D. Map

time limit per test: 2 seconds

memory limit per test: 128 megabytes

input: standard input

output: standard output

There is an area map that is a rectangular matrix $n \times m$, each cell of the matrix contains the average height of a corresponding area part. Peter works for a company that has to build several cities within this area, each of the cities will occupy a rectangle $a \times b$ cells on the map. To start construction works in a particular place Peter needs to remove excess ground from the construction site where a new city will be built. To do so he chooses a cell of the minimum height within this site, and removes excess ground from other cells of the site down to this minimum level. Let's consider that to lower the ground level from h_2 to h_1 ($h_1 \le h_2$) they need to remove h_2 - h_1 ground units.

Let's call a site's position optimal, if the amount of the ground removed from this site is minimal compared to other possible positions. Peter constructs cities according to the following algorithm: from all the optimum site's positions he chooses the uppermost one. If this position is not unique, he chooses the leftmost one. Then he builds a city on this site. Peter repeats this process untill he can build at least one more city. For sure, he cannot carry out construction works on the occupied cells. Would you, please, help Peter place cities according to the algorithm?

Input

The first line contains four space-separated integers: map sizes n, m and city sizes a, b ($1 \le a \le n \le 1000$, $1 \le b \le m \le 1000$). Then there follow n lines, each contains m non-negative space-separated numbers, describing the height matrix. Each number doesn't exceed 10^9.

Output

In the first line output k — the amount of constructed cities. In each of the following k lines output 3 space-separated numbers — the row number and the column number of the upper-left corner of a subsequent construction site, and the amount of the ground to remove from it. Output the sites in the order of their building up.

Examples

input Copy

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

output Copy

```
2
1 1 1
2 1 2
```

input Copy

```
4 4 2 2
1 5 3 4
2 7 6 1
1 1 2 2
2 2 1 2
```

output Copy

```
3
3 1 2
```

```
3 3 3
1 2 9
```