```
Advent of Code [About] [Events] [Shop] [Settings] [Log Out] jhillierdavis 20*
 var y=2020;  [Calendar] [AoC++] [Sponsors] [Leaderboard] [Stats]
--- Day 11: Seating System ---
Your plane lands with plenty of time to spare. The final leg of your
journey is a ferry that goes directly to the tropical island where you can
finally start your vacation. As you reach the waiting area to board the
ferry, you realize you're so early, nobody else has even arrived yet!
By modeling the process people use to choose (or abandon) their seat in the
waiting area, you're pretty sure you can predict the best place to sit. You
make a quick map of the seat layout (your puzzle input).
The seat layout fits neatly on a grid. Each position is either floor (.),
an empty seat (L), or an occupied seat (#). For example, the initial seat
layout might look like this:
L.LL.LL.LL
LLLLLLL.LL
L.L.L..L..
LLLL.LL.LL
L.LL.LL.LL
L.LLLL.LL
. . L . L . . . . .
LLLLLLLLL
L.LLLLL.L
L.LLLLL.LL
Now, you just need to model the people who will be arriving shortly.
Fortunately, people are entirely predictable and always follow a simple set
of rules. All decisions are based on the number of occupied seats adjacent
to a given seat (one of the eight positions immediately up, down, left,
right, or diagonal from the seat). The following rules are applied to every
seat simultaneously:
 - If a seat is empty (L) and there are no occupied seats adjacent to it,
    the seat becomes occupied.
  - If a seat is occupied (\overline{\#}) and four or more seats adjacent to it are
    also occupied, the seat becomes empty.
  - Otherwise, the seat's state does not change.
Floor (.) never changes; seats don't move, and nobody sits on the floor.
After one round of these rules, every seat in the example layout becomes
occupied:
#.##.##.##
#######.##
#.#.#..#..
####.##.##
#.##.##.##
#.####.##
. . # . # . . . . .
|#########
#.#####.#
#.####.##
After a second round, the seats with four or more occupied adjacent seats
become empty again:
#.LL.L#.##
# L L L L L L . L #
L.L.L..L..
#LLL.LL.L#
#.LL.LL.LL
#.LLLL#.##
. . L . L . . . . .
# L L L L L L L L #
#.LLLLLL.L
```

#.#LLLL.##

#.##.L#.##

#L###LL.L#

L.#.#..#..

L # # . # # . L

#.##.LL.LL

This process continues for three more rounds:

|#.##||#.##| ..#.#.... # L # # # # # # L # #.LL###L.L #.#L###.## #.#L.L#.## # L L L # L L . L # L.L.L.#.. #LLL.##.L# # . LL . LL . LL #.LL#L#.## . . L . L # L # L L L L # L # #.LLLLLL.L #.#L#L#.##

#.#L.L#.## # L L L # L L . L # L.#.L..#.. # L # # . # # . L # #.#L.LL.LL #.#L#L#.## ..L.L.... # L # L # # L # L # #.LLLLLL.L #.#L#L#.##

At this point, something interesting happens: the chaos stabilizes and further applications of these rules cause no seats to change state! Once people stop moving around, you count 37 occupied seats.

Simulate your seating area by applying the seating rules repeatedly until no seats change state. How many seats end up occupied?

To begin, get your puzzle input. Answer: You can also [Share] this puzzle. Retool - Build internal tools, remarkably fast.