

[Advent of Code](#)
[\[About\]](#)
[\[AoC++\]](#)
[\[Events\]](#)
[\[Settings\]](#)
[\[Log Out\]](#)
 Roland Tritesch (AoC++) 34\*  
 0x0000|2017
 [\[Calendar\]](#)
[\[Leaderboard\]](#)
[\[Stats\]](#)
[\[Sponsors\]](#)

--- Day 15: Dueling Generators ---

Here, you encounter a pair of dueling generators. The generators, called generator A and generator B, are trying to agree on a sequence of numbers. However, one of them is malfunctioning, and so the sequences don't always match.

As they do this, a judge waits for each of them to generate its next value, compares the lowest 16 bits of both values, and keeps track of the number of times those parts of the values match.

The generators both work on the same principle. To create its next value, a generator will take the previous value it produced, multiply it by a factor (generator A uses `16807`; generator B uses `48271`), and then keep the remainder of dividing that resulting product by `2147483647`. That final remainder is the value it produces next.

To calculate each generator's first value, it instead uses a specific starting value as its "previous value" (as listed in your puzzle input).

For example, suppose that for starting values, generator A uses `65`, while generator B uses `8921`. Then, the first five pairs of generated values are:

--Gen. A--	--Gen. B--
1092455	430625591
1181022009	1233683848
245556042	1431495498
1744312007	137874439
1352636452	285222916

In binary, these pairs are (with generator A's value first in each pair):

000000000000100001010101101100111	00011001101010101101001100110111
01000110011001001111011100111001	01001001100010001000010110001000
00001110101000101110001101001010	01010101010100101110001101001010
01100111111110000001011011000111	00001000001101111100110000000111
01010000100111111001100000100100	000100010000000000010100000000100

Here, you can see that the lowest (here, rightmost) 16 bits of the third value match: `1110001101001010`. Because of this one match, after processing these five pairs, the judge would have added only `1` to its total.

To get a significant sample, the judge would like to consider 40 million pairs. (In the example above, the judge would eventually find a total of `588` pairs that match in their lowest 16 bits.)

After 40 million pairs, what is the judge's final count?

Your puzzle answer was `594`.

--- Part Two ---

In the interest of trying to align a little better, the generators get more picky about the numbers they actually give to the judge.

Our [sponsors](#) help make Advent of Code possible:

[SmartyStreets](#) -  
 U2VuZGluZyBDaH  
 Jpc3RtYXMGy2Fy  
 ZHMgdG8gYmFkIG  
 FkZHJlc3Nlc38K

By popular demand, there are now AoC-themed objects available (until Jan. 3rd). Get them shipped from the US or from Europe.

They still generate values in the same way, but now they only hand a value to the judge when it meets their criteria:

- Generator A looks for values that are multiples of `4`.
- Generator B looks for values that are multiples of `8`.

Each generator functions completely independently: they both go through values entirely on their own, only occasionally handing an acceptable value to the judge, and otherwise working through the same sequence of values as before until they find one.

The judge still waits for each generator to provide it with a value before comparing them (using the same comparison method as before). It keeps track of the order it receives values; the first values from each generator are compared, then the second values from each generator, then the third values, and so on.

Using the example starting values given above, the generators now produce the following first five values each:

--Gen. A--	--Gen. B--
1352636452	1233683848
1992081072	862516352
530830436	1159784568
1980017072	1616057672
740335192	412269392

These values have the following corresponding binary values:

01010000100111111001100000100100	01001001100010001000010110001000
0111011010111100101111010110000	00110011011010001111010010000000
00011111101000111101010001100100	01000101001000001110100001111000
01110110000001001010100110110000	01100000010100110001010101001000
00101100001000001001111001011000	00011000100100101011101101010000

Unfortunately, even though this change makes more bits similar on average, none of these values' lowest 16 bits match. Now, it's not until the 1056th pair that the judge finds the first match:

--Gen. A--	--Gen. B--
1023762912	896885216
00111101000001010110000111100000	00110101011101010110000111100000

This change makes the generators much slower, and the judge is getting impatient; it is now only willing to consider 5 million pairs. (Using the values from the example above, after five million pairs, the judge would eventually find a total of `309` pairs that match in their lowest 16 bits.)

After 5 million pairs, but using this new generator logic, what is the judge's final count?

Your puzzle answer was `328`.

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.