Dress Rehearsal Problem A
Preludes
Problem ID: chopin

Frederic Chopin was a Polish music composer who lived from 1810-1839. One of his most famous works was his set of preludes. These 24 pieces span the 24 musical keys (there are musically distinct 12 scale notes, and each may use major or minor tonality). The 12 distinct scale notes are:

1	2	3	4	5	6	7	8	9	10	11	12
A	$A\sharp = B\flat$	B	C	$C\sharp = D\flat$	D	$D\sharp = E\flat$	E	F	$F\sharp = G\flat$	G	$G\sharp = A\flat$

Five of the notes have two alternate names, as is indicated above with the equals sign (e.g. $C\sharp = D\flat$ means that note has two names, $C\sharp$ and $D\flat$). Thus, there are 17 possible names for the scale notes, but only 12 musically distinct notes. When using one of these as the keynote for a musical key, we can further distinguish between the major and minor tonalities. This gives 34 possible keys, of which 24 are musically distinct.

In naming his preludes, Chopin used all the keys except for the following 10 (which were named instead by their alternate names):

$A\flat$ minor $A\sharp$ major $A\sharp$ minor $C\sharp$ major $D\flat$ minor
$D\sharp$ major $D\sharp$ minor $G\flat$ major $G\flat$ minor $G\sharp$ major

Write a program that, given the name of a key, will give an alternate name (if it has an alternate) or report that the key name is unique.

Input

Each test case is described by one line of input having the format "`note tonality`", where `note` is one of the 17 names for the scale notes given above, and `tonality` is either `major` or `minor`. All note names will be upper-case, and the two accidentals ($\sharp$ and $\flat$) will be written as `#` and `b`, respectively.

Output

For each case, display the case number followed by the alternate key name, if it exists, or print `UNIQUE` if the key name is unique. Follow the format of the sample output.

Sample Input	Output for Sample Input
Ab minor D# major G minor	Case 1: G# minor Case 2: Eb major Case 3: UNIQUE

This page is intentionally left blank.

Dress Rehearsal Problem B
Limited Correspondence
Problem ID: correspondence

Emil, a Polish mathematician, sent a simple puzzle by post to his British friend, Alan. Alan sent a reply saying he didn't have an infinite amount of time he could spend on such non-essential things. Emil modified his puzzle (making it a bit more restricted) and sent it back to Alan. Alan then solved the puzzle.

Here is the original puzzle Emil sent: given a sequence of pairs of strings $(a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)$, find a non-empty sequence $s_1, s_2, \ldots, s_m$ such that the following is true:

$$a_{s_1} a_{s_2} \ldots a_{s_m} = b_{s_1} b_{s_2} \ldots b_{s_m}$$

where $a_{s_1} a_{s_2} \ldots$ indicates string concatenation. The modified puzzle that Emil sent added the following restriction: for all $i \neq j$, $s_i \neq s_j$.

You don't have enough time to solve Emil's original puzzle. Can you solve the modified version?

Input

Each test case starts with a line containing an integer $1 \leq k \leq 11$, followed by k lines. Each of the k lines contains two space-separated lowercase alphabetic strings which represent a pair. Each individual string will be at most 100 characters long.

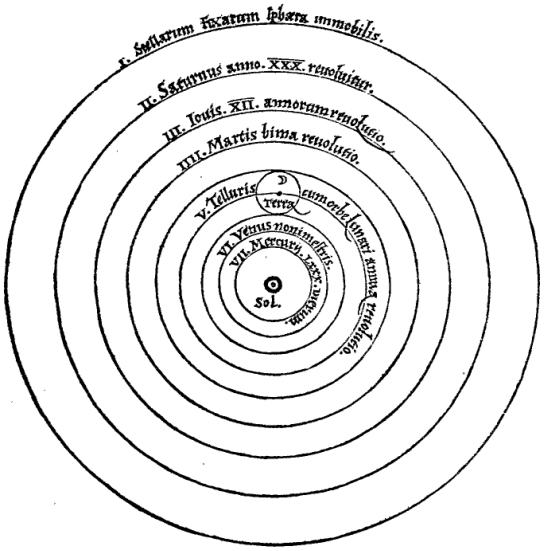
Output

For each case, display the case number followed by the sequence found (if it is possible to form one) or "`IMPOSSIBLE`" (if it is not possible to solve the problem). If it is possible but there are multiple sequences, you should prefer the shortest one (in terms of the number of characters output). If there are multiple shortest sequences, choose the one that is lexicographically first. Follow the format of the sample output.

