Problem SOLITAIRE: Peg Solitaire

The game of peg solitaire, popular at the court of the French king Louis XIV, has the following rules. Given a two-dimensional board with a mesh of holes, each hole can contain one peg (pin). The only legal move of a peg is a vertical or horizontal jump over an adjacent peg into the empty hole next to the jumped peg in line with it, which is then removed. The original goal of the game was to leave a single peg in the predefined position on the board by performing only legal moves. Obviously, such a solution is possible only for certain board forms and starting configurations. To drop this restriction, we slightly redefine the problem: Given the starting configuration of the board, determine the minimum number of pegs that can be achieved by means of legal moves as well as the minimum number of moves required to reach that number of pegs.

Input

The first line of the input contains one number, $1 \leq N \leq 100$ which represents the number of test cases. Each test case is described by the following lines of input that represent the initial state of the solitaire board.
In this representation '`.`' denotes an empty hole and '$\circ$' a hole with a peg in it. '#' indicates a part of the board without a hole. All boards have the same shape, see sample input (that includes position of holes). In its initial state, the board can contain at most 8 pegs. There is an empty line between two consecutive test cases.

Output

For each test case your program should output a line with two numbers separated by a single whitespace, with the first one denoting the minimum number of pegs achievable by legal moves starting with the given initial state, and the second one providing the minimum required number of moves.

Sample Input 1

```
3
###...###
..oo.....
.....oo..
.........
###...###

###...###
..oo.o...
...o.oo..
...oo....
###...###

###o..###
.o.oo....
o.o......
.o.o.....
###...###
```

Sample Output 1

```
1 3
1 7
1 7
```