Problem A. Three Colors (High Div Only!)

Input file:	`3colors.in`
Output file:	`3colors.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Three is like magic number in computer science. For example, consider vertex coloring problem. It is easy to detect whether you can color vertices of an undirected graph using two colors so that no vertices of the same color are connected by an edge, but it is very hard to do it if you are allowed to use three colors. Asking for more? It is easy to solve 2-SAT but it is hard to solve 3-SAT.

It is unfair, so in this problem we consider an easy problem with three colors while the problem with two colors is more difficult.

Consider coloring edges of an undirected G using k colors from $\{0, 1, \ldots, k-1\}$. Denote the sum of colors of edges incident to vertex u as $s(u)$. The coloring is called *neighbor distinguishing* if for any two vertices u and v connected by an edge $s(u) \neq s(v)$.

Given a bipartite graph G you must find its neighbor distinguishing 3-coloring.

Input

The first line of the input file contains three integer numbers n_1, n_2 and m — the number of vertices in each part and the number of edges, respectively ($1 \le n_1, n_2 \le 1500$, $1 \le m \le 10000$). The following m lines describe edges, each edge is described by two integer numbers — the numbers of vertices it connects. Vertices in each part are independently numbered starting from 1.

Output

If the given graph has no neighbor distinguishing 3-coloring, output "-1".

In the other case output m integer numbers — the colors of the edges in order they are described in the input file.

Examples

3colors.in	3colors.out
3 3 7 1 1 1 2 1 3 2 2 3 1 3 2 3 3	0 0 0 1 2 2 2

Problem B. Antipalindromic Numbers

Input file:	`anti.in`
Output file:	`anti.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Palindrome is a word that is read the same way in either direction. Similarly let us define *antipalindrome* as a word that has no equal characters on the same positions when read from front to back and from back to front. For example "computer" is antipalindrome, but "information" is not ('m' is on the 6-th position in both "information" and "noitamrofni").

A number is called antipalindromic if its decimal notation is antipalindrome.

You are given a number x. Find the smallest antipalindromic number greater than x.

Input

Input file contains several test cases. Each test case is the number x ($1 \le x \le 10^{100}$). The last test case is followed by 0, it must not be processed.

Output

For each number in the input print the smallest antipalindromic number greater than it on a line by itself.

Examples

anti.in	anti.out
5 20 99 0	10 21 1010

Problem C. Arbitrage

Input file:	`arbitrage.in`
Output file:	`arbitrage.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

As the financial crisis got stronger, currency exchange rates started to fluctuate too much, so the central banks of several countries decided to fix the conversion rate between their currencies. After that the currency exchange market will be closed for good, and the new common currency will be introduced in these countries.

However, there are some restrictions on the exchange rates that will be fixed for the transitional period. We call a matrix of exchange rates *legible* if it satisfies the following conditions:

1. For any pair of currencies u and v the exchange rate c_{uv} (1 unit of u currency is exchanged to c_{uv} units of v currency) must be inside the corridor defined using exchange rates on the market at the day before the reform. So it must satisfy inequalities $m_{uv} \le c_{uv} \le M_{uv}$ for given m_{uv} and M_{uv}.
2. For each pair of currencies u and v it must be $c_{uv} = 1/c_{vu}$.
3. There must be no arbitrage possibility — it must be impossible to perform a sequence of exchanges starting with some amount in currency u and finishing with larger amount of the same currency. That is, for any sequence of currencies $u_1, u_2, \dots u_k$ it must be $c_{u_1u_2}c_{u_2u_3}\dots c_{u_ku_1} \le 1$.
4. All exchange rates c_{uv} must be rational.

Given corridors for all pairs of currencies, help the central banks to establish legible matrix of exchange rates.

Input

The first line of the input file contains n — the number of countries ($2 \le n \le 50$). The following n^2 lines contain two rational numbers each: m_{11} and M_{11}, m_{12} and M_{12}, ..., m_{1n} and M_{1n}, m_{21} and M_{21}, ..., m_{nn} and M_{nn} ($m_{ii} = M_{ii} = 1$; $1/100 \le m_{ij} \le M_{ij} \le 100$). All numbers are written as a fraction of a form p/q.

