

Problem C. Fibonacci number's ratio (the hard version)

Input file: `stdin`
Output file: `stdout`
Time limit: 2 s
Memory limit: 512 Mb

Fibonacci numbers have many remarkable properties. For example, the ratio of two successive Fibonacci numbers tends to the golden ratio:

$$\lim_{n \rightarrow \infty} \frac{F_{n+1}}{F_n} = \frac{1 + \sqrt{5}}{2}.$$

You are going to test this property for generalized Fibonacci numbers

$$G_n = aG_{n-1} + bG_{n-2}, \quad n \geq 2.$$

For given a , b , G_0 and G_1 calculate the limit

$$\lim_{n \rightarrow \infty} \frac{G_{n+1}}{G_n},$$

if it exists.

Input

The first line contains four integers a , b , G_0 , G_1 from -1000 to 1000 .

Output

Print “YES” in the first line, if the limit exists, and “NO” otherwise. In the case of the positive result, print the limit itself in the second line. Absolute or relative error must not exceed 10^{-6} .

Examples

stdin	stdout
1 1 1 1	YES 1.6180339887
0 0 1 1	NO

Note

If the sequence G_n contains an infinite number of zeros, the limit does not exist.

Problem D. Generalized Fibonacci sequence

Input file: `stdin`
Output file: `stdout`
Time limit: 2 s
Memory limit: 512 Mb

Let us say that a generalized Fibonacci sequence is a sequence, where the first two elements are equal to a and b (a and b are positive integers), and every next element is a sum of the two previous ones. Thus, every such sequence can be uniquely determined by its first two elements. For example, if $a = 3$, $b = 4$, then the first sequence members are 3, 4, 7, 11, 18, 29, The standard Fibonacci sequence is obtained when $a = b = 1$.

You have a number n . You want to find the number of generalized Fibonacci sequences, where n appears as an i -th element, where $i > 2$. Generalized sequence's elements are numbered from 1.

Input

Input consists of a single integer n ($1 \leq n \leq 10^6$).

Output

Print a single integer, the answer to the problem.

Examples

<code>stdin</code>	<code>stdout</code>
3	3
1	0

Problem E. Addition in the Fibonacci number system

Input file: `stdin`
Output file: `stdout`
Time limit: 2 s
Memory limit: 512 Mb

Fibonacci numbers are defined by the recurrence:

$$F_0 = 0, \quad F_1 = 1, \quad F_n = F_{n-1} + F_{n-2}, \text{ for } n > 1.$$

Fibonacci number system (FNS) is based on Zeckendorf's theorem, which states that any positive integer has a unique representation in the following form:

$$n = F_{k_1} + F_{k_2} + \dots + F_{k_r},$$

where F_{k_i} — Fibonacci numbers, and $k_1 \gg k_2 \gg \dots \gg k_r \gg 0$. Here $i \gg j$ implies that $i \geq j + 2$. Non-negative integer can be written as a sequence of ones and zeros:

$$n = (b_m b_{m-1} \dots b_2)_F \Leftrightarrow n = \sum_{k=2}^m b_k F_k,$$

where $b_i = 1$ if F_i occurs in the representation, and 0 otherwise. For example, $1000000 = 832040 + 121393 + 46368 + 144 + 55 = F_{30} + F_{26} + F_{24} + F_{12} + F_{10}$, or $(1000000)_{10} = (100010100000000000010100000000)_F$. This number system looks a lot like the binary representation, except that it never has consecutive ones.

When we add 1 to the number in FNS two cases arise. If the lowest digit is zero, it is replaced by one (because $F_2 = 1$), otherwise two lowest digits will be 01, and they turn into 10 (because $F_3 = F_2 + 1$). Then we have to replace set 011 with 100 (because $F_n = F_{n-1} + F_{n-2}$) while the string of numbers has consecutive ones.

Write a program for addition of two numbers in the FNS.

Input

You have two non-negative integers in the FNS, one number per line. The number of digits in the numbers can be different, but does not exceed 5000.

Output

Print the result of addition in the FNS.

Examples

<code>stdin</code>	<code>stdout</code>
1010 100	10010

Problem F. Playing with coins

Input file: `stdin`
Output file: `stdout`
Time limit: 2 s
Memory limit: 512 Mb

Petya and Vasya came up with a game with coins. They have n coins of values a_i . There can be several coins of the same value. At first, all the coins lie on the table. Players take one coin from the table by turns. Petya goes first. He wins if after his turn he has preassigned and known to both players sum s . Other coins can still be on the table at that moment. Vasya wins if he manages to prevent Petya from getting specified sum.

Petya considers the game unfair, because there are quite a few sums s , that can provide him a victory. Help the guys determine all the possible sums s , that Petya can win regardless of what Vasya does.

Input

First line contains integer n ($1 \leq n \leq 100$) — the number of coins on the table. Next line contains n positive integers — the values of coins a_i . Their sum does not exceed 100.

Output

In the first line output the number of sums s which are winning for Petya. In the next line print sums themselves in ascending order.

