Codeforces Round #170 (Div. 1)

A. Learning Languages

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

The "BerCorp" company has got n employees. These employees can use m approved official languages for the formal correspondence. The languages are numbered with integers from 1 to m. For each employee we have the list of languages, which he knows. This list could be empty, i. e. an employee may know no official languages. But the employees are willing to learn any number of official languages, as long as the company pays their lessons. A study course in one language for one employee costs 1 berdollar.

Find the minimum sum of money the company needs to spend so as any employee could correspond to any other one (their correspondence can be indirect, i. e. other employees can help out translating).

Input

The first line contains two integers n and m ($2 \le n, m \le 100$) — the number of employees and the number of languages.

Then n lines follow — each employee's language list. At the beginning of the i-th line is integer k_i ($0 \le k_i \le m$) — the number of languages the i-th employee knows. Next, the i-th line contains k_i integers — a_{ij} ($1 \le a_{ij} \le m$) — the identifiers of languages the i-th employee knows. It is guaranteed that all the identifiers in one list are distinct. Note that an employee may know zero languages.

The numbers in the lines are separated by single spaces.

Output

Print a single integer — the minimum amount of money to pay so that in the end every employee could write a letter to every other one (other employees can help out translating).

Sample test(s)

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

input
8 7 0 3 1 2 3 1 1 2 5 4 2 6 7 1 3 2 7 4 1 1
output
2

input
2 2 1 2 0
output
1

Note

In the second sample the employee 1 can learn language 2, and employee 8 can learn language 4.

In the third sample employee 2 must learn language 2.

B. Set of Points

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

Convexity of a set of points on the plane is the size of the largest subset of points that form a convex polygon. Your task is to build a set of n points with the convexity of exactly m. Your set of points should not contain three points that lie on a straight line.

Input

The single line contains two integers n and m ($3 \le m \le 100$, $m \le n \le 2m$).

Output

If there is no solution, print "-1". Otherwise, print n pairs of integers — the coordinates of points of any set with the convexity of m. The coordinates shouldn't exceed 10^8 in their absolute value.

Sample test(s)

input
4 3

output
0 0 3 0 0 3 1 1

input
6 3

output
-1

input
6 6

output
10 0 -10 0 10 1 9 1 9 -1 0 -2

input
7 4

output
176166 6377 709276 539564 654734 174109 910147 434207 790497 366519 606663 21061 859328 886001

C. Game

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

Two players play the following game. Initially, the players have a knife and a rectangular sheet of paper, divided into equal square grid cells of unit size. The players make moves in turn, the player who can't make a move loses. In one move, a player can take the knife and cut the paper along any segment of the grid line (not necessarily from border to border). The part of the paper, that touches the knife at least once, is considered cut. There is one limit not to turn the game into an infinite cycle: each move has to cut the paper, that is the knife has to touch the part of the paper that is not cut before.

Obviously, the game ends when the entire sheet is cut into 1×1 blocks. During the game, the pieces of the sheet are not allowed to move. It is also prohibited to cut along the border. The coordinates of the ends of each cut must be integers.

You are given an $n \times m$ piece of paper, somebody has already made k cuts there. Your task is to determine who will win if the players start to play on this sheet. You can consider that both players play optimally well. If the first player wins, you also need to find the winning first move.

Input

The first line contains three integers n, m, k ($1 \le n, m \le 10^9, 0 \le k \le 10^5$) — the sizes of the piece of paper and the number of cuts. Then follow k lines, each containing 4 integers xb_i, yb_i, xe_i, ye_i ($0 \le xb_i, xe_i \le n, 0 \le yb_i, ye_i \le m$) — the coordinates of the ends of the existing cuts.

It is guaranteed that each cut has a non-zero length, is either vertical or horizontal and doesn't go along the sheet border.

The cuts may intersect, overlap and even be the same. That is, it is not guaranteed that the cuts were obtained during any correct game.

Output

If the second player wins, print "SECOND". Otherwise, in the first line print "FIRST", and in the second line print any winning move of the first player (the coordinates of the cut ends, follow input format to print them).

