1. Further bit checking
   a. Increase Hamming distance between valid code words with more parity bits
   b. Example: 4-bit word

7	6	5	4	3	2	1	Bit position
D3	D2	D1	C2	D0	C1	C0	Bit type (D = data, C = check / parity)

      i. C0 is the parity bit over bits 3, 5, 7
      ii. C1 is the parity bit over bits 3, 6, 7
      iii. C2 is the parity bit over bits 5, 6, 7
   c. Original data: 0110. Let's calculate the code word associated with it.
      i. Fill in table, then calculate check bits

7	6	5	4	3	2	1
0	1	1	C2	0	C1	C0

      ii. For C0, bits 3, 5, 7 are 0, 1, 0. XOR(010) = 1, so C0 = 1.
      iii. For C1, bits 3, 6, 7 are 0, 1, 0. XOR(010) = 1, so C1 = 1.
      iv. For C2, bits 5, 6, 7 are 1, 1, 0. XOR(110) = 0, so C2 = 0.
      v. Putting these together, we get 0110011.
   d. Let's flip one of the bits now. Is 0010011 valid?
      i. Fill in the table, then verify check bits

7	6	5	4	3	2	1
D3	D2	D1	C2	D0	C1	C0
0	0	1	0	0	1	1

      ii. For C0, bits 3, 5, 7 are 0, 1, 0. XOR(010) = 1, so C0 = 1.
      iii. For C1, bits 3, 6, 7 are 0, 0, 0. XOR(000) = 0, so C1 = 0.
      iv. For C2, bits 5, 6, 7 are 1, 0, 0. XOR(100) = 1, so C2 = 1.
      v. We have a mismatch with C1 and C2, thus the codeword was invalid.
         1. We calculated C1 = 0, but the bit received was 1.
         2. Same applies for C2 = 1, but bit received was 0.
   e. Error correction – we know there was an error, how do we fix it?
      i. XOR the generated check bits with the check bits from the received word
      ii. Result tells us exactly where the error occurred
      iii. This is possible because of the way we've laid out the check bits
         1. This is also why we started with 1 instead of 0 when numbering the bits
      iv. XORing each bit together:

Received:	011
Calculated:	<u>101</u>
XOR:	110

      v. So we know bit position 6 was the error, we correct that one
         1. We get 0110011, which was our original code word before we flipped anything
         2. Extracting the data, we get 0110, which was our original data
2. Why this works
   a. Each data bit must be covered (checked) by at least 2 parity bits
      i. Doesn't matter which parity bits cover which data bits
      ii. However, doing it in the way we did above allows you to determine the error location easily
   b. Why we start numbering from 1
      i. Need a bit pattern that indicates there were no errors
      ii. If the recreated check bits match the original check bits, XOR returns 0
      iii. We use that position to indicate no errors
   c. Which parity bits check which data bits
      i. C0 is in location 1

        1. Checks all bits with bit 1 high (except for itself)
        2. 3, 5, 7, 8, 11, 13, so on
    ii. C1 is in location 2
        1. Checks all bits with bit 2 high (except for itself)
        2. 3, 6, 7, 10, 11, 14, 15, 18, 19, so on
    iii. C2 is in location 4
        1. Checks all bits with bit 4 high (except for itself)
        2. 5, 6, 7, 12, 13, 14, 15, 20, 21, so on
    iv. If we had a C3, it would be in location 8
        1. Would check all bits with bit 8 high (except for itself)
        2. 9, 10, 11, 12, 13, 14, 15, 24, 25, so on