Output

Output n lines, each line must contain n rational numbers separated by spaces. Each rational number must be written as an irreducible fraction. Its numerator and denominator must not exceed 10^{1000}.

If there is no solution, print "`Impossible`" at the first line of the output file instead.

Examples

arbitrage.in	arbitrage.out
3 1/1 1/1 7/10 8/10 25/1 35/1 12/10 14/10 1/1 1/1 30/10 50/1 1/33 1/28 1/45 1/35 1/1 1/1	1/1 10/13 30/1 13/10 1/1 39/1 1/30 1/39 1/1
2 1/1 1/1 1/3 1/2 1/3 1/2 1/1 1/1	Impossible

Problem D. Border Correlation (High Div Only!)

Input file:	`border.in`
Output file:	`border.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Border correlation is used in statistical word analysis when determining structural similarity of words.

Consider a word w of length n composed of lowercase letters of the English alphabet. A word is said to be *bordered* if its prefix function is greater than zero. Recall that a prefix function of a word is the length of its longest prefix that isn't equal to the word itself and is also the word's suffix. For example, the word "`abababa`" is bordered, because it has prefix "`ababa`" that is also its suffix, but the word "`aababab`" is unbordered because none of its prefixes is its suffix except two degenerate cases (ε and the word itself).

Let us number cyclic shifts of w from 1 to n, the i-th cyclic shifts starts from the i-th character of the original word. For example, the first cyclic shift of "`abababa`" is "`abababa`", the second one is "`bababaa`", etc, the seventh one is "`aababab`".

Border correlation $B(w)$ is a word composed of n characters '`0`' and '`1`'. The i-th character of $B(w)$ is '`1`' if the i-th cyclic shift of w is bordered, and '`0`' if it is unbordered. For example, the border correlation of "`abababa`" is "`1011110`".

Given a word w find its border correlation.

Input

Input file contains a word w consisting of lowercase letters of the English alphabet. Its length doesn't exceed 100 000.

Output

Output border correlation of the word in the input file.

Examples

border.in	border.out
abababa	1011110

Problem E. Chipmunks

Input file:	`chipmunks.in`
Output file:	`chipmunks.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Andrew and Ann work in a medical laboratory. They make tests of new medicines on chipmunks. Initially there were two chipmunks in the laboratory, and they were numbered 1 and 2. It was very convenient — you can say "chipmunks with total number 2" and understand that this is about chipmunk 2, or "chipmunks with total number 3" and understand that this is about both chipmunks.

However, soon the third chipmunk was bought for the laboratory. Now when you say "chipmunks with total number 3" it is not clear whether this is about chipmunk 3 or chipmunks 1 and 2. However, Andrew and Ann found the solution. Now they say "1 chipmunk with total number 3" or "2 chipmunks with total number 3" and it is clear what set of chipmunks is referred.

But the problems came when the fourth chipmunk was bought. If it were assigned number 4, the phrase "2 chipmunks with total number 5" would be ambiguous. So Andrew and Ann had to find another solution, and of course it was found. The chipmunk was assigned number 5. And again specifying the number of chipmunks in the set and their total number uniquely identified the set of chipmunks.

Help Andrew and Ann to continue assigning numbers to chipmunks as more and more of them are bought. Let first $n-1$ chipmunks be already assigned numbers in the following way: each new chipmunk is assigned the smallest possible positive integer number such that the number of chipmunks in a set and sum of their numbers uniquely identifies the set. Suppose the n-th chipmunk is bought. Find out what is the smallest possible number that can be assigned to the n-th chipmunk so that the above property was preserved.

Input

The input file contains one integer number n ($1 \le n \le 20$).

Output

Output one number — the number that must be assigned to the n-th chipmunk.

Examples

chipmunks.in	chipmunks.out
1	1
4	5
5	8

Problem F. Coins Game

Input file:	coins.in
Output file:	coins.out
Time limit:	4 seconds
Memory limit:	256 megabytes

Ann and Betty play a game with coins. Initially $m \times n$ coins are placed on an $m \times n$ grid, some heads up, some tails up. The cells of the grid are indexed by two numbers (x, y) where $1 \le x \le m$ and $1 \le y \le n$. Ann moves first.

