Problem A. Lightsabre

Input file:	`standard input`
Output file:	`standard output`
C++ time limit:	1 second
Memory limit:	256 megabytes

You are about to take part in a lightsabre duel! You have been granted Yoda's lightsabre as well as several pieces of armour of varying strengths. Unfortunately, the stronger the armour, the more it weighs. You are only allowed to take one piece of armour into the duel. You conclude that you want to take the lightest piece of armour such that Yoda's lightsabre cannot break through it. This way, your armour will be strong enough to avoid accidentally cutting yourself with the sabre whilst still light enough to wear.

You are given N armour pieces numbered 1 to N in order of strength (with 1 being the weakest and N the strongest). Using Yoda's lightsabre, you may try to cut through any armour piece. If you end up cutting through an armour piece, it breaks, and you know that armour piece would have been too weak to take with you. However you would like to break no more than M armour pieces in this process, and you would like to determine the armour of **minimum** strength such that you **cannot** cut through that armour. It is guaranteed such an armour piece exists.

Thinking about this problem, you realise it would be great to work out the **minimum** number of times you would need to try to cut an armour piece to solve this problem such that you are guaranteed you will never break more than M armour pieces. Given N and M, output the minimum number of times needed.

Input

Input consists of a single line containing two space-separated integers: N, M where $1 \leq N \leq 10^6$ and $1 \leq M \leq 10^6$.

Output

Output a single integer, denoting the answer to the problem.

Scoring

Subtask 1: (4 points) $N \leq 1\,000, M = 1$

Subtask 2: (6 points) $N \leq 1\,000, M = N$

Subtask 3: (18 points) $N \leq 10, M \leq N$

Subtask 4: (30 points) $N \leq 1\,000, M \leq 1\,000$

Subtask 5: (42 points) $N \leq 10^6, M \leq 10^6$

Examples

standard input	standard output
10 1	10
10 2	4
7 3	3

Note

In the first example given above, you must determine the solution without breaking more than 1 piece of armour. The only way to do this is by sequentially trying each piece of armour from strongest to weakest until you find the one which breaks.

In the second example, you may break at most 2 pieces of armour. This allows for a more efficient solution which only requires checking at most 4 pieces of armour.

Problem B. Coin flips

Input file:	`standard input`
Output file:	`standard output`
C++ time limit:	5 seconds
Memory limit:	256 megabytes

Compute the expected (average) score in the following game of chance. N coins are placed in a row. Each coin has the same probability P of coming up heads when flipped. Each coin has some value, not necessarily all the same. The score is initially zero, and the following steps are repeated until no coins are left.

1. Flip all the coins (keeping them in their original positions).
2. If they all came up tails, go back to step 1. Otherwise, identify the leftmost head. Call this coin C.
3. If C is neither the leftmost nor rightmost coin, add the product of the values of the two coins adjacent to C to your score.
4. Remove C from the row.

The answer can be shown to be a fraction, but the numerator and denominator may be very large, so the output will be given in a special form. Refer to the output format section for details.

Input

The first line of input contains N ($3 \leq N \leq 2000$), the number of coins, two integers s and t ($1 \leq s < t \leq 10000$), with $P = \frac{s}{t}$, and a prime number M ($\max\{2N+1, 2t\} < M \leq 10^9$), which is explained in the output section. This is followed by N lines each containing an integer a_i ($1 \leq a_i \leq 1000$), the values of the coins from left to right.

It is guaranteed that $\frac{M-1}{2}$ is also a prime.

Output

Let the expected score be $\frac{x}{y}$, where x and y have no common factors. Output the integer z such that $0 \leq z < M$ and $yz - x$ is a multiple of M (it can be shown that under the given constraints, exactly one such z exists). For example, if the expected score is $\frac{2}{3}$ and $M = 107$, then output 72 because $72 \cdot 3 - 2$ is a multiple of 107.

Scoring

Subtask 1 (10 points): $N = 3$ and $M \leq 10\,000$.

Subtask 2 (15 points): $N \leq 12$ and $M \leq 10\,000$.

Subtask 3 (10 points): All coins have the same value and $M \leq 10\,000$.

Subtask 4 (25 points): $N \leq 100$.

Subtask 5 (40 points): No further restrictions.

