Task 1: MATRIX

Two $m \times n$ matrices A and B are given. Matrix B is obtained from matrix A by row-addition operations and column-subtraction operations. A row-addition operation adds 1 to each entry of a row. A column-subtraction operation subtracts 1 from each entry of a column.

In this task, you have to find the numbers of row-addition operations r_1, $\cdots$, r_m to be applied to row 1, $\cdots$, row m of A respectively such that the following properties hold.

- There correspond c_1, $\cdots$, c_n column-subtraction operations to be applied to column 1, $\cdots$, column n of A respectively so that these row and column operations transform the given matrix A to the given matrix B.
- The number of any row and column operations is between 0 and 9 inclusively; that is, $0 \leq r_i \leq 9$, $i = 1, \cdots, m$ and $0 \leq c_j \leq 9$, $j = 1, \cdots, n$.
- The value $r_1 \cdots r_m$, considered as an integer, is as small as possible.

You should concatenate the values r_1, $\cdots$, r_m and output it as a single m-digit integer $r_1 \cdots r_m$ (with possibly leading zeros). If the given matrix B cannot be obtained from the given matrix A with 0 to 9 row-addition operations on each row and 0 to 9 column-subtraction operations on each column, your program should output the value -1.

Example 1

Let

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & -1 \end{bmatrix}.$$

The required row-additions and column-subtractions are shown as:

	-4	-6	-7
$+4$	1	2	3
$+0$	4	5	6

Since there are 4 and 0 row-additions operations for row 1 and row 2 respectively, the required output is

40

after checking that 4 is the smallest possible number of row operations for row 1.

Example 2

Let

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}.$$

The required row-additions and column-subtractions are shown as:

	-4	-2	-0
$+4$	1	2	3
$+2$	4	5	6
$+0$	7	8	9

Since there are 4, 2, and 0 row-additions operations for row 1, row 2, and row 3, the required output is

420

after checking that 4 is the smallest possible number of row operations for row 1.

Example 3

Let

$$A = \begin{bmatrix} 0 \\ -9 \end{bmatrix}, \quad B = \begin{bmatrix} 9 \\ 9 \end{bmatrix}.$$

It can be checked that it is impossible to obtain B from A with 0 to 9 row-additions on each row and 0 to 9 column-subtractions on each column. So the output should be -1.

Input File: `MATRIX.IN`

The first line contains two integers m and n separated by a space, $1 \leq m \leq 100$, $1 \leq n \leq 100$. The matrix A is given by the next m lines for row 1 to row m. Each of these m lines contains n integers, with a space between two adjacent integers. Similarly, the matrix B is given by the next m lines. Each entry of the matrices is an integer between -1000 and 1000 inclusively.

For Example 1, the input file contains

```
2 3
1 2 3
4 5 6
1 0 0
0 -1 -1
```

Output File: `MATRIX.OUT`

The output file contains an m-digit integer. For Example 1, the output file contains

```
40
```

Task 2: RECT

A rectangle is *axis-parallel* if its top and bottom sides are parallel to the x-axis and its left and right sides are parallel to the y-axis. From now on by a rectangle we mean an axis-parallel rectangle.

A rectangle will be specified by a 4-tuple (x_1, y_1, x_2, y_2) in which (x_1, y_1) is the bottom-left corner and (x_2, y_2) is the top-right corner of the rectangle. We will also say the rectangle (x_1, y_1, x_2, y_2) is a $(x_2 - x_1) \times (y_2 - y_1)$ rectangle.

Two rectangles are *incomparable* if neither will fit inside the other possibly with translation and 90° rotation; otherwise, the two rectangles are comparable. Given a list of rectangles, you are to output the number of pairs of incomparable rectangles.

Example 1

Consider three rectangles:

- the 2×2 rectangle $A : (0, 0, 2, 2)$;
- the 3×1 rectangle $B : (0, 0, 3, 1)$;
- the 2×3 rectangle $C : (1, 1, 3, 4)$.

Rectangle pair AB is incomparable (neither can fit inside the other). But rectangle pair AC is comparable (A fits inside C after a translation), so is rectangle pair BC (B fits inside C after a 90° rotation and a translation).

