have to pick them from the stock.

The game ends when one player wins by playing his last dominoes, or when the game is blocked - so-called "fish" - when in the hands of the players still have tiles, but it is impossible to make a move.

Write a program, which by the given n tiles in a line and the number of k and m tiles in the hands of each two players will determine whether there is blocking game that is called "fish".

For example, let the line on the table consists of the 10 tiles and the first player has on his hands 2 dominoes and the second one 3 dominoes. The rest of the dominoes are in the stock. In this case the game is blocked ("fish"), because all dominoes with one spot are used and, therefore, none of the players can make a move.

Note. Game situation is not called "fish" when at the hands of one player there are no dominoes.



Input

The first line of the input file contains integers: n– the number of tiles in the line, k – the number of first player's tiles, m – the number of second player's tiles.

The other n lines contain $2n$ integers – the number of spots in n tiles laid out in a line.

Limitations

$n > 0$, $k >= 0$, $m >= 0$, $n+k+m \le 28$;

$0 \le p_1, p_2, p_3, p_4, \ldots, p_{2n-1}, p_{2n} \le 6$; $p_{2*i} = p_{2*i+1}$ where $1 \le i < n$.

Output

The output file should contain a single word (without quotation marks): "YES", if the game is blocked ("fish" situation occurs) or "NO" otherwise.

Example

Input.txt	Output.txt
10 2 3 1 2 2 4 4 1 1 1 1 5 5 6 6 1 1 0 0 3 3 1	YES
1 6 7 2 2	NO

B. Image Processing (64 Mb, 1 sec / test)

One of the most peculiar filtering methods applied in image processing is the so-called morphological filtering. One of the basic operations in this group of transformations is dilatation.

Let us assume we have a rectangular image sized n x m pixels, and each pixel of the image has a non-negative integer number weight. Therefore, the weight of each pixel in the new filtered image will be the maximum among the weights of adjacent pixels in the source image. If we denote the source image as matrix A and the output, as matrix B, the weights will be calculated according to the formula:

$$B[x, y] = \max (A[x, y], A[x + 1, y], A[x, y + 1], A[x – 1, y], A[x, y – 1]);$$

At the same time, the weights of the pixels beyond the edges of the image are equaled to zero.

Your task is to write a program that will calculate dilatation B of source image A.

Limitations

$0 < m, n < 100$; $0 <= A[x, y] <= 256$.

Input

The first line contains positive integers n and m.

The next n lines each contain m non-negative integer numbers $A[x, y]$.

Output

n lines containing m numbers each (processed image).

Example

Input	Output
5 6 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 3 0 0 0 0 3 3 5 0 0 0 0 5 5 5 0 0 0 0 5 0 0 0 0 0 0 0 0

C. The Game (64 Mb, 3 sec / test)

Petya and Vova often play board games. The boys are quite inventive, so each time they come up with new rules. They are especially fond of the game that they made up the last time.

Petya has a gameboard containing ***n*** fields marked with numbers 1 to ***n***. Some fields on the board are connected by arrows, and there is at least one arrow starting in each field. At the beginning of the game, Petya places his counter on one of the fields. Vova has an exact copy of the gameboard, but he cannot see Petya's board and he doesn't know where the counter is.

After each move, Vova calls out a field number – ***k***. If Petya's counter is in field ***k***, Petya says "hit" and the game is over. If the counter is in another field, Petya says "miss" and moves the counter to another field, following one of the arrows.

Your task is to write a program that will help Vova choose the sequence of moves to finish the game as fast as possible based on a copy of the gameboard, without seeing where the counter is.

Limitations

$1 < n \le 20$; $0 \le m \le (n-1)*n$.

Input

The first line in the input file contains two integer numbers ***n*** and ***m*** separated by spaces, where ***n*** is the number of fields on the board and ***m*** is the number of arrows. The next ***m*** lines describe the arrows. Each line contains two different numbers separated by a space – numbers of the starting and the ending field.

Output

The first line in the output file contains a integer number ***k*** that equals to the number of moves, or the message "No solution" if there is no solution. If a solution exists, the second line contains ***k*** integer numbers separated by spaces – the game plan for Vova. The first number is the first move, the second number is the second move, etc.

Example

Input	Output
3 3 1 2 2 3 3 1	3 1 1 1

D. Teletype (64 Mb, 1 sec / test)

In an organization, a teletype broke down. When it prints a message, some letters are occasionally missing. There is no apparent pattern in how symbols are skipped – any one of the letters can be possibly omitted when a message is received. For instance, when sending a line that reads "abracadabra," the teletype might as well receive "abraaabra" or "abacadaba." The machine could not be fixed quickly; therefore, to make it work somehow, it was decided to have all the messages sent twice and then try to restore the initial text. For restoring purposes, let us assume that none of the letters can be skipped both times.

