--- Day 24: Arithmetic Logic Unit ---

Magic smoke starts leaking from the submarine's arithmetic logic unit (ALU). Without the ability to perform basic arithmetic and logic funct the submarine can't produce cool patterns with its Christmas lights!

It also can't navigate. Or run the oxygen system.

Don't worry, though - you probably have enough oxygen left to give you enough time to build a new ALU.

The ALU is a four-dimensional processing unit: it has integer variables `x`, `y`, and `z`. These variables all start with the value `0`. The ALU also supports six instructions:

- `inp a` - Read an input value and write it to variable `a`.
- `add a b` - Add the value of `a` to the value of `b`, then store the res in variable `a`.
- `mul a b` - Multiply the value of `a` by the value of `b`, then store th result in variable `a`.
- `div a b` - Divide the value of `a` by the value of `b`, truncate the re to an integer, then store the result in variable `a`. (Here, "trunca means to round the value toward zero.)
- `mod a b` - Divide the value of `a` by the value of `b`, then store the remainder in variable `a`. (This is also called the modulo operation
- `eql a b` - If the value of `a` and `b` are equal, then store the value variable `a`. Otherwise, store the value `0` in variable `a`.

In all of these instructions, `a` and `b` are placeholders; `a` will always the variable where the result of the operation is stored (one of `w`, `x`, or `z`), while `b` can be either a variable or a number. Numbers can be positive or negative, but will always be integers.

The ALU has no jump instructions; in an ALU program, every instruction run exactly once in order from top to bottom. The program halts after last instruction has finished executing.

(Program authors should be especially cautious; attempting to execute `d` with `b=0` or attempting to execute `mod` with `a<0` or `b<=0` will cause the program to crash and might even damage the ALU. These operations are n intended in any serious ALU program.)

For example, here is an ALU program which takes an input number, negat it, and stores it in `x`:

```
inp x
mul x -1
```

Here is an ALU program which takes two input numbers, then sets `z` to `1` the second input number is three times larger than the first input numb or sets `z` to `0` otherwise:

```
inp z
inp x
mul z 3
eql z x
```

Here is an ALU program which takes a non-negative integer as input, converts it into binary, and stores the lowest (1's) bit in `z`, the seco lowest (2's) bit in `y`, the third-lowest (4's) bit in `x`, and the fourth- lowest (8's) bit in `w`:

```
inp w
add z w
mod z 2
div w 2
add y w
mod y 2
div w 2
add x w
mod x 2
div w 2
mod w 2
```

Once you have built a replacement ALU, you can install it in the subma which will immediately resume what it was doing when the ALU failed: validating the submarine's **model number**. To do this, the ALU will run MOdel Number Automatic Detector program (MONAD, your puzzle input).

Submarine model numbers are always **fourteen-digit numbers** consisting o of digits `1` through `9`. The digit `0` **cannot** appear in a model number.

When MONAD checks a hypothetical fourteen-digit model number, it uses fourteen separate `inp` instructions, each expecting a **single digit** of t model number in order of most to least significant. (So, to check the number `13579246899999`, you would give `1` to the first `inp` instruction, the second `inp` instruction, `5` to the third `inp` instruction, and so on. This means that when operating MONAD, each input instruction should on ever be given an integer value of at least `1` and at most `9`.

Then, after MONAD has finished running all of its instructions, it wil indicate that the model number was **valid** by leaving a `0` in variable `z`. However, if the model number was **invalid**, it will leave some other non

value in `z`.

MONAD imposes additional, mysterious restrictions on model numbers, and
legend says the last copy of the MONAD documentation was eaten by a tar
You'll need to figure out what MONAD does some other way.

To enable as many submarine features as possible, find the largest valid
fourteen-digit model number that contains no `0` digits. What is the larg
model number accepted by MONAD?

Your puzzle answer was `36969794979199`.

--- Part Two ---

As the submarine starts booting up things like the Retro Encabulator, y
realize that maybe you don't need all these submarine features after al

What is the smallest model number accepted by MONAD?

Your puzzle answer was `11419161313147`.

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

At this point, you should return to your Advent calendar and try anothe

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

You can also [Share] this puzzle.