Example 2

Consider four rectangles:

- the 1×3 rectangle A : (3,3,4,6);
- the 3×1 rectangle B : (1,2,4,3);
- the 2×2 rectangle C : (5,6,7,8);
- the 2×2 rectangle D : (10,10,12,12).

There are four incomparable rectangle pairs AC, AD, BC, BD. Thus the answer is 4.

Input File: `RECT.IN`

The first input line contains an integer which is the number of rectangles n (where $0 \leq n \leq 10000$). Each of the next n lines describes a rectangle and contains four integers x_1, y_1, x_2, y_2, a space separates two adjacent integers. Recall that coordinates (x_1, y_1), (x_2, y_2) specify respectively the bottom-left and top-right corner of the rectangle. All coordinate values are in the range of 0 to 10,000; that is, $0 \leq x_1 \leq 10000$, $0 \leq y_1 \leq 10000$, $0 \leq x_2 \leq 10000$, $0 \leq y_2 \leq 10000$.

For Example 1, the input file contains

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

Output File: `RECT.OUT`

The output file contains a single integer which is the number of pairs of incomparable rectangles. For Example 1, the output file contains

```
1
```

Task 3: WIDTH

We consider a set S of n points in the plane. The *width* w of S is the minimum distance between two parallel lines that enclose S. For instance, in Figure 1, the set

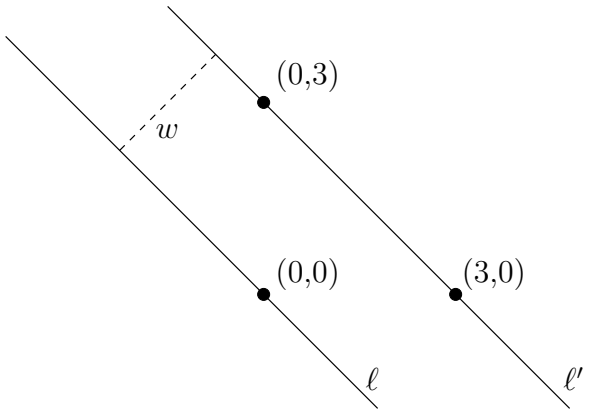

Figure 1: The width w of a set of three points.

S consists of $n = 3$ points $(0, 0)$, $(0, 3)$ and $(3, 0)$. The width is achieved by the two lines ℓ and ℓ', their distance is $w = 3\sqrt{2}/2 \simeq 2.12$. In this task, you are given a set of points in the input file and you need to compute the integer part of w^2 and write it in the output file. For instance, in Figure 1, we have $w = 3\sqrt{2}/2$, so $w^2 = 4.5$, and thus you need to output the integer part of 4.5, which is 4.

We give you a useful formula to help you solve this problem. Let $A = (x_a, y_a)$, $B = (x_b, y_b)$ and $C = (x_c, y_c)$ be three points. The height h (see Figure 2) of the triangle ABC is given by the following formula

$$h = \sigma \frac{(x_a - x_c)(y_b - y_c) - (x_b - x_c)(y_a - y_c)}{\sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}}$$

where $\sigma = 1$ if ABC is counterclockwise (as in Figure 2) and $\sigma = -1$ if ABC is clockwise.

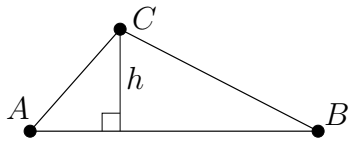

Figure 2: Triangle ABC.

Clearly, if all the points of S lie on a straight line, the width w is zero.

Input File: `WIDTH.IN`

The first input line contains the integer n, the number of points in S. Each of the next n lines contains the x coordinate and the y coordinate of an input point separated by a space. For instance, the point set in Figure 1 corresponds to the following input file:

```
3
0 0
3 0
0 3
```

Note that the coordinates of the points are integers ranging from 0 to 199 inclusively. There are at most 100000 input points. A point may appear several times in the input file. For instance, the following input file is possible:

```
4
0 0
3 0
0 3
3 0
```

In this case, the answer to the problem is still 4.