Your task is to write a program that will attempt to select the shortest option among the possible versions of the initial text based on two printed messages with missing letters. If there are several possible options for the initial message, any one of the shortest options can be regarded as the correct solution.

Limitations

Messages contain lower-case letters of the Latin alphabet only.

The length of messages received does not exceed 1 000 characters.

The length of any restored message is less than or equal to the sum of the lengths of the messages received.

Each letter of the initial message is always present at least in one of the messages received.

Input.txt

The input file contains two lines, each one of them is a received variation of the initial message.

Output.txt

The output file contains a single line with a restored version of the initial message.

Examples

Input	Output
abraaabra abacadaba	abracadabra

E. Curious Visitor (64 Mb, 1 sec / test)

Software engineer Vovochka graduated from college and got a job. Now he travels to other cities a lot to deploy software.

Lately, he has been traveling from city ***s*** to city ***t*** very often. During the trip, he has to make several stop-overs in other cities. Vovochka is a curious person, so it's boring for him to follow exactly the same route every time. Therefore, when he buys tickets, he makes sure his new route is different from the previous trips.

Unfortunately, the accountants know the price of the cheapest route from ***s*** to ***t***, and they are unwilling to compensate the extras. However, they don't mind if Vovochka chooses a different route as long as it costs the same.

Your task is to write a program that will count how many times Vovochka can travel from city ***s*** to city ***t*** using a different route each time based on a matrix of travel costs between ***k*** cities. Two routes are considered different if at least one city on the route is not the same as on the other route. The answer should be printed in module 10^9.

Limitations

$1 < k \le 1000$; $1 \le s, t \le k$; $s \ne t$;

$0 \le C[i, j] \le 10^9$; $C[i, i] = 0$ where $C[i, j]$ is the travel costs from city i to city j;

$C[i, j]$ can be not equal to $C[j, i]$ for $i \ne j$.

Input

The first line in the input file contains three integer numbers ***k***, ***s***, ***t*** separated by spaces. The next ***k*** lines each contain ***k*** integer numbers $C[i, j]$ separated by spaces. Line number $i + 1$ contains the cost of travelling from city ***i*** to cities 1, 2, 3, …, ***k***. The cities are numbered from 1 to ***k***.

Output

The output line must contain a single integer number, equal to the number of alternative shortest routes from ***s*** to ***t*** in module 10^9.

Example

Input	Output
4 2 3 0 1 2 1 1 0 3 2 2 3 0 1 1 2 1 0	4

F. Land Division (64 Mb, 1 sec / test)

After another victorious war, King Skinflint has occupied an area that can be circumscribed by irregular ***n***-sided polygon ***M*** without self-intersections. Skinflint decided to divide the captured land into convex districts. To improve its defensive capacity, each district must be separated from the adjacent ones by stone walls. The vertices of each district always match the vertices of the occupied territory.
The walls can be described by a set of diagonals of polygon M without shared inner points.

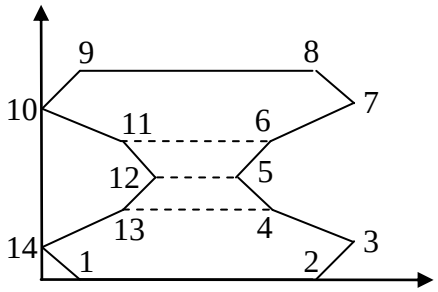
Building each wall costs exactly one chest of gold. The captured area must be divided into districts in a way that would minimize the total cost of building the walls.
Help King Skinflint optimize the land division layout.

Limitations

$3 \le n \le 200$. All the coordinates are integer numbers less than or equal to 32,000 in absolute value.

Input

The first line contains ***n*** – the number of vertices in ***n***-sided polygon ***M***. The next ***n*** lines contain coordinates of vertices of ***n***-sided polygon ***M*** in the clockwise order. It is known that no three vertices are collinear.



Output

The first line must contain one number ***k*** – the minimum number of walls that must be built to divide the captured area into convex districts.
The next ***k*** lines must contain pairs of vertex numbers of ***n***-sided polygon ***M*** connected by the corresponding wall.
If there are several solutions for the determined ***k***, any one of them can be displayed.

