Problem A. Number of Close Strings

Input file:	close.in
Output file:	close.out
Time limit:	10 seconds
Memory limit:	256 mebibytes

Your password for the new website Taciturn is a string with n characters, each character is either '0' or '1'. The website requires you to change the password every day, so you've already accumulated k such past passwords, each being a string of n characters, each either '0' or '1'.

Today they have introduced a new requirement: your new password must differ from *each* of your previous passwords in at least m positions. In other words, for each previous password the number of positions i such that the character in i-th position of the previous password is different from the character in i-th position of the new password is at least m.

Note that this is a new requirement, so your old passwords might not be so different from each other and some of them might even be identical.

You're wondering how many passwords are now *NOT* usable for you. It's hard to find this number exactly, so you want to estimate it with at most 1% error.

Input

The first line of the input file contains three integers n ($1 \le n \le 50$) — the length of the password, k ($1 \le k \le 50$) — the number of your old passwords, and m ($1 \le m \le n$) — the minimum allowed difference between your new password and one of your old passwords.

The next k lines contain n characters each, without spaces — the old passwords.

Output

Output the number of passwords that are *not* usable as a new password for you as a floating-point number. Of course, this number is an integer, but since we're only interested in finding it approximately, you can output it as floating-point. Exponential form (like 2.34567e+89) is fine. Your answer will be accepted if it's from 0.99 times the true answer to 1.01 times the true answer.

Examples

close.in	close.out
5 3 2 00000 00001 11111	15.9

Problem B. Random Domino Placements

Input file:	domino.in
Output file:	domino.out
Time limit:	10 seconds
Memory limit:	256 mebibytes

Consider a $n \times n$ grid. You want to place some (possibly none) dominoes on this grid. Each domino occupies two adjacent cells of the grid, either horizontally or vertically. Two cells are adjacent when they share a side. At most one domino can occupy each cell.

There are many such placements. More precisely, let m be the number of different such placements. For example, when $n = 2$, there are $m = 7$ possible placements.

You need to pick k placements independently and uniformly at random from those m placements, so that for each pick each placement is used with probability $\frac{1}{m}$. See the output format for the details on how will your solution be checked.

Input

The input file contains two space-separated integers n and k.

There are exactly two testcases in this problem. The first testcase has $n = 2$, $k = 2$, the second testcase has $n = 30$, $k = 30\,000$.

Output

Output k independently and uniformly random placements. Each placement should occupy $n + 1$ lines. The first n lines should contain n characters each, describing the placement, the $n + 1$-st line should be empty.

The cells that are not occupied by any domino should be marked with '`.`' character (dot). The left cell of a horizontal domino should be marked with '`L`' character (capital letter L), the right cell of a horizontal domino — '`R`' character (capital letter R), the top cell of a vertical domino — '`T`' character (capital letter T), and the bottom cell of a vertical domino — '`B`' character (capital letter B).

In the first testcase, we'll just check that you've output two valid 2×2 placements. In the second testcase, we'll compute some statistics over the 30 000 placements you output to verify that the placements are (almost) independently and uniformly random. More specifically, we'll count how many times each possible 1×1, 2×2 and 3×3 square of characters (there are 9 918 such squares) appears in total in all placements you output, and compare that number to the expected number of appearances of such square of characters in an independently and uniformly random set of placements. Your solution will be accepted if for each square the number of its appearances is within six standard deviations of the expected one. In other words, if your placements are independently and uniformly random, your solution will be accepted with probability more than 99.99%.

Examples

domino.in	domino.out
2 2	LR .. TT BB

Problem C. Two Equal Squares

Input file:	`equal-squares.in`
Output file:	`equal-squares.out`
Time limit:	1 second
Memory limit:	256 mebibytes

You are given two equal squares on the plane. The squares have zero intersection area (but they might still touch or even share a segment). How many common tangent lines do they have?

A line a is a *tangent line* for a figure b if and only if the following two conditions hold:

1. All points belonging to figure b are either on a or to exactly one side of a. In other words, one of the half-planes formed by a must have no points from b.
2. At least one point belonging to figure b is on a.

