Problem A. Cut!

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

You have a checkered paper. You need to cut this paper to pieces such that every peace contains exactly k grid cells of source paper. You can cut only by grid lines.

Input

The first line of the input file contains two integer numbers n and m ($1 \leq n, m \leq 100$) — height and width of paper in cells respectively. Second line contains number k ($1 \leq k \leq 1000$).

Output

If it is impossible to cut a given paper in described way print «-1». Otherwise, print n lines with m integer numbers each. Every number stands for a number of piece which contain this cell. Pieces should be enumerated with consecutive natural numbers starting from one.

Examples

`cut.in`	`cut.out`

`2 2`
`1`
	`1 2`
	`4 3`

`2 2`
`2`
	`1 1`
	`2 2`

`2 2`
`3`
	`-1`

`4 4`
`4`
	`1 1 1 4`
	`1 4 4 4`
	`2 2 3 3`
	`2 2 3 3`

Problem B. DNA

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

Deoxyribonucleic acid (DNA) — is a molecule that carries the genetic instructions used in the growth, development, functioning and reproduction of all known living organisms and many viruses.

The two DNA strands are represent two polynucleotides since. Each of which are composed of simpler monomer units called nucleotides. Each nucleotide is composed of one of four nitrogen-containing nucleobases: cytosine (C), guanine (G), adenine (A), or thymine (T). Every strand has an orientation. Two strands in DNA always have opposite orientations. So, the first nucleotide of the first strand is connected with the last nucleotide of the other strand, second with the one before last and so on.

Also, DNA satisfies the *complementary principle*: each type of the nucleobase on one strand specifies uniquely the type of the connected nucleobase on the other strand. For example, for adenine on the one strand always corresponds thymine on the other. For example, the strand AGC *complementary* to the strand GCT.

Michael knows only one strand s of his DNA but wants to know if there is a superman gene t in his DNA. DNA has gene if it contains as substring in one of it's strands.

Input

First line of the input file contains the string s with length no more than 200 characters.

Second line of the input file contains the string t with length no more than 20 characters.

Both strings contain only letters «ATGC».

Output

Print «Yes» if Michael has the superman gene or «No» otherwise.

Examples

dna.in	dna.out
`ATGCATGC` `TGC`	`Yes`
`ATGCATGC` `GCATGCAT`	`Yes`
`ATGCATGC` `TTT`	`No`

Problem C. Bandits

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

After robbing a caravan on a road, m bandits have obtained a loot of n similar diamonds. They decided to divide the loot. To do so they have ordered themselves by their birthdate, and make proposals in turn.

When the proposal is made, the bandits vote either for, or against it. If the strict majority of bandits votes for the proposal, it is accepted, in the other case the author of the proposal is killed.

The proposal states for each of the still alive bandits, what number of diamonds should he get. All bandits are smart, greedy, bloodthirsty and careful. That means, that the bandit votes against the proposal if and only if he is certain that he will survive and get the same number of diamonds or more in future assuming that all the other bandits also vote optimally.

Find out what is the maximal number of diamonds the youngest bandit (the one who makes the first proposal) can get. If he is killed regardless of his proposal, output "-1".

Input

Input file contains m and n ($1 \le m \le 2000$, $1 \le n \le 2000$).

Output

Print the maximal number of diamonds the first bandit can get, or "-1" if he is killed regardless of his proposal.

Example

bandits.in	bandits.out
`5 1000`	`997`
`2 1000`	`-1`

Problem D. Network Wars

Input file:	`network.in`
Output file:	`network.out`
Time limit:	3 seconds
Memory limit:	64 megabytes

Network of Byteland consists of n servers, connected by m optical cables. Each cable connects two servers and can transmit data in both directions. Two servers of the network are especially important — they are connected to global world network and president palace network respectively.

The server connected to the president palace network has number 1, and the server connected to the global world network has number n.

Recently the company *Max Traffic* has decided to take control over some cables so that it could see what data is transmitted by the president palace users. Of course they want to control such set of cables, that it is impossible to download any data from the global network to the president palace without transmitting it over at least one of the cables from the set.

To put its plans into practice the company needs to buy corresponding cables from their current owners. Each cable has some cost. Since the company's main business is not spying, but providing internet connection to home users, its management wants to make the operation a good investment. So it wants to buy such a set of cables, that cables *mean cost* is minimal possible.

That is, if the company buys k cables of the total cost c, it wants to minimize the value of c/k.

Input

The first line of the input file contains n and m ($2 \le n \le 100$, $1 \le m \le 400$). Next m lines describe cables — each cable is described with three integer numbers: servers it connects and the cost of the cable. Cost of each cable is positive and does not exceed 10^7.

