Prelab Question:

1.

Decimal	Binary for original Decimal	Changed binary after mod 2	Decimal(index from matrix H based on changed binary)
0	0000	0000	Ham_ok
1	0001	1000	4
2	0010	0100	5
3	0011	1100	Ham_Error
4	0100	0010	6
5	0101	1010	Ham_Error
6	0110	1001	Ham_Error
7	0111	1110	3
8	1000	0001	7
9	1001	1001	Ham_Error
10	1010	0101	Ham_Error
11	1011	1101	2
12	1100	0011	Ham_Error
13	1101	1011	1
14	1110	0111	0
15	1111	1111	Ham_Error

2.

2a) $(1110\,0011)$ $\begin{pmatrix} 0&1&1&1\\ 1&0&1&1\\ 1&1&0&1\\ 1&1&1&0\\ 1&0&0&0\\ 0&1&0&0\\ 0&0&0&1 \end{pmatrix} = (2,2,3,4)$

mod 2

$= \boxed{(0,0,1,0)}$

matched

*

HAM_CORRECT

Fix: $(1110001\underline{1}) = (1110000\underline{0}1)$

2b) $(1101\,1000)$ $\begin{pmatrix} 0&1&1&1\\ 1&0&1&1\\ 1&1&0&1\\ 1&1&1&0\\ 1&0&0&0\\ 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1 \end{pmatrix} = (3,2,3,2)$

mod 2

$= \boxed{(1,0,1,0)}$

∴ doesn't appear in matrix

∴ HAM_ERR

Purpose of the lab:
Write encode and decode programs to check and correct errors for text caused by random bit-flip(noise). Specifically, we will multiply binary from a string in text with the G matrix provided in pdf to encode and multiply H matrix for decode. If decode result (0, 0, 0, 0) meaning no error, else if the result is 4 bits appear in the matrix, meaning it's possible to fix. Otherwise if the result 4 bits not 0,0,0,0 and not appear in the matrix, then it's an error and not possible to fix.

Command options:
Encode:
-h: print help message
-i: input for input file
-o: output for output file
Decode:
-h: print help message
-i: input for input file
-o: output for output file
-v: print the statics of error stat(Total bytes processed, uncorrected errors, corrected errors, and error rate)
Error:
-h program usage and help
-s input random seed
-e input error_rate

Files:
Decode.c: decode main function
Encode.c: encode main function
Error.c: provided file, help printing out errors rate
Bm.c: create memories for matrices, set up functions like matrix multiplication get bit ,set bit etc.
Bv.c: helper functions for bit matrx.c
Hamming.c: create functions to set up matrices and multiply the matrices

Pseudocode:
Encode: 1 string has 8 bits, encode will take 4 bits and multiply G to become 8 bits hamming code.
Decode: 0 will be correct, ht matrix 8 rows represent 8 bits hamming code. If decode result (0, 0, 0, 0) meaning no error, else if the result is 4 bits appear in the matrix, meaning it's possible to fix. Otherwise if the result 4 bits not 0,0,0,0 and not appear in the matrix, then it's an error and not possible to fix.

Pseudocode:
Bv.c:
Bv create:
Create pointer *bv with malloc

Initial vector memory
And length variable

bv length: just return length

Bv setbit (*v, i)
Location = i/8(because 8 bytes each group, check where the group is)
Position = i % 8 (check where the bit is in the group(8 bits 1 group))
Set_on = set 1 to the position(find the position, and set 1 to it)
Vector[location] or set_on to place the bit into right place

Bv clear bit
Location = i/8(because 8 bytes each group, check where the group is)
Position = i % 8 (check where the bit is in the group(8 bits 1 group))
Set_on = set 0 to the position(find the position, and set 0 to it), (set 0 means ~1)
Vector[location] & set_on to place the bit into right place

Bv get bit
Location = i/8(because 8 bytes each group, check where the group is)
Position = i % 8 (check where the bit is in the group(8 bits 1 group))
Get bit = set 1 to the position
If vector[location] & get bit = 0, then return 0, otherwise return 1(so we actually know 0 or 1 in that specific vector position)

Bv_xor_bits
Location = i/8(because 8 bytes each group, check where the group is)
Position = i % 8 (check where the bit is in the group(8 bits 1 group))
Setxor = 1 * bit to the position
Vector[location] ^- set xor to place the bit

Bm.c
Bm create:
Create pointer bm and use malloc to allocate memory
Initial variables rows, cols, and vector(with bv create(rows * cols))

Bm delete:
Delete the content in vector first, and then free the vector memory
Free pointer, set pointer to free

Bm row and bm cols, just return rows and cols

Bm set bit, bm get bit, and bm clear bit, just call bv functions

Bm from data:

Check if allocated memory successfully first, if not free memory
Create for loop with 8 (8 bits)
Call bv get bit
If bv get bit = 1
Set bit

Bm multiply(matrix A, and matrix B):
Pointer C = bm create (A rows, and B cols)
For loop for matrix A row i++
    For loop for matrix B cols
        For loop for matrix A cols
            Variable += bmgetbit A * bm getbit B
        Variable %=2
        If variable == 1
            Then bm set bit
Return C

Hamming.c
Create lookup table(check prelab)
Lookup[16] = {Ham OK, 4, 5 HamErr ……}

Ham_encode(Gmatrix, msg)
Bitmatrix Pointer A = bm from data(msg, 4)// get the bits
Bitmatrix Pointer B = bm multiply (A, G)//multiply the bits to matrix G
Return Bm_to data(b)

Ham_decode(matrix Ht, code, msg)
Bitmatrix Pointer A = bm from data(code, 8)// get the bits from encode
Bitmatrix Pointer B = bm multiply (A,Ht)//multiply the bits to matrix Ht
Use bm to data to check B
If !== 0
    If lookup table is not == Ham_err
        If bm get bit == 1, clear bit
        Else set bit
        Return hamcorrect
    Else return Ham_err

Return ham ok

Encode.c
Create 2x2 G matrix using array
Main function:

Command options for help message, input file, output file
Handle file permission code(fstat, fchmod)
Bm create, for loop rows and for loop cols, if array == 1, set bits(set bits for G into bit matrix)

Create for loop to encode(when inputfile not feof)
fuptc (read characters from input file)
1 character will separate to two part, both parts need to ham_encode(G, lowernibble/highnibble)
Fputc to outputfile
Close file and free memory

Decode.c
//same idea like encode.c
Create Ht matrix using 2x2 array first and then bm create and use for loop to set bits in bit matrix
Command line stuffs
Create for loop to read all characters in the file(encode outfile)
Total bytes+=2
Ham decode twice(prob create 2 variables) to collect 8 bits to form a character(4 bits each from encode)
Free memory, close files
If -v, print error stat