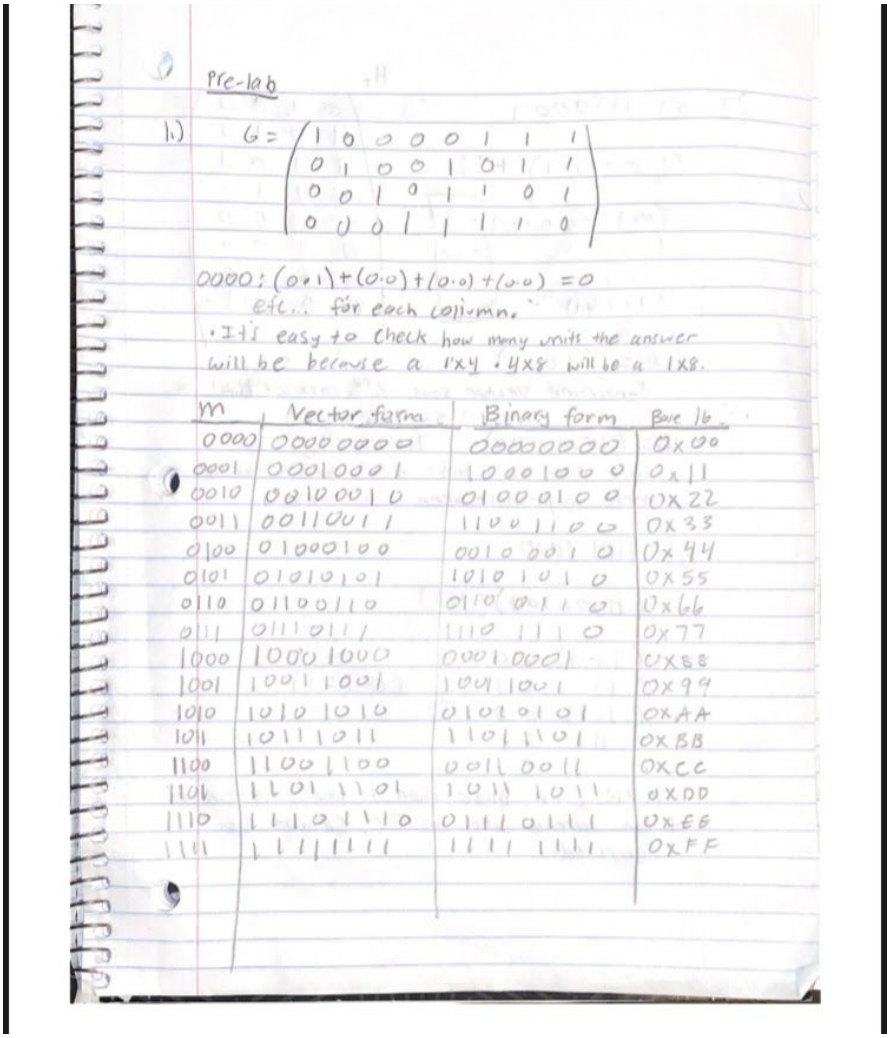
Version 1.0

Section 0: Pre-Lab

- Matrix Multiplication depends on how matrix columns and rows match. We need a 1x4 nibble times a 4x8 generator to generate a suitable code. The cols and rows MUST match, else the bounds will not work.



Pre-lab

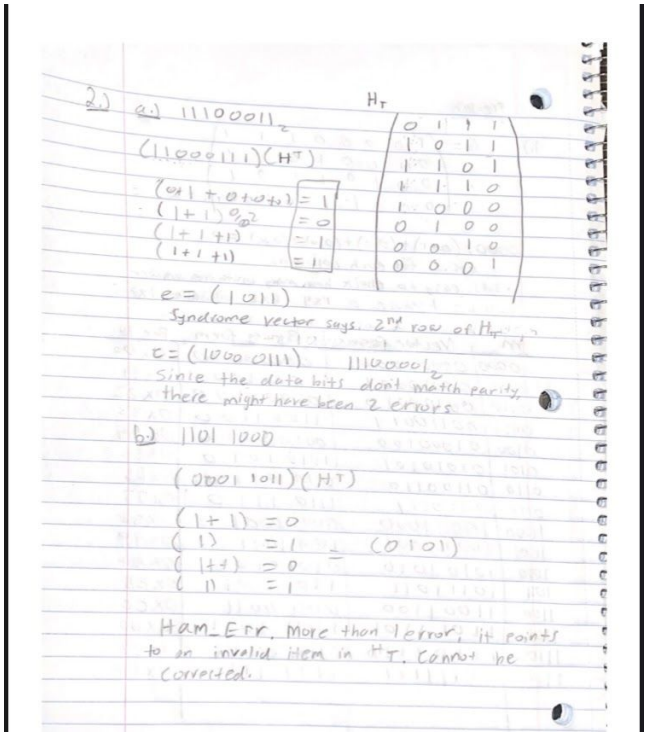
1.) $G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$

0000: $(0\cdot1)+(0\cdot0)+(0\cdot0)+(0\cdot0)=0$

etc.. for each column.

