Blackout

Input File: *standard input*
Output File: *standard output*

This time they had planned for every possible failure and made every system redundant: two independent electric fences, two control centers, two power lines connected to two separate power plants, and two GPS tracking devices attached to each beast. Nothing could possibly go wrong with this amount of security and redundancy.

What they never predicted is now happening: a national strike in every non-renewable energy power plant in the country, following the announcement of a new target of producing 100% of the country's energy from renewable sources within 5 years. The ensuing blackout is there to last, and the backup generators can only power the electric fences for a few more hours, before every Tyrannosaurus Rex of the brand new park is free to explore the neighboring residential areas.

You have been asked to participate in implementing the emergency plan. Paleontologists have recently discovered that T-Rexes won't eat salty food, and conjectured that salt can be used as a T-Rex repellent. Further, the brand new park is conveniently arranged as a $S \times S$ grid, broken up into unit squares. The plan is to use helicopters to drop sea water on certain squares of land so that the T-Rexes will avoid them and stay in the park. Your job is to write a program that will determine where to pour sea water each hour after the blackout in order to minimise the number of escaped T-Rexes.

There are two types of squares in the park; empty squares that T-Rexes can enter, and squares with obstacles, which they can not. The GPS trackers will give you the last position of each T-Rex in this grid before the power goes down. While you won't be able to track their movement, you do know that every hour each T-Rex can move to any of the 4 squares adjacent to its previous position (provided the square is not an obstacle, nor has had sea water dropped onto it). Each hour, you will be able to drop sea water on W squares, permanently preventing any T-Rex from entering these squares.

Input

- The first line of input will contain three integers S, W and T. S represents the number of lines and rows of the grid, W represents the number of squares you can drop sea-water on every hour, and T represents the number of T-Rexes in the park.
- The following S lines of input will each contain S space-separated integers, 0 representing an empty square, and 1 representing an obsacle.
- The following T lines of input will each contain two integers, C_i and R_i ($0 \leq C_i, R_i < S$), representing the column and row of the last known position of a T-Rex before the black-out. Rows are numbered from top to bottom, columns are numbered from left to right (so position"0 0" would be the top-left square).

Output

- The first line of output should consist of a single integer D, the total number of times water will be dropped on a cell. As you can not drop water on to the same cell multiple times (this would be useless), $D \leq S^2$.
- Each of the D following lines should contain two integers, DC_i and DR_i ($0 \leq DC_i, DR_i < S$), representing the column and the row of a square to drop sea water on. The locations are given in the order the water should be dropped, which means the first W lines correspond to the squares targeted during the first hour of the blackout, the next W lines to the squares targeted during the second hour, and so on.

Sample Input

```
5 2 1
0 1 0 0 0
0 0 0 0 0
0 0 0 1 0
0 0 0 0 0
0 1 1 1 0
2 3
```

Sample Output

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

Explanation

The park is a 5×5 grid, with only one T-Rex. You can drop water on 2 squares every hour. The grid is represented below, where T is the initial location of the T-Rex.

```
0 1 0 0 0
0 0 0 0 0
0 0 0 1 0
0 0 T 0 0
0 1 1 1 0
```

One way to prevent the T-Rex from escaping the park is to drop water on two squares in the first hour, on the left $(1, 3)$ and above $(2, 2)$ the T-Rex's initial location. This means the T-Rex can either stay where it is, or move to the right. At the end of the first hour, the situation is the following, where W marks the squares where water has been dropped, and T marks the two squares where the T-Rex may now be located.

```
0 1 0 0 0
0 0 0 0 0
0 0 W 1 0
0 W T T 0
0 1 1 1 0
```

Dropping water on square $(4, 3)$ will prevent the T-Rex from ever escaping the 2-square area that it is now confined to.

Constraints & Subtasks

- Subtask 1 (20 points): $S = 5$, $W = 1$, $T = 1$.
- Subtask 2 (20 points): $S = 10$, $W = 2$, $T = 1$.
- Subtask 3 (20 points): $S = 15$, $W = 3$, $T = 2$.
- Subtask 4 (20 points): $S = 20$, $W = 4$, $T = 5$.
- Subtask 5 (20 points): $S = 30$, $W = 5$, $T = 10$.

For each test case of each subtask, you will score 100% if the output of your program enables the capture of the most T-Rexes among all judges' solutions. Otherwise, your score for that test will be :

$$\frac{100 \times [\text{Number of T-Rexes your output captured}]}{\text{Maximum number of T-Rexes captured by judges' solutions}}$$

If your program gives an invalid output, your score will be 0 for that test case.

Your total score for a subtask will be the **sum** of your score for each of its test cases.