Each turn a player to move does the following: she selects the coin which is heads up and flips it. Let the coin to be flipped reside at cell (i, j) of the grid ($1 \le i \le m$, $1 \le j \le n$). After that the same player also selects two numbers i_1 and j_1 such that $0 \le i_1 < i$, $0 \le j_1 < j$ and also flips the coins at positions (i_1, j), (i, j_1) and (i_1, j_1) (if some of them don't exist because one of the coordinates is zero, the corresponding positions are ignored).

The player who cannot make a move because all coins are tails up loses.

Given the initial configuration of coins find out who wins the game, and if it is Ann what is her winning first move.

Input

The first line of the input file contains m and n ($1 \le m, n \le 50$). The following m lines contain n characters each, the j-th character of the i-th of these lines is '1' if the coin at (i, j) is heads up and '0' if it is tails up.

Output

The first line of the output file must contain the name of the winning player. If it is Ann, the second line must contain i and j and the third line must contain i_1 and j_1. These numbers must describe the winning first move. If there are several possible winning moves, output any one.

Examples

coins.in	coins.out
3 3 000 000 001	Ann 3 3 0 0
2 2 11 01	Betty

Problem G. Machine Learning

Input file:	`learning.in`
Output file:	`learning.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Machine learning studies the ability of algorithms and computer programs to "learn". Usually machine learning is achieved through "training" with a large set of samples. In this problem we consider a simple example of machine learning when the deterministic automaton is trained to recognize some language given by a set of samples.

Deterministic finite automaton (DFA) is an ordered set $\langle \Sigma, U, s, T, \varphi \rangle$ where Σ is the finite set called *input alphabet* ($\Sigma = \{0, 1\}$ in this problem), U is the finite set of *states*, $s \in U$ is the *initial state*, $T \subset U$ is the set of *terminal states* and $\varphi : U \times \Sigma \to U$ is the *transition function*.

The input of the automaton is a word α over Σ. Initially the automaton is in state s. Each step it reads the first character c of the input word and changes its state to $\varphi(u, c)$ where u is the current state. After that the first character of the input word is removed and the step repeats. If after its input word is empty the automaton is in the terminal state, it accepts the initial word α, in the other case it rejects it.

All words of length from 0 to n are divided to two sets S^+ and S^-. The automaton is said to be correct with respect to this division if it accepts all words from S^+ and rejects all words from S^-. Find the correct automaton with the minimal number of states. Note that the behavior of the automaton on words from neither S^+ nor S^- can be arbitrary.

Input

The first line of the input file contains n ($1 \le n \le 12$). The following $2^n - 1$ lines describe S^+ and S^-. Each line contains one word prefixed with '+' if it is in S^+ or with '-' if it is in S^-. Words are listed ordered by length, and then lexicographically.

Output

The first line of the output file must contain u — the number of states of the automaton ($u \ge 1$), and s — the initial state (states are numbered from 1 to u).

The second line must contain t — the number of terminal states, followed by t integer numbers — the terminal states themselves.

The following u lines must contain two numbers each — the i-th of these lines must contain $\varphi(i, 0)$ and $\varphi(i, 1)$.

Examples

learning.in	learning.out
2 + +0 -1 +00 -01 -10 -11	2 1 1 1 1 2 2 2

Problem H. Peaks

Input file:	`peaks.in`
Output file:	`peaks.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Permutation $\langle a_1, a_2, \ldots, a_n \rangle$ of integer numbers from 1 to n is said to have k peaks if inequalities $a_{i-1} < a_i > a_{i+1}$ are satisfied for exactly k different indices i (we consider $a_0 = a_{n+1} = 0$ for the purpose of the above inequalities).

For example, permutation $\langle 3, 1, 4, 5, 2 \rangle$ has two peaks at $i = 1$ and $i = 4$.

Given n and k find the number of permutations of numbers from 1 to n with exactly k peaks. Return this number modulo 239.

Input

Input file contains two integer numbers: n and k ($1 \le n \le 10^{15}$, $1 \le k \le 30$).

Output

Output one integer number — the number of permutations of numbers from 1 to n with exactly k peaks, modulo 239.

