Problem A. Circles On The Toral Screen

Input file:	`circles.in`
Output file:	`circles.out`
Time limit:	1 second
Memory limit:	64 megabytes

Peter's computer is attached to screen 256×256. This screen has very unusual form. It is a tore! That's so beatiful! If his window is too large to be displayed, it just covers itself and the operating system allows to switch between the first part of the window and the second one.

Now Peter is writing a graphical editor for this screen. He decided to use the screen form in a very special way in his editor. Of course, if we go, for example, from point $(255, 255)$ to the right direction, we will come to point $(0, 255)$. So Peter decided to allow any possible integer values of coordinates so he will be able to draw very long curves without any checks as if he would have a very-very long screen. So, getting point coordinates (x, y), the editor always take them modulo 256. Note that it must use the positive remainder, for example, point $(-239, -17)$ corresponds to the physical point $(17, 239)$.

Now Peter has decided to implement circles drawing procedure in his editor. He thought for a very long time about it. Now he knows that the *representation* of the circle on the (usual flat, but possibly very-very long) screen is a non-empty set of points (x, y) with two conditions satisfied:

- Exactly two of eight neighbours of any fixed point from this set (x, y) belong to this set.
- The second condition is much harder to explain. Let us fix any point from the set (x, y). Denote with M the set of all its eight neighbours except the neighbours that belong to the representation. Let us include the point (x, y) itself to M also, so M will always contain seven points. The point (x, y) must be the *true nearest point to the circle* from this set. That is, the point (x, y) gives a true minimum to the absolute value of the difference of the Euclid distance of the point to the center of the circle and the radius of the circle (i. e. the function $|\sqrt{(x - x_0)^2 + (y - y_0)^2} - r|$ for the point (x, y) must be less than the value of this function for all other points of M, where (x_0, y_0) are coordinates of the center of the circle).

Now Peter asks you to help him to implement the creation of this representation to display it on his toral screen in his editor. The screen is empty at the beginning. Your program will read descriptions of the circles from the input file and draw them on the screen. If it is not possible to build representation of any of the given circles, your program must detect this fact.

Input

The first line of the input file contains the number of circles to be drawn $1 \le K \le 1000$. The next K lines contain the circles descriptions. Each circle is described by three numbers x, y and r — coordinates of the center and the radius of the circle. Coordinates are integers with absolute value not greater than 10^8, and radius is a non-negative integer which does not exceed 10^7. Also it is always guaranteed that the total sum of all radii does not exceed 10^7.

Output

If there were circles which do not have representation, the first line must contain message "`Circles without representation were detected!`". Then l lines must follow, each of which must correspond to one circle without representation in order they appear in input file (l is the total number of circles without representation). These lines must have format "`Unable to build circle` x y r". All messages are to be displayed without quotes.

In the other case, the file must contain the contents of the screen. The screen is represented by 256 lines of 256 characters each. You must use color (i mod 10) for i-th circle. For example, first circle from the input file must be displayed with the color '`1`', hundredth — with color '`0`', and two hundred thirty ninth — with '`9`'. The circles must be drawn in order they appear. For pixels which were left empty, use symbol '`.`'. Note that x coordinate grows right and the y coordinate grows down.

Note

The dots in ends of lines of the example output are removed for paper saving purposes. Your program must output lines with exact length of 256 characters.

Example

circles.in	circles.out
2 5 2 3 10 10 8	...1...1 ..1.....1 ..1.....22222 ..1...221....22 ...1.2.1.......2 ....211.........2 ...2.............2 ...2.............2 ..2...............2 ..2...............2 ..2...............2 ..2...............2 ..2...............2 ...2.............2 ...2.............2 ....2...........2 .....2.........2 ......22.....22 ....22222 *236 lines of 256 dots follow* 111

Problem B. How Many Programs?

Input file:	`programs.in`
Output file:	`programs.out`
Time limit:	7 seconds
Memory limit:	64 megabytes

Little Jaina loves programming. She already knows N programming constructions! Now Jaina thinks how many programs of a given length L she can compose using these constructions. Help her to count them!

Note that Jaina ignores any spaces and blank lines in her program and counts only printable characters. Also Jaina considers different two programs if they are equal as strings, but different sequences of constructions were used to create them.

Of course Jaina does not require that the program must compile. She will learn about compilers on the next lesson.

Input

The first line of the input file contains number of constructions $1 \le N \le 1000$. The next N lines contain constructions themselves. Length of any of these lines does not exceed 255 characters. If a construction occures more than once, all occurences are considered different, so there are exactly N different constructions Jaina knows. The strings contain only characters with ASCII codes from 32 to 126. Jaina wants that all blanks in the strings must be ignored.

The last line of the input contains a single integer $1 \le L \le 1000$.

Output

Output the number of different programs Jaina can compose.

Example

programs.in	programs.out
3 begin clrscr end 17	20

Problem C. The Middlest Number

Input file:	`midnum.in`
Output file:	`midnum.out`
Time limit:	9 seconds
Memory limit:	64 megabytes

One can perform the following operations to get the middlest number of a given sequence $a_1, a_2, \ldots, a_n$ written on the blackboard. He must select the minimal number a_{min}, the maximal number a_{max}, erase them and write down $\frac{a_{min}+a_{max}}{2}$. After repeating this $n-1$ times, there will be only one number on the blackboard. We will call it *the middlest number*.

Your task is to find the middlest number of a given sequence of integers.

Input

The first line contains integer $1 \le n \le 239017$. The next n lines contain numbers a_i.

