Problem A. Binary Tree

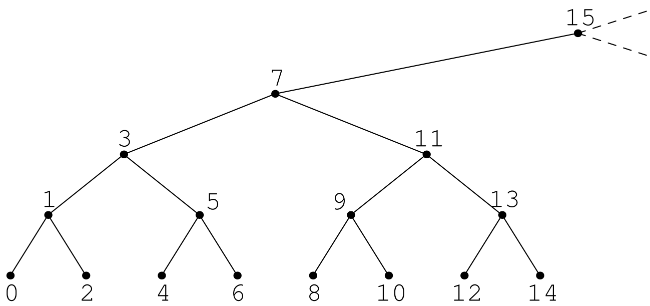
Input file:	`binary.in`
Output file:	`binary.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Never underestimate the power of two!

quote of **7ania7** on www.topcoder.com

In Speciality of Interval Trees and Heap (SITH) everyone knows that it is easy to enumerate vertices from top to bottom: root has a number of 1, its children are 2 and 3 and so on. It is really easy: you have to only divide vertex number by two to get its father's number.

But, as you know, everything is backwards in mathematics, and in this problem vertices are numbered from left to the right (see figure). Your task is very simple: you have to find sum of all numbers on vertices on a simple path from vertex a to vertex b. You can assume that root has a number of $55\,213\,970\,774\,324\,510\,299\,478\,046\,898\,216\,203\,619\,608\,871\,777\,363\,092\,441\,300\,193\,790\,394\,367$.



Input

First and only line of the input file will have two vertex numbers: a and b ($0 \leqslant a, b \leqslant 10^{15}$).

Output

The only line of output file should contain one number, the answer to the problem.

Examples

binary.in	binary.out
1 5	9
3 4	12

Problem B. Cars

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

Vasya is responsible for the annual competitions at IRCC (Institute of Remote Control Cars). The competition is held as follws. Beforehand, starting positions are chosen and zones are fixed. Each zone is a rectangle with sides parallel to the sides of the game area. Players' cars start from the starting positions and have to visit each zone.

The actual competition will be held tomorrow, so it's time for Vasya to carry out the testing. A single model car is available for this purpose. Vasya is going to drive the car sequentially from every starting position to every zone (if a starting position is inside some zone, movement is not needed).

Naturally, each time Vasya will choose the shortest path. It is known that it takes x^2 seconds for the model car to travel x meters (it needs time for acceleration and so on, though we won't tire you with physical details).

Your task is to calculate the number of seconds Vasya needs to carry out the whole testing (assume that he moves the car from a starting position to another and from a zone to a starting position immediately).

Input

The first line of input contains integer numbers n and k—the number of starting positions and zones, respectively ($1 \leqslant n, k \leqslant 100\,000$). The next n lines contain two integer numbers x_i and y_i each—the coordinates of starting positions. The next k lines contain four integer numbers each—the coordinates of bottom left and top right corners of the zones.

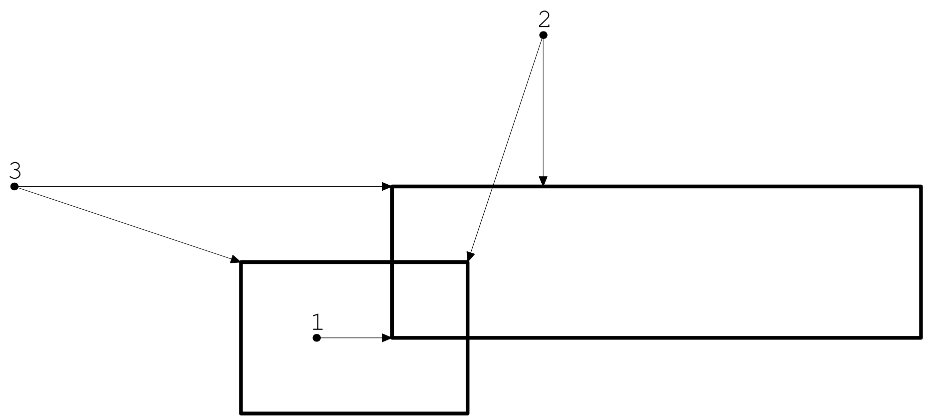
All coordinates do not exceed 10^6 by absolute value.

