Problem A. Binary strings

Input file:	`binstr.in`
Output file:	`binstr.out`

Eventually, Vasya has become a student of the Computer Science Department of a university. As you have guessed the first lecture was on the binary system. But since Vasya had known it from long ago, it seemed very boring to him. To kill this bore he wrote a binary string in his notebook: 1001011001. Then he started to write out all the prefixes of the mentioned string and next to them the suffixes of the same length.

1	1
10	01
100	001
1001	1001
10010	11001
100101	011001
1001011	1011001
10010110	01011001
100101100	001011001
1001011001	1001011001

It seemed very amusing to him because it possessed a very peculiar property. He noticed that there were only three "suffix-prefix equal" pairs (of length 1, 4 and 10). Naturally he was very much interested by the question: how many strings of length N are there with only one such pair. Surely this number can be pretty big, so it's satisfying for Vasya to know only the remainder of this number modulo P.

Input

Input file contains two integer numbers: N and P ($1 \leq N \leq 10^6$, $2 \leq P \leq 10^6$).

Output

Output file must contain only one nonnegative integer number — the answer.

Example

binstr.in	binstr.out
3 3	1

Problem B. Chords

Input file:	`chords.in`
Output file:	`chords.out`

Consider a circle. There are $2N$ different points marked on it and numbered with integer numbers in range from 1 to N so that each number from the interval has two points corresponding to it.

Points with the same numbers are connected by segments. Thus we've got N chords. Futhermore, the chords are numbered as well: the chord number "i" connects the two different points with number "i". Apparently some of the chords may intersect. Now it would be nice to find out for each single chord the number of chords it intersects.

Input

In the first line of the input file there is a single integer number N ($1 \leq N \leq 10^5$). In the next line there are $2N$ integers in range from 1 to N — the numbers assigned to the points in the order of traversal. Each number is written exactly twice. All the numbers in the line are separated with spaces.

Output

The output file is supposed to contain exactly N lines: on the i-th line write the number of chords that i-th chord intersects.

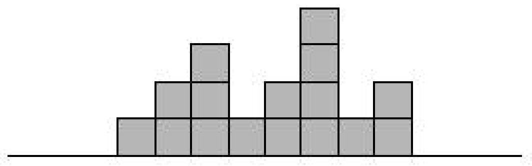
Example

chords.in	chords.out
5 1 2 3 1 4 2 5 5 3 4	2 3 3 2 0

Problem C. Cubes

Input file:	`cubes.in`
Output file:	`cubes.out`

Consider a wall built up of cubes of the same size: it can be represented by a sequence of n stacks numbered from 1 to n constructed of these cubes. The total number of cubes s is divisible by n. An example can be seen at the picture:



Let us define an elementary operation as follows: take up the topmost cube in the stack i ($1 \leq i \leq n$) and put it atop the stack j ($1 \leq j \leq n$). The cost of such an operation equals $g_{i,j}$. You task is to find out what is the minimum total cost of an elementary operation sequence that makes the wall *even*, i.e. such that in every stack there are exactly s/n cubes.

Input

In the first line of the input file you will find an integer n ($1 \leq n \leq 100$). The next line contains n integer numbers which correspond to the heights of the stacks of the wall (1^{st} number — the height of the 1^{st} stack, 2^{nd} number — the height of the 2^{nd} stack etc.) provided that $1 \leq s \leq 10^7$. The following n lines contain n numbers each: j-th number in the $(i+2)$-th line equals $g_{i,j}$ ($0 \leq g_{i,j} \leq 100$).

Output

Output the answer for the problem in the first line.

