

Problem 1: Map Check

Great County Comprehensive Internet Services (GCCIS), a leading local provider of information technology, is planning a new network. Each server will be connected to a certain number of clients which will be located exactly north, south, east, or west of their server. Lines will connect the server to one or more clients located in the same direction away from the server, but may not cross each other or pass through another server or client served by another server; a server may have lines extending into any number of the four directions.

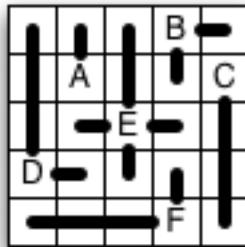
The administration of GCCIS does not believe that its employees are apt enough to produce a correct network map and have turned to you for help. You have been asked to write a program which can validate a map proposal.

Input

see file ~/1.txt

Input to your program is a map specification immediately followed by a map proposal consisting of lines with positive integers, single-letter server names, and periods, separated by white space. The input below corresponds to the map shown next to it:

```
6 5 5
A 2 2 1
B 1 4 2
C 2 5 3
D 4 1 4
E 3 3 5
F 5 4 4
D A E . B
D . E B .
D E . E C
. D E F C
F F F . C
```



The first line contains s , the number of servers (≤ 52), followed by r and c , the number of rows and columns in the map grid (each ≤ 25). Rows and columns will be numbered top-down and left-to-right, beginning with one.

Each of the next s lines contains a server name, the row and column number where the server is located in the grid, and the number of clients which the server should be connected to.

The next r lines contain c words each, which form the map proposal. Each word is either a server name to represent a client connected to that server, or a period to represent a server in the location specified earlier, or a minus sign if the location contains neither a server nor a client.

Output

Output from your program should be one line with one of the words `yes` or `no`, depending on whether the map proposal satisfies the specification or not. For the example:

`yes`

Problem 2: Map Revisited

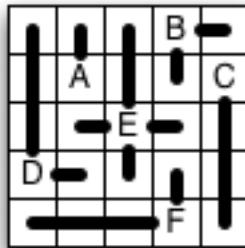
The fears of the administration of Great County Comprehensive Internet Services (GCCIS) have been confirmed: its employees are unable to produce correct network maps as described in problem 1. The vice president, Wiley Coyote, was so impressed by your earlier work for GCCIS that he is now offering a day of unlimited junk food if you can deliver a program which produces a map proposal from a map specification.

Input

see file ~/2.txt

Input to your program is a map specification consisting of lines with positive integers, single-letter server names, and periods, separated by white space. The input below could produce the map shown next to it:

```
6 5 5
A 2 2 1
B 1 4 2
C 2 5 3
D 4 1 4
E 3 3 5
F 5 4 4
```



As in problem 1, the first line contains s (≤ 52), the number of servers, followed by r and c , the number of rows and columns in the map grid (each ≤ 25). Rows and columns will be numbered top-down and left-to-right, beginning with one.

Each of the next s lines contains a server name, the row and column number where the server is located in the grid, and the number of clients which the server should be connected to.

Output

Output from your program should be a map proposal consisting of r lines containing c words each, separated by single blanks. Each word is either a server name to represent a client connected to that server, or a period to represent a server in the location given by the map specification, or a minus sign if the location contains neither a server nor a client. For the input above:

```
D A E . B
D . E B .
D E . E C
. D E F C
F F F . C
```

The input will always be a valid specification; therefore, the output should always be a valid proposal as described in problem 1.

Note that if you concatenate an input and the corresponding output for this problem you get an input for problem 1 which should produce the output *yes*.

Problem 3: Falling Factorial Form

The falling factorial function $x^{[n]}$ is defined as follows.

$$x^{[n]} = x(x-1) \dots (x-n+1)$$

For example:

$$x^{[0]} = 1$$

$$x^{[1]} = x$$

$$x^{[2]} = x(x-1) = x^2 - x$$

$$x^{[3]} = x(x-1)(x-2) = x^3 - 3x^2 + 2x$$

The falling factorial function suggests a new way to write polynomials, namely as integer combinations of falling factorial functions. For example, here are some polynomials and their falling factorial forms:

$$x^3 - 3x^2 + 2x = x^{[3]}$$

$$3x^2 - 3x = 3x^{[2]}$$

$$x = x^{[1]}$$

$$x^3 = x^{[3]} + 3x^{[2]} + x^{[1]}$$

The first three examples are just the falling factorial functions defined above, the last example results by adding the first three.