Output

Write the only integer number—the number of seconds required for the testing.

Example

cars.in	cars.out
3 2 2 2 5 6 -2 4 1 1 4 3 3 2 10 4	50



Problem C. Sign of Determinant

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

Research Institute of Linear Algebra (abbr. RILA) assigned Vasya a task of finding the sign of determinant of a very sparse square matrix. Fortunately, Vasya soon realized there's no more than one non-zero element in each column of the matrix. Unfortunately, the matrix is really huge! Help Vasya to write a program that will compute the sign of the determinant for him.

Recall two equivalent definitions of $\det A$, a *determinant* of a square matrix A of size $n \times n$: by row expansion and using permutations.

$$\textbf{1.}\ \det A = \sum_{i=1}^{n} (-1)^{1+i} a_{1,i} \det A_i^1,$$

where A_q^p is a $(n-1) \times (n-1)$ matrix that is a result of cutting p-th row and q-th column from matrix A (here, rows and columns are numbered starting from one).

$$\textbf{2.}\ \det A = \sum_{p \in S_n} (-1)^{N(p)} a_{1,p_1} a_{2,p_2} a_{3,p_3} \cdots a_{n,p_n},$$

where S_n is the group of all permutations of order n, and $N(p)$ is the number of inversions in p, which in turn is the number of pairs of indices i and j ($1 \leqslant i < j \leqslant n$) such that $p_i > p_j$.

Input

The first line of the input file contains two positive integers m and n where n is the size of the matrix ($1 \leqslant n \leqslant 5\,000\,000$). Next m lines contain the shortened matrix description: i-th of them contains four integers k_i, r_i, d_i and a_i ($1 \leqslant r_i \leqslant n$, $0 \leqslant d_i < n$, $|a_i| \leqslant 10^9$; it is guaranteed that the sum of all k_i is equal to n). The first of these lines gives the numbers in the first k_1 columns: one should put a_1 in each of the cells $(r_1, 1)$, $((r_1 + d_1) \overline{\bmod}\ n, 2)$, $((r_1 + 2d_1) \overline{\bmod}\ n, 3)$, $\ldots$, $((r_1 + (k_1 - 1)d_1) \overline{\bmod}\ n, k_1)$; the second one gives the numbers in the next k_2 columns: one should put a_2 in each of the cells $(r_2, k_1 + 1)$, $((r_2 + d_2) \overline{\bmod}\ n, k_1 + 2)$, $((r_2 + 2d_2) \overline{\bmod}\ n, k_1 + 3)$, $\ldots$, $((r_2 + (k_2 - 1)d_2) \overline{\bmod}\ n, k_1 + k_2)$; and so on. The last of these lines gives the numbers in the last k_m columns: one should put a_m in each of the cells $(r_m, n - k_m + 1)$, $((r_m + d_m) \overline{\bmod}\ n, n - k_m + 2)$, $((r_m + 2d_m) \overline{\bmod}\ n, n - k_m + 3)$, $\ldots$, $((r_m + (k_m - 1)d_m) \overline{\bmod}\ n, n)$. Here, the first number in parentheses is the row number, and the second one is the column number; $a \overline{\bmod}\, b$ means $((a-1) \bmod b) + 1$. All other cells of the matrix contain zeroes. It is guaranteed that the input size will not exceed 1 mebibyte.

Output

On the first line of the output file, write one symbol '`0`' if the determinant is zero, and its sign ('`+`' or '`-`') otherwise.

Examples

detsign.in	detsign.out
`1 30` `30 1 1 1`	`+`
`1 239` `239 1 1 -1`	`-`
`1 10` `10 1 0 1`	`0`

`2 2`
`1 1 0 0`
`1 2 0 1`
	`0`

Problem D. Roman Fraction

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

Vasya has an important project at the Research Institute of Given Strings (abbr. RIGS). He should develop an effective way of representing real numbers by strings that do not contain Arabic numerals. After quite some time spent thinking on the problem, Vasya decided to approximate real numbers by rational fractions which have numerator and denominator written in Roman numerals.