Output File: `WIDTH.OUT`

The output file should contain the integer part of w^2. For instance, the output file for the point set in Figure 1 is:

```
4
```

Task 4: PATHS

A graph is made up of a set of nodes and a set of links. A link connects two nodes. For example, Figure 1 shows a simple graph with 4 nodes and 5 links. In the figure, each link has a specific direction, going from the originating node to the destination node. Each link has a cost attached to it. The m nodes of a graph are identified by the integers $0, 1, ..., m-1$.

A path connects one node to another, following the direction of links from node to node. The length of a path is the number of links used. The cost of a path is the sum of link cost over the entire path. For a given graph, your task is to find the minimum cost among the costs of all the shortest paths between Node 0 and Node 1. A shortest path with the minimum cost is called a minimum-cost shortest path.

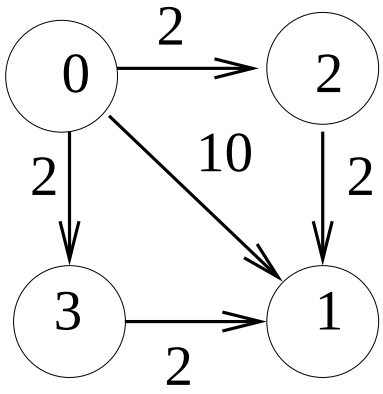

Figure 1

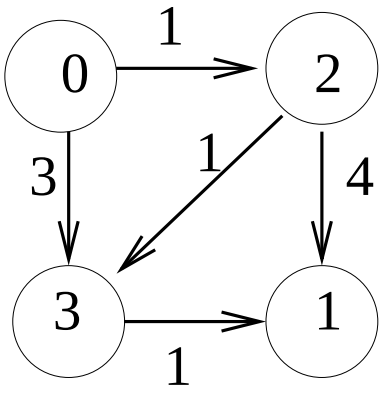

Figure 2

For example, again consider Figure 1. The shortest path from Node 0 to Node 1 is the 1-link path $0 \rightarrow 1$ with a path cost of 10. While there are cheaper paths: $0 \rightarrow 2 \rightarrow 1$ and $0 \rightarrow 3 \rightarrow 1$, they are longer (2 links). Therefore, $0 \rightarrow 1$ is the minimum-cost shortest path.

Consider another example, Figure 2. There are two shortest paths of length 2, the path $0 \rightarrow 3 \rightarrow 1$ has a lower cost (cost = 4) than the path $0 \rightarrow 2 \rightarrow 1$ (cost = 5). Another path $0 \rightarrow 2 \rightarrow 3 \rightarrow 1$ (cost = 3) is cheaper but longer. Therefore, the minimum-cost shortest path is $0 \rightarrow 3 \rightarrow 1$.

Input File: PATHS.IN

The first line contains two integers m and n separated by a space, where m is the number of nodes and n is the number of links. The links of the graph and their costs are given by the next n lines. Each line has three integers (with a space between two adjacent integers), denoting *Source-Node Destination-Node Link-Cost*.

For example, the input data for the graph in Figure 1 is:

```
4 5
0 2 2
0 3 2
0 1 10
2 1 2
3 1 2
```

There are at most 100 nodes, and nodes are numbered from 0 to 99. There are at most 1000 links and link cost varies between 0 and $2^{15} - 1$.

Output File: PATHS.OUT

The output file contains a single integer which is the path cost of a minimum-cost shortest path from Node 0 to Node 1. For example, the output file of Figure 1 contains

```
10
```

Note that while there may be multiple minimum-cost shortest paths, the cost of these paths are the same.

Task 5: PAIR

A popular solitaire board game is played on a rectangular $m \times n$ game board consisting of mn squares. The board initially comes entirely populated with either animals or obstacles. An 'X' in a square denotes an obstacle, a digit '0'–'9' in a square denotes the species of the animal.

