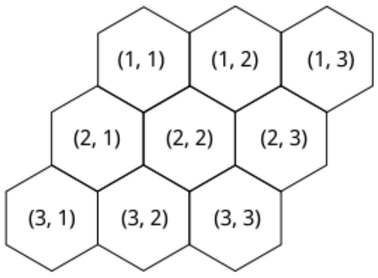
Problem A. Bees

Input filename:	`bees.in`
Output filename:	`bees.out`
Time limit:	1 second
Memory limit:	256 Mb

Honeyland is a small honey-producing country, inhabited by bees, and consisting of honeycombs. Specifically, there are n rows of m honeycombs each. Every honeycomb is a partially autonomous subject of the Honey Federation, and there is a hot topic of the connection between them. To resolve the existing problems, the government decided to dig a network of honey rivers, connecting all federation subjects. One such river connects the two honeycombs that have it as their side. To construct such a river, a lot of honey is needed, and the honey is the result of the hard work of regular working-class bees. The government, therefore, wants to minimize the amount of honey used. Unfortunately, bees are hardworking but not particularly analytically-skilled from nature, so they cannot solve this problem themselves. But You can!

The amount of honey necessary to construct a river between two neighboring federation subjects is $(x_1x_2 + p_1y_1y_2) \pmod{p_2}$, where x_1, y_1, x_2, y_2 are the coordinates of first and second subjects respectively, and p_1, p_2 are the given prime numbers. The first coordinate denotes the row number, and the second denotes the honeycomb number in the corresponding row. Rows are indexed top to bottom, and honeycombs are indexed left to right. The upper left corner of the Honeyland, therefore, has the coordinates $(1, 1)$. In the picture below, a map of the Federation for $n - 3$, $m = 3$ is shown:



Input file format

Input file contains four integers: $1 \leq n, m, p_1, p_2 \leq 1000$. It is guaranteed that numbers p_1 and p_2 are prime.

Output file format

Output the single integer — the minimal amount of honey necessary to connect all the federation subjects.

Sample tests

bees.in	bees.out
3 3 11 23	28

Problem B. Maximum in a Cycle

Input filename:	`maxincycle.in`
Output filename:	`maxincycle.out`
Time limit:	2 seconds
Memory limit:	256 Mb

Many of us are not particularly satisfied with the quality of the road, and the professor of FiGA (Falsificational and Geospace Academy) is not much different. Specifically, he assigns a unique value of *badness* to every road he knows.

Once upon the time, wandering after 2 a.m., the professor came up with a definition of an *awful* road: the road is called awful if there is a circular path, in which this road has the highest value of badness.

The professor next got terrified by the prospect that all the roads are awful, so he decided to go to bed instead and redirected this task to the programmers.

Input file format

The first line contains two non-negative integers n and m — the number of intersections, and the number of roads respectively, $n, m \leq 10^5$. The following m lines contain the description of the bidirectional roads in the format of pair of numbers — the indices of the intersections that this road connects. Intersections are indexed with integers from 1 to n, and the roads are given in the increasing order of badness.

It is guaranteed, that every intersection is reachable from every other intersection and that there is no road connecting an intersection to itself.

Output file format

On the first line of the output file print the number of roads, which are not awful. On the next line, print the space-separated numbers of these roads. The roads are indexed 1 to m in the input order.

Sample tests

`maxincycle.in`	`maxincycle.out`

4 6
1 2
2 1
2 3
2 3
3 4
4 3
	3
	1 3 5

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

Problem C. Connect Points

Input filename:	`connect.in`
Output filename:	`connect.out`
Time limit:	2 seconds
Memory limit:	256 Mb

You are given N points in the Euclidean plane. You want to connect some pairs of these points by straight segments in such a way that (i) all points become connected (reachable from each other via the selected segments) and (ii) the total distances of the segments used is minimal.

Input file format

The first line of the input file contains the number $1 \leq N \leq 200$ of points. The following N lines contain the space-separated integer coordinates $-1000 \leq X_i, Y_i \leq 1000$ of the i-th point. No two points coincide, and no three are colinear.

Output file format

On the first line of the output file, print the total length L of the selected segments, with the precision of at least 6 decimal digits after the decimal point. On the second line print the number K of the selected segments. On the next K lines, print two space-separated numbers A_j, B_j, denoting the endpoint indices of the j-th selected edge. Points are indexed from 1 to N in the input order. If there are multiple answers with the same value of L then you can output any of them.

Sample tests

connect.in	connect.out

4
0 0
0 1
1 0
1 1
	3
	3
	1 2
	2 4
	4 3

5
0 0
0 2
1 1
3 0
3 2
	7.064495
	4
	3 1
	3 2
	3 4
	4 5

Problem D. Union Day

Input filename:	`unionday.in`
Output filename:	`unionday.out`
Time limit:	2 seconds
Memory limit:	256 Mb

In Byteland, there are n cities, but not a single road. King Waldemar de Béart decided to solve this issue and connect several pairs of cities by roads in such a way that all the cities become connected (reachable from each other). When the construction completes, the king plans to celebrate Union Day. Unfortunately, road construction is by no means cheap, and the Byteland's treasury is on its last legs. The king, therefore, wants to minimize the total length of all constructed roads.

Input file format

The first line of the input file contains the number $1 \leq n \leq 5000$ of cities in the Byteland. The following n lines contain two integer coordinates $-10000 \leq x_i, y_i \leq 10000$ of the i-th city. It is guaranteed that no two cities have the same location.

Output file format

On the first line of the output file print the minimal total road length. Output precision has to be at least 10^{-3}.

Sample tests

unionday.in	unionday.out

6
1 1
7 1
2 2
6 2
1 3
7 3
	9.65685