Codeforces Round #277.5 (Div. 2)

A. SwapSort

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

In this problem your goal is to sort an array consisting of n integers in at most n swaps. For the given array find the sequence of swaps that makes the array sorted in the non-descending order. Swaps are performed consecutively, one after another.

Note that in this problem you do not have to minimize the number of swaps — your task is to find any sequence that is no longer than n.

Input

The first line of the input contains integer n ($1 \le n \le 3000$) — the number of array elements. The second line contains elements of array: $a_0, a_1, ..., a_{n-1}$ ($-10^9 \le a_i \le 10^9$), where a_i is the i-th element of the array. The elements are numerated from 0 to $n - 1$ from left to right. Some integers may appear in the array more than once.

Output

In the first line print k ($0 \le k \le n$) — the number of swaps. Next k lines must contain the descriptions of the k swaps, one per line. Each swap should be printed as a pair of integers i, j ($0 \le i, j \le n - 1$), representing the swap of elements a_i and a_j. You can print indices in the pairs in any order. The swaps are performed in the order they appear in the output, from the first to the last. It is allowed to print $i = j$ and swap the same pair of elements multiple times.

If there are multiple answers, print any of them. It is guaranteed that at least one answer exists.

Sample test(s)

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

input
6 10 20 20 40 60 60
output
0

input
2 101 100
output
1 0 1

B. BerSU Ball

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

The Berland State University is hosting a ballroom dance in celebration of its 100500-th anniversary! n boys and m girls are already busy rehearsing waltz, minuet, polonaise and quadrille moves.

We know that several boy&girl pairs are going to be invited to the ball. However, the partners' dancing skill in each pair must differ by at most one.

For each boy, we know his dancing skills. Similarly, for each girl we know her dancing skills. Write a code that can determine the largest possible number of pairs that can be formed from n boys and m girls.

Input

The first line contains an integer n ($1 \le n \le 100$) — the number of boys. The second line contains sequence $a_1, a_2, ..., a_n$ ($1 \le a_i \le 100$), where a_i is the i-th boy's dancing skill.

Similarly, the third line contains an integer m ($1 \le m \le 100$) — the number of girls. The fourth line contains sequence $b_1, b_2, ..., b_m$ ($1 \le b_j \le 100$), where b_j is the j-th girl's dancing skill.

Output

Print a single number — the required maximum possible number of pairs.

Sample test(s)

input
4 1 4 6 2 5 5 1 5 7 9
output
3

input
4 1 2 3 4 4 10 11 12 13
output
0

input
5 1 1 1 1 1 3 1 2 3
output
2

C. Given Length and Sum of Digits...

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

You have a positive integer m and a non-negative integer s. Your task is to find the smallest and the largest of the numbers that have length m and sum of digits s. The required numbers should be non-negative integers written in the decimal base without leading zeroes.

Input

The single line of the input contains a pair of integers m, s ($1 \le m \le 100, 0 \le s \le 900$) — the length and the sum of the digits of the required numbers.

Output

In the output print the pair of the required non-negative integer numbers — first the minimum possible number, then — the maximum possible number. If no numbers satisfying conditions required exist, print the pair of numbers "`-1 -1`" (without the quotes).

Sample test(s)

input
2 15
output
69 96

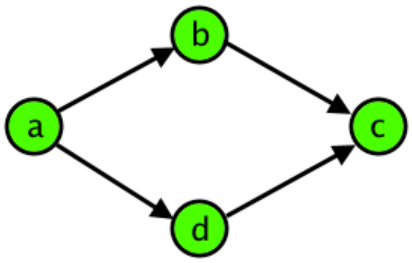
input
3 0
output
-1 -1

D. Unbearable Controversy of Being

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

Tomash keeps wandering off and getting lost while he is walking along the streets of Berland. It's no surprise! In his home town, for any pair of intersections there is exactly one way to walk from one intersection to the other one. The capital of Berland is very different!