Sample Input	Output for Sample Input
5 are yo you u how nhoware alan arala dear de 8 i ie ing ding resp orres ond pon oyc y hello hi enj njo or c 3 efgh efgh d cd abc ab 3 a ab b bb c cc	Case 1: dearalanhowareyou Case 2: ienjoycorresponding Case 3: abcd Case 4: IMPOSSIBLE

Dress Rehearsal Problem C
Heliocentric
Problem ID: heliocentric

Nicolaus Copernicus, born in Toruń in the Kingdom of Poland in 1473, was the first to collect sufficient evidence to show that the heliocentric view of the solar system (that the Sun was at the center of the planetary orbits) could compete with the Ptolemaic view (that the Earth was at the center). Copernicus viewed each planet as revolving in a circular orbit with the Sun as the common center of each circle. This new model, which was later improved by many other astronomers, made it much simpler to understand and predict the motions of the planets. Pictured below is his rendering of the planetary orbits.



Consider two of the planets in Copernicus' orbital system: Earth and Mars. Assume the Earth orbits the Sun in exactly 365 Earth days, and Mars orbits the Sun in exactly 687 Earth days. Thus the Earth's orbit starts at day 0 and continues to day 364, and then starts over at day 0. Mars orbits similarly, but on a 687 day time scale. We would like to find out how long it will take until both planets are on day 0 of their orbits simultanously. Write a program that can determine this.

Input

Each test case is described by a single line containing two integers $0 \leq e < 365$ and $0 \leq m < 687$. These indicate which days Earth and Mars are at their respective orbits.

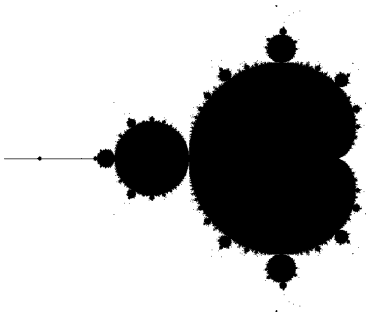
Output

For each case, display the case number followed by the smallest number of days until the two planets will both be on day 0 of their orbits. Follow the format of the sample output.

Sample Input	Output for Sample Input
0 0 364 686 360 682 0 1 1 0	Case 1: 0 Case 2: 1 Case 3: 5 Case 4: 239075 Case 5: 11679

Dress Rehearsal Problem D
In or Out
Problem ID: mandelbrot

Born in Warsaw, Benoît Mandelbrot (1924-2010) is considered the father of fractal geometry. He studied mathematical processes that described self-similar and natural shapes known as fractals. Perhaps his most well-known contribution is the Mandelbrot set, which is pictured below (the set contains the black points):



The Mandelbrot set is typically drawn on the complex plane, a 2-dimensional plane representing all complex numbers. The horizontal axis represents the real portion of the number, and the vertical axis represents the imaginary portion. A complex number $c = x + yi$ (at position (x, y) on the complex plane) is *not* in the Mandelbrot set if the following sequence diverges:

$$z_{n+1} \leftarrow z_n^2 + c$$

beginning with $z_0 = 0$. That is, $\lim_{n\to\infty} |z_n| = \infty$. If the sequence does not diverge, then c is in the set.

Recall the following facts about imaginary numbers and their arithmetic:

$$i = \sqrt{-1}, \quad i^2 = -1, \quad (x + yi)^2 = x^2 - y^2 + 2xyi, \quad |x + yi| = \sqrt{x^2 + y^2}$$

where x and y are real numbers, and $|\cdot|$ is known as the *modulus* of a complex number (in the complex plane, the modulus of $x + yi$ is equal to the straight-line distance from the origin to the the point (x, y)).

Write a program which determines if the sequence z_n diverges for a given value c within a fixed number of iterations. That is, is c in the Mandelbrot set or not? To detect divergence, just check to see if $|z_n| > 2$ for any z_n that we compute – if this happens, the sequence is guaranteed to diverge.

Input

Each test case is described by a single line containing three numbers: two real numbers $-3 \le x \le 3$ and $-3 \le y \le 3$, and an integer $0 \le r \le 10{,}000$. The value of c for this case is $x + yi$, and r is the maximum number of iterations to compute.