Example

Input	Output
14 1 0 9 0 10 1 8 2 7 3 8 4 10 5 9 6 1 6 0 5 2 4 3 3 2 2 0 1	3 4 13 5 12 6 11

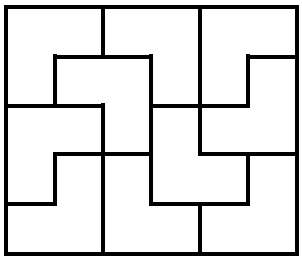
G. Strong Pavement (64 Mb, 1 sec / test)

A factory in Burlakovsk town producing paving slabs started to manufacture L-shaped items made up of three identical squares (see figure) Such slabs are excellent for plane surfaces. Thanks to the flat shape, slabs interlock tightly with each other, which helps prevent cracks in pavements of sidewalks and squares. These new slabs became popular very quickly, and soon all the streets in Burlakovsk were paved with the factory's products. The town's new mayor decided to add a second layer of slabs in the most important sections of streets. According to the plan, none of the slabs in the second layer could be positioned in a way that would match the pattern of the first layer. This would help avoid dents on the surface.

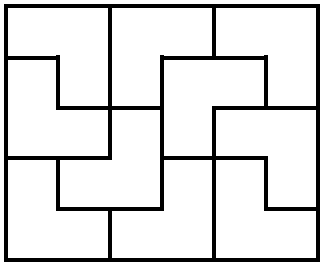
Your task is to write a program that will design the pattern for the second layer based on the information about the positions of slabs in the first layer of a rectangle sized $m \times n$. If there are several solutions, the program can output any one of them. If there is no solution, the program must output the message "`NO SOLUTION`" (without quotation marks).

Example:

The first layer



The second layer



Limitations

$2 \le m, n \le 1\,000$; either one of the two numbers is even.

Input

The first line contains positive integers m and n.

The next m lines each contain n numbers corresponding to the index of the slab covering the relevant square.

Output

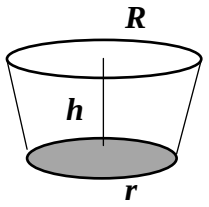
The output is m lines with n numbers in each, specifying the index of the slab covering the relevant square.

Example

Input	Output
5 6 1 1 2 2 3 3 1 4 4 2 3 5 6 6 4 7 5 5 6 8 9 7 7 10 8 8 9 9 10 10	1 1 2 2 3 3 4 1 2 5 5 3 4 4 6 5 7 7 8 6 6 9 10 7 8 8 9 9 10 10

H. Pots for Plants (64 Mb, 3 sec / test)

In every home, there are plenty of useless flower pots. Vasya's house is no exception. Right now, all of his ***N*** pots are lined up against the wall in the non-descending order based on their height. Vasya wants to pile all the pots up neatly and stow them away in a cabinet. However, since the height of the cabinet is limited, Vasya decides to throw some of the pots away. All Vasya's pots are hollow upside-down frustums (truncated cones) with bottom radius ***r***, top radius ***R***, and height ***h*** (***r*** <= ***R***).



The piling process is the following: Vasya takes each pot starting with the first one and decides whether to add it to the pile or throw it away. If Vasya chooses to add it to the pile, he sets it bottom down so that its center matches the pile's axial line. The final height of the pile shall not exceed the height of the cabinet.

Your task is to help Vasya calculate the maximum number of pots in the pile that can fit into the cabinet.

Limitation

1 <= ***N*** <= 500, 1 <= ***H*** <= 10^6
1 <= ***h***, ***r***, ***R*** <= 10^6

Input

The first line specifies the numbers ***N*** and ***H***, the following lines specify 3 integer numbers ***h***, ***r***, ***R***, which are the dimensions of pot ***i***.

Output

Output the maximum number of pots that Vasya can pile up and stow away in the cabinet.

Example

Input	Output
2 3 1 2 3 2 1 2	2

I. Vedic Square (64 Mb, 1 sec / test)

It is well known, that the digital root of a number is the sum of the digits in that number. If the resulting sum consists of more than one digit, the digits must be summed up again, and so on until the sum is a single-digit number. For example, let us calculate the digital root of 123454:

123454 => 1 + 2 + 3 + 4 + 5 + 4 = 19

19 => 1 + 9 = 10

10 => 1 + 0 = **1**.

Thus, the digital root of 123454 is 1: DigitRoot(123454) = 1.