More precisely, right now Vasya wants to find a rational number $\frac{A}{B}$ on a segment $[\alpha, \beta]$ so that the following is true:

- numbers A and B are integers from the range $[1, 999\,999]$,
- the sum of lengths of A and B as strings is minimal possible.

In this problem, integers 1 through 999 are written thus: first goes the number of hundreds, then the number of tens, and after that the number of ones. Numbers 1 through 9 are written as `I`, `II`, `III`, `IV`, `V`, `VI`, `VII`, `VIII`, `IX`. Tens (10 through 90) are written as `X`, `XX`, `XXX`, `XL`, `L`, `LX`, `LXX`, `LXXX`, `XC`. Hundreds (100 through 900) are written as `C`, `CC`, `CCC`, `CD`, `D`, `DC`, `DCC`, `DCCC`, `CM`. If there is a zero in some decimal position, it is omitted.

Integers 1 through 999 999 are written as $\overline{S}T$ where S and T are strings representing numbers 1 through 999. The number itself is then $1000 \cdot num(S) + num(T)$ where num is the number being represented.

Example usage of Roman numerals:

- $\overline{XXX}CCXXXIX = 30\,239$
- $\overline{DCCCLXXXVIII} = 888\,000$
- $\overline{CMXCIX}CMXCIX = 999\,999$

Help Vasya write a program that solves this problem.

Input

The first line of the input file contains two real numbers α and β ($0 < \alpha \leqslant \beta < 1$). They are given with no more than nine digits after decimal point.

Output

The output file should contain one integer: the minimal sum of lengths of A and B in Roman numerals. If it is impossible to find such A and B, write "`IMPOSSIBLE`" instead.

Examples

fraction.in	fraction.out
`0.2 0.2`	`2`
`0.123456789 0.123456789`	`IMPOSSIBLE`
`0.1 0.9`	`2`

Problem E. Fair Division

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

Vasya works in Research Institute of Division of Edibles (abbr. RIDE). Once upon a time, his boss invited him to tea. There, they had a rectangular cake (size $W \times H$). The cake had two cherries on it, with coordinates (x_1, y_1) and (x_2, y_2) if the origin was placed in the lower left corner of the cake. To check Vasya's skill, the boss asked him to divide that cake in a fair way with one straight cut. Here, a fair division is one which divides the cake into two parts of equal area, and each of the parts has a cherry on it. One cannot cut through a cherry. Help Vasya!

Input

The first line of the input file contains two integers W and H ($2 \leqslant W, H \leqslant 10\,000$). The second line contains coordinates of the first cherry: integers x_1 and y_1 ($0 < x_1 < W$, $0 < y_1 < H$). Finally, the third line contains coordinates of the second cherry: integers x_2 and y_2 ($0 < x_2 < W$, $0 < y_2 < H$). It is guaranteed that the cherries are in different points of the cake.

Output

If there is a fair way to divide the cake, write "`YES`" on the first line of the output file. On the second and third lines, write four numbers in the range from -10^9 to 10^9, two numbers per line: the coordinates of two distinct points on the line of the required straight cut. If there is no fair way of division, write "`NO`" on the first line of the output file. It is guaranteed that if there exists a way to fairly divide the cake, there is also such a way that fits into output constraints.

Example

honest.in	honest.out
6 6 1 1 5 5	YES 3 0 3 1

Problem F. Juice

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

Vasya works as a household manager at JUICE (Juice Usage Institute at Central Europe). Unsurprisingly, one of his most common assignments is supplying certain departments of the institute with juice.

Vasya knows that juice is sold in boxes of $a_1, a_2, \ldots, a_n$ liters. There are k providers of juice. The provider number i sells each a_i-liter box at a price of $x_i a_j + y_i$ roubles. After careful investigation, Vasya also found out that provider i has no boxes per a_{q_i} liters (and all other boxes are available for order in any amount).

Help Vasya to choose a single provider to make a deal for exactly w liters of juice, minimizing the total cost.