Output

Output the middlest number of the given sequence with six digits after the decimal point.

Example

midnum.in	midnum.out
3 2 3 9	4.250000

Problem D. Logical Expression Calculator

Input file:	`logcalc.in`
Output file:	`logcalc.out`
Time limit:	9 seconds
Memory limit:	64 megabytes

Your task is to implement the calculator of logical expressions. The operations allowed are (from highest priority to lowest):

- '`~`' (not). Unary operation, which converts true to false and vice versa.
- '`&`' (and). Binary operation which is true only if both operands are true.
- '`|`' (or). Binary operation which is true if at least one of operands is true.
- '`^`' (xor — exclusive or). Binary operation which is true if exactly one of operands is true.

The expression can contain variables and boolean constants `0` (false) and `1` (true). There are 52 different variables allowed: capital and small English letters. Parentheses are also allowed with standard effect.

Input

The first line of the input contains an expression. The next line contains the number of sets of values of variables for which the expression must be calculated ($1 \le M \le 4063$). The next M lines contain 52 numbers each — the values of all variables '`A`'...'`Z`' and '`a`'...'`z`'. The length of expression does not exceed 50000 characters.

Output

For each set of input data output the value of expression on a single line. Output 0 if expression is false and 1 otherwise.

Example

logcalc.in	logcalc.out
A&B|C 4 1 1 0 0 0 *...47 zeros follow...* 1 0 1 0 0 *...47 zeros follow...* 1 0 0 0 0 *...47 zeros follow...* 0 0 0 0 0 *...47 zeros follow...*	1 1 0 0

Problem E. Travelling Salesman Returns!

Input file:	`salesman.in`
Output file:	`salesman.out`
Time limit:	10 seconds
Memory limit:	64 megabytes

Travelling Salesman plans to return to the Alpha Centauri system! All the people wait it! They want new best goods from other systems!

But the Salesman as usual wants to minimize the travel expenses. He selects any starting planet, flies there on the intergalactic spaceship, visits all planets in the system in order which minimizes the total cost, and then flies on the intergalactic spaceship away. Of course he does not want to visit any planet more than once. Your task is to calculate the optimal route for the Salesman. The people can wait no longer!

Input

The Alpha Centauri system contains n planets. This number is written on the first line of the input file ($1 \le n \le 19$). The next n lines contain n numbers each: j-th number of the i-th line is the travel cost from i-th planet to j-th. The numbers are separated by spaces. Numbers a_{ii} should be ignored. All numbers are positive integers which do not exceed 10^8.

Output

Output the minimal total cost in the first line. In the second line output n numbers — the route on which the total cost is minimized.

Example

salesman.in	salesman.out
3 8 1 6 3 5 7 4 9 2	5 3 1 2

Problem F. Unstable Systems

Input file:	`unstable.in`
Output file:	`unstable.out`
Time limit:	6 seconds
Memory limit:	64 megabytes

Of course you know that some operating systems are not stable. Sasha learnt it only few days ago. Now there are very bad days of his life. He is an adminstrator of the network of n computers with different versions of such systems. Each computer is a workstation which is usually used to run a single program. But the programs often crash with a message "`The system is busy or unstable`". Sasha has determined some *unsafety value* corresponding to the frequency of program crash for each program on each workstation (the larger values correspond to more often crashes). Now he plans to arrange programs in such a way that the maximal unsafety value of all workstations will become minimal possible (because crashes slow down all the work!). Help him!

Input

The first line of the input file contains the number of workstations n ($1 \le n \le 500$) which is equal to number of programs. The next n lines contain n numbers each — j-th number of i-th line contains the unsafety value for a program j on i-th computer. All numbers do not exceed 10^6 by their absolute values.

Output

Write the maximal unsafety value on the first line. Then output n lines each corresponding to one program in format i j — i-th computer must run j-th program.

Example

unstable.in	unstable.out
2 1 3 4 5	4 1 2 2 1

Problem G. Number Container

Input file:	`contain.in`
Output file:	`contain.out`
Time limit:	0.5 seconds
Memory limit:	64 megabytes

Consider all M^N N-digit numbers in the scale of notation with base M (leading zeros are allowed). Your task is to generate the lexicographically smallest string of digits of minimal possible length which will contain all these numbers as substrings if this string is infinitely appended to itself.

Input

There are only two integers in input file: M and N. It is guaranteed that $2 \le M \le 10$ and $1 \le N \le 10$. It is always true that $M^N \le 1000000$.

Output

Output the lexcographically smallest string of minimal length which solves this task.

Example

contain.in	contain.out
2 2	0011

Problem H. Reversive Inversions II

Input file:	`invers.in`
Output file:	`invers.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Inversion table for a permutation P of numbers $\{1, 2, \ldots, N\}$ is the table $A = (A_i)_{1 \le i \le N}$ which maps each $i = P_j$ into the number of indices j' such that $j' \le j$ but $P_{j'} > P_j = i$.

Given an inversion table for a permutation P, calculate the inversion table for the inverse permutation P^{-1}.

Input

File consists only of N integer numbers, delimited by spaces and newline characters, that form the inversion table of a permutation. You may assume that $1 \le N \le 262144$.

Output

Output N integer numbers separated by single spaces — inversion table for the inverse permutation. Leave no trailing spaces at the end of the single line of output.

If there are several possible answers, output any of them. If there are no answers, output the first N primes instead.

Example

invers.in	invers.out
5 0 1 3 2 1 0	1 5 1 3 2 0 0