Problem Set 1 - General

Question C

I remember Raveen telling me that Problem Set 1 would be a good one to do because it allows you to get more into the actual act of coding, as opposed to the rest of the course, where you just kind of plug things into a black box (i.e. network flow….).

And this question felt a lot like that, and was quite a cheeky one to code. The actual methodology wasn't the easiest thing to get either. I think the first thought I went down was that numbers a and d probably had an impact on the number of 0's and 1's we needed in the final string. After some thinking, it wasn't hard to figure out that the number of 0's needed to satisfy A was ANS such that:

```
// ANSc2 = a
```

The same follows for b. If a and b are not whole numbers from this, it is not possible to construct a string.

But the harder part was understanding how to order them based on the b and c, in order to satisfy the conditions of b and c. Lots of racking my brain about, if we were to arrange the 1's all on the left, and the 0's all on the right, what that would mean, and if we were to shift a 1 into the territory of 0's what that would mean.

The key thing to get here, was that there were more cases where we weren't able to create a string than not. The number of 01 and 10 subsequences (given by b and c) had to match the product of the number of 0s and 1s (num_0 * num_1). If this wasn't satisfied, it was also impossible to construct the string, because it would imply an imbalance in how the subsequences could be distributed.

Once this product check is complete, also notice that once we've gotten the 01 count down properly, the 10 counts will follow. If you think about it, everytime we put down a 0, we guarantee that all following 1's will contribute to the count of 01's. So, if we haven't put down any 1's yet, each 0 will contribute num_1's amount of 01 sequences.

So, just need to know how many initial 0's to put down (how many times num_1 fits into b). Once we've put down the 0's we've guaranteed all but the remainder of b / num_1 01 sequences handled. We just need to put down 1s until putting down the next 0 will cover the remainder of b / num_1. This ensures that we've satisfied all the 01 sequences. Once the remainder is handled, we can pad the string with the remaining 1s followed by the remaining 0s.

Finally, we handle some edge cases if b and c are 0, and if a and d are 0, prior to the string constructions, and then we are complete.