--- Day 21: Springdroid Adventure ---

You lift off from Pluto and start flying in the direction of Santa.

While experimenting further with the tractor beam, you accidentally pull an asteroid directly into your ship! It deals significant damage to your hull and causes your ship to begin tumbling violently.

You can send a droid out to investigate, but the tumbling is causing enough artificial gravity that one wrong step could send the droid through a hole in the hull and flying out into space.

The clear choice for this mission is a droid that can jump over the holes in the hull - a springdroid.

You can use an Intcode program (your puzzle input) running on an ASCII-capable computer to program the springdroid. However, springdroids don't run Intcode; instead, they run a simplified assembly language called springscript.

While a springdroid is certainly capable of navigating the artificial gravity and giant holes, it has one downside: it can only remember at most 15 springscript instructions.

The springdroid will move forward automatically, constantly thinking about whether to jump. The springscript program defines the logic for this decision.

Springscript programs only use Boolean values, not numbers or strings. Two registers are available: `T`, the temporary value register, and `J`, the jump register. If the jump register is true at the end of the springscript program, the springdroid will try to jump. Both of these registers start with the value false.

Springdroids have a sensor that can detect whether there is ground at various distances in the direction it is facing; these values are provided in read-only registers. Your springdroid can detect ground at four distances: one tile away (`A`), two tiles away (`B`), three tiles away (`C`), and four tiles away (`D`). If there is ground at the given distance, the register will be true; if there is a hole, the register will be false.

There are only three instructions available in springscript:

- `AND X Y` sets `Y` to true if both `X` and `Y` are true; otherwise, it sets `Y` to false.
- `OR X Y` sets `Y` to true if at least one of `X` or `Y` is true; otherwise, it sets `Y` to false.
- `NOT X Y` sets `Y` to true if `X` is false; otherwise, it sets `Y` to false.

In all three instructions, the second argument (`Y`) needs to be a writable register (either `T` or `J`). The first argument (`X`) can be any register (including `A`, `B`, `C`, or `D`).

For example, the one-instruction program `NOT A J` means "if the tile immediately in front of me is not ground, jump".

Or, here is a program that jumps if a three-tile-wide hole (with ground on the other side of the hole) is detected:

```
NOT A J
NOT B T
AND T J
NOT C T
AND T J
AND D J
```

The Intcode program expects ASCII inputs and outputs. It will begin by displaying a prompt; then, input the desired instructions one per line. End each line with a newline (ASCII code `10`). When you have finished entering your program, provide the command `WALK` followed by a newline to instruct the springdroid to begin surveying the hull.

If the springdroid falls into space, an ASCII rendering of the last moments of its life will be produced. In these, `@` is the springdroid, `#` is hull, and `.` is empty space. For example, suppose you program the springdroid like this:

```
NOT D J
WALK
```

This one-instruction program sets `J` to `true` if and only if there is no ground four tiles away. In other words, it attempts to jump into any hole it finds:

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

.................
.................
.@...............
#####.############

.................
..@..............
.................
#####.############

...@.............
.................
.................
#####.############

.................
....@............
.................
#####.############

.................
.................
.....@...........
#####.############

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

However, if the springdroid successfully makes it across, it will use an output instruction to indicate the amount of damage to the hull as a single giant integer outside the normal ASCII range.

Program the springdroid with logic that allows it to survey the hull without falling into space. What amount of hull damage does it report?

Your puzzle answer was `19351175`.

The first half of this puzzle is complete! It provides one gold star: *

--- Part Two ---

There are many areas the springdroid can't reach. You flip through the manual and discover a way to increase its sensor range.

Instead of ending your springcode program with `WALK`, use `RUN`. Doing this will enable extended sensor mode, capable of sensing ground up to nine tiles away. This data is available in five new read-only registers:

- Register `E` indicates whether there is ground five tiles away.
- Register `F` indicates whether there is ground six tiles away.
- Register `G` indicates whether there is ground seven tiles away.
- Register `H` indicates whether there is ground eight tiles away.
- Register `I` indicates whether there is ground nine tiles away.

All other functions remain the same.

Successfully survey the rest of the hull by ending your program with `RUN`. What amount of hull damage does the springdroid now report?

Although it hasn't changed, you can still get your puzzle input.

Answer: [Submit]

You can also [Share] this puzzle.