A. 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 $s_{i1}, s_{i2}, \ldots, s_{ik_i}$ ($10^4 \le s_{ij} \le 10^6$) — 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 $q_1, q_2, \ldots, q_p$ ($1 \le q_i \le n$) — the numbers of the sellers with who Valera can make a deal. Print the numbers of the sellers **in the increasing order**.

Examples

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.

B. Valera and Fruits

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

Valera loves his garden, where n fruit trees grow.

This year he will enjoy a great harvest! On the i-th tree b_i fruit grow, they will ripen on a day number a_i. Unfortunately, the fruit on the tree get withered, so they can only be collected on day a_i and day $a_i + 1$ (all fruits that are not collected in these two days, become unfit to eat).

Valera is not very fast, but there are some positive points. Valera is ready to work every day. In one day, Valera can collect no more than v fruits. The fruits may be either from the same tree, or from different ones. What is the maximum amount of fruit Valera can collect for all time, if he operates optimally well?

Input

The first line contains two space-separated integers n and v $(1 \le n, v \le 3000)$ — the number of fruit trees in the garden and the number of fruits that Valera can collect in a day.

Next n lines contain the description of trees in the garden. The i-th line contains two space-separated integers a_i and b_i $(1 \le a_i, b_i \le 3000)$ — the day the fruits ripen on the i-th tree and the number of fruits on the i-th tree.

Output

Print a single integer — the maximum number of fruit that Valera can collect.

Examples

input
2 3 1 5 2 3
output
8

input
5 10 3 20 2 20 1 20 4 20 5 20
output
60

Note

In the first sample, in order to obtain the optimal answer, you should act as follows.

- On the first day collect 3 fruits from the 1-st tree.
- On the second day collect 1 fruit from the 2-nd tree and 2 fruits from the 1-st tree.
- On the third day collect the remaining fruits from the 2-nd tree.

In the second sample, you can only collect 60 fruits, the remaining fruit will simply wither.

C. Valera and Tubes

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

Valera has got a rectangle table consisting of n rows and m columns. Valera numbered the table rows starting from one, from top to bottom and the columns – starting from one, from left to right. We will represent cell that is on the intersection of row x and column y by a pair of integers (x, y).

Valera wants to place exactly k tubes on his rectangle table. A tube is such sequence of table cells (x_1, y_1), (x_2, y_2), ..., (x_r, y_r), that:

- $r \geq 2$;
- for any integer i $(1 \leq i \leq r - 1)$ the following equation $|x_i - x_{i+1}| + |y_i - y_{i+1}| = 1$ holds;
- each table cell, which belongs to the tube, must occur exactly once in the sequence.

Valera thinks that the tubes are arranged in a fancy manner if the following conditions are fulfilled:

- no pair of tubes has common cells;
- each cell of the table belongs to some tube.

Help Valera to arrange k tubes on his rectangle table in a fancy manner.

Input

The first line contains three space-separated integers n, m, k $(2 \leq n, m \leq 300; 2 \leq 2k \leq n \cdot m)$ — the number of rows, the number of columns and the number of tubes, correspondingly.

Output

Print k lines. In the i-th line print the description of the i-th tube: first print integer r_i (the number of tube cells), then print $2r_i$ integers $x_{i1}, y_{i1}, x_{i2}, y_{i2}, \ldots, x_{ir_i}, y_{ir_i}$ (the sequence of table cells).

If there are multiple solutions, you can print any of them. It is guaranteed that at least one solution exists.

Examples

input
3 3 3

output
3 1 1 1 2 1 3 3 2 1 2 2 2 3 3 3 1 3 2 3 3

input
2 3 1

output
6 1 1 1 2 1 3 2 3 2 2 2 1

Note

Picture for the first sample:

Picture for the second sample:

D. Valera and Swaps

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

A *permutation* p of length n is a sequence of distinct integers $p_1, p_2, \ldots, p_n$ $(1 \le p_i \le n)$. A permutation is an identity permutation, if for any i the following equation holds $p_i = i$.

A *swap* (i, j) is the operation that swaps elements p_i and p_j in the permutation. Let's assume that $f(p)$ is the minimum number of swaps that you need to make the permutation p an identity permutation.

Valera wonders, how he can transform permutation p into any permutation q, such that $f(q) = m$, using the minimum number of swaps. Help him do that.

Input

The first line contains integer n $(1 \le n \le 3000)$ — the length of permutation p. The second line contains n distinct integers $p_1, p_2, \ldots, p_n$ $(1 \le p_i \le n)$ — Valera's initial permutation. The last line contains integer m $(0 \le m < n)$.

Output

In the first line, print integer k — the minimum number of swaps.

In the second line, print $2k$ integers $x_1, x_2, \ldots, x_{2k}$ — the description of the swap sequence. The printed numbers show that you need to consecutively make swaps (x_1, x_2), (x_3, x_4), ..., (x_{2k-1}, x_{2k}).

If there are multiple sequence swaps of the minimum length, print the lexicographically minimum one.

Examples

input
5 1 2 3 4 5 2

output
2 1 2 1 3

input
5 2 1 4 5 3 2

output
1 1 2

Note

Sequence $x_1, x_2, \ldots, x_s$ is lexicographically smaller than sequence $y_1, y_2, \ldots, y_s$, if there is such integer r $(1 \le r \le s)$, that $x_1 = y_1, x_2 = y_2, \ldots, x_{r-1} = y_{r-1}$ and $x_r < y_r$.

E. Valera and Number

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

Valera is a coder. Recently he wrote a funny program. The pseudo code for this program is given below:

```
//input: integers x, k, p
a = x;
for(step = 1; step <= k; step = step + 1){
   rnd = [random integer from 1 to 100];
   if(rnd <= p)
      a = a * 2;
   else
      a = a + 1;
}

s = 0;

while(remainder after dividing a by 2 equals 0){
   a = a / 2;
   s = s + 1;
}
```

Now Valera wonders: given the values x, k and p, what is the expected value of the resulting number s?

Input

The first line of the input contains three integers x, k, p ($1 \le x \le 10^9$; $1 \le k \le 200$; $0 \le p \le 100$).

Output

Print the required expected value. Your answer will be considered correct if the absolute or relative error doesn't exceed 10^{-6}.

Examples

input
1 1 50
output
1.0000000000000

input
5 3 0
output
3.0000000000000

input
5 3 25
output
1.9218750000000

Note

If the concept of expected value is new to you, you can read about it by the link:

http://en.wikipedia.org/wiki/Expected_value