Decoration

Christmas is one of the most favorite holyday for all of us. This time Ivancho wants to decorate the windows in his school in Christmas theme. He has forms that contain coloured and transperent parts which can use for the decoration. Ivancho can neither put one colored part over another nor rotate the forms. In addition, he can't put a form in a way that it is out of the boundaries of the window and can use each form not more than once. However, Ivancho's classmates don't like every possible layout. For each form Ivancho knows the number of his classmates who like it. The total beauty of a window is equal to the product of the number of forms put on it and the total sum of likes of these forms.

Now, Ivancho asks you as good programmers to save the Christmas decoration and his reputation. Write a program **decoration** which get the sizes of the windows and the forms and returns the decoration of the windows in which the sum of beauties of all windows is optimal. To make your task easier the windows and the forms will be presented as matrixes. The description of each form is realized by a bool table. The colored part of the form is the connected component of ones in the table.

In other words we can imagine the windows as tables filled with zeroes in the beginning and the forms as tables filled with zeroes and ones. Attaching a form on a window is copying the form matrix over the window matrix. You can' copy one matrix over another if this includes copying a cell filled with one in the form matrix over a cell filled with one in the window matrix or the form matrix goes out of the window matrix. You can see one decoration of a window 7x7 below. The different forms are coloured in different colors.

0	0	0	1	0	0	0
0	1	1	1	0	0	0
0	1	1	1	0	0	0
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	0	0	0
1	1	1	0	0	0	0

Input: On the first line of the input file **decoration.in** will be written two integers **N** and **M** - the number of windows and the number of forms. The following **N** rows will contain description

of the windows with two integers **Ai** and **Bi** - the number of rows and columns of the matrix which represents the window with number **i**. Next is the description of the **M** forms. For each of them are written **Pj** and **Qj** - the sizes of the matrix and **Cj** the number of likes of the current form followed by **Pj** rows with **Qj** columns - the matrix of form with number **j**. Each row is a consequence of ones and zeroes divided by spaces. It is guaranteed that each matrix contains exactly one connected component of ones and there isn't a row or column filled with zeroes only.

The numeration of the windows, the forms and the matrix rows and columns starts from one.

Output: The output file **decoration.out** must contain **M** rows. The **i-th** row must contain three numbers - **Ti, Xi, Yi -** the number of the window on which you place the **i-th** form and the coordinates of the cell in the window matrix where you place the upper left corner of the form matrix. If you don't want to use the **i-th** form, you must print a row containing **Ti=Xi=Yi=-1**.

Constraints:

The number of the cells in all window matrixes will be less than 10 000

The number of the cells in all form matrixes will be less than 10 000

1 <= **Ai, Bi,Pj,Qj,Cj <= 100**

In 20% of the tests there won't be transperent parts in any of the forms (all form matrixes will be filled with ones).

In the rest 80% of the tests the **M** forms will be able to be divided in groups of disjoint sets so that the coloured parts of the forms in each set can be arranged in a way that they form a rectangle without transperent parts in it and without overlapping coloured cells.

Grading:

If the output file is incorrect or the answer includes overlapping coloured parts you will receive 0 points for the current test.

Otherwise, the grading will be done by the formula $(yours/best)^2$. *Where yours* is the sum of the beauty of all windows in your answer and *best* is the maximal beauty achieved by any of the participants on the current test.

Example:

decoration.in	**decoration.out**
2 3 3 3 1 1 3 3 3 1 1 1 1 0 1 1 1 1 1 1 3	1 1 1 1 2 2 -1 -1 -1

1 2 2 1 1 1 1 1	