Advent of Code   [About] [Events] [Shop] [Settings] [Log Out]   wiwiwashere 34*
      λy.2024   [Calendar] [AoC++] [Sponsors] [Leaderboard] [Stats]

--- Day 17: Chronospatial Computer ---

The Historians push the button on their strange device, but this time, you
all just feel like you're falling.

"Situation critical", the device announces in a familiar voice.
"Bootstrapping process failed. Initializing debugger...."

The small handheld device suddenly unfolds into an entire computer! The
Historians look around nervously before one of them tosses it to you.

This seems to be a 3-bit computer: its program is a list of 3-bit numbers
(0 through 7), like 0,1,2,3. The computer also has three registers named A,
B, and C, but these registers aren't limited to 3 bits and can instead hold
any integer.

The computer knows eight instructions, each identified by a 3-bit number
(called the instruction's opcode). Each instruction also reads the 3-bit
number after it as an input; this is called its operand.

A number called the instruction pointer identifies the position in the
program from which the next opcode will be read; it starts at 0, pointing
at the first 3-bit number in the program. Except for jump instructions, the
instruction pointer increases by 2 after each instruction is processed (to
move past the instruction's opcode and its operand). If the computer tries
to read an opcode past the end of the program, it instead halts.

So, the program 0,1,2,3 would run the instruction whose opcode is 0 and
pass it the operand 1, then run the instruction having opcode 2 and pass it
the operand 3, then halt.

There are two types of operands; each instruction specifies the type of its
operand. The value of a literal operand is the operand itself. For example,
the value of the literal operand 7 is the number 7. The value of a combo
operand can be found as follows:

  - Combo operands 0 through 3 represent literal values 0 through 3.
  - Combo operand 4 represents the value of register A.
  - Combo operand 5 represents the value of register B.
  - Combo operand 6 represents the value of register C.
  - Combo operand 7 is reserved and will not appear in valid programs.

The eight instructions are as follows:

The adv instruction (opcode 0) performs division. The numerator is the
value in the A register. The denominator is found by raising 2 to the power
of the instruction's combo operand. (So, an operand of 2 would divide A by
4 (2^2); an operand of 5 would divide A by 2^B.) The result of the division
operation is truncated to an integer and then written to the A register.

The bxl instruction (opcode 1) calculates the **bitwise XOR** of register B and
the instruction's literal operand, then stores the result in register B.

The bst instruction (opcode 2) calculates the value of its combo operand
modulo 8 (thereby keeping only its lowest 3 bits), then writes that value
to the B register.

The jnz instruction (opcode 3) does nothing if the A register is 0.
However, if the A register is not zero, it jumps by setting the instruction
pointer to the value of its literal operand; if this instruction jumps, the
instruction pointer is not increased by 2 after this instruction.

The bxc instruction (opcode 4) calculates the bitwise XOR of register B and register C, then stores the result in register B. (For legacy reasons, this instruction reads an operand but ignores it.)

The out instruction (opcode 5) calculates the value of its combo operand modulo 8, then outputs that value. (If a program outputs multiple values, they are separated by commas.)

The bdv instruction (opcode 6) works exactly like the adv instruction except that the result is stored in the B register. (The numerator is still read from the A register.)

The cdv instruction (opcode 7) works exactly like the adv instruction except that the result is stored in the C register. (The numerator is still read from the A register.)

Here are some examples of instruction operation:

- If register C contains 9, the program 2,6 would set register B to 1.
- If register A contains 10, the program 5,0,5,1,5,4 would output 0,1,2.
- If register A contains 2024, the program 0,1,5,4,3,0 would output 4,2,5,6,7,7,7,3,1,0 and leave 0 in register A.
- If register B contains 29, the program 1,7 would set register B to 26.
- If register B contains 2024 and register C contains 43690, the program 4,0 would set register B to 44354.

The Historians' strange device has finished initializing its debugger and is displaying some information about the program it is trying to run (your puzzle input). For example:

```
Register A: 729
Register B: 0
Register C: 0

Program: 0,1,5,4,3,0
```

Your first task is to determine what the program is trying to output. To do this, initialize the registers to the given values, then run the given program, collecting any output produced by out instructions. (Always join the values produced by out instructions with commas.) After the above program halts, its final output will be 4,6,3,5,6,3,5,2,1,0.

Using the information provided by the debugger, initialize the registers to the given values, then run the program. Once it halts, what do you get if you use commas to join the values it output into a single string?

Your puzzle answer was 2,0,4,2,7,0,1,0,3.

--- Part Two ---

Digging deeper in the device's manual, you discover the problem: this program is supposed to output another copy of the program! Unfortunately, the value in register A seems to have been corrupted. You'll need to find a new value to which you can initialize register A so that the program's output instructions produce an exact copy of the program itself.

For example:

```
Register A: 2024
Register B: 0
Register C: 0

Program: 0,3,5,4,3,0
```

This program outputs a copy of itself if register A is instead initialized to 117440. (The original initial value of register A, 2024, is ignored.)

What is the lowest positive initial value for register A that causes the program to output a copy of itself?

Your puzzle answer was 265601188299675.

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.