--- Day 18: Duet ---

You discover a tablet containing some strange assembly code labeled simply "Duet". Rather than bother the sound card with it, you decide to run the code yourself. Unfortunately, you don't see any documentation, so you're left to figure out what the instructions mean on your own.

It seems like the assembly is meant to operate on a set of registers that are each named with a single letter and that can each hold a single integer. You suppose each register should start with a value of 0.

There aren't that many instructions, so it shouldn't be hard to figure out what they do. Here's what you determine:

- snd X plays a sound with a frequency equal to the value of X.
- set X Y sets register X to the value of Y.
- add X Y increases register X by the value of Y.
- mul X Y sets register X to the result of multiplying the value contained in register X by the value of Y.
- mod X Y sets register X to the remainder of dividing the value contained in register X by the value of Y (that is, it sets X to the result of X modulo Y).
- rcv X recovers the frequency of the last sound played, but only when the value of X is not zero. (If it is zero, the command does nothing.)
- jgz X Y jumps with an offset of the value of Y, but only if the value of X is greater than zero. (An offset of 2 skips the next instruction, an offset of -1 jumps to the previous instruction, and so on.)

Many of the instructions can take either a register (a single letter) or a number. The value of a register is the integer it contains; the value of a number is that number.

After each jump instruction, the program continues with the instruction to which the jump jumped. After any other instruction, the program continues with the next instruction. Continuing (or jumping) off either end of the program terminates it.

For example:

```
set a 1
add a 2
mul a a
mod a 5
snd a
set a 0
rcv a
jgz a -1
set a 1
jgz a -2
```

- The first four instructions set a to 1, add 2 to it, square it, and then set it to itself modulo 5, resulting in a value of 4.
- Then, a sound with frequency 4 (the value of a) is played.
- After that, a is set to 0, causing the subsequent rcv and jgz instructions to both be skipped (rcv because a is 0, and jgz because a is not greater than 0).
- Finally, a is set to 1, causing the next jgz instruction to activate, jumping back two instructions to another jump, which jumps again to the rcv, which ultimately triggers the recover operation.

At the time the recover operation is executed, the frequency of the last sound played is 4.

What is the value of the recovered frequency (the value of the most recently played sound) the first time a rcv instruction is executed with a non-zero value?

To begin, get your puzzle input.

Answer: [Submit]

You can also [Share] this puzzle.