You are expected to write a program which reads the coefficients of polynomials and outputs the coefficients of their falling factorial forms.

Input

see file ~/3.txt

Input to your program consists of lines with ≤ 25 integers, separated by white space:

```
0 2 -3 1
0 -3 3
0 1
0 0 0 1
7 0 5 1
```

Each line defines the coefficients of one polynomial, starting with the *constant* coefficient.

Output

For each input line there should be one output line with the coefficients of the falling factorial form, starting with the coefficient of $x^{[0]}$. For the input above:

```
0 0 0 1
0 0 3
0 1
0 1 3 1
7 6 8 1
```

Problem 4: Graffiti

While on a walk our robotic friend Dezider finds a rather peculiar drawing. The drawing consists of numbers in a 3x3 grid. Each cell in the grid appears to contain a single digit:

4	7	8
5	1	4
8	2	5

Upon closer examination Dezider discovers that some of the cells originally contained 2-digit numbers. Apparently someone has painted over some of the digits in the drawing, preserving just one of the original digits in each cell. A passerby mentions to Dezider that originally the sum of the numbers in each row and each column was exactly 100 and that there had not been leading zeros.

Dezider would like to restore the drawing to its original form but cannot determine the values of the missing digits. He has asked you to write a program to determine them.

Input

see file ~/4.txt

The input contains three lines, each line specifies one row of the grid. Each line contains three single digits, separated by white space. For example:

```
4 7 8
5 1 4
8 2 5
```

Output

If it is possible to find the missing digits, the output consists of the three lines of a restored grid (which need not be unique). Each line contains three one- or two-digit numbers, separated by single spaces. For the example above:

```
14 78 8
58 1 41
28 21 51
```

If no solution exists, the program should output one line with the text `No solution.` For example, for the input

```
1 5 7
2 4 5
1 7 0
```

the output would be

```
No solution.
```

Problem 5: Rational Decimals

The decimal expansion of a non-negative rational number x will end in zeros (or in nines) to infinity if, and only if, the denominator of x is of the form $2^n 5^m$, where m and n are non-negative integers. Otherwise, x has a decimal expansion which eventually gets into a loop, endlessly repeating a sequence of one or more digits:

$$1/3 = 0.33333\dots$$

$$1/7 = 0.142857142857\dots$$

$$1318/185 = 7.1243243243\dots$$

You should write a program which reads non-negative rational numbers and outputs their decimal expansions.

Input

see file `~/5.txt`

Input to your program consists of lines with two positive integers (each $< 2^{31}$), separated by white space:

```
1 3
1000 7
1318 185
123 10
```

Each line contains the numerator followed by the denominator of a non-negative rational number. The denominator will not be zero and the number of repeating digits will not exceed 100 digits.

Output

For each input line there should be one output line with two numbers, separated by one blank, describing the decimal expansion of the rational number. For the input above:

```
0.3
142.857142
7.1243
12.30
```

The first number must contain or end with a decimal point and must be the prefix of the expansion with all those digits (at least up to the decimal point) which are not repeated.

The second number must consist of one or more digits, must be the recurring part of the expansion, must be as short as possible, and must not be just a single nine.

Problem 6: The Tile Chase

Žofka and Filip set up a crawling game in the kitchen. The kitchen is tiled with square tiles forming a perfect grid pattern. The children have placed raisins on some of the tiles. The game is played by crawling from tile to tile collecting the raisins on some tiles that are visited. The children start crawling from the northwest corner and work their way to the southeast corner. The objective of the game is to collect as many raisins as possible. The catch is, they have to crawl and collect raisins according to the following rules:

- start in the northwest corner, end in the southeast corner,
- only the following two moves are allowed: move south (not southeast or southwest) to the neighboring tile, or move east (not northeast or southeast) to the neighboring tile, and
- collect raisins from a tile only if you are changing direction on that tile. For example, if you moved east to get to the current tile and your next move is to the south, you collect all raisins on the current tile.

Filip, just one year old, suspects that he needs a bit of help. So, he is asking you to write a program that tells him which route to take to collect the largest possible number of raisins.