Output

For each case, display the case number followed by whether the given c is in the Mandelbrot set, using `IN` or `OUT`.

Sample Input	Output for Sample Input
0 0 100 1.264 -1.109 100 1.264 -1.109 10 1.264 -1.109 1 -2.914 -1.783 200 0.124 0.369 200	Case 1: IN Case 2: OUT Case 3: OUT Case 4: IN Case 5: OUT Case 6: IN

Dress Rehearsal Problem E
Polish Notation
Problem ID: polish

One way of representing mathematical expressions is with Polish (or prefix) notation, initially developed by Jan Łukasiewicz (1878-1956) for use in logic. In this notation, the operator appears before its operands. For example, rather than writing

$$3 + 4 * 7$$

using so-called infix notation, we could write the equivalent expression in Polish notation

$$+ 3 * 4 7$$

In this example, the addition applies to the values of the two following expressions, '3' and '* 4 7'. Further, the multiplication applies to two values that follow it, 4 and 7.

In general, we can write an expression in Polish notation using the following rules:

- A constant integer is an expression (e.g. 3).
- A lowercase single character is an expression (a variable, e.g. x).
- If E1 and E2 are expressions and OP is an operator, then the following is an expression: 'OP E1 E2'. The meaning of this expression in infix notation is 'E1 OP E2'.

The operators that we will support are all binary: addition (+), subtraction (-), and multiplication (*).

Polish notation has the advantage of being easy to parse for a computer. Write a program that reads an expression in Polish notation, simplifies the expression as much as possible, and prints it out in Polish notation. The simplification procedure is just replacing subexpressions that contain no variables with their values wherever possible.

Input

Each test case is described by a single line containing an expression. The expression is a sequence of space-separated tokens (integers, operators, and single-character lowercase variables). Each expression will be correctly formatted in Polish notation, with at most 2048 symbols (operators and operands combined). Integer constants will be in the range $[-10, 10]$.

Output

For each case, display the case number followed by the simplified expression in Polish notation, in the same format as the input. Do not modify the expression other than the simplification step described earlier. All intermediate computations will fit within a signed 32-bit integer. Follow the format of the sample output.

Sample Input	Output for Sample Input
+ 3 4 - x x * - 6 + x -6 - - 9 6 * 0 c	Case 1: 7 Case 2: - x x Case 3: * - 6 + x -6 - 3 * 0 c

This page is intentionally left blank.

Dress Rehearsal Problem F

Prime Spiral

Problem ID: spiral

Boredom can be good for creativity. Polish mathematician Stanislaw Ulam (1909-1984) discovered the eponymous Ulam spiral while listening to a "long and very boring paper". He started by writing down the positive integers in a spiral on a grid, one number per grid cell. Then he eliminated the composite numbers (i.e. non-primes). An interesting property he discovered was that the remaining prime numbers seem to form along many diagonals of the grid:

All positive integers

101	100	99	98	97	96	95	94	93	92	91
102	65	64	63	62	61	60	59	58	57	90
103	66	37	36	35	34	33	32	31	56	89
104	67	38	17	16	15	14	13	30	55	88
105	68	39	18	5	4	3	12	29	54	87
106	69	40	19	6	1	2	11	28	53	86
107	70	41	20	7	8	9	10	27	52	85
108	71	42	21	22	23	24	25	26	51	84
109	72	43	44	45	46	47	48	49	50	83
110	73	74	75	76	77	78	79	80	81	82
111	112	113	114	115	116	117	118	119	...	

Only primes

101				97						
					61		59			
103		37						31		89
	67		17				13			
				5		3		29		
			19			2	11		53	
107		41		7						
	71				23					
109		43				47				83
	73						79			
		113							...	

Note that both of these are infinitely large grids, but due to physical constraints only a finite subset of the grid will fit in this space.

Let's consider traveling around the second grid above (the Ulam spiral), where you are free to travel to any cell containing a composite number, but traveling to any cell containing a prime number is disallowed. You can travel up, down, left, or right, but not diagonally. Write a program to find the length of the shortest path between pairs composite numbers, if it is possible. Note, for example, that it is impossible to travel from the cells numbered 12 and 72 to any other composite cell. The length of a path is the number of steps on the path.