Input

The input file contains 8 space-separated integers: x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4. Points (x_1, y_1) and (x_2, y_2) are opposite corners of the first square, while (x_3, y_3) and (x_4, y_4) are opposite corners of the second square. We guarantee that the squares are equal and that the area of their intersection is zero. All coordinates in the input file are between -10^4 and 10^4.

Note that the sides of the squares might not be parallel to coordinate axes.

Output

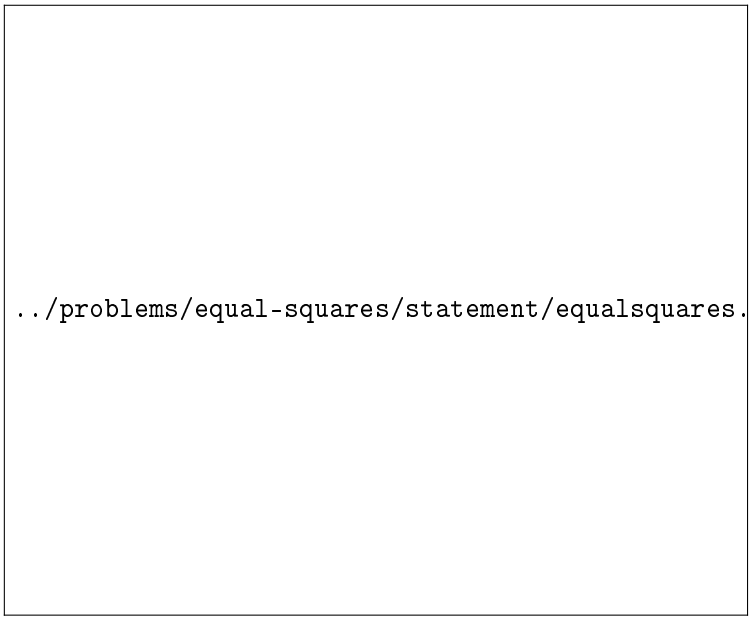
Print the number of common tangent lines for these two squares. In case there are infinitely many, output '`Infinity`' (without quotes).

Examples

equal-squares.in	equal-squares.out
0 0 3 4 5 5 10 5	4

Note

Here's how the example case looks like:



Problem D. Interleave Binary Strings

Input file:	`interleave.in`
Output file:	`interleave.out`
Time limit:	1 second
Memory limit:	256 mebibytes

You are given two random strings a and b, each containing just characters '`0`' and '`1`'. You're going to *interleave* the two strings: combine all their characters in such a way that the characters from each string still appear in order. For example, one could interleave strings '`0110`' and '`100`' to get '**1**`0`**10**`1`**0**`0`' — the characters from the first string are highlighted in bold.

How many different strings can you obtain by interleaving the two given strings? Output this number modulo $1\,000\,000\,007$ ($10^9 + 7$).

Input

The input file contains two lines. The first line contains the string a, and the second line contains the string b, each containing at least one and at most 50 characters, and all characters are either '`0`' or '`1`'.

The strings will be generated randomly: to generate each string, we will first pick a random length uniformly at random between 1 and 50, and then generate this many random characters, again uniformly.

There are 100 testcases in this problem. The first testcase corresponds to the sample input and output, the 99 remaining testcases are generated randomly.

Output

Output one integer — the number of different strings that can be obtained by interleaving the two given strings, modulo $1\,000\,000\,007$ ($10^9 + 7$).

Examples

interleave.in	interleave.out
0110 100	12

Problem E. Alternate Jumping

Input file:	jumping.in
Output file:	jumping.out
Time limit:	3 seconds
Memory limit:	256 mebibytes

Rita and Pasha are playing a cooperative game on a stripe of n cells, numbered from a to b ($n = b-a+1$). They start from a and their goal is to reach b.

The players make alternating moves: Rita moves, then Pasha moves, then Rita moves again, and so on. At each move Rita can jump from cell number x to cell number y if and only if $d(x) \le y-x \le \frac{x}{d(x)}$, where $d(x)$ is the smallest divisor of x greater than 1 (and thus $\frac{x}{d(x)}$ is the largest divisor of x less than x). In particular, Rita can make no jumps from a prime number, and thus they lose the game if it happens that it's Rita's turn and the current number is prime.

