--- Day 2: 1202 Program Alarm ---

On the way to your gravity assist around the Moon, your ship computer beeps angrily about a "1202 program alarm". On the radio, an Elf is already explaining how to handle the situation: "Don't worry, that's perfectly norma--" The ship computer bursts into flames.

You notify the Elves that the computer's magic smoke seems to have escaped. "That computer ran Intcode programs like the gravity assist program it was working on; surely there are enough spare parts up there to build a new Intcode computer!"

An Intcode program is a list of integers separated by commas (like `1,0,0,3,99`). To run one, start by looking at the first integer (called position `0`). Here, you will find an opcode - either `1`, `2`, or `99`. The opcode indicates what to do; for example, `99` means that the program is finished and should immediately halt. Encountering an unknown opcode means something went wrong.

Opcode `1` adds together numbers read from two positions and stores the result in a third position. The three integers immediately after the opcode tell you these three positions - the first two indicate the positions from which you should read the input values, and the third indicates the position at which the output should be stored.

For example, if your Intcode computer encounters `1,10,20,30`, it should read the values at positions `10` and `20`, add those values, and then overwrite the value at position `30` with their sum.

Opcode `2` works exactly like opcode `1`, except it multiplies the two inputs instead of adding them. Again, the three integers after the opcode indicate where the inputs and outputs are, not their values.

Once you're done processing an opcode, move to the next one by stepping forward `4` positions.

For example, suppose you have the following program:

```
1,9,10,3,2,3,11,0,99,30,40,50
```

For the purposes of illustration, here is the same program split into multiple lines:

```
1,9,10,3,
2,3,11,0,
99,
30,40,50
```

The first four integers, `1,9,10,3`, are at positions `0`, `1`, `2`, and `3`. Together, they represent the first opcode (`1`, addition), the positions of the two inputs (`9` and `10`), and the position of the output (`3`). To handle this opcode, you first need to get the values at the input positions: position `9` contains `30`, and position `10` contains `40`. Add these numbers together to get `70`. Then, store this value at the output position; here, the output position (`3`) is at position `3`, so it overwrites itself. Afterward, the program looks like this:

```
1,9,10,70,
2,3,11,0,
99,
30,40,50
```

Step forward `4` positions to reach the next opcode, `2`. This opcode works just like the previous, but it multiplies instead of adding. The inputs are at positions `3` and `11`; these positions contain `70` and `50` respectively. Multiplying these produces `3500`; this is stored at position `0`:

```
3500,9,10,70,
2,3,11,0,
99,
30,40,50
```

Stepping forward `4` more positions arrives at opcode `99`, halting the program.

Here are the initial and final states of a few more small programs:

- `1,0,0,0,99` becomes `2,0,0,0,99` (`1 + 1 = 2`).
- `2,3,0,3,99` becomes `2,3,0,6,99` (`3 * 2 = 6`).
- `2,4,4,5,99,0` becomes `2,4,4,5,99,9801` (`99 * 99 = 9801`).
- `1,1,1,4,99,5,6,0,99` becomes `30,1,1,4,2,5,6,0,99`.

Once you have a working computer, the first step is to restore the gravity assist program (your puzzle input) to the "1202 program alarm" state it had just before the last computer caught fire. To do this, **before running the program**, replace position `1` with the value `12` and replace position `2` with the value `2`. **What value is left at position `0`** after the program halts?

Your puzzle answer was `11590668`.

--- Part Two ---

"Good, the new computer seems to be working correctly! **Keep it nearby** during this mission - you'll probably use it again. Real Intcode computers support many more features than your new one, but we'll let you know what they are as you need them."

"However, your current priority should be to complete your gravity assist around the Moon. For this mission to succeed, we should settle on some terminology for the parts you've already built."

Intcode programs are given as a list of integers; these values are used as the initial state for the computer's **memory**. When you run an Intcode program, make sure to start by initializing memory to the program's values. A position in memory is called an **address** (for example, the first value in memory is at "address 0").

Opcodes (like `1`, `2`, or `99`) mark the beginning of an **instruction**. The values used immediately after an opcode, if any, are called the instruction's **parameters**. For example, in the instruction `1,2,3,4`, `1` is the opcode; `2`, `3`, and `4` are the parameters. The instruction `99` contains only an opcode and has no parameters.

The address of the current instruction is called the **instruction pointer**; it starts at `0`. After an instruction finishes, the instruction pointer increases by **the number of values in the instruction**; until you add more

instructions to the computer, this is always `4` (`1` opcode + `3` parameters) for the add and multiply instructions. (The halt instruction would increase the instruction pointer by `1`, but it halts the program instead.)

"With terminology out of the way, we're ready to proceed. To complete the gravity assist, you need to **determine what pair of inputs produces the output `19690720`**."

The inputs should still be provided to the program by replacing the values at addresses `1` and `2`, just like before. In this program, the value placed in address `1` is called the **noun**, and the value placed in address `2` is called the **verb**. Each of the two input values will be between `0` and `99`, inclusive.

Once the program has halted, its output is available at address `0`, also just like before. Each time you try a pair of inputs, make sure you first **reset the computer's memory to the values in the program** (your puzzle input) - in other words, don't reuse memory from a previous attempt.

Find the input **noun** and **verb** that cause the program to produce the output `19690720`. **What is `100 * noun + verb`?** (For example, if `noun=12` and `verb=2`, the answer would be `1202`.)

Your puzzle answer was `2254`.

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.