Example

cubes.in	cubes.out

4
1 3 2 2
1 1 1 1
1 1 2 4
1 2 3 4
2 3 4 5
	1

Problem D. Examinator

Input file:	`examinator.in`
Output file:	`examinator.out`

Recently Vasya had mastered counting, but his elder brother Petya didn't believe him and decided to check his skills in the following way: Petya took several cubes, each of which had an unique number written on it. Then he started building a tower out of them, placing only one cube on each tower level. While doing that he could insert cubes into arbitrary position of the tower or remove arbitrary cubes from it. As well, he would ask Vasya questions in the following format: how many cubes are there in the tower between cubes A and B inclusive, where A and B are the numbers written on some cubes, and Vasya, who observed the process, should give the replies immediately without getting stuck. Certainly Vasya did this task very easily. However, being hurt by the way Petya unbelieved him, he decided to switch with his elder brother and in his turn check Petya's counting skills with the same task on cubes. Moreover, since Petya was older, Vasya made a decision to make the task a bit more complex: while building the tower Petya would not be present in the room and would not be able to see the tower, and Vasya would make a comment on each of his actions. Having understood what was expected from him, Petya realized very soon that his counting skills did not cope with the problem and decided to ask you to help him.

Input

On the first line of the input file you will find a single integer number N — the number of actions by Vasya ($1 \leq N \leq 200000$). Next there will be N lines, describing the actions themselves. The first number in each of them will be the code of the operation to be executed ($0 \leq X \leq 2$). Consider each of the operations separately:

0 — ask how many cubes there are between cubes A and B inclusively in the tower (in this case there will be two more integers A and B in the line, denoting the numbers of cubes);

1 — insert cube A upon cube B in the tower (in this case there will be two more integers A and B in the line, denoting the numbers of cubes; it's guaranteed that cube A is absent while cube B is present in the tower);

2 — remove cube A from tower (in this case there will be one more integer A in the line, denoting the number of the cube; it's guaranteed that cube A is present in the tower).

The numbers of the cubes are integer numbers in range from 1 to 10^5. In operation 1 (cube insertion) B can be 0, that describes the case when the cube being inserted is put on the floor, and all the tower built stacks upon this cube.

All the numbers in the input file are separated by arbitrary number of spaces and line feeds.

Output

The output file must contain as many lines as many operations 0 were there in the input file. For each of those operations one integer number should be output — the answer to be given by Petya. In case when any (or both) of the cubes is absent in the tower, the output should be 0.

Example

examinator.in	examinator.out

11
0 3 5
1 18 0
1 13 0
1 15 13
0 13 18
0 18 13
1 25 13
1 1 0
2 15
2 13
0 1 25
	0
	3
	3
	2

Problem E. Fibonacci sum

Input file: fibsum.in
Output file: fibsum.out

Widely known Fibonacci number sequence is constructed as follows: $F_0 = 1$, $F_1 = 1$, $F_i = F_{i-2} + F_{i-1}$. At first glance it is very amusing and is not deprived of some interesting properties. But what makes a problemsetter worrying most is their thorough analysis that's been made so far by many and many outstanding researchers. For this reason we will not keep to the tradition of starting the sequence with 1, 1 and let it start with some arbitrary integer numbers. Even more, we will not sum only last two numbers to get next one — we will sum k last numbers. More formally, so-called "generalized Fibonacci sequence" is a sequence $\{F'_i\}$ where $F'_i = F'_{i-1} + F'_{i-2} + ... + F'_{i-k}$ for $i \geq k$, and F'_0, F'_1, ... F'_{k-1} are given integer numbers. Given a and b, you must find $F'_a + F'_{a+1} + ... + F'_{b-1} + F'_b$ modulo p.

Input

In the first line of the input file there is a single integer number N — the number of test cases in this file ($1 \leq N \leq 100$). Each test case consists of three lines:

— first line contains one integer number k ($1 \leq k \leq 10$);

— second line contains k integers: F'_0, F'_1, ... F'_{k-1} ($|F_i| \leq 10^9$);

— third line contains three integers: a, b and p ($0 \leq a \leq b \leq 10^{18}$, $2 \leq p \leq 10^6$).

All numbers in the input are separated by one or more spaces.

Output

For each test case output the answer on a single line.