Examples

peaks.in	peaks.out
3 1	4
10 3	131

Problem I. Tour

Input file:	`tour.in`
Output file:	`tour.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

There are n cities in Flatland, some of them are connected by bidirectional roads. The system of roads is organized in such way that there is exactly one way to get from any city to any other city by the roads. Recently the king of Flatland has decided that he would like to travel around the country and visit all the cities. He would like to start travelling from the capital of the country, travel along the roads and return to the capital visiting each city exactly once.

The minister of transport of Flatland told the king that it is impossible to do so, because there are no cycles in the road system of Flatland, in particular there is no way to travel from the capital back to the capital without visiting a city more than once. But the king insisted that he would like to perform the tour. So the minister ordered to build new roads in order to make such tour possible.

Help the minister to find the minimal number of roads to build so that the king could perform his tour.

Input

The first line of the input file contains n — the number of cities ($3 \le n \le 100\,000$). The following $n-1$ lines describe roads. Each road is described by two integer numbers — the numbers of cities it connects. The cities are numbered from 1 to n.

Output

Output one integer number — the minimal number of roads that must be built.

Examples

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

In the given example roads $3-4$ and $4-5$ can be built, the king's tour would then be $1-2-3-4-5-1$.

Problem J. The Wall (High Div Only!)

Input file:	`wall.in`
Output file:	`wall.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

After conquering Edgeland, Flatland and Allies decided to divide its capital to areas of influence. After some negotiations it was decided to build the wall to separate Flatland area from Allies area. Due to perfectionism of the leaders of the parties, it was decided that the wall would have form of a circle.

Each party marked several points of interest on the map that it would like to have in its area of influence. Now the wall must have all Flatland points at one side of the wall, and all Allies points on the at the other side. The wall would be very thin, so it can pass at any side of the points exactly on it.

Help the parties to build the wall. Given points of interest, find some circle that satisfies the requirements above.

Input

The first line of the input file contains n and m — the number of points of interest of Flatland and the number of points of interest of Allies ($2 \le n, m \le 120$). The following n lines contain two integer numbers each — the coordinates of points of interest of Flatland. The following m lines describe points of interest of Allies. All coordinates do not exceed 10^4 by their absolute values. All points of interest are different.

Output

If it is possible to build the wall, print "`YES`" at the first line of the input file. The second line must contain three real numbers: coordinates of the center and radius of the wall. Your answer must be accurate up to 10^{-6}, but print as many digits after the decimal point as possible.

If there is no solution, print "`NO`" at the first line of the output file.

Examples

wall.in	wall.out
2 2 0 0 0 1 1 0 1 1	YES 0 0.5 0.5
2 2 0 0 1 1 1 0 0 1	YES 0.5 0.5 0.70710678118654752
4 2 0 0 2 2 2 0 0 2 1 1 5 5	NO

Problem K. Little Brackets (First Div Only!)

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

onsider all regular bracket sequences with one type of brackets. Let us call the *depth* of the sequence the maximal difference between the number of opening and the number of closing brackets in a sequence prefix. For example, the depth of the sequence "`()()(())`" is 2, and the depth of "`(((()(()())))`" is 4.

Find out the number of regular bracket sequences with n opening brackets that have the depth equal to k. For example, for $n = 3$ and $k = 2$ there are three such sequences: "`()(())`", "`(()())`", "`(())()`".

Input

Input file contains several test cases. Each test case is described with n and k ($1 \le k \le n \le 50$).

Last testcase is followed by two zeroes. They should not be processed.

Output

For each testcase output the number of regular bracket sequences with n opening brackets that have the depth equal to k.

Separate output for different testcases by a blank line. Adhere to the format of the sample output.

Example

*

brackets.in	brackets.out
3 2 37 23 0 0	Case 1: 3 Case 2: 203685956218528

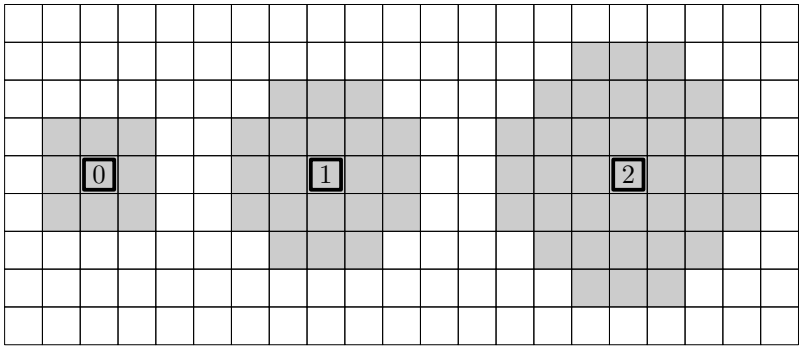
Problem L. Under Control (First Div Only!)