- It is easy to check how many units the answer will be because a 1x4 • 4x8 will be a 1x8.

m	Vector form	Binary form	Base 16
0000	00000000	00000000	0x00
0001	00010001	10001000	0x11
0010	00100010	01000100	0x22
0011	00110011	11001100	0x33
0100	01000100	00100010	0x44
0101	01010101	10101010	0x55
0110	01100110	01100110	0x66
0111	01110111	11101110	0x77
1000	10001000	00010001	0x88
1001	10011001	10011001	0x99
1010	10101010	01010101	0xAA
1011	10111011	11011101	0xBB
1100	11001100	00110011	0xCC
1101	11011101	10111011	0xDD
1110	11101110	01110111	0xEE
1111	11111111	11111111	0xFF



2.) a.) 11100011_2

$(11100011)(H^T)$

$(1+1+0+0+0) = 1$

$(1+1)\%2 = 0$

$(1+1+1) = 1$

$(1+1+1) = 1$

H_T

0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

$e = (1011)$

Syndrome vector says 2nd row of H_T

$c = (1000\ 0111)$ 11100001_2

Since the data bits dont match parity, there might have been 2 errors

b.) 1101 1000

(0001 1011)(H^T)

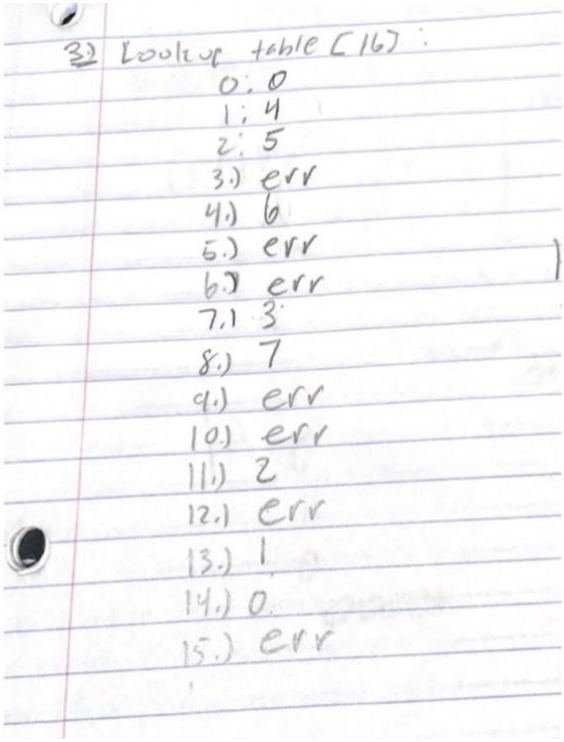
$(1+1) = 0$

$(1) = 1$ (0101)

$(1+1) = 0$

$(1) = 1$

Ham_Err. More than 1 error, it points to an invalid Hem in H_T. Cannot be corrected.



3.) Lookup table [16]:

0: 0
1: 4
2: 5
3.) err
4.) 6
5.) err
6.) err
7.) 3
8.) 7
9.) err
10.) err
11.) 2
12.) err
13.) 1
14.) 0
15.) err

Section 1: Bm.c

- Since there will be a lot of segmenting into the correct byte for this ADT, I created a function based off Euegene's idea in his section that takes in a certain number of bits and returns the correct number of bytes. This function in particular will be very useful for selecting the correct byte for a row.
- For bm_create, make sure to allocate the correct amount of memory for each row, not the number of cols rather the number of bytes necessary to fit the number of cols. 0-8 cols should correspond to 1 byte allocated.
- For set/clr/get bit apply binary manipulation. Create a manipulator byte, 0x01 which can be used to access a byte and return a value at a certain index
- For bm_print, pretty simple just get the bit at the iterator i,j combo.

Section 2: Hamming.c

- For practice purposes I'm going to create temporary ADT's G and H to test my code. Also add professors upper and lower nibble code, which is extremely handy for byte manipulation.
- Ham_init will create a 4x8 bitmap G and a 8x4 bitmap H

- To create G, iterate through the number of cols then 4 times which can be used for the generator matrix's 4 rows and the 4 bits of data. H is the same but swap some things.
- To encode the data,
    - First we get the lower nibble of the data byte
    - Create 2 variables: 1 to hold the result which will be or'd with the other variable which holds the value of the current matrix vector row/col multiplication.
- To decode the generated 8 bit hamming code
    - Multiply the inputed code by H transpose to get the syndrome vector
    - 8 times, for each row in H transpose, use bitwise manipulation to get the the value for the syndrome vector, and if it corresponds to a value in H transpose, the error is in the location at the row value in H transpose.

Section 3: Generator.c and Decoder.c

- After initializing the G and H matrices, Receive input via fgetc() and apply ham_encode and ham_decode respectively.
- For decoder.c, create variables for printing Ham_Err, Ham_err_ok, and Ham_ok to print to output.