Example

fibsum.in	fibsum.out

5
6
0 4 1 2 3 7
2 8 19
8
4 2 5 6 8 3 4 0
2 5 3
9
7 4 2 7 1 4 2 6 1
2 9 91
7
4 7 9 2 7 1 5
0 1 72
9
9 8 2 1 8 7 6 5 1
0 0 48
	14
	1
	57
	11
	9

Problem F. Foremen

Input file: `foremen.in`
Output file: `foremen.out`

A factory employs a staff of N ($1 \le N \le 100$) workers. The management has decided to split all of them into K brigades. In every brigade there will be elected a foreman — the leader of the brigade who apparently is a member of the brigade. A foreman in addition to his wages will receive a salary rise that depends on his skills and the size of his brigade. For every worker the management knows how much the rise will be if he becomes a foreman. Needless to say that the management wants, as it usually happens, to save some money. Pursuing this purpose they are trying to split all the workers into the brigades and elect the foremen so that the total rise amount (the sum of all rise amounts of all the foremen) is minimal possible. Unfortunately they are good neither in maths nor in backtracking. So you are the one to help them.

Input

In the first line of the input file there are two integers N and K ($1 \le K \le N$). Each of the following N lines contains N integers — the "rise" table. j-th number in the $(i+1)$-th line shows the rise that is required by the worker i provided that the size of his brigade is j (certainly, if he is the foreman of that brigade). The rise is an integer from the interval $[0, 1000]$. All the numbers in the lines are separated by spaces.

Output

In the first line of the output file write the minimal total rise amount. In the second line output K different numbers — i-th number should be the number of the worker who is elected to become the foreman in the i-th brigade. In the third line output N different numbers in range from 1 to K: i-th of them must be the number of the brigade to which i-th worker belongs.

Example

foremen.in	foremen.out

5 3
5 1 2 3 4
1 5 4 3 2
1 2 3 4 5
4 5 1 2 3
2 1 5 4 3
	3
	1 3 5
	1 3 2 1 3

Problem G. Game

Input file: `game.in`
Output file: `game.out`

Recently the widely known "Silly Board Games Inc." issued a new board game called "Ultimate chip". The game board looks like a stripe of N adjacent cells numbered 1 to N from left to right. The so-called main acting item of the game is a chip which may be driven along the board only rightwards. Each cell i has a number a_i prescribed to it — the maximum number of cells by which the chip can be moved from this cell. However, in a single move the chip must be advanced by at least K positions to the right. (Certainly, whenever $a_i < K$, the chip cannot be moved from this cell at all). Thus, from the cell number i the chip can be moved by any number of cells dx satisfying $K \le dx \le a_i$. The goal in the game is (as you may have guessed) to drive the chip to the last cell of the board (number N). Initially the chip is set on the first cell (number 1).

Input

The first line of the input file contains two integer numbers N and K separated by a space. Here N is the board size, K is the minimum number of cells by which the chip must be moved ($0 < N < 10^5, 0 < K \le N$). On the second line there are N numbers — a_i.

Output

If it is possible to drive the chip from the first cell to the last cell satisfying the restrictions specified, output "POSSIBLE" on the first line of the output file, otherwise write "IMPOSSIBLE". In case of positive answer, on the second line, output M — the number of moves necessary to drive the chip to the cell number N; in the following M lines output the moves themselves consequently — the integer pairs in the format "cell", "number of cells by which to move". If there are multiple solutions anyone is appropriate.