Any two servers are connected by at most one cable. No cable connects a server to itself. The network is guaranteed to be connected, it is possible to transmit data from any server to any other one.

Output

First output k — the number of cables to buy. After that output the cables to buy themselves. Cables are numbered starting from one in order they are given in the input file.

Example

network.in	network.out

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

4 5
1 2 2
1 3 2
2 3 1
2 4 2
3 4 2
	3
	1 2 3

Problem E. Numbers to Numbers

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

Numbers in English are written down in the following way (only numbers less then 10^9 are considered). Number $\overline{abc\,def\,ghi}$ is written as "$[abc]$ million $[def]$ thousand $[ghi]$". Here "$[xyz]$" means the written down number $\overline{xyz}$.

In the written down number the part "$[abc]$ million" is omitted if $\overline{abc} = 0$, "$[def]$ thousand" is omitted if $\overline{def} = 0$, and "$[ghi]$" is omitted if $\overline{ghi} = 0$. If the whole number is equal to 0 it is written down as "zero". Note that words "million" and "thousand" are singular even if the number of millions or thousands respectively is greater than one.

Numbers under one thousand are written down in the following way. The number $\overline{xyz}$ is written as "$[x]$ hundred and $[yz]$. Here "$[x]$ hundred and" is omitted if $x = 0$. Note that "hundred" is also always singular.

Numbers under 20 are written down as "zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine", "ten", "eleven", "twelve", "thirteen", "fourteen", "fifteen", "sixteen", "seventeen", "eighteen", and "nineteen" respectively. Numbers from 20 to 99 are written down in the following way. Number $\overline{xy}$ is written as "$[x0]$ $[y]$", and numbers divisible by ten are written as "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty", and "ninety" respectively.

For example, number 987 654 312 is written down as "nine hundred and eighty seven million six hundred and fifty four thousand three hundred and twelve", number 100 000 037 as "one hundred million thirty seven", number 1 000 as "one thousand". Note that "one" is never omitted for millions, thousands and hundreds.

Given English text with all numbers written as words, transform it to the text where all numbers are written as numbers. For example, the text "he had one hundred and ninety five dogs" must be tranformed to "he had 195 dogs". You must perform a transformation in such a way that as many words as possible are transformed to numbers, so, for example, "three thousand" must be transformed to "3000", not to "3 thousand".

If there are several way to perform the transformation in such a way, you must do it so that the first number is as great as possible, for example "two thousand thirty two" must be transformed to "2032", not "2030 2". If there are still several ways, maximize the second number, and so on.

Input

Input file contains an English text that only contains English letters, punctuation marks (',', '.', '!', '?', ':', ';', '(', ')'), and blanks (spaces and line feeds).

You must perform the transformation in the way described. Words in numbers to be transformed are separated with blanks only (so, for example, "thirty, three" may only be transformed as "30, 3", not as "33").

When performing transformation you must replace characters starting from the first letter of the first word in the number to the last one in the last word with digits.

Input file is not empty and its size does not exceed 20000 bytes.

Output

Output the result of the transformation.

Example

numbers.in

```
From three thousand one hundred and fifty teams selected
from one thousand four hundred and eleven universities
in seventy five countries competing at one hundred and
twenty seven sites and hundreds more competing at
preliminary contests worldwide, seventy three teams
of students competed for bragging rights and prizes
at The Twenty Eighth Annual ACM International Collegiate
Programming Contest World Finals sponsored by IBM on
March Thirty One, Two Thousand Four, and hosted at the
Obecni Dum, Prague by Czech Technical University in Prague.
```

numbers.out

```
From 3150 teams selected
from 1411 universities
in 75 countries competing at 127 sites and hundreds more competing at
preliminary contests worldwide, 73 teams
of students competed for bragging rights and prizes
at The 20 Eighth Annual ACM International Collegiate
Programming Contest World Finals sponsored by IBM on
March 31, 2004, and hosted at the
Obecni Dum, Prague by Czech Technical University in Prague.
```

Problem F. Beautiful Permutation

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

Consider a permutation of integer numbers from 1 to n. Let us call length of the longest monotonic subsequence of the permutation its *ugliness*.

For example, the ugliness of the permutation $\langle 1, 2, 5, 3, 4\rangle$ is 4 because it has a monotonic subsequence $(1, 2, 3, 4)$ of length 4, but has none of length 5. The ugliness of the permutation $\langle 5, 6, 3, 4, 1, 2\rangle$ is 3 because it has a monotonic subsequence $(5, 3, 1)$ of length 3.