Examples

standard input	standard output
3 1 2 107 5 2 7	10
5 3 8 9887 5 6 3 2 8	8875

Problem C. Minesweeper

Input file:	`standard input`
Output file:	`standard output`
C++ time limit:	10 seconds
Memory limit:	1024 megabytes

Let's play Minesweeper! The game consists of an M by N rectangular board (M rows numbered top to bottom from 0 to $M-1$ and N columns numbered left to right from 0 to $N-1$) consisting of $M \times N$ cells. Some of these cells, initially unknown to you, contain mines. Your job is to determine which cells are empty of mines and which contain mines. In fact, this game can be found on your machine right now. Go ahead and try it out.

This is an interactive task. You interact with the grader by querying a cell on the board. If the cell does not contain a mine, you obtain the number of neighbouring cells which contain a mine (each cell has up to 8 neighbours). Else, if the cell contains a mine, you gain one penalty. The goal is to query **all** cells which do **not** contain mines whilst gaining as few penalties as possible.

Note that there is no limit on the number of penalties you may acquire during interaction with the grader, but your score exponentially decreases as you get more penalties (see **Scoring** below).

It is **guaranteed** that all cells along the boundaries of the rectangular board do **not** contain any mines. Furthermore, for any two cells which do not contain a mine, there will always exist a path from one cell to the other where each cell along the path does not contain any mines. We define a path as a sequence of cells $(a_1, a_2, \ldots, a_k)$ where the i-th cell is adjacent (either horizontally, vertically, or diagonally) to the $i+1$-th cell for all $1 \leq i < k$.

Input

The first line of input consists of two space-separated integers M, N, $(3 \leq M, N \leq 1\,000)$, after which your program is required to interact with the grader.

Interaction Protocol

After taking in one line of input containing M, N, you must output two space-separated integers `x y`, which denotes a cell in the 0-indexed position (x, y). You will then obtain a single integer. If your queried cell contains a mine, you get `-1`. Else, you get the number of neighbouring cells which contain mines.

The interaction is done by implementing two functions: `init` and `query`. The signatures for each function are as follows:

Python:

- `def init(M, N)`
- `def query(num_adj_mines)`

C++:

- `pair<int, int> init(int M, int N)`
- `pair<int, int> query(int num_adj_mines)`

Java:

- `public static int[] init(int M, int N)`
- `public static int[] query(int num_adj_mines)`

Template code for each language implementing the function signatures are given as attachments.

The `init` function will be called exactly once at the beginning of the interaction with the two integers M and N. The function should return your first cell query (x, y) (If C++, as a `pair<int, int>`, if Java as an array `int[]`).

The grader will then deliver the answer of your query by calling `query`. The function must then subsequently call the next cell you wish to query. Once all cells without mines have been queried, the interaction will automatically finish (no termination is required from your side).

Scoring

Let Q denote the total number of mines on the board and let L denote the number of penalties gained (i.e. L is the number of mines explicitly revealed during interaction with the grader.) Your partial score for test case c will then be: $\exp(-L/5)$ if $L < Q$. Otherwise, your score will be 0.

If P points are allocated to a particular subtask containing l test cases: $c_1, c_2, \ldots, c_l$, then your score S for that subtask is:

$$S = P * \min(c_1, c_2, \ldots, c_l)$$

Subtask 1 (3 points): $M, N \leq 4$

Subtask 2 (6 points): $M, N \leq 5$

Subtask 3 (11 points): There is only 1 mine.

Subtask 4 (15 points): No cell has more than 1 mine adjacent to it.

Subtask 5 (20 points): No cell has more than 3 mines adjacent to it.

Subtask 6 (20 points): $M, N \leq 50$

Subtask 7 (15 points): $M, N \leq 250$

Subtask 8 (10 points): No further constraints

Example

standard input	standard output
4 4 1 2 2 1 2 1 2 1 1 1 1 0 3	0 0 0 1 0 2 0 3 1 0 1 3 2 0 2 3 3 0 3 1 3 2 3 3 2 2

Note

In the example interaction given above, a 4 by 4 board is given. The user then queries the 12 boundary cells in reading order. Note that the query for cell $(3, 3)$ returns 0, thus one can deduce that cell $(2, 2)$ does not contain any mines. The 13th query is then done querying cell $(2, 2)$, after which the user obtains that 3 mines are adjacent to it. As there are only 3 cells left unqueried, these three cells must all contain mines and thus the user code terminates, finishing the game.