Input

Each test case is described by a line of input containing two integers $1 \leq x, y \leq 10{,}000$ which indicate two cells in the grid. Note that while there are limits on the input values, the intervening path between x and y is not limited in any way.

Output

For each case, display the case number followed by the length of the shortest path between the cells x and y, or "impossible" if no such path is possible.

Sample Input	Output for Sample Input
1 4 9 32 10 12	Case 1: 1 Case 2: 7 Case 3: impossible

Dress Rehearsal Problem G
Sierpiński circumference
Problem ID: triangle

Polish mathematician Wacław Sierpiński (1882-1969) described the 2D geometric figure known as the Sierpiński triangle as part of his work on set theory in 1915. The triangle, which is really an infinite collection of points, can be constructed by the following algorithm:

1. The initial shape is a solid triangle.
2. Shrink the current shape to half its dimensions (both height and width), and make two more copies of it (giving three copies total).
3. Arrange the three copies so that each touches the two others at their corners. Set the current shape to be the union of these three.
4. Repeat from step 2.

Here is an illustration of the first few iterations:

As the iterations go to infinity, this process creates an infinite number of connected points. However, consider the case of a finite number of iterations. If the initial triangle has a circumference of 3, what is the sum of the circumferences of all (black) triangles at a given iteration? Write a program to find out not the exact circumference, but the number of decimal digits required to represent its integer portion. That is, find the number of decimal digits required to represent the largest integer that is at most as large as the circumference.

Input

Each test case is a line containing a non-negative integer $0 \leq n \leq 10{,}000$ indicating the number of iterations.

Output

For each case, display the case number followed by the number of decimal digits required to represent the integer portion of the circumference for the given number of iterations. Follow the format of the sample output.

Sample Input	Output for Sample Input
0 1 5 10 100	Case 1: 1 Case 2: 1 Case 3: 2 Case 4: 3 Case 5: 19

Dress Rehearsal Problem H

Towers of Powers 2: Power Harder

Problem ID: towers

In this problem we will be exploring the concept of sorting numbers. Most of you have likely written a sorting algorithm before, but just in case, here is a tutorial: You reorder the numbers to put the smaller ones in front and larger ones in the back.

Having dealt with the theory, we come to practice. You will be given some integers. Please output the same integers in sorted order. Where is the trick, you ask? Well, you might have to deal with equality cases. If two numbers are equal, the one that was the first in the input should also be the first on the output. Other than that, there are no tricks, no complications. Just sort the numbers from lowest to highest. Simple!

Oh, one more thing. When we say *some* integers, we mean they might be somewhat large integers. To make life easier for you, we will express them in a simple "tower of powers" form. Each integer will be in the format

$$a_1^{a_2^{\cdot^{\cdot^{a_n}}}},$$

where $1 \leq a_1, a_2, \ldots, a_n \leq 100$, and $1 \leq n \leq 100$. Note that evaluation is from top to bottom, thus

$$a_1^{a_2^{a_3}} = a_1^{\left(a_2^{a_3}\right)}.$$

Are you done yet? No? Better start working!

Input

The first line of each test case contains the number M of integers in this test case, $1 \leq M \leq 100$. Each of the next M lines describes one of the M integers. The integer is written as described above, using the carat (^) to represent power, with no whitespace. There will be between 1 and 100 numbers in the representation of each integer, and each of these numbers will be an integer between 1 and 100.

Output

For each test case, display the case number followed by the sorted list of integers, one per line, in the original form.

Sample Input	Output for Sample Input
4 2^2^2 3^4 15 9^2	Case 1: 15 2^2^2 3^4 9^2

This page is intentionally left blank.

Dress Rehearsal Problem I

Matrix Inverse

Problem ID: matrix

Given a square $n \times n$ matrix A, the definition of its inverse $B = A^{-1}$ is the matrix that fulfills the equality:

$$AB = I$$

where B and I are $n \times n$ matrices, and I is the identity matrix with ones along the diagonal, and zeros everywhere else:

$$I = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

For this problem, you should implement a matrix inverse solver for 2×2 matrices.

Input

Each test case is described by two lines of input. Each line has two 32-bit signed integers, and the integers given in order a, b, c, d represent the values of the matrix to invert:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

A blank line follows each test case.

Output

For each case, display the case number followed by two lines containing the inverse of the given matrix. Each test case is guaranteed to have an inverse (i.e. no matrix is singular), and that inverse is integer-valued. Follow the format of the sample output.

