--- Day 15: Warehouse Woes ---

You appear back inside your own mini submarine! Each Historian drives their
mini submarine in a different direction; maybe the Chief has his own
submarine down here somewhere as well?

You look up to see a vast school of lanternfish swimming past you. On
closer inspection, they seem quite anxious, so you drive your mini
submarine over to see if you can help.

Because lanternfish populations grow rapidly, they need a lot of food, and
that food needs to be stored somewhere. That's why these lanternfish have
built elaborate warehouse complexes operated by robots!

These lanternfish seem so anxious because they have lost control of the
robot that operates one of their most important warehouses! It is currently
running amok, pushing around boxes in the warehouse with no regard for
lanternfish logistics or lanternfish inventory management strategies.

Right now, none of the lanternfish are brave enough to swim up to an
unpredictable robot so they could shut it off. However, if you could
anticipate the robot's movements, maybe they could find a safe option.

The lanternfish already have a map of the warehouse and a list of movements
the robot will attempt to make (your puzzle input). The problem is that the
movements will sometimes fail as boxes are shifted around, making the
actual movements of the robot difficult to predict.

For example:

```
##########
#..O..O.O#
#......O.#
#.OO..O.O#
#..O@..O.#
#O#..O...#
#O..O..O.#
#.OO.O.OO#
#....O...#
##########

<vv>^<v^>v>^vv^v>v<>v^v<v<^vv<<<^><<><>>v<vvv<>^v^>^<<<>><<v<<<v^vv^v>^
vvv<<^>^v^^><<>>><>^<<><^vv^^<>vvv<>><^^v>^>vv<>v<<<<v<^v>^<^^>>>^<v<v
><<v>v^v^<>><>>>><^^>vv>v<^^^>>v^v^<^^>v^^>v^<^v>v<>v^v^<v>v^^<^^vv<
<<v<^>>^^^^>>>v^<>vvv^><v<<<>^^^vv^<vvv>^>v<^^^^v<>^>vvvv><>>v^<<^^^^^
^><^><>>><>^^<<^^v>>><^<v>^^>vv>^v>>><^^>^^v<>^v<>v<^v>v<>v^^<^^vv<
>^>>^v>vv>^<<<^v<>><<><<v<<v><>v<^vv<<<>^^v^>^^>>><<^v>>v^v><^^>>^<>vv^
<>><^^>^^^<><vvvvv^v<v<<>^v<v>v<<^><<><<<^^<<<^<<>><<><^^^>^^<>^>v<>
^^>vv<^v^v<vv>^<><v<^v>^^^>>>^^vvv^>vvv<>>>^<^>>>>>^<<^v>^vvv<>^<><<v>
v^^>>><<^^<>>^v^<v^vv<>v^<<>^<^v^v><^<<<><<^<v><v<>vv>>v><v^<vv<>v^<<^
```

As the robot (@) attempts to move, if there are any boxes (O) in the way,
the robot will also attempt to push those boxes. However, if this action
would cause the robot or a box to move into a wall (#), nothing moves
instead, including the robot. The initial positions of these are shown on
the map at the top of the document the lanternfish gave you.

The rest of the document describes the moves (^ for up, v for down, < for
left, > for right) that the robot will attempt to make, in order. (The
moves form a single giant sequence; they are broken into multiple lines

just to make copy-pasting easier. Newlines within the move sequence should
be ignored.)

Here is a smaller example to get started:

```
#######
#..O.O.#
##@.O..#
#...O..#
#.#.O..#
#...O..#
#......#
#######

<^^>>>vv<v>>v<<
```

Were the robot to attempt the given sequence of moves, it would push around
the boxes as follows:

```
Initial state:
#######
#..O.O.#
##@.O..#
#...O..#
#.#.O..#
#...O..#
#......#
#######

Move <:
#######
#..O.O.#
##@.O..#
#...O..#
#.#.O..#
#...O..#
#......#
#######

Move ^:
#######
#.@O.O.#
##..O..#
#...O..#
#.#.O..#
#...O..#
#......#
#######

Move ^:
#######
#.@O.O.#
##..O..#
#...O..#
#.#.O..#
#...O..#
#......#
#######

Move >:
#######
#..@OO.#
##..O..#
#...O..#
#.#.O..#
#...O..#
#......#
#######

Move >:
#######
#...@OO#
##..O..#
#...O..#
#.#.O..#
#...O..#
#......#
#######

Move >:
#######
#...@OO#
##..O..#
```

```
#...O..#
#.#.O..#
#...O..#
#......#
#######

Move v:
#######
#....OO#
##..@..#
#...O..#
#.#.O..#
#...O..#
#...O..#
#######

Move v:
#######
#....OO#
##..@..#
#...O..#
#.#.O..#
#...O..#
#...O..#
#######

Move <:
#######
#....OO#
##.@...#
#...O..#
#.#.O..#
#...O..#
#...O..#
#######

Move v:
#######
#....OO#
##.....#
#..@O..#
#.#.O..#
#...O..#
#...O..#
#######

Move >:
#######
#....OO#
##.....#
#...@O.#
#.#.O..#
#...O..#
#...O..#
#######

Move >:
#######
#....OO#
##.....#
#....@O#
#.#.O..#
#...O..#
#...O..#
```

```
#######

Move v:
#######
#....OO#
##.....#
#.....O#
#.#.O@.#
#...O..#
#...O..#
#######

Move <:
#######
#....OO#
##.....#
#.....O#
#.#O@..#
#...O..#
#...O..#
#######

Move <:
#######
#....OO#
##.....#
#.....O#
#.#O@..#
#...O..#
#...O..#
#######
```