Examples

<code>stdin</code>	<code>stdout</code>
3 2 2 3	3 2 3 5

Problem G. Wythoff's game with three piles

Input file: `stdin`
Output file: `stdout`
Time limit: `2 s`
Memory limit: `512 Mb`

There are three piles of stones. Two players take turns removing stones from these piles; during each move, a player can remove any number of stones from one of the piles or the same number of stones from all three piles. A player wins, if he takes the last stone. Determine who wins if both players play optimally.

Input

The input contains multiple test cases. The first line contains their number t ($1 \leq t \leq 1000$). The next t lines describe the test cases. Each of them contains three integers a, b, c ($1 \leq a, b, c \leq 10^{12}$), the numbers of stones in the piles.

Output

For each test case, in a separate line output the number of the player (1 or 2), who wins if they play optimally. If the first player wins, in the same line after a space output the first move leading him to the win. A move is described by two integers: a number of a pile and a strictly positive number of stones, that should be taken from the pile. The piles are numbered by integers 1, 2, 3. If the same number of stones should be taken from all three piles, output 0 instead of the number of a pile. If there are many winning moves, output any of them. If the second player wins, do not print a move.

Examples

<code>stdin</code>	<code>stdout</code>
3	1 0 1
1 1 1	2
1 2 3	1 1 4
5 2 3	

Problem H. Matches are not toys for children

Input file: `stdin`
Output file: `stdout`
Time limit: 2 s
Memory limit: 512 Mb

Petya and Vasya play with matches again. They have a pile of matches, from which the players remove matches in turns by certain rules. A player loses, if he can not make a valid move.

The permanent characters of olympiad problems are fed up with variants of the game, when one can take no more than a fixed amount of matches during a move. That's why today they agreed, that the number of matches, that can be removed during a move, is from one to $\lfloor \log_2 n \rfloor$ (the rounded down binary logarithm of the number of matches n *before the current move*).

Although the game promises to be interesting, Petya and Vasya have no chance to finish it. At any moment, Granny can come, and she strongly disapproves games with matches. Help the boys to determine, who of them wins, if they both play optimally.

Input

The input contains multiple test cases. The first line contains their number t ($1 \leq t \leq 1000$). The next t lines describe the test cases. Each of them contains an integer n ($1 \leq n \leq 10^{18}$), the number of matches in the pile at the beginning of the game.

Output

For each test case, in a separate line output the number of the player (1 or 2), who wins if they play optimally. If the first player wins, in the same line after a space output the first move leading him to the win, that is the number of matches, that should be taken from the pile. If there are many choices of a winning move, output any of them. If the second player wins, do not print a move.

Examples

stdin	stdout
5	2
1	1 1
2	2
3	1 1
4	1 2
5	

Note

$\log_2 1 = 0$, therefore, if only one match remains in the pile, the game ends.

Problem I. Move the queen to the corner!

Input file: `stdin`
Output file: `stdout`
Time limit: `2 s`
Memory limit: `512 Mb`

There is a queen in the upper left corner of the $n \times m$ chessboard. Two players play the following game. They take turns. During each move, the queen is moved vertically down, horizontally right or diagonally right and down to any non-zero number of squares. The player, after whose move the queen will be in the lower right corner of the board, loses the game.

As you have already guessed, your job is to determine, who wins, if both players play optimally.

Input

The input contains multiple test cases. The first line contains their number t ($1 \leq t \leq 1000$). The next t lines describe the test cases. Each of them contains a pair of integers n and m ($2 \leq n, m \leq 10^{12}$), where n is a height and m is a width of a board.

Output

For each test case, in a separate line output the number of the player (1 or 2), who wins if they play optimally. If the first player wins, in the same line output the first move leading him to the win. A move is described by two non-negative integers, a vertical shift a and a horizontal shift b . Note that for a correct move of the queen we have $a = 0$, or $b = 0$, or $a = b$, but a and b can not be zeros together. If there are multiple choices of a winning move, output any of them. If the second player wins, do not print a move.

Examples

<code>stdin</code>	<code>stdout</code>
3	2
3 3	1 1 1
5 7	1 9 0
10 2	

Problem J. Move the (p, q)-knight to the corner!

Input file: `stdin`
Output file: `stdout`
Time limit: `2 s`
Memory limit: `512 Mb`

The (p, q) -knight is a figure that moves p squares in one direction (vertical or horizontal) and q squares in the perpendicular one. For the standard chess knight, $p = 1$, $q = 2$.

The (p, q) -knight stays in the upper left corner $(1, 1)$ of the chessboard $n \times m$ and needs to reach the lower right corner (n, m) . In this problem, the knight can perform only such moves, that bring it closer to the target in both horizontal and vertical directions. Thus it can move p squares right and q squares down, or vice versa, q squares right and p squares down. Of course, the knight can not leave the board. Moreover, the board contains forbidden squares. The knight is not allowed move into them, but can jump over them. The squares $(1, 1)$ and (n, m) can not be forbidden.

Determine the number of ways to get from the square $(1, 1)$ to the square (n, m) , without visiting the forbidden squares. Since this number can be very large, output it modulo d (d is prime).