Let us take an ascending sequence of 9 positive integers a_1, a_2, …, a_9 less than or equal to 1,000. The Vedic square for these integers will be table T sized 9 x 9 cells, and cell T[i, j] contains the digital root of the product of $a_i \times a_j$:

$$T[i, j] = \text{DigitRoot}(a_i \times a_j).$$

The task is to write a program that will attempt to restore numbers a_1, a_2, … a_9 based on random table T so that $T[i, j] = \text{DigitRoot}(a_i \times a_j)$ or report that there is no solution. If there are several solutions, any one of them can be regarded as the correct one.

Limitations

$a_i < a_{i+1}$ for $i = 1..8$; $1 \le a_i \le 1000$; $1 \le T[i, j] \le 9$.

Input

The input file consists of 9 lines, each one containing 9 digits 1 to 9 separated by spaces. Line i contains digital roots of the products $a_i \times a_1$, $a_i \times a_2$, …, $a_i \times a_9$.

Ouput

The output file must contain an ascending sequence of 9 positive integers a_1, a_2, …, a_9 or the message "`No solution`" (without quotation marks), if there is no such sequence.

Example

Input.txt	Output.txt
1 2 3 4 5 6 7 8 9 2 4 6 8 1 3 5 7 9 3 6 9 3 6 9 3 6 9 4 8 3 7 2 6 1 5 9 5 1 6 2 7 3 8 4 9 6 3 9 6 3 9 6 3 9 7 5 3 1 8 6 4 2 9 8 7 6 5 4 3 2 1 9 9 9 9 9 9 9 9 9 9	1 2 3 4 5 6 7 8 9

J. Recurrence (64 Mb, 1 sec / test)

Recurrent equations can be found in many problems, and they have a number of peculiar properties.

Let us consider the following recurrence as an example: $x_{i+1} = a/x_i + b$ where a and b are integer numbers. It turns out that if an infinite sequence $x_0, x_1, x_2, x_3, x_4, \ldots$ based on this formula consists of integer numbers only, it will always be a periodic sequence.

For example: let $a = -10800$, $b = 180$, $x_0 = 30$. In this case we have a periodic sequence: 30, −180, 240, 135, 100, 72, 30, … with the period length of 6.

Your task is to write a program that will calculate the maximum possible period of a integer-number sequence based on the recurrence $x_{i+1} = a/x_i + b$, given integer numbers a and b.

Limitations

$-2^{30} \le a, b \le 2^{30}$.

Input

The input file contains two integer numbers a and b separated by a space.

Output

The first line must contain either a integer number k representing the maximum length of the period or 0 if the given recurrence does not produce any integer-number sequence.

If a integer-number sequence does exist, print a integer number x – any member of any integer-number sequence with the period k – in the second line.

Example

Input	Output
-10800 180	6 30

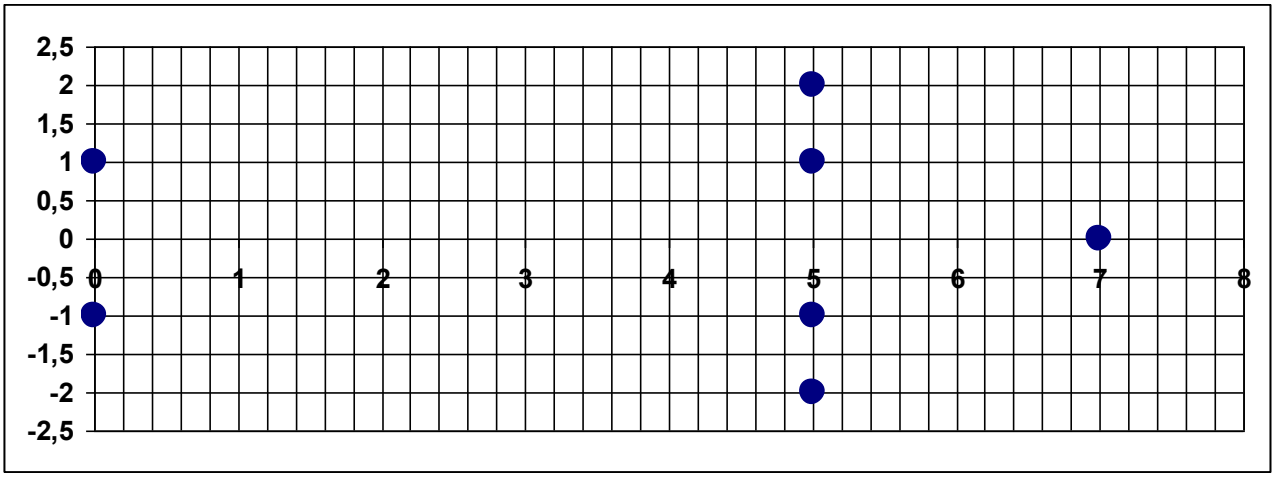
K. Arrow (64 Mb, 1 sec / test)

Seven points with integer-number coordinates are plotted on a plane.

