Tilted Towers

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

points: 100 · problem code: tiltedtowers · author: glipR

Your friend Nancy has made a computer simulation of the popular game Jenga, where groups of 3 planks are stacked on top of one another in a criss-cross fashion.



However, her simulation isn't the most sophisticated: the condition for the tower falling over is relatively simple. She defines a particular layer of the tower to be *uneven* if the average position of the planks is not in the center - this is a fancy way of saying that there are more blocks on one side than the other.

For example, from the standard 3 wide layer, removing the centre plank would keep the layer even, while removing either the left of right plank would result in an uneven layer.

Nancy's simulation determines that the tower will fall if two consecutive layers in the tower are uneven, or if any layer is empty. (This result obviously doesn't line up with reality (in both the positive and negative case), but ignore that.)

You've been given a current game state, and wonder about the maximum number of planks you can remove before the tower will fall. In particular, at no point while you are removing the planks can the tower be in a falling state.

Input

Input will begin with a single integer n, representing the height (number of layers) of the tower.

$3n$ lines will then follow, with groups of 3 lines representing a pictogram of a layer of the tower, from top to bottom.

Each line will contain 3 characters, each an `X` or a `.`, representing the presence (or lack) of a plank.

The first layer will always represent horizontal lines, and then it will alternate between vertical and horizontal.

Output

Output should begin with a single integer p - How many planks you can remove from the tower without it falling.

p lines should follow, each containing two numbers i and j. i represents the layer the plank is being removed from (top layer is 1, and increases), and j represents the plank to remove in that layer (for vertical plank layers, $1, 2, 3$ are the left/middle/right planks, and for horizontal plank layers, $1, 2, 3$ are the top/middle/bottom planks.)

If there are multiple possible ways to select planks to remove, simply print any possible order / collection.

Constraints

- $1 \leq n \leq 1000$
- The input tower will always be safe (no two adjacent layers will start uneven)

Example

For the following input:

```
5
XXX
...
...
XXX
XXX
XXX
...
XXX
XXX
X.X
X.X
X.X
XXX
XXX
XXX
```

This represents the following tower:



This can be reduced to the following tower (paper wedge used to appease real-world gravity):



By removing these planks:



So one possible output for the problem could be:

```
5
3 3
2 2
5 3
5 1
4 3
```