Advent of Code [About] [Events] [Shop] [Settings] [Log Out] Sergio Tallo Torres 20* int y=2022; --- Day 10: Cathode-Ray Tube ---You avoid the ropes, plunge into the river, and swim to shore. Code possible: The Elves yell something about meeting back up with them upriver, but the Axis - All we river is too loud to tell exactly what they're saying. They finish want for crossing the bridge and disappear from view. Christmas is your application, pls! Situations like this must be why the Elves prioritized getting the communication system on your handheld device working. You pull it out of <embedded, cloud,</pre> your pack, but the amount of water slowly draining from a big crack in Machine learning, its screen tells you it probably won't be of much immediate use. fullstack> our cameras require Unless, that is, you can design a replacement for the device's video it all system! It seems to be some kind of cathode-ray tube screen and simple CPU that are both driven by a precise clock circuit. The clock circuit ticks at a constant rate; each tick is called a cycle. Start by figuring out the signal being sent by the CPU. The CPU has a single register, X, which starts with the value 1. It supports only two instructions: - addx V takes two cycles to complete. After two cycles, the X register is increased by the value \overline{V} . (\overline{V} can be negative.) - noop takes one cycle to complete. It has no other effect. The CPU uses these instructions in a program (your puzzle input) to, somehow, tell the screen what to draw. Consider the following small program: noop addx 3 addx -5 Execution of this program proceeds as follows: - At the start of the first cycle, the noop instruction begins execution. During the first cycle, X is 1. After the first cycle, the noop instruction finishes execution, doing nothing. - At the start of the second cycle, the addx 3 instruction begins execution. During the second cycle, X is still 1. - During the third cycle, X is still 1. After the third cycle, the addx 3 instruction finishes execution, setting X to 4. - At the start of the fourth cycle, the addx -5 instruction begins execution. During the fourth cycle, X is still 4. - During the fifth cycle, X is still 4. After the fifth cycle, the addx -5 instruction finishes execution, setting X to -1. Maybe you can learn something by looking at the value of the X register throughout execution. For now, consider the signal strength (the cycle number multiplied by the value of the \overline{X} register) during the 20th cycle and every 40 cycles after that (that is, during the 20th, 60th, 100th, 140th, 180th, and 220th cycles). For example, consider this larger program: addx 15 addx -11 addx 6 addx -3 addx 5 addx -1 laddx -8 addx 13 addx 4 noop addx −1 addx 5 addx -1 addx 5 addx -1 addx 5 addx -1 addx 5 addx -1 addx -35 addx 1 addx 24 addx -19 addx 1 addx 16 addx -11 noop noop addx 21 addx -15 noop noop addx -3 addx 9 addx 1 addx -3 addx 8 addx 1 addx 5 noop noop noop noop noop addx -36 noop addx 1 addx 7 noop noop noop addx 2 addx 6 noop noop noop noop noop addx 1 noop noop addx 7 addx 1 noop addx -13 addx 13 addx 7 noop addx 1 addx -33 noop noop noop addx 2 noop noop noop addx 8 noop addx -1 addx 2 addx 1 noop addx 17 addx -9 addx 1 addx 1 addx -3 addx 11 noop noop addx 1 noop addx 1 noop noop addx -13 addx -19 addx 1 addx 3 addx 26 addx -30 addx 12 addx -1 addx 3 addx 1 noop noop noop addx -9 addx 18 addx 1 addx 2 noop noop addx 9 noop noop noop addx -1 addx 2 addx -37 addx 1 addx 3 noop addx 15 addx -21 addx 22 addx -6 laddx 1 noop addx 2 addx 1 noop addx -10 noop noop addx 20 addx 1 laddx 2 laddx 2 laddx -6 addx -11 noop noop noop The interesting signal strengths can be determined as follows: - During the 20th cycle, register X has the value 21, so the signal strength is 20 * 21 = 420. (The 20th cycle occurs in the middle of the second addx -1, so the value of register X is the starting value, 1, plus all of the other addx values up to that point: 1 + 15 - 11 + 6 - 3 + 5 - 1 - 8 + 13 + 4 = 21.- During the 60th cycle, register X has the value 19, so the signal strength is 60 * 19 = 1140. - During the 100th cycle, register X has the value 18, so the signal strength is 100 * 18 = 1800. - During the 140th cycle, register X has the value 21, so the signal strength is $140 \times 21 = 2940$. - During the 180th cycle, register X has the value 16, so the signal strength is $180 \times 16 = 2880$. - During the 220th cycle, register X has the value 18, so the signal strength is $220 \times 18 = 3960$. The sum of these signal strengths is 13140. Find the signal strength during the 20th, 60th, 100th, 140th, 180th, and 220th cycles. What is the sum of these six signal strengths? Your puzzle answer was 12840. --- Part Two ---It seems