Input

The first line of input contains integer numbers n, k and w—the number of juice boxes values, the number of provides and the exact number of liters to be purchased ($1 \leqslant n, k, w \leqslant 5\,000$). The following line contains n different integer numbers a_i—box volumes ($1 \leqslant a_i \leqslant 5\,000$). The next k lines contain three integer numbers x_i, y_i и q_i each—price coefficients and the number of lacking box volume ($0 \leqslant x_i, y_i \leqslant 10^4$, $1 \leqslant q_i \leqslant n$).

Output

In a case if it is impossible to buy exactly w liters of juice, output two zeroes.

Otherwise write two lines: first, containing the total cost of the deal and the number of the provider to make deal with, and second, containing n integer numbers—the amounts of boxes of each volume to be ordered. Providers are numbered from 1 to k in the order they are described in the input data.

If there are several optimal solutions, output any of them.

Examples

juice.in	juice.out
3 4 29 1 5 10 1 0 1 1 11 2 1 2 3 1 4 2	47 3 4 5 0
2 1 5 2 3 0 0 1	0 0

Problem G. Polar Fox

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

There are n polar foxes running on a circle around the north pole (with constant latitude). Initially, i-th fox is on longitude a_i and runs clockwise with a constant speed of v_i degrees per second.

There might be cases when a faster fox runs into another slower one. In that case, the first fox will eat the second one and continues to run with its initial speed. Foxes are very hugry and eating takes no time. After all such events, the remaining foxes will run with their speed until the end of the long polar day.

But before this happens, you have to catch as many foxes as you can, it's part of your work for RIFF (Research Institute of Furry Foxes). To accomplish this task, you should select a point on the circle and start to run from that point clockwise with your constant speed of v_{you} degrees per second. You cannot select a point with a fox—it's too dangerous. When you catch a fox, you put in into your bag.

You should leave the circle when either one of the foxes will tries to eat you (it is *very* painful), or there are no more foxes you can catch.

Input

First line of input file contains one positive integer n ($n \leqslant 100\,000$) and your speed v_{you} with exactly three digits after the decimal point ($0.001 \leqslant v_{you} \leqslant 360.000$). Each of the next n lines contains description of one fox. It consists of two numbers: the initial longitude a_i and speed v_i, both with exactly three digits after the decimal point ($0.001 \leqslant a_i, v_i \leqslant 360.000$). No two foxes start at the same point.

Output

Output exactly one integer: the maximal number of foxes you can catch. Remember, you may not enter circle in a point with a fox.

Examples

polarfox.in	polarfox.out
2 0.500 90.000 0.300 270.000 0.700	1
4 12.000 60.000 11.000 61.000 10.000 242.000 11.000 243.000 10.000	3

Note

In this problem, longitude is measured from east to west and can be from 0 to 360 degrees. So, a fox running with speed 1.0 degree/second could run in one second from point with longitude 30.0 to point with longitude 31.0 degrees.

Problem H. Rebus

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

Do you like rebuses? Vasya certainly does. No wonder—he's a researcher at Research Institute of Puzzles, Enigmas and Rebuses (abbr. RIPER). Recently, Vasya and his colleagues investigated an especially hard rebus. Its left part is a word of N ($N \leqslant 10^6$) letters. But suddenly the domesticated roach Petya came out and ate the right part of the rebus! All that is now left from it is that Vasya recalls it was divisible by k. Help Vasya find the minimal value the left part can hold.

Recall that rebus is the following type of puzzle. First, one writes down an equation consisting of non-negative integers. Then, each digit is substituted by a letter (same digits by same letters, different by different ones). To solve a rebus is to restore the digit values of the letters so that the equation holds once again. There can be no leading zeroes in the numbers in the rebus; the number 0 is written using exactly one digit.

Input

The first line of the input file contains a string consisting of uppercase Latin letters. The length of that string is from 1 to 10^6. The second line contains an integer k ($1 \leqslant k \leqslant 10^6$).

Output

Output a single number: the minimal value the left part can take. If there is no such value, output -1 instead.