Sample Input	Output for Sample Input
1 0 0 1 30 29 1 1 -7 -16 4 9	Case 1: 1 0 0 1 Case 2: 1 -29 -1 30 Case 3: 9 16 -4 -7

This page is intentionally left blank.

Dress Rehearsal Problem J
Reversing Roads
Problem ID: roads

You work for the city of One-Direction-Ville. The city mandates that every road in its limits be one direction only. You are evaluating proposals for a new subdivision and its road network. One problem you've observed in some early proposals is that it is impossible to get to certain locations from others along the proposed roads. In order to speed up evaluation of subsequent proposals, you want to write a program to determine if it is possible to get to any location from any other location; you call this a *valid* proposal. And if a proposal is not valid, then your program should find out if there is an easy way to fix it by reversing the direction of one of the roads.

Input

Each test case begins with a line containing two integers, $1 \leq m \leq 50$ and $0 \leq n \leq m(m-1)/2$. m indicates the number of locations in the proposal, and n indicates the number of roads connecting these locations. Following this are n lines. Each line contains two space-separated integers a and b, where $0 \leq a, b < m$ and $a \neq b$. This indicates that there is a road from location a to location b. If there is a road from a to b, then there will be *no* road from b to a. Also, there will never be more than one road between two locations.

Output

For each case, display the case number followed by an indication of whether the proposal is valid or not. If the proposal is valid, output `valid`. If it is not valid, but by reversing the direction of one roads it can become valid, print the two locations which describe the existing road that should be reversed. If more than one road reversal can create a valid proposal, print the first one that appears in the input. If the proposal is not valid and impossible to become valid by reversing one road, print `invalid`. Follow the format of the sample output.

Sample Input	Output for Sample Input
3 3 0 1 1 2 2 0 3 3 0 1 1 2 0 2 3 2 1 2 0 2 4 4 0 1 1 2 2 3 0 3	Case 1: valid Case 2: 0 2 Case 3: invalid Case 4: 0 3

This page is intentionally left blank.

Dress Rehearsal Problem K
Counting Stars
Problem ID: stars

The field of astronomy has been significantly advanced through the use of computer technology. Algorithms can automatically survey digital images of the night sky, looking for new patterns.

For this problem, you should write such an analysis program which counts the number of stars visible in an bitmap image. An image consists of pixels, and each pixel is either black or white (represented by the characters # and −, respectively). All black pixels are considered to be part of the sky, and each white pixel is considered to be part of a star. White pixels that are adjacent vertically or horizontally are part of the same star.

Input

Each test case begins with a line containing a pair of integers $1 \leq m, n \leq 100$. This is followed by m lines, each of which contains exactly n pixels.

Output

For each case, display the case number followed by the number of stars that are visible in the corresponding image. Follow the format of the sample output.

Sample Input	Output for Sample Input
10 20 ##################--- ##-################-- #---################ ##-################ ########---######### #######-----######## ########---######### ###################-- ###################--- ####################-# 3 10 #-######## ---------- #-########	Case 1: 4 Case 2: 1

This page is intentionally left blank.

Dress Rehearsal Problem L
Statistics
Problem ID: statistics

Research often involves dealing with large quantities of data, and those data are often too massive to examine manually. Statistical descriptions of data can help humans understand their basic properties. Consider a sample of n numbers $X = (x_1, x_2, \ldots, x_n)$. Of many statistics that can be computed on X, some of the most important are the following:

- $\min(X)$: the smallest value in X
- $\max(X)$: the largest value in X
- range(X): $\max(X) - \min(X)$

Write a program that will analyze samples of data and report these values for each sample.

Input

Each test case is described by one line of input, which begins with an integer $1 \leq n \leq 30$ and is followed by n integers which make up the sample to be analyzed. Each value in the sample will be in the range -1,000,000 to 1,000,000.

Output

For each case, display the case number followed by the min, max, and range of the sample (in that order). Follow the format of the sample output.

Sample Input	Output for Sample Input
2 4 10 9 2 5 6 4 5 9 2 1 4 7 6 10 1 2 5 10 9 1 9	Case 1: 4 10 6 Case 2: 1 9 8 Case 3: 1 10 9 Case 4: 9 9 0

This page is intentionally left blank.