Let us call *beautiful* those permutations that have a smallest possible ugliness for given n. Given n, you must find the first in lexicographic order beautiful permutation of size n.

Input

Input file contains n ($1 \le n \le 10\,000$).

Output

Output the first in lexicographic order beautiful permutation of size n.

Example

perm.in	perm.out
2	1 2
4	2 1 4 3

Problem G. Flipping Bits

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Goose Tattarrattat has a sequence b of bits. The length of the sequence is n. Tattarrattat also has a favorite integer m which is between 1 and n, inclusive.

A sequence of n bits is called a rotator sequence if it has the following property: its prefix of length $n-m$ is equal to its suffix of length $n-m$.

For example, let $m = 2$. Consider the sequence $b = 10101010$. Its length is $n = 8$, so we have $n - m = 6$. The prefix of length 6 is "101010", the suffix of length 6 is "101010". They are the same, so this b is a rotator sequence. Now consider $b = 11010100$. For this b we compare the prefix "110101" to the suffix "010100". They differ, so this b is not a rotator sequence.

Tattarrattat wants to change her sequence b into some rotator sequence. She will produce such a sequence in a sequence of steps. In each step she can do one of the following two types of changes to the sequence:

- Flip any bit (from 1 to 0 or from 0 to 1).
- Flip the first $k \cdot m$ bits, for any positive integer k.

You are given b: each of its characters will be either '0' or '1'. Find the minimal number of steps required to obtain some rotator sequence.

Input

The first line contains sequence b $(1 \leq |b| \leq 300)$. Sequence b consists of n characters '0' and '1'. The next line contains integer m $(1 \leq m \leq n)$.

Output

Output the minimal number of steps required to obtain some rotator sequence.

Examples

standard input	standard output

`00111000`
`1`
	`2`

`101100001101`
`3`
	`2`

`11111111`
`4`
	`0`

`1101001000`
`8`
	`1`

`11011100101110`
`5`
	`4`

Problem H. Puzzle

Input file:	`puzzle.in`
Output file:	`puzzle.out`
Time limit:	2 секунды
Memory limit:	64 megabytes

Board games with tokens and large paper maps were quite popular a few years ago.

Recenlty Vasya had found k maps for such games on the attic. Without descriptions he was not able to understand the rules of the games, so he has invented his own rules.

On each of the maps there is a number of circles — possible token positions. One of these positions is marked as “Start” position and some other position — as “Finish”. Some of the circles are connected by segments and two positions can be connected by more than one segment. A player can move token from one position to another if and only if they are connected by a segment.

To play the game invented by Vasya one needs all the maps Vasya found. On each map the player puts a token to the “Start” position. On each move the player should move tokens on all maps. Each token should be move to position connected by a segment with the current position of the token. The token should be moved even it is in the “Finish” position.

Vasya wonders what is the minimal number of moves needed to put tokens an all maps to “Finish” positions.

It is guaranteed that on each map every position is reachable from all other positions.

Input

First line of the input file contains an integer number k ($1 \le k \le 10$).

Descriptions of these k maps follow. First string of each description contains two integer numbers n_i and m_i ($2 \le n_i \le 50$, $1 \le m_i \le 1500$) — number of positions and segments of i-th map respectively. Positions are numbered with integer numbers from 1 to n, “Start” position is position number 1, “Finish” position has the number n_i. Each of the next m_i lines contains two integer numbers — numbers of positions connected by a segment.

Output

If there is a sequence a moves leading to the situation in which all tokens are in “Finish” positions, output the minimal length of such a sequence. If such a sequence does not exist, output “`Impossible`”.

Example

puzzle.in	puzzle.out

2
5 4
1 2
2 3
3 4
3 5
3 3
1 2
2 3
3 1
	3

Problem I. String

Input file:	`string.in`
Output file:	`string.out`
Time limit:	2 секунды
Memory limit:	64 megabytes

So called *recurrent equations* are widely used in mathematics. Usually such equations are used to specify integer sequences. An example of such sequence is the Fibonacci sequence ($F_i = F_{i-1} + F_{i-2}$).

Using the recurrent equations one can define not only integer sequences, but also sequences of strings. In this problem we consider a sequence of strings s_0, s_1, ..., defined in the following way.

String s_0 is an empty string and each string s_i ($i \geq 1$) is obtained from s_{i-1} using the following rule: if decimal representation of i is a substring of s_{i-1}, then $s_i = s_{i-1}$, otherwise s_i is the concatenation of s_{i-1} and decimal representation of i.

You are given the number n. Find the string s_n.