like the $\overline{\mathsf{X}}$ register controls the horizontal position of a sprite. Specifically, the sprite is 3 pixels wide, and the \overline{X} register sets the horizontal position of the middle of that sprite. (In this system, there is no such thing as "vertical position": if the sprite's horizontal position puts its pixels where the CRT is currently drawing, then those pixels will be drawn.) You count the pixels on the CRT: 40 wide and 6 high. This CRT screen draws the top row of pixels left-to-right, then the row below that, and so on. The left-most pixel in each row is in position 0, and the rightmost pixel in each row is in position 39. Like the CPU, the CRT is tied closely to the clock circuit: the CRT draws a single pixel during each cycle. Representing each pixel of the screen as a #, here are the cycles during which the first and last pixel in each row are drawn: Cycle So, by carefully timing the CPU instructions and the CRT drawing operations, you should be able to determine whether the sprite is visible the instant each pixel is drawn. If the sprite is positioned such that one of its three pixels is the pixel currently being drawn, the screen produces a lit pixel (#); otherwise, the screen leaves the pixel dark (...)The first few pixels from the larger example above are drawn as follows: Start cycle 1: begin executing addx 15 During cycle 1: CRT draws pixel in position 0 Current CRT row: # During cycle 2: CRT draws pixel in position 1 Current CRT row: ## End of cycle 2: finish executing addx 15 (Register X is now 16) Sprite position:###......###.... Start cycle 3: begin executing addx -11 During cycle 3: CRT draws pixel in position 2 Current CRT row: ##. During cycle 4: CRT draws pixel in position 3 Current CRT row: ##.. End of cycle 4: finish executing addx -11 (Register X is now 5) Sprite position:###........................ Start cycle 5: begin executing addx 6 During cycle 5: CRT draws pixel in position 4 Current CRT row: ##..# During cycle 6: CRT draws pixel in position 5 Current CRT row: ##..## End of cycle 6: finish executing addx 6 (Register X is now 11) Sprite position:###......... Start cycle 7: begin executing addx -3 During cycle 7: CRT draws pixel in position 6 Current CRT row: ##..##. During cycle 8: CRT draws pixel in position 7 Current CRT row: ##..##.. End of cycle 8: finish executing addx -3 (Register X is now 8) Sprite position:###................. Start cycle 9: begin executing addx 5 During cycle 9: CRT draws pixel in position 8 Current CRT row: ##..##..# During cycle 10: CRT draws pixel in position 9 Current CRT row: ##..##..## End of cycle 10: finish executing addx 5 (Register X is now 13) Sprite position:###........ Start cycle 11: begin executing addx -1 During cycle 11: CRT draws pixel in position 10 Current CRT row: ##..##..##. During cycle 12: CRT draws pixel in position 11 Current CRT row: ##..##..##.. End of cycle 12: finish executing addx -1 (Register X is now 12) Sprite position:###........ Start cycle 13: begin executing addx -8 During cycle 13: CRT draws pixel in position 12 Current CRT row: ##..##..# During cycle 14: CRT draws pixel in position 13 Current CRT row: ##..##..## End of cycle 14: finish executing addx -8 (Register X is now 4) Start cycle 15: begin executing addx 13 During cycle 15: CRT draws pixel in position 14 Current CRT row: ##..##..##. During cycle 16: CRT draws pixel in position 15 Current CRT row: ##..##..##.. End of cycle 16: finish executing addx 13 (Register X is now 17) Sprite position:###.....###..... Start cycle 17: begin executing addx 4 During cycle 17: CRT draws pixel in position 16 Current CRT row: ##..##..##..# During cycle 18: CRT draws pixel in position 17 Current CRT row: ##..##..##..## End of cycle 18: finish executing addx 4 (Register X is now 21) Sprite position:###....###.... Start cycle 19: begin executing noop During cycle 19: CRT draws pixel in position 18 Current CRT row: ##..##..##..##. End of cycle 19: finish executing noop Start cycle 20: begin executing addx -1 During cycle 20: CRT draws pixel in position 19 Current CRT row: ##..##..##..##.. During cycle 21: CRT draws pixel in position 20 Current CRT row: ##..##..##..##..# End of cycle 21: finish executing addx -1 (Register X is now 20) Sprite position:###.....###.... Allowing the program to run to completion causes the CRT to produce the following image: ##..##..##..##..##..##..##..##..##.. ###...###...###...###...###...###...###.. ####....####....####....####.... ######....#####....#####.....#####.... #######.....######.....######...##### ########......#######.....######.... Render the image given by your program. What eight capital letters appear on your CRT? Your puzzle answer was ZKJFBJFZ. 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.