Week 1 C

This is a combinatorial problem. Notice that a, d must be the number of 0s and 1s respectively choose 2, i.e.

$$a = \binom{n_0}{2}, b = \binom{n_0}{2}$$

. However checking if a, b are combinational numers are slow (equivalent to finding square root), we could instead do a single binary search on a, then utilize the fact that $b + c = n_0 \times n_1$, since the total pairs of 10 and 01 are equal to the cross product of number of 0s and 1s. A sum and division is much faster than O(log n) for finding d. However this has created problems as I discovered in debugging.

Since my method of calculation bases entirely on a, the output string (if it exists) will be constructed as 5 parts,

- Blocks of 0s, the number of 0s is the quotient of b divided by n_1
- Block of 1s, the number of 1s is the difference between total number of 1s and the remainder of b divided by n_1,
- A singular 0, followed by
- Block of 1s, the length is the the remainder of b divided by n_1,
- Block of 0s, the length is the number of 0s minus all the 0s beforehand (1st and 3rd block)

The above string construction method is in the style of "brute force" since the question accepts any valid string. For any 0 that occur ahead of every 1, the numer of 01 increases by n_1. Therefore i could measure the length of "block 1" above by quotient of the division above. The remainder of the division will be the number of 1 that has a non-leading 0 somewhere between block 2 and 4. The lenght of block 4 is equivalent to the remainder, and therefore block 2 and 3 are formed as is. Finally block 5 is all the zeroes that are left over and could be accounted for.

One of the first bugs in submission was the runtime error, which was discovered to be a division by 0, when a or d happen to be 0 at calculating the above division. This also connected out a whole string of edge cases, for example we could have a valid $\{0, 0, 0, 1\}$ input, which is 11, a $\{1, 1, 1, 0\}$ which is 010, and even $\{0, 0, 0, 0\}$ which is either 0 or 1. I have hence integrated a whole edge cases for cases of a or d being 0. This is a forseen side effect of using only a as the basis of calculations.

And a final runtime error occured at string construction, where a -1 length string could be assigned at block 5 due to my calculation. The edge case here is that if $a == b$, i.e. the string is consisted of purely a block of 0 followed by a block of 1.

Overall this problem is very fun to attempt, also a starter on what debugging hell will be like for later questions (as I am writing this, the completed Pset 2 C).

Apart from combinatorial way to construct the string, a purely algorithmic way to construct the strings are also viable (which is how the output in the samples are constructed), where we would place 0 and 1 based on if a or b is larger than each other, or if one has depleted. This method wouldn't be implemented.

The runtime of the algorithm would be O(log n + n), from the binary search and physical construction of the string.