

Advent of Code [About] [AoC++] [Events] [Settings] [Log Out] bildzeitung 50*
 //2017 [Calendar] [Leaderboard] [Stats] [Sponsors]

--- Day 21: Fractal Art ---

You find a program trying to generate some art. It uses a strange process that involves repeatedly enhancing the detail of an image through a set of rules.

The image consists of a two-dimensional square grid of pixels that are either on (#) or off (.). The program always begins with this pattern:

```

.#.
..#
###

```

Because the pattern is both 3 pixels wide and 3 pixels tall, it is said to have a size of 3.

Then, the program repeats the following process:

- If the size is evenly divisible by 2, break the pixels up into 2x2 squares, and convert each 2x2 square into a 3x3 square by following the corresponding **enhancement rule**.
- Otherwise, the size is evenly divisible by 3; break the pixels up into 3x3 squares, and convert each 3x3 square into a 4x4 square by following the corresponding **enhancement rule**.

Because each square of pixels is replaced by a larger one, the image gains pixels and so its size increases.

The artist's book of enhancement rules is nearby (your puzzle input); however, it seems to be missing rules. The artist explains that sometimes, one must **rotate** or **flip** the input pattern to find a match. (Never rotate or flip the output pattern, though.) Each pattern is written concisely: rows are listed as single units, ordered top-down, and separated by slashes. For example, the following rules correspond to the adjacent patterns:

```

../.# = ..
        .#

.#./..#/# = .#.
        ..#
        ##

#..#/#.../#.../.## = #..#
        ....
        #..#
        .##.

```

When searching for a rule to use, rotate and flip the pattern as necessary. For example, all of the following patterns match the same rule:

```

.#.  .#.  #..  ###
..#  #..  #.#  ..#
###  ###  ##.  .#.

```

Suppose the book contained the following two rules:

```

../.# => ##./#.../...
.#./..#/# => #..#/#.../#...

```

As before, the program begins with this pattern:

Our **sponsors** help make Advent of Code possible:

Winton - a data science and investment management company

By popular demand, there are now AoC-themed objects available (until Jan. 3rd)! Get them shipped from the US or from Europe.

```

.#.
..#
###

```

The size of the grid (3) is not divisible by 2, but it is divisible by 3. It divides evenly into a single square; the square matches the second rule, which produces:

```

#..#
....
....
#..#

```

The size of this enhanced grid (4) is evenly divisible by 2, so that rule is used. It divides evenly into four squares:

```

#.|.#
..|..
--+--
..|..
#.|.#

```

Each of these squares matches the same rule (`../.# => ##./#../...`), three of which require some flipping and rotation to line up with the rule. The output for the rule is the same in all four cases:

```

##.|##.
#..|#..
...|...
---+---
##.|##.
#..|#..
...|...

```

Finally, the squares are joined into a new grid:

```

####.
#..#..
.....
####.
#..#..
.....

```

Thus, after 2 iterations, the grid contains 12 pixels that are on.

How many pixels stay on after 5 iterations?

Your puzzle answer was 194.

--- Part Two ---

How many pixels stay on after 18 iterations?

Your puzzle answer was 2536879.

Both parts of this puzzle are complete! They provide two gold stars: **

At this point, all that is left is for you to admire your advent calendar.

If you still want to see it, you can get your puzzle input.

You can also [Share] this puzzle.