Problem 2: The Castle

```
     1   2   3   4   5   6   7
   #############################
 1 #   |   #   |   #   |   |   #
   #####---#####---#---#####---#
 2 #   #   |   #   #   #   #   #
   #---#####---#####---#####---#
 3 #   |   |   #   #   #   #   #
   #---#########---#####---#---#
 4 # ->#   |   |   |   |   #   #
   #############################   (Figure 1)

#  = Wall
|  = No wall
-  = No wall
-> = It points to the wall to remove according to the example output.
```

Figure 1 shows the map of a castle. Write a program that calculates

1. how many rooms the castle has
2. how big the largest room is
3. which wall to remove from the castle to make as large a room as possible.
   The castle is divided into m * n (m<=50, n<=50) square modules. Each such module can have between zero and four walls.

Input Data

The map is stored in the `INPUT.TXT` file in the form of numbers, one for each module.

- The file starts with the number of modules in the north-south direction and the number of modules in the east-west direction.
- In the following lines each module is described by a number (0<=p<=15). This number is the sum of: 1 (= wall to the west), 2 (= wall to the north), 4 (= wall to the east), 8 (= wall to the south). Inner walls are defined twice; a wall to the south in module 1,1 is also indicated as a wall to the north in module 2,1.
- The castle always has at least two rooms.
  `INPUT.TXT` for our example:

```
4
7
11 6 11 6 3 10 6
7 9 6 13 5 15 5
1 10 12 7 13 7 5
13 11 10 8 10 12 13
```

Output Data

In the `OUTPUT.TXT` file, the following are written on three lines: First the number of rooms, then the area of the largest room (counted in modules) and a suggestion of which wall to remove (first the row and then the column of the module next to the wall and finally the compass direction that points to the wall). In our example ("4 1 E" is one of several possibilities, you need only produce one):

```
5
9
4 1 E
```