Tomash has noticed that even simple cases of ambiguity confuse him. So, when he sees a group of four distinct intersections a, b, c and d, such that there are two paths from a to c — one through b and the other one through d, he calls the group a "damn rhombus". Note that pairs (a, b), (b, c), (a, d), (d, c) should be directly connected by the roads. Schematically, a damn rhombus is shown on the figure below:



Other roads between any of the intersections don't make the rhombus any more appealing to Tomash, so the four intersections remain a "damn rhombus" for him.

Given that the capital of Berland has n intersections and m roads and all roads are unidirectional and are known in advance, find the number of "damn rhombi" in the city.

When rhombi are compared, the order of intersections b and d doesn't matter.

Input

The first line of the input contains a pair of integers n, m $(1 \le n \le 3000, 0 \le m \le 30000)$ — the number of intersections and roads, respectively. Next m lines list the roads, one per line. Each of the roads is given by a pair of integers a_i, b_i $(1 \le a_i, b_i \le n; a_i \ne b_i)$ — the number of the intersection it goes out from and the number of the intersection it leads to. Between a pair of intersections there is at most one road in each of the two directions.

It is not guaranteed that you can get from any intersection to any other one.

Output

Print the required number of "damn rhombi".

Sample test(s)

input
5 4 1 2 2 3 1 4 4 3

output
1

input
4 12 1 2 1 3 1 4 2 1 2 3 2 4 3 1 3 2 3 4 4 1 4 2 4 3

output
12

E. Hiking

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

A traveler is planning a water hike along the river. He noted the suitable rest points for the night and wrote out their distances from the starting point. Each of these locations is further characterized by its *picturesqueness*, so for the i-th rest point the distance from the start equals x_i, and its *picturesqueness* equals b_i. The traveler will move down the river in one direction, we can assume that he will start from point 0 on the coordinate axis and rest points are points with coordinates x_i.

Every day the traveler wants to cover the distance l. In practice, it turns out that this is not always possible, because he needs to end each day at one of the resting points. In addition, the traveler is choosing between two desires: cover distance l every day and visit the most picturesque places.

Let's assume that if the traveler covers distance r_j in a day, then he feels frustration $\sqrt{|r_j - l|}$, and his total frustration over the hike is calculated as the total frustration on all days.

Help him plan the route so as to minimize the *relative total frustration*: the total frustration divided by the total picturesqueness of all the rest points he used.

The traveler's path must end in the farthest rest point.

Input

The first line of the input contains integers n, l $(1 \le n \le 1000, 1 \le l \le 10^5)$ — the number of rest points and the optimal length of one day path.

Then n lines follow, each line describes one rest point as a pair of integers x_i, b_i $(1 \le x_i, b_i \le 10^6)$. No two rest points have the same x_i, the lines are given in the order of strictly increasing x_i.

Output

Print the traveler's path as a sequence of the numbers of the resting points he used in the order he used them. Number the points from 1 to n in the order of increasing x_i. The last printed number must be equal to n.

Sample test(s)

input
5 9 10 10 20 10 30 1 31 5 40 10

output
1 2 4 5

Note

In the sample test the minimum value of *relative total frustration* approximately equals 0.097549. This value can be calculated as $(1 + 1 + \sqrt{2} + 0)/(10 + 10 + 5 + 10)$.

F. Special Matrices

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

An $n \times n$ square matrix is *special*, if:

- it is binary, that is, each cell contains either a 0, or a 1;
- the number of ones in each row and column equals 2.

You are given n and the first m rows of the matrix. Print the number of special $n \times n$ matrices, such that the first m rows coincide with the given ones.

As the required value can be rather large, print the remainder after dividing the value by the given number mod.

Input

The first line of the input contains three integers n, m, mod ($2 \le n \le 500$, $0 \le m \le n$, $2 \le mod \le 10^9$). Then m lines follow, each of them contains n characters — the first rows of the required special matrices. Each of these lines contains exactly two characters '1', the rest characters are '0'. Each column of the given $m \times n$ table contains at most two numbers one.

Output

Print the remainder after dividing the required value by number mod.

Sample test(s)

input
3 1 1000 011

output
2

input
4 4 100500 0110 1010 0101 1001

output
1

Note

For the first test the required matrices are:

```
011
101
110

011
110
101
```

In the second test the required matrix is already fully given, so the answer is 1.