Input

The first line contains knight's parameters p and q and a prime module d ($1 \leq p < q \leq 20$, $2 \leq d \leq 10^6$). In the second line, the board sizes n and m ($2 \leq n, m \leq 10^9$) are given. The next line contains the only integer k , the number of forbidden squares ($0 \leq k \leq 10$). The next k lines describe forbidden squares by pairs of coordinates " $x_i \ y_i$ " ($1 \leq x_i \leq n$, $1 \leq y_i \leq m$). All forbidden squares are distinct from each other and from start and finish squares $(1, 1)$ and (n, m) .

Output

Output the single integer — the answer modulo d .

Examples

stdin	stdout
1 2 23 4 4 0	2
1 2 23 4 4 1 3 2	1

Problem K. Game

Input file: `stdin`
Output file: `stdout`
Time limit: `2 s`
Memory limit: `512 Mb`

In aristocratic circles of Berland, the following game is widespread. A field for this game contains l cells arranged on a circle. Some cells contain white and black chips, one cells can not contain more than one chip. The first player moves white chips, and the second player moves black ones. The players take turns. The player with white chips starts the game. During one move, a player can take any chip of his color and move it by non-zero number of cells clockwise or counter-clockwise. It is not allowed to jump over other chips or occupy a cell with another chip. A player, who can not perform a valid move, loses the game. Determine the outcome of the game, if both opponents play optimally. There can be a draw (i.e. the game can last infinitely).

Input

The input consists of multiple test cases. In the first line, the integer t is given, describing the number of the test cases. Description of each test case starts with three integers l , n and m ($1 \leq l \leq 10^9, 1 \leq n, m \leq 10^6$), the number of cells, the number of white chips and the number of black chips, respectively. Then an increasing sequence of n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq l$) follows. They are the numbers of cells occupied by white chips. Then an increasing sequence of m integers $b_1, b_2, \dots, b_m$ ($1 \leq b_i \leq l$) follows. They are the numbers of cells occupied by black chips. Cells are numbered by integers from 1 to l in clockwise order. All numbers a_i and b_j are different. Total sums of n and m over all the test cases do not exceed 10^6 .

Output

For each test case, output an answer in a separate line. If the player wins, who moves white chips, output “**white**”. If the player wins, who moves black chips, output “**black**”. In the case of a draw, output “**draw**”.

Examples

stdin	stdout
4	black
2 1 1	draw
1	white
2	white
4 1 1	
1	
2	
4 2 1	
1 4	
2	
6 2 2	
5 6	
2 4	

Problem L. Game with a string

Input file: `stdin`
Output file: `stdout`
Time limit: 2 s
Memory limit: 512 Mb

This problem is interactive. Your program has to read data from the standard input and write the data to the standard output. After writing each line, be sure to use the stream flushing function. For example, you must use the `fflush(stdout)` function in C++, call `System.out.flush()` in Java, `flush(output)` in Pascal, `sys.stdout.flush()` in Python, `Console.Out.Flush()` in C#.

Suppose there is a string written on a blackboard, and two player play the following game with it. The players take turns alternately. A move consists of erasing a non-empty subset of symbols of the string. It is not allowed to erase two equal symbols during one move. For example, from the string “abacaba”, it is allowed to erase symbols with numbers 1, 2 and 4, but not allowed to erase symbols with numbers 1, 3, 5, 6, because, for example, the first and the third symbols are equal (“a”). A player loses, if he can not make a move, i.e. if before his move, there are no symbols on the blackboard.

Your task is to choose, for which player (the first one or the second one) you would like to play, and beat the jury’s program (the interactor) in this game.

Input

The input contains a non-empty string s of length up to 500, consisting of lowercase latin letters. Symbols in the string are numbered by integers from 1 to its length. After erasing symbols, the numbers of remaining symbols *do not change*.

During information exchange with the interactor, you get opponent’s moves in the input, one per line. Description of a move contains a positive integer — a number of elements in an erased subset. Then the same string contains the numbers of erased symbols in an arbitrary order. Numbers in the description of the move a separated by spaces. It is guaranteed, that the interactor’s moves are correct — all erased symbols are different, every number is printed only once and does not coincide with a number of some previously erased symbol. Your moves should satisfy these conditions too.

If you play for the first player, output your first move, then read a move of the second player, than output your move and so on. If you play for the second player, at first read an opponent’s move, then output your move, then read an opponent’s move again, and so on.

When your opponent loses, he outputs the single integer «-1» instead of his move. After reading this move, your program should terminate the execution.

Output

At the beginning of cooperation with the interactor, output the number of the player (1 or 2), for whom you want to play. Note that the interactor always plays optimally, and you must win to solve this problem.

Then output descriptions of your moves, one per line (the order of cooperation with the interactor is described in the input data format). After writing each line, do not forget to do “flush”.

Examples

stdin	stdout
abacaba	1 1 4 2 2 5 2 1 6 1 7 1 3 -1
pp	2 1 2 1 1 -1

Note

Output data for the examples contain full protocols of communication between the solution and the interactor, i.e. the first number is a number of a player, then moves of both players follow alternately.