Pasha plays a bit differently: he can jump from cell number x to cell number y if and only if $d(y) \le y-x \le \frac{y}{d(y)}$. In particular, Pasha can make no jumps *to* a prime number.

How many ways are there for the players to reach the number b starting from number a? Rita moves first from a, and either Rita or Pasha can reach b. You need to output the answer modulo 10^9+7.

Input

The input file contains two space-separated integers a and b, $2 \le a < b \le 10^9$, $b-a \le 200\,000$.

Output

Print the number of ways to reach b from a, modulo 10^9+7.

Examples

jumping.in	jumping.out
4 10	2

Problem F. Recognize Power of Two

Input file:	power.in
Output file:	power.out
Time limit:	5 seconds
Memory limit:	256 mebibytes

Andrew has noticed that 7 and 9 are quite unusual digits. There are small powers of two that start with 1 in decimal notation ($2^0 = 1$), with 2 ($2^1 = 2$), ..., with 6 ($2^6 = 64$), and with 8 ($2^3 = 8$). However, the smallest power of 2 that starts with 7 in decimal notation is the enormous $2^{46} = 70\,368\,744\,177\,664$, and the smallest power of 2 that starts with 9 is even bigger: $2^{53} = 9\,007\,199\,254\,740\,992$.

Having learned this, Andrew wants to study the powers of two a bit more. More precisely, he will generate a random number, and ask you: what is the smallest power of two that starts with this number in decimal notation?

Prepare for answering such requests! Note that it's possible to prove that there exists a power of two that starts with any given prefix.

Input

The input file will contain one positive integer p with at most 50 digits, without leading zeroes.

The testcases will be generated randomly. More precisely, for each testcase except the first one, we will pick the length of the number uniformly at random between 1 and 50, and then pick a random number without leading zeroes with such length — again, uniformly at random.

There are 100 testcases in this problem. The first testcase corresponds to the example below, the 99 remaining testcases are generated randomly according to the above description.

Output

Output the smallest non-negative integer k such that 2^k starts with p in decimal notation.

Examples

power.in	power.out
7	46

Problem G. Power Sum Graphs

Input file:	`powersum.in`
Output file:	`powersum.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Consider a directed graph with n nodes, numbered from 1 to n. For each pair of nodes a and b, the graph either has an arc from a to b or does not. In this problem, there's at most one arc from a to b (but there might also be an arc from b to a), and there can be at most one arc from a to a — from a vertex to itself.

A special type of such graph is a *power sum graph*. It is defined like this: each of n nodes has an *outgoing power* s_i and an *incoming power* t_i assigned to it. Both powers are integers. There's also one fixed integer c for the entire graph called the *power level*. For each pair of nodes a and b there's an arc from a to b if $s_a + t_b \geq c$, and there's no arc from a to b if $s_a + t_b < c$.

Given a graph, check if it is a power sum graph. In case it is, you also need to output the incoming and outgoing powers of each node, as well as the power level. In case there are several solutions (and there usually are several solutions), output any.

Input

The first line of the input file contains one integer n, $1 \leq n \leq 5\,000$ — the number of nodes in the given graph. The next n lines contain n characters each (without spaces), the j-th character in the i-th line is equal to '`1`' if there's an arc from node i to node j, and '`0`' otherwise.

Output

On the first line of output, print '`YES`' (without quotes) if the given graph is a power sum graph, or '`NO`' if it's not. In case it is, output two integers on each of the next n lines — the outgoing and incoming powers of the corresponding node, and finally the power level on the next line of output. All numbers you output must be integers and between -10^9 and 10^9, inclusive. You can output any set of powers that corresponds to the given graph.

Examples

powersum.in	powersum.out
5 00000 11011 00001 11111 11111	YES 0 1 2 1 1 0 3 1 3 2 3

Problem H. Simplicity is the Ultimate Sophistication

Input file:	simplicity.in
Output file:	simplicity.out
Time limit:	5 seconds
Memory limit:	256 mebibytes

There are n types of candies in the world, numbered from 1 to n, and you love all of them! However, you love some of them more than the others. More specifically, for each two different types of candies $i \neq j$ you either prefer i to j, or prefer j to i.