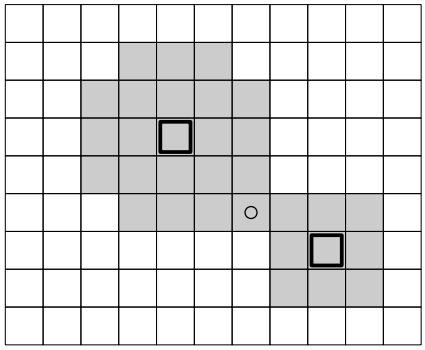
Input file:	`control.in`
Output file:	`control.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

n a game of *Civilization III* the area controlled by a city is defined by its culture level. The game proceeds on a rectangular grid. A city occupies one grid square. Each city has a *culture level* which is a non-negative integer number.

A city with a culture level 0 controls its own square and eight adjacent squares. A city with a culture level 1 additionally controls all squares that share a side with those squares (a total of $9 + 12 = 21$ squares). Generally, if a city with a culture level i controls the set A of squares, a city with the same location and a culture level $i + 1$ would control all these squares and also squares that share a side with at least one square from A.

The picture on the left shows the sets of squares controlled by cities with culture levels of 0, 1 and 2.





The area controlled by the civilization is defined as follows. Consider the total area controlled by all its cities. The civilization area is the smallest set of squares, such that it contains all the squares controlled by some city, and its complement contains no hanging squares. A square x of a set B is called *hanging* if there is no 2×2 square in B that contains square x.

Calculate the total area controlled by a civilization, given the locations of all its cities on a map. You may consider that the map is infinite and that there are no other civilizations.

Input

The first line of the input file contains an integer number n — the number of the cities of a civilization ($1 \le n \le 50$). Next n lines describe cities. Each city is described with its integer coordinates (x_i, y_i) and its culture level c_i. Coordinates do not exceed 10^9 by their absolute value, culture level does not exceed 10.

Output

Output the total number of squares controlled by a civilization.

Example

*

control.in	control.out
2 0 0 1 4 -3 0	31

The squares controlled by the civilization in the example are shown on the right picture. The square marked by a small circle is not controlled by any city, however it belongs to the area controlled by the civilization because otherwise it would be hanging.

Problem M. Puzzle (First Div Only!)

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

ittle Georgie likes puzzles very much. Recently he has found a wooden triangle in the box with old toys. The side of the triangle is n inches long. The triangle is divided into n^2 unit triangles with lines drawn on his surface.

The interesting fact about that triangle is that it is not solid — it consists of two parts. Each of the parts is a connected set of unit triangles. Georgie has put his triangle onto the table and now wonders whether he can separate the parts. He wants to separate them without taking any part of the triangle off the table, just moving the parts by the table surface. The triangle has a small but non-zero thickness, so while being moved the parts must not intersect.

For example, if the triangle is divided into parts as it is shown on the top picture below, Georgie can separate the parts the way he wants. However in the case displayed on the bottom picture, he cannot separate the parts without lifting one of them.

Help Georgie to determine whether he can separate the parts moving them by the surface of the table.

Input

Input file contains one or more testcases. The first line of each testcase contains n ($2 \le n \le 50$). Next n lines contain the description of the triangle, i-th of these lines contains $2i - 1$ characters, describing unit triangles in the i-th row, from left to right. Character '0' means that the triangle belongs to the first part of the main triangle, '1' means that it belongs to the second one.

Testcases are separated with blank lines. Testcase with $n = 0$ designates the end of the test data, this testcase must not be processed.

Output

For each puzzle output the line with its number followed by the line that states whether the parts can be separated. Separate output for different cases with a blank line.

Example

*

puzzle.in	puzzle.out
6 0 001 00011 0000011 000111111 00111111111 6 0 001 00111 0011011 000000111 00111111111 0	Puzzle 1 Parts can be separated Puzzle 2 Parts cannot be separated