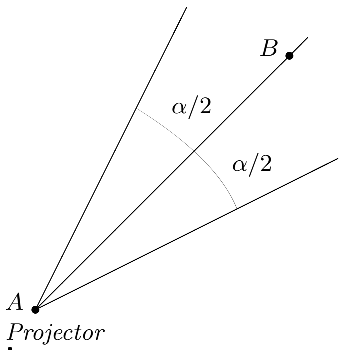
Example

game.in	game.out
5 2 3 1 2 4 1	POSSIBLE 2 1 2 3 2
5 3 3 1 2 4 1	IMPOSSIBLE

Problem H. Illumination

Input file: `illumination.in`
Output file: `illumination.out`

Recently a theater has bought N same floodlights for the scene illumination. The floodlights are installed in the way that they cannot be moved or turned around but we can adjust their illumination angle (on the picture: angle α, $a \le \alpha \le b$) provided that the angle is the same for all of the floodlights. The theater administration wishes that it's as bright on the scene as possible. However, the engineers are aware that whenever any positive square area is lit by all of the N floodlights it causes some unpleasant effects and irritates the spectators' eyes. You as one of the most famous professionals in solving "light" problems are to help them: find out the maximum illumination angle the floodlights will be set up for provided there is no positive square area lit by all of N floodlights simultaneously.



Input

In the first line of the input file there are three integers N, a, b — the number of floodlights, and the bounds for the illumination interval in degrees, correspondingly ($1 \le N \le 100, 1 \le a \le b \le 179$). In the following N lines there will be given 4 integer numbers in each: x_1, y_1, x_2, y_2 — the coordinates of the point where the corresponding floodlight is installed (point A on the picture) and the coordinates of another point that belongs to the ray issuing from the installation point and dividing the illumination interval in half (*bisector*) (point B on the picture) ($|x_i|, |y_i| \le 1000$). The length of the segment (x_1, y_1) — (x_2, y_2) is strictly greater than zero. All the numbers on the line are separated with spaces. Several floodlights can be installed in the same point.

Output

In the single line of the output file write out a real number — the maximum illumination angle with the accuracy of 10^{-3} if the problem has a solution and "IMPOSSIBLE" otherwise.

Example

illumination.in	illumination.out

2 1 179
0 0 0 3
3 3 3 0
	90.000000

2 1 179
0 0 3 0
3 0 0 0
	IMPOSSIBLE

Problem I. Interval

Input file: `interval.in`
Output file: `interval.out`

Given the interval $[L, R]$, the interval set is a set of numbers produced by the recurrence relation:

$$X = (Y + Z)/2,$$

provided the sum $(Y + Z)$ is even, and Y, Z are in the interval set. Initially the interval set consists of the numbers L and R. Given L and R compute the size of the interval set.

Input

Problem input contains multiple test cases in the form L and R on each line ($0 \le L, R \le 10^{18}$).

Output

For each interval $[L, R]$ print the cardinality of the interval set.

Example

interval.in	interval.out
0 1	2
0 2	3

Problem J. Matrix

Input file: `matrix.in`
Output file: `matrix.out`

Consider a 3×3 pattern-matrix. Each of its elements equals either 0, 1 or X. You are to find the number of binary $3 \times n$ matrices, which do not contain this pattern, modulo p. X acts as a wildcard, i.e. it can be either 0 or 1. (*Note*: the pattern cannot be rotated.)

Input

In the first line of the input file there are two numbers: n and p ($3 \le n \le 10^9, 3 \le p \le 10^6$). In the following three lines you will find the pattern-matrix. Each of them will contain exactly three elements. The elements are separated by a space.

Output

In the first line of the output file write the required number.

Example

matrix.in	matrix.out
3 83 1 X X X 0 1 1 X 0	81

Problem K. Strings

Input file: `strings.in`
Output file: `strings.out`

Given two strings S_1 and S_2 and a sequence of queries (i, j), where i is some position in S_1 and j is some position in S_2, output for each query the length of coinciding prefix of suffixes starting from i in S_1 and from j in S_2.

Input

On the first two lines you will find S_1 and S_2, which consist of the first 10 lowercase english letters. Their lengths (N_1 and N_2 for S_1 and S_2 correspondingly) do not exceed 10^5. On the third line there will be the number of queries N ($1 \le N \le 10^5$). The following N lines will contain pairs i, j ($1 \le i \le N_1, 1 \le j \le N_2$).

Output

Output N numbers — the answers for the queries. Separate numbers by spaces or line feeds.

Example

strings.in	strings.out
dabda dacda 2 1 1 5 2	2 1