Your preferences are sometimes giving you headaches. When you are given some set of candies to choose from, you want to pick one that you love the most from them. But your preferences can sometimes form a cycle, making it so that there's no single type of candy you prefer to all the rest. For example, suppose you prefer type 1 to type 3, prefer type 3 to type 7, and type 7 to type 1. Now, if you're given three candies of type 1, 3 and 7 to choose from, there's no good choice! If you pick a candy of type 1, you would be left wishing you picked candy of type 7; but if you actually pick the candy of type 7, you start longing for the candy of type 3, and so on.

We'll call a set of preferences *logical*, if the above situation cannot happen — in other words, if the set of preferences does not contain any cycles.

You've found that if you ban some small set x (with at most 15 elements) of the candy types, your preferences among the remaining candy types are logical. However, you don't want to ban too many candy types, so you want to find a (possibly different) set y of the candy types that has the minimum number of candy types and such that banning all types from y results in remaining preferences becoming logical.

Input

The first line of the input file contains one integer n ($1 \leq n \leq 500$) — the number of candy types.

The next n lines contain n characters each, without spaces: the j-th character of the i-th line is '1' if you prefer type i to type j, and '0' if you prefer type j to type i or if $i = j$. The i-th character of j-th line is always different from the j-th character of the i-th line when $i \neq j$.

The following line contains one integer k ($1 \leq k \leq 15$) — the number of elements in the set x. The following line contains k different space-separated integers, each between 1 and n — the candy types in x. We guarantee that the preferences between the candy types *not* in x are logical (also known as acyclic).

Output

On the first line of the output file, print m — the number of elements in the set y. On the second line of the output file print m different space-separated integers, each between 1 and n — the candy types in y. The preferences between the candy types *not* in y must be logical, and m must be minimum possible.

Examples

simplicity.in	simplicity.out
4 0110 0011 0001 1000 2 2 3	1 4

Problem I. Lies, Damned Lies, and Statistics

Input file:	stats.in
Output file:	stats.out
Time limit:	1 second
Memory limit:	256 mebibytes

Statistics allows you to make conclusions about a big world by looking at a relatively small sample. The usual way to do that is to look at some metrics of the sampled values, often called *statistics* themselves.

Suppose you have a sample of n students' test scores - real values $a_1, a_2, \ldots, a_n$, each between 0 and 100. Here are three commonly used statistics that one would use to understand the scores better:

- The *mean* score is the average of the individual scores: $m = \frac{1}{n}\sum_i a_i$.
- The *standard deviation* of the scores is the square root of the average squared difference from the mean: $s = \sqrt{\frac{1}{n}\sum_i (a_i - m)^2}$.
- The *median* of the scores is the middle score in the sorted order. For simplicity, we will assume that n is an odd number (does not divide evenly by 2), and we'll define the median h as the $\frac{n+1}{2}$-th score in sorted order, counting from 1.

You're given three values m, s and h. Could they be the mean, standard deviation, and median of some n test scores, where n is an odd number between 1 and 19, and each score is between 0 and 100? If yes, you need to find at least one set of scores with such statistics.

Input

The first line of the input file contains the number t of testcases to solve, $1 \le t \le 10\,000$. Each of the next t lines describes one testcase and contains three integers m, s and h, each between 0 and 100, inclusive.

Output

Output one line for each testcase. If there exist n scores, such that n is an odd number between 1 and 19, each score is a (possibly non-integral) number between 0 and 100, and the scores have the given mean, standard deviation, and median, then output n followed by those n scores, separated with spaces. In case there are several solutions, output any. If there's no solution, output -1.

Your output will be considered correct if all scores are between -10^{-7} and $100 + 10^{-7}$, and computing the mean, standard deviation and median of your set of scores results in statistics that differ from the given ones by at most 10^{-7}.

Examples

stats.in	stats.out
1 42 36 40	5 0.0 10.0 40.0 60.0 100.0