Input

see file `~/6.txt`

Input to your program consists of lines with non-negative integers, separated by white space:

```
3 5
0 2 5 0 6
1 2 0 3 7
0 0 2 8 0
```

The first line contains r and c , the number of rows and columns in the grid (each ≤ 1000).

Each of the next r lines contains c numbers, the number of raisins on each tile in one row of the grid (each ≤ 1000).

The race starts in the northwest corner, i.e., first column of the first row; this tile contains no raisins. The race ends in the last column of the last row — no raisins there, either.

Output

Output from your program should be one line with a number and a string, separated by a single space:

```
16 EESESE
```

The number defines how many raisins are on the path and the string defines the path with each move designated by a letter (E for east and S for south). The string length will be $r + c - 2$.

Problem 7: Vertical Sums

One of Buster's favorite pastimes (he usually does this while waiting for his dog biscuits to bake) is playing the game of vertical sums. To play the game, Buster writes all the numbers from 1 to 19, in order, into a grid. He places the numbers into the grid by starting at the cell in the upper left corner of the grid and works his way cell by cell, from left to right, across the row. Once he finishes the top row he moves to the row immediately below it and repeats the process. As he works his way through the grid he places numbers into some of the cells he visits along the way. Single digit numbers fit into single cells; however, two-digit numbers are placed into two adjacent cells in the same row. Other than to represent a two-digit number, digits cannot share an edge of a cell, horizontally or vertically. An example of a correctly filled grid is shown below:

1		2		3		4		5	
			6		7		8		9
1	0								
		1	1			1	2		
1	3			1	4			1	5
		1	6			1	7		
1	8			1	9				
4	11	4	13	5	20	6	17	6	14

$\Sigma = 100$

Buster's friend Arnie loves arithmetic and after Buster fills in one of his grids, Arnie computes the sum of the numbers in each column (in the grid above Arnie's sums appear below the grid in the row that is shaded gray). One day over dog biscuits Arnie suggests to Buster that another way to play the game would be to start out with the sums and then fill in the grid, using the same rules as before, so that sum of the digits in each column match the sums.

Heidi and Sammy, while waiting for the next set of dog biscuits, overhear this conversation. They decide that it is possible to write a program that, given the size of the grid and the sums, fills in the grid if possible. Once they settle on an algorithm they realize that between napping, protecting the house, and walking, they do not have time to write the program. So, they decide to outsource the project to you.

Input

see file `~/7.txt`

Input to your program consists of positive integers, separated by white space. The first line contains two integers, r and c , the number of rows and columns of the grid to be filled (each ≤ 15). The second line contains c integers — each is the sum of the digits

in the corresponding column of the grid. (These sums add up to 100.) For the example above:

```
7 10
4 11 4 13 5 20 6 17 6 14
```

Output

Your output is either a single line with `No solution.` or the grid of digits, r lines with c characters each, where you represent each cell of the grid by a digit or a period. For the example above:

```
1.2.3.4.5.
...6.7.8.9
10.....
..11..12..
13..14..15
..16..17..
18..19.....
```

Problem 8: Gothic

What does blood have to do with a party? Very simple:

```
blood brood brook brock broch broth booth booty borty porty party
```

This is a sequence of minimal length, taken from the 233614 words in Webster's Second International Dictionary, which starts with `blood` and ends with `party` and where two successive words differ by exactly one letter.

Your job is to compute such sequences.

Input

see file `~/8.txt`

The first line of input contains a positive integer n . Each of the next n lines contains two words separated by white space. The remaining lines of the input contain all words from Webster's Second International Dictionary. For this problem words consist of lower-case letters from `a` to `z`. For example:

```
5
blood party
barn burr
aal aam
bottom bottom
automotive laboratory
a
aa
aal
aalii
aam
aani
...
```

Output

The output must consist of n lines, each one discussing one pair of words in the input: a line must contain a single word from the dictionary equal to both input words; a minimal-length sequence of dictionary words as described above, separated by single blanks starting with the first input word and ending with the second; or the statement `cannot morph word into word`. For the input above:

```
blood brood brook brock broch broth booth booty borty porty party
barn burn burr
aal aam
bottom
cannot morph automotive into laboratory
```