**Problem C**

**Snakes**

The Cartesian plane is covered with snakes. You will be given a rectangular portion of the plane, divided into a grid of squares. Each square will either contain a segment of a snake or a barrier.

Each snake occupies a chain of adjacent squares, connected horizontally and/or vertically. Snakes are at least 2 segments long (one segment per square) and cannot overlap each other or themselves. Each snake must meet at least one of the following two conditions (see the figures below for examples):

1. Both endpoints must be in squares on the edge of the rectangle.
2. Both endpoints must be horizontally or vertically adjacent to each other, and the snake is at least 4 segments long (so the snake forms a loop).

The portion of the plane will be given as a String grid, where each character represents one square. A '#' represents a barrier and a '.' (period) represents a square that contains a segment of a snake. Fill the grid with snakes so that all of the non-barrier squares are filled. Do this in such a way that minimizes n, the number of snakes whose endpoints are not adjacent (and therefore, must be on the edge of the grid). Return n, or return -1 if there is no way to fill every non-barrier square.

{ "......",

".#.##.",

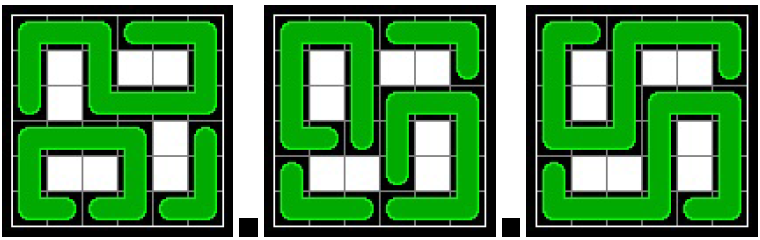
".#....",

"....#.",

".##.#.",

"......" }

can be filled with snakes in several ways and a few are shown in the figures below:



**Input**

The input contains several cases, until the end of file. Input for each test case is given in the following order:

* A single line containing the length of grid
* The elements of the array grid each in a single line

The grid will contain between 1 and 12 elements, inclusive. Each element of grid will contain the same number of characters, between 1 and 12, inclusive. Each element of grid will contain only the characters '#' and '.' (period).

**Output**

Output contains the result in a single line.

|  |  |
| --- | --- |
| **Sample Input**  6  ......  .#.##.  .#....  ....#.  .##.#.  ......  5  ###.###  ###.###  .......  ###.###  ###.###  7  #.....##  ...#....  ........  ..#.#.#.  ......#.  ...#..#.  #.......  8  ###########  #......#...  #.####.#...  #.#..#.####  #.#..#.####  #.####.#..#  #......#..#  ########### | **Sample Output**  2  -1  1  0 |