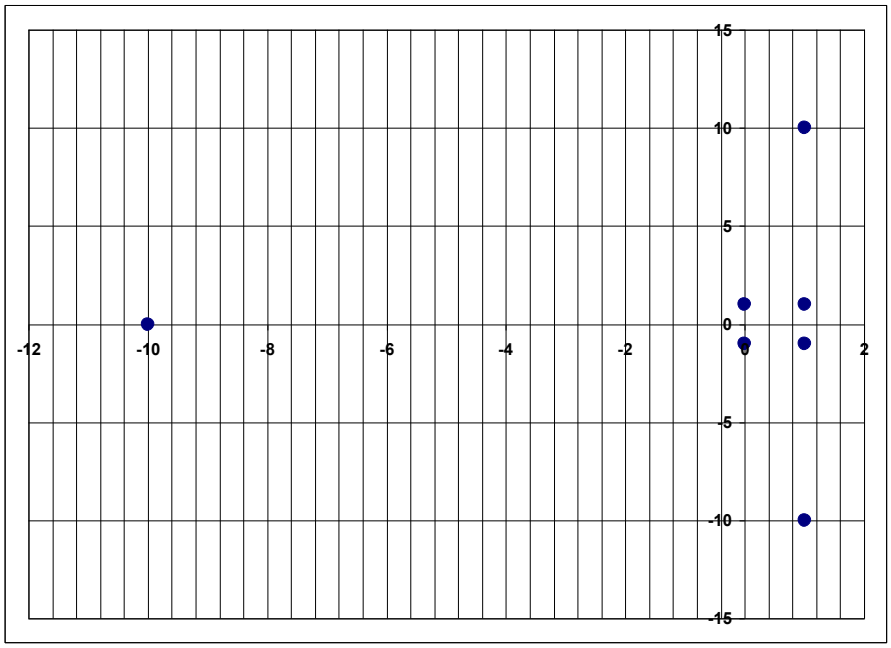
Let us assume that the seven given points form an arrow if the following conditions are satisfied:

- The angles formed by the given points can produce a rectangle and a triangle, so that one of the rectangle's sides lies along the triangle's side, and the rectangle and the triangle don't have any more common points.
- Each of the given points is a vertex – either of the rectangle or the triangle.
- The base of the triangle is always longer than the rectangle's side that the triangle is adjacent to; the other two sides of the triangle have equal lengths.
- The figure has an axis of symmetry that passes through the tip of the arrow (the vertex not lying on the extension of the rectangle's side) and the center of the rectangle's opposite side.

Example 1. Arrow



Example 2. No arrow



Your task is to write a program that will check whether it is possible to connect the points with line segments to produce a circumference of an arrow-shaped figure. If it is possible, display the numbers of the circumscribing points in the clockwise order, starting with the arrow's tip.

Limitations

$|x_i, y_i| \le 100$; $1 \le i \le 7$

Input

The input file consists of seven lines, each one containing a pair of integer numbers x_i and y_i – the coordinates of the relevant point.

Output

Print "`Yes`" (without quotation marks) in the first line if an arrow can be plotted based on the given points. In the second line, print the numbers (separated by spaces) of the circumscribing points from the input file in the clockwise order, starting with the arrow's tip.

The number of the point corresponding to the tip of the arrow shall not be repeated at the end of the sequence.

If it is impossible to plot an arrow based on the given points, print "`No`" (without quotation marks).

Example

Input	Output
5 -1 0 -1 5 2 0 1 5 -2 7 0 5 1	Yes 6 5 1 2 4 7 3
1 -1 0 -1 1 10 0 1 -10 0 1 -10 1 1	No

Rybinsk State Aviation Technical University

Task authors:
© Sergey G. Volchenkov, 2015 (volchenkov@yandex.ru)
© Vladimir N. Pinaev, 2015 (vpinaev@mail.ru)
© Michael Y. Kopachev, 2015 (rybkmu@mail.ru)
© Oleg Strekalovsky, 2015 (o.strekalovsky@yandex.ru)
© Andrey Mirzoyan, 2015 (amspectre@gmail.com)
© Sergei Dobrikov, 2015