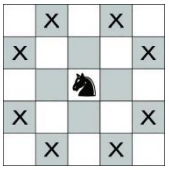

You are given a chess table *NxM* (*N* rows and *M* columns) and some of the cells are **inaccessible**. Rows are numbered from top to bottom from 0 and columns are numbered from left to right from 0. This way the top left cell has coordinates (0,0). On this table there is a chess knight, which can make **at most K jumps**. It cannot be in a cell which is inaccessible. *Jump means a valid move of a chess knight on the table (shown on the picture)*.

Your task is to destroy the knight. For the task you have an artillery weapon, which can with one shot hit any cell you desire. If in the moment the shot is fired, the knight is in the cell fired upon, then the knight is destroyed. After the shot the cell remains as it was before – accessible or inaccessible. The knight can make 0 or 1 jumps between two shots of your weapon. You never know where the knight is.

Write a program **chess**, which calculates the minimal number of shots with which one can be certain that the knight is destroyed and finds one such list of shots.

Input

On the first line of the standard input are 3 integers separated with a space - N, M and K.

On each of the next N lines there are M symbols describing the table, where '.' - describes an accessible cell and '#' - describes an inaccessible cell.

Output

On the first line of the standard output print a single integer – the minimal found number of shots, required to be sure that the knight is destroyed.

On each of the following lines (as many as the found minimal number) output two non-negative integers seperated with a space – the coordinates of the current struck cell. The coordinates of the hit cells should be printed in the order they are shot at.

If there is more than one solution print any of them.

Constraints

$1 \le N,M,K \le 100$
In test cases worth 20% of the points: $M=2$;
In other test cases worth 10% of the points: all cells are accessible;
In other test cases worth 20% of the points: K is even.

Grading

Each test case is evaluated separately.

Example 1

Input	*Output*
3 5 1 ..... ##### .#.#.	10 0 0 0 1 0 2 0 3 0 4 2 0 2 2 2 4 0 1 0 3

***Explanation of example 1*:** In this example the knight can make at most one jump. With the first 5 shots it is guaranteed that the knight will be destroyed if it was on the top row and did not jump to the bottom row. With the next 3 shots it is guaranteed that the knight will be destroyed if it was on the top row and jumped to the

bottom row during the first 5 shots or if the knight was initially on the bottom row and did not jump. With the last 2 shots we guarantee that the knight will be destroyed if it was initially on the bottom row and during the past 3 shots has jumped to the top row.

Example 2

Input	*Output*
3 3 1	13 0 0 1 0 2 0 2 1 2 2 1 2 0 2 0 1 1 1 2 1 2 0 1 0 0 0