Pairs of animals can only be removed if they are of the same species. If a pair of animals are removed, the squares in which they were located become empty and stay empty for the rest of the game. To be removed, a candidate pair must either be adjacent to each other or have a path between them. Two squares are adjacent if they are side by side horizontally or vertically. A path is a sequence of adjacent *empty* squares. The path length of a path is simply the number of empty squares in the path.

You should output the maximal number of pairs of animals it can remove from the board and the minimal cumulative path length needed to achieve this.

Example 1

Consider a 3×4 board with the following configuration:

X	X	0	X
X	1	1	X
X	0	X	X

The two animals of species 1 can be removed because they are adjacent and the path length between them is 0. After their removal, there is a path of length 2 between the two animals of species 0, thus these two animals can be removed too. To remove these two pairs of animals, the cumulative path length is $0 + 2 = 2$. This is also the minimal cumulative path length as there is only one way to remove both pairs of animals. Thus the answer is 2 2.

Example 2

Consider a 4×1 board with the following configuration:

9
9
9
9

If we first remove the two middle animals of species 9 and then remove the top and bottom animals of species 9, two pairs of animals are removed with a cumulative path length $0 + 2 = 2$.

But we can remove the top two animals of species 9 and then remove the bottom two animals of species 9. This also removes two pairs of animals with a cumulative path length $0 + 0 = 0$.

Clearly a zero cumulative path length is minimal. Thus the answer is 2 0.

Input File: PAIR.IN

The first line of the input file contains the integers m and n separated by a space, $1 \leq m \leq 5$, $1 \leq n \leq 5$. Each of the subsequent m lines contains n characters, each character is one of 'X', '0', '1', $\cdots$, '9'. There is *no* space between adjacent characters.

For Example 1, the input file looks like:

```
3 4
XX0X
X11X
X0XX
```

Output File: PAIR.OUT

The output file contains two integers separated by a space. The first integer is the maximal number of pairs of animals that can be removed. The second integer is the minimal cumulative path length needed to achieve the maximal number of pairs.

For Example 1, the output file contains

```
2 2
```

Task 6: WORD

Consider a set of k strings $\{S_1, S_2, \ldots, S_k\}$ where every character used in the k strings is either a space or any of the 26 characters in { 'a', 'b', 'c', ..., 'z' }. For some constants ℓ and d, our aim is to compute an (ℓ, d)-pattern for $\{S_1, S_2, \ldots, S_k\}$. An (ℓ, d)-pattern is a length-ℓ string $W = W[1]W[2]\ldots W[\ell]$ which satisfies the following property:

- For every string S_i $(i = 1, 2, \ldots, k)$, there exists a length-ℓ substring $X = X[1]X[2]\ldots X[\ell]$ of S_i such that the hamming distance of X and W is less than or equal to d. (The hamming distance of X and W is the number of pairs of $(X[j], W[j])$ such that $X[j] \neq W[j]$, for $j = 1, 2, \ldots, \ell$.)

In this task, you are given numbers ℓ and d and a set of strings; you need to compute an (ℓ, d)-pattern for the given set of strings. You can assume that an (ℓ, d)-pattern exists and is unique.

Example 1

Consider the following 3 strings, the corresponding $(3, 0)$-pattern is "oil".

- oil is expensive
- we have three oilers
- be more oily

Example 2

Consider the following 4 strings, the corresponding $(5, 1)$-pattern is "apple".

- you have two applas
- i am an ppple
- we are acples
- adples are good for health

Input File `WORD.IN`

The first line contains two integers ℓ and d separated by a space, where $1 \leq \ell \leq 10$ and $0 \leq d \leq 2$. The second line contains the integer k, where $1 \leq k \leq 30$. The remaining k lines contain the k strings $S_1, S_2, \ldots, S_k$. (Each string is of length at most 50.) For Example 2, the input file looks like:

```
5 1
4
you have two applas
i am an ppple
we are acples
adples are good for health
```

Output File `WORD.OUT`

The output file contains a string of length ℓ. For Example 2, the output file looks like:

```
apple
```

This string represents an (ℓ, d)-pattern for the set of strings and ℓ and d given in the input file. The input is always such that there exists exactly one (ℓ, d) pattern.