Input

Input file contains an integer number n ($1 \leq n \leq 500$).

Output

Output s_n.

Examples

string.in	string.out
1	1
3	123
13	123456789101113

Problem J. Shooting Game

Input file:	`shooting.in`
Output file:	`shooting.out`
Time limit:	5 seconds
Memory limit:	64 megabytes

Alice and Bob play shooting game on a plane.

The game proceeds as follows. There are n obstacles on the plane, each obstacle is a segment. First, Alice selects some point on the plane that doesn't belong to a line containing an obstacle and stands there. Then Bob selects some other point and stands at it. After that Alice shoots at Bob from the gun.

Bob always selects such point that there are as many obstacles as possible between him and Alice. If the line connecting Alice and Bob passes through the end of an obstacle, this obstacle is considered to be between Alice and Bob.

Alice would like to shoot Bob very much, so she tries to select such point that Bob couldn't hide behind too many obstacles. Help Alice to choose such point on the plane, that after Bob selects his point, the number of obstacles between Alice and Bob was minimal possible.

Input

The first line of the input file contains n — the number of obstacles ($1 \le n \le 8$). The following n lines contain four integer numbers x_1, y_1, x_2, y_2 each — the coordinates of the ends of the corresponding obstacle. Obstacles have no common points. Coordinates do not exceed 100 by their absolute values.

Output

The first line of the output file must contain k — the minimal number of obstacles Alice can ensure to be between her and Bob. The second line of the output file must contain two real numbers — coordinates of the point Alice should choose. The point must not belong to any line containing an obstacle. Print as many digits after the decimal point as possible.

Examples

shooting.in	shooting.out

2
0 0 2 0
0 2 2 2
	1
	1.000000000000000000
	1.000000000000000000

Problem K. String Equations

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 megabytes

We all understand equations such as:

$$3 + 8 = 4 + 7$$

But what happens if we look at equations with strings instead of numbers? What would addition and equality mean?

Given two strings x and y, we define $x + y$ to be the concatenation of the two strings. We also define $x = y$ to mean that x is an anagram of y. That is, the characters in x can be permuted to form y.

You are given n distinct nonempty strings, each containing at most 10 lowercase characters. You may also assume that at most 10 distinct characters appear in all the strings. You need to determine if you can choose strings to put on both sides of an equation such that the "sums" on each side are "equal" (by our definitions above). You may use each string on either side 0 or more times, but no string may be used on both sides.

Input

The input consists of a number of cases. Each case starts with a line containing the integer n ($2 \le n \le 100$). The next n lines contain the n strings. The input is terminated with $n = 0$.

Output

For each case, print either "`yes`" or "`no`" on one line indicating whether it is possible to form an equation as described above. If it is possible, print on each of the next n lines how many times each string is used, with the strings listed in the same order as the input. On each line, print the string, followed by a space, followed by the letter "`L`", "`R`", or "`N`" indicating whether the string appears on the left side, the right side, or neither side in the equation. Finally, this is followed by a space and an integer indicating how many times the string appears in the equation. Each numeric output should fit in a 64-bit integer.

If there are multiple solutions, any solution is acceptable.

Example

stdin	stdout
2 hello world 7 i am lord voldemort tom marvolo riddle 0	no yes i L 1 am L 1 lord L 1 voldemort L 1 tom R 1 marvolo R 1 riddle R 1

Problem L. Doubly-sorted Grid

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	512 megabytes

A rectangular grid with lower case English letters in each cell is called **doubly sorted** if in each row the letters are non-decreasing from the left to the right, and in each column the letters are non-decreasing from the top to the bottom. In the following examples, the first two grids are doubly sorted, while the other two are not:

```
abc   ace   aceg   base
def   ade   cdef   base
ghi   bdg   xxyy   base
```

You are given a partially-filled grid, where some of the cells are filled with letters. Your task is to compute the number of ways you can fill the rest of the cells so that the resulting grid is doubly sorted. The answer might be a big number; you need to output the number of ways modulo 10007.

Input

The first line of the input contains two integers R and C ($1 \le R, C \le 10$) — the number of rows and the number of columns, respectively. Next R lines contains the description of partially-filled grid. Each of the next R lines contains a string of length C. Each character of the string is either a lower-case English letter, or '.', indicating that the cell is not filled yet.

Output

The sole line of the output should contain the answer to the problem.

Examples

standard input	standard output

`2 2`
`ad`
`c.`
	`23`

`3 3`
`.a.`
`a.z`
`.z.`
	`7569`

`4 4`
`....`
`.g..`
`.cj.`
`....`
	`0`