The larger example has many more moves; after the robot has finished those
moves, the warehouse would look like this:

```
##########
#.O.O.OOO#
#........#
#OO......#
#OO@.....#
#O#....O#
#O.....OO#
#O.....OO#
#OO....OO#
##########
```

The lanternfish use their own custom Goods Positioning System (GPS for
short) to track the locations of the boxes. The GPS coordinate of a box is
equal to 100 times its distance from the top edge of the map plus its
distance from the left edge of the map. (This process does not stop at wall
tiles; measure all the way to the edges of the map.)

So, the box shown below has a distance of 1 from the top edge of the map
and 4 from the left edge of the map, resulting in a GPS coordinate of
100 * 1 + 4 = 104.

```
#######
#...O..
#......
```

The lanternfish would like to know the sum of all boxes' GPS coordinates
after the robot finishes moving. In the larger example, the sum of all
boxes' GPS coordinates is 10092. In the smaller example, the sum is 2028.

Predict the motion of the robot and boxes in the warehouse. After the robot
is finished moving, what is the sum of all boxes' GPS coordinates?

Your puzzle answer was 1436690.


--- Part Two ---

The lanternfish use your information to find a safe moment to swim in and
turn off the malfunctioning robot! Just as they start preparing a festival
in your honor, reports start coming in that a second warehouse's robot is
also malfunctioning.

This warehouse's layout is surprisingly similar to the one you just helped.
There is one key difference: everything except the robot is twice as wide!
The robot's list of movements doesn't change.

To get the wider warehouse's map, start with your original map and, for
each tile, make the following changes:

  - If the tile is #, the new map contains ## instead.
  - If the tile is O, the new map contains [] instead.
  - If the tile is ., the new map contains .. instead.
  - If the tile is @, the new map contains @. instead.

This will produce a new warehouse map which is twice as wide and with wide
boxes that are represented by []. (The robot does not change size.)

The larger example from before would now look like this:

```
####################
##....[]....[]..[]##
##............[]..##
##..[][]....[]..[]##
##....[]@.....[]..##
##[]##....[]......##
##[]....[]....[]..##
##..[][]..[]..[][]##
##........[]......##
####################
```

Because boxes are now twice as wide but the robot is still the same size
and speed, boxes can be aligned such that they directly push two other
boxes at once. For example, consider this situation:

```
######
#...#.#
#.....#
#..OO@#
#..O..#
#.....#
######

<vv<<^^<<^^
```

After appropriately resizing this map, the robot would push around these
boxes as follows:

```
Initial state:
##############
##......##..##
##..........##
##...[][]@.##
##....[]....##
##..........##
##############

Move <:
##############
##......##..##
##..........##
##...[][]@..##
##....[]....##
##..........##
##############

Move v:
##############
##......##..##
##..........##
##...[][]...##
##....[].@..##
##..........##
##############

Move v:
##############
##......##..##
##..........##
##...[][]...##
##....[]....##
##.......@..##
##############

Move <:
##############
##......##..##
##..........##
##...[][]...##
##....[]....##
##......@...##
##############

Move <:
##############
##......##..##
##..........##
##...[][]...##
##....[]....##
##.....@....##
##############

Move ^:
##############
##......##..##
##...[][]...##
##....[]....##
##.....@....##
##.........##
##############

Move ^:
```

```
##############
##......##..##
##...[][]...##
##....[]....##
##.....@....##
##..........##
##############

Move <:
##############
##......##..##
##...[][]...##
##....[]....##
##....@.....##
##..........##
##############

Move <:
##############
##......##..##
##...[][]...##
##....[]....##
##...@......##
##..........##
##############

Move ^:
##############
##......##..##
##...[][]...##
##...@[]....##
##..........##
##..........##
##############

Move ^:
##############
##...[].##..##
##...@.[]...##
##....[]....##
##..........##
##..........##
##############
```

This warehouse also uses GPS to locate the boxes. For these larger boxes,
distances are measured from the edge of the map to the closest edge of the
box in question. So, the box shown below has a distance of 1 from the top
edge of the map and 5 from the left edge of the map, resulting in a GPS
coordinate of 100 * 1 + 5 = 105.

```
##########
##...[]...
##........
```

In the scaled-up version of the larger example from above, after the robot
has finished all of its moves, the warehouse would look like this:

```
####################
##[].......[].[][]##
##[]...........[].##
##[]........[][][]##
##[]......[]....[]##
##..##......[]....##
##..[]............##
##..@......[].[][]##
##......[][]..[]..##
####################
```

The sum of these boxes' GPS coordinates is 9021.

Predict the motion of the robot and boxes in this new, scaled-up warehouse.
What is the sum of all boxes' final GPS coordinates?

Your puzzle answer was 1482350.

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

At this point, you should return to your Advent calendar and try another puzzle.

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

You can also [Share] this puzzle.