Examples

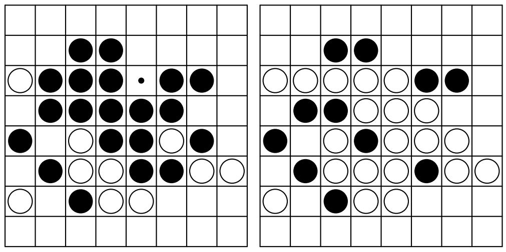
rebus.in	rebus.out
MAMA 3	1212
SPBSUCHAMPIONSHIPXXVI 1	-1
I 1	0

Problem I. Reversi

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

The game of Reversi is played on a 8×8 board with 64 pieces. Each piece has two sides, one white and one black. One player puts the pieces black side upwards, and the other one white side upwards.

At the beginning of the game, each player puts two pieces at the center of the board. Then players make moves in turns, starting from the black player. A move consists in putting a new piece in such a position that there exists at least one straight (horizontal, vertical, or diagonal) occupied line between the new piece and another piece of that player, with one or more opponent's pieces between them and without any gaps. After placing the piece, the player flips all such lines **simultaneously**, so that they become his own pieces.



If one player cannot make a valid move, play passes back to the other player. When neither player can move, the game ends. This occurs when the grid has filled up, or when one player has no more pieces on the board, or when neither player can legally place a piece in any of the remaining squares. However, a player cannot pass if he can make a valid move. The player with the most pieces on the board at the end of the game wins. If the number of pieces is the same for both players, the game ends in a draw.

Help Vasya at Research Institute of Board Games (RIBG) to write a program that will, given a position, determine who wins if both players play optimally.

Input

The first eight lines of the input file describe the current position in the game. Here, symbol '`W`' means a white piece, '`B`' a black one, and '`.`' denotes an empty cell. The ninth line contains one symbol, the color of the player who should make the next move. It is guaranteed that the number of empty cells on the board does not exceed 12.

Output

If the optimal game ends in a draw, write "`DRAW`". If the white player will win, write "`WHITE`". If the black player will win, write "`BLACK`".

Example

reversi.in	reversi.out

`..BBBBB.`
`W.BWBW.W`
`WWWWWWWW`
`WBBBBBWW`
`WBBWBBWW`
`WWWWWBWW`
`W.WBWW.W`
`..WBBW..`
`B`
	`BLACK`

Problem J. Sport

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

There are n sportsmen taking part in sports competitions held by Independent Department of Different Qualitative Defences (IDDQD). On the first day, they all stand in one line.

Too bad! Sportsmen should stand in the order of non-increasing height, and now they stand in some random order: the first one (from left) has height a_1, the second one has a_2, and so on. Nevertheless, there is a solution. No need to rearrange the sportsmen! Instead, they can be divided (virtually) into *several* lines.

For example, if four sportsmen with heights of 176, 174, 178, 168 centimeters stand in that order, one can say that there are two virtual lines with two sportsmen in each. And in this case, the heights of sportsmen in each of these lines will be in non-increasing order—just as planned!

Your task is to help organizers to calculate the number of such partitions. Every line in each partition must consist of several (at least one) sportsmen. Lines mustn't be mixed: if there are two sportsmen from one line, all sportsmen between them must be in the same line. In every line, sportsmen heights must be in non-increasing order from left to right. Every sportsman must belong to exactly one line.

Input

First line of input file will contain a non-negative integer n, the number of sportsmen ($n \leqslant 100\,000$). Second line will contain n integers $a_1, a_2, \ldots, a_n$: heights of sportsmen ($1 \leqslant a_i \leqslant 10^9$).

Output

The only line of output file should contain one integer, the answer to the problem.

Example

sport.in	sport.out
4 176 174 178 168	4

Problem K. Triangle

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

Vasya works in Terran Research Institute of Planar and Linear Entities (TRIPLE). Right now he must solve the following problem. Given n points on a plane, construct a triangle with vertices in three of these points such that its area is minimal possible, but greater than zero.

Input

First line of input file contains one positive integer n ($3 \leqslant n \leqslant 2\,000$). Each of the next n lines contains coordinates of one point in the format x_i y_i. All coordinates are integer and don't exceed 10^9 by absolute value.

Output

Output exactly one integer: minimal possible area. The error should be less than 0.1. It is guaranteed that there exists at least one triangle with positive area.

Example

triang.in	triang.out
4 0 0 10 0 5 10 5 6	10.0