CODEFORCES β
Sponsored by Telegram

acmASCIS Contest#4

A. Boots

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

A boot shop has received a shipment from the factory consisting of N left boots and N right boots. Each boot has some integer size, and a left and right boot will form a proper pair if they have equal sizes. Each boot can only belong to a single pair. The employees of the boot store want to create N proper pairs of boots. Fortunately, the factory has offered to exchange any number of boots in the shipment with new boots of different sizes.

Input
You are given an integer N (1 <= N <= 50). The next two lines each contains N elements not greater than 1000, the sizes of the left boots and right boots, respectively.

Output
Output the least number of boots that must be exchanged.

Sample test(s)

input
3 1 3 1 2 1 3
output
1

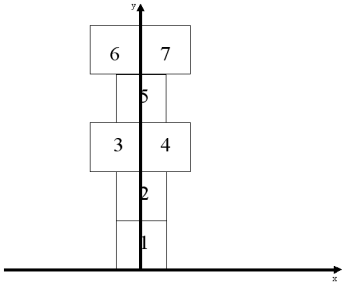
input
2 1 3 2 2
output
2

input
7 1 2 3 4 5 6 7 2 4 6 1 3 7 5
output
0

B. Hopscotch

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

So nearly half of the winter is over and Maria is dreaming about summer. She's fed up with skates and sleds, she was dreaming about Hopscotch all night long. It's a very popular children's game. The game field, the court, looks as is shown in the figure (all blocks are square and are numbered from bottom to top, blocks in the same row are numbered from left to right). Let us describe the hopscotch with numbers that denote the number of squares in the row, staring from the lowest one: 1-1-2-1-2-1-2-(1-2)..., where then the period is repeated (1-2).



The coordinate system is defined as shown in the figure. Side of all the squares are equal and have length a.

Maria is a very smart and clever girl, and she is concerned with quite serious issues: if she throws a stone into a point with coordinates (x, y), then will she hit some square? If the answer is positive, you are also required to determine the number of the square.

It is believed that the stone has fallen into the square if it is located **strictly** inside it. In other words a stone that has fallen on the square border is not considered a to hit a square.

Input

The only input line contains three integers: a, x, y, where a ($1 \le a \le 100$) is the side of the square, x and y ($-10^6 \le x \le 10^6, 0 \le y \le 10^6$) are coordinates of the stone.

Output

Print the number of the square, inside which the stone fell. If the stone is on a border of some stone or outside the court, print "-1" without the quotes.

Sample test(s)

input
1 0 0
output
-1

input
3 1 1
output
1

input
3 0 10
output
5

input
3 0 7
output
-1

input
3 4 0

output
-1

C. Valera and Antique Items

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Valera is a collector. Once he wanted to expand his collection with exactly one antique item.

Valera knows n sellers of antiques, the i-th of them auctioned k_i items. Currently the auction price of the j-th object of the i-th seller is s_{ij}. Valera gets on well with each of the n sellers. He is perfectly sure that if he outbids the current price of one of the items in the auction (in other words, offers the seller the money that is strictly greater than the current price of the item at the auction), the seller of the object will immediately sign a contract with him.

Unfortunately, Valera has only v units of money. Help him to determine which of the n sellers he can make a deal with.

Input

The first line contains two space-separated integers n, v $(1 \le n \le 50;\ 10^4 \le v \le 10^6)$ — the number of sellers and the units of money the Valera has.

Then n lines follow. The i-th line first contains integer k_i $(1 \le k_i \le 50)$ the number of items of the i-th seller. Then go k_i space-separated integers $s_{i1}, s_{i2}, ..., s_{ik_i}$ $(10^4 \le s_{ij} \le 10^6)$ — the current prices of the items of the i-th seller.

Output

In the first line, print integer p — the number of sellers with who Valera can make a deal.

In the second line print p space-separated integers $q_1, q_2, ..., q_p$ $(1 \le q_i \le n)$ — the numbers of the sellers with who Valera can make a deal. Print the numbers of the sellers **in the increasing order**.

Sample test(s)

input
3 50000 1 40000 2 20000 60000 3 10000 70000 190000
output
3 1 2 3

input
3 50000 1 50000 3 100000 120000 110000 3 120000 110000 120000
output
0

Note

In the first sample Valera can bargain with each of the sellers. He can outbid the following items: a 40000 item from the first seller, a 20000 item from the second seller, and a 10000 item from the third seller.

In the second sample Valera can not make a deal with any of the sellers, as the prices of all items in the auction too big for him.

D. Fixed-Point Theorem

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

The fixed-point theorem is one of those cornerstones of mathematics that reaches towards all disciplines, and oddly enough it is also closely related to the ability of any program to Quine itself (or to print out its own source code). Put simply, the fixed-point theorem states that with certain restrictions on a real-valued function F, there is always a point such that X=F(X). Taking the fixed-point theorem further, you can show that any function that meets certain restrictions will start to cycle through values if you keep on feeding it its own output (doing this with programs and their output is one way of producing programs that Quine themselves). One simple function that does this is the logistic function F(X)=R*X*(1-X) in the interval [0,1] for certain values of R. For example, if you start with the value X=.25 and feed it into F to get a new X, then feed that value into F to get yet another X, and so on, the values of X that are produced will converge to a small set of values that will eventually repeat forever, called a cycle. Your program will be given a double R between 0.1 and 3.569 inclusive. Starting with X=.25, generate the first 200,000 iterations of F using the given value of R, which will stabilize values of X. Then generate 1000 more values, and return the range of these values (highest value - lowest value). In other words, you will be finding the range of the values produced between iterations 200,001 and 201,000 inclusive.

Input
The input consists of one double R (0.1 <= R <= 3.569).

Output
Output the required range rounded to 4 decimal places.

Sample test(s)

input
0.1
output
0.0000

input
3.05
output
0.1475

input
3.4499
output
0.4176

E. Rounder

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

Most of the time when rounding a given number, it is customary to round to some multiple of a power of 10. However, there is no reason why we cannot use another multiple to do our rounding to. For example, you could round to the nearest multiple of 7, or the nearest multiple of 3. Given an int n and an int b, round n to the nearest value which is a multiple of b. If n is exactly halfway between two multiples of b, return the larger value.

Input

You are given two integers n(1 <= n <= 1000000) and b(2 <= b <= 500).

Output

Output n rounded to the nearest value which is a multiple of b. If n is exactly halfway between two multiples of b, output the larger value.

Sample test(s)

input
5 10
output
10

input
4 10
output
0

input
100 3
output
99

input
123456 7
output
123459