Sample test(s)

input

```
2 1 0
```

output

```
FIRST
1 0 1 1
```

input

```
2 2 4
0 1 2 1
0 1 2 1
1 2 1 0
1 1 1 2
```

output

```
SECOND
```

D. Google Code Jam

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

Many of you must be familiar with the Google Code Jam round rules. Let us remind you of some key moments that are crucial to solving this problem. During the round, the participants are suggested to solve several problems, each divided into two subproblems: an easy one with small limits (Small input), and a hard one with large limits (Large input). You can submit a solution for Large input only after you've solved the Small input for this problem. There are no other restrictions on the order of solving inputs. In particular, the participant can first solve the Small input, then switch to another problem, and then return to the Large input. Solving each input gives the participant some number of points (usually different for each problem). This takes into account only complete solutions that work correctly on all tests of the input. The participant gets the test result of a Small input right after he submits it, but the test result of a Large input are out only after the round's over. In the final results table the participants are sorted by non-increasing of received points. If the points are equal, the participants are sorted by ascending of time penalty. By the Google Code Jam rules the time penalty is the **time when the last correct solution was submitted**.

Vasya decided to check out a new tactics on another round. As soon as the round begins, the boy quickly read all the problems and accurately evaluated the time it takes to solve them. Specifically, for each one of the n problems Vasya knows five values:

- Solving the Small input of the i-th problem gives to the participant $scoreSmall_i$ points, and solving the Large input gives $scoreLarge_i$ more points. That is, the maximum number of points you can get for the i-th problem equals $scoreSmall_i + scoreLarge_i$.
- Writing the solution for the Small input of the i-th problem takes exactly $timeSmall_i$ minutes for Vasya. Improving this code and turning it into the solution of the Large input takes another $timeLarge_i$ minutes.
- Vasya's had much practice, so he solves all Small inputs from the first attempt. But it's not so easy with the Large input: there is the $probFail_i$ probability that the solution to the Large input will turn out to be wrong at the end of the round. Please keep in mind that these solutions do not affect the participants' points and the time penalty.

A round lasts for t minutes. The time for reading problems and submitting solutions can be considered to equal zero. Vasya is allowed to submit a solution exactly at the moment when the round ends.

Vasya wants to choose a set of inputs and the order of their solution so as to make the expectation of the total received points maximum possible. If there are multiple ways to do this, he needs to minimize the expectation of the time penalty. Help Vasya to cope with this problem.

Input

The first line contains two integers n and t ($1 \le n \le 1000$, $1 \le t \le 1560$). Then follow n lines, each containing 5 numbers:
$scoreSmall_i$, $scoreLarge_i$, $timeSmall_i$, $timeLarge_i$, $probFail_i$
($1 \le scoreSmall_i, scoreLarge_i \le 10^9$, $1 \le timeSmall_i, timeLarge_i \le 1560$, $0 \le probFail_i \le 1$).

$probFail_i$ are real numbers, given with at most 6 digits after the decimal point. All other numbers in the input are integers.

Output

Print two real numbers — the maximum expectation of the total points and the corresponding minimum possible time penalty expectation. The answer will be considered correct if the absolute or relative error doesn't exceed 10^{-9}.

Sample test(s)

input
3 40 10 20 15 4 0.5 4 100 21 1 0.99 1 4 1 1 0.25
output
24.0 18.875

input
1 1 100000000 200000000 1 1 0
output
100000000 1

Note

In the first sample one of the optimal orders of solving problems is:

1. The Small input of the third problem.
2. The Small input of the first problem.
3. The Large input of the third problem.
4. The Large input of the first problem.

Note that if you solve the Small input of the second problem instead of two inputs of the third one, then total score expectation will be the same but the time penalty expectation will be worse (38).

E. Binary Tree on Plane

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A root tree is a directed acyclic graph that contains one node (root), from which there is exactly one path to any other node.

A root tree is binary if each node has at most two outgoing arcs.

When a binary tree is painted on the plane, all arcs should be directed from top to bottom. That is, each arc going from u to v must meet the condition $y_u > y_v$.

You've been given the coordinates of all tree nodes. Your task is to connect these nodes by arcs so as to get the binary root tree and make the total length of the arcs minimum. All arcs of the built tree must be directed from top to bottom.

Input

The first line contains a single integer n $(2 \le n \le 400)$ — the number of nodes in the tree. Then follow n lines, two integers per line: x_i, y_i $(|x_i|, |y_i| \le 10^3)$ — coordinates of the nodes. It is guaranteed that all points are distinct.

Output

If it is impossible to build a binary root tree on the given points, print "-1". Otherwise, print a single real number — the total length of the arcs in the minimum binary tree. The answer will be considered correct if the absolute or relative error doesn't exceed 10^{-6}.

Sample test(s)

input
3 0 0 1 0 2 1

output
3.650281539872885

input
4 0 0 1 0 2 1 2 0

output
-1