Pre-lab

$$G = \begin{pmatrix} 1000 & 0111 \\ 0100 & 1011 \\ 0010 & 1101 \\ 0001 & 1110 \end{pmatrix}$$

$\vec{c} = \vec{m} \cdot G$

from $0000_2 \rightarrow 1111_2$

1. $\vec{c}_{(0000)_2} = (00000000)$ $\quad \vec{c}_{(0001)_2} = (00011110)$ $\quad \vec{c}_{(0010)_2} = (00101101)$

$\vec{c}_{(0011)_2} = (00110011)$ $\quad \vec{c}_{(0100)_2} = (01001011)$ $\quad \vec{c}_{(0101)_2} = (01010101)$

$\vec{c}_{(0110)_2} = (01100110)$ $\quad \vec{c}_{(0111)_2} = (01111000)$ $\quad \vec{c}_{(1000)_2} = (10000111)$

$\vec{c}_{(1001)_2} = (10011001)$ $\quad \vec{c}_{(1010)_2} = (10101010)$ $\quad \vec{c}_{(1011)_2} = (10110100)$

$\vec{c}_{(1100)_2} = (11001100)$ $\quad \vec{c}_{(1101)_2} = (11010010)$ $\quad \vec{c}_{(1110)_2} = (11100001)$

$\vec{c}_{(1111)_2} = (11111111)$

pseudocode: m[rows][cols] G[rows][cols]

```
for ( int i=0 ; i < m_rows ; i++){
    for( int j=0 ; j< G_cols; j++){
        for (int k =0 ; k < m_cols; k++){
            c[i][j] += m[i][k] x G[k][i];
        }
    }
}
```

2. $\vec{e} = \vec{c} \cdot H^T$

$$H^T = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

a) $11100011_2 \Rightarrow \vec{c} = (11000111) \longrightarrow$ then $\vec{e} = (1011)$. This contains an error which can be matched up with $H^T[1]$.

(1st bit need to be flipped, and the original message becomes 0001_2)

So it can be corrected by flipping over the 1st bit. Return HAM_ERR_OK

b) $11011000_2 \Rightarrow \vec{c} = (00011011) \longrightarrow \vec{e} = (0101)$. This contains an error and cannot be fixed for there're more than one bits are flipped.

Return HAM_ERR.

Decoding pseudocode: $\vec{e}[0][cols]$ $H^T[rows][cols]$

```
int find;
for(int i=0; i<H^T_rows; i++){
    for(int j=0; j<H^T_cols; j++){
        if(e[0][j] != H^T[i][j]){
            find = -1;
            break the loop}
        find = i; ← the ith bit need to be flipped
    }
}
if(find == -1) {return HAM_ERR}
else{ check the ith bit's value (0 or 1)
    then use bm_set_bit() / bm_clr_bit() }
```

3.

0	0
1	4
2	5
3	HAM_ERR
4	6
5	HAM_ERR
6	HAM_ERR
7	3
8	7
9	HAM_ERR
10	HAM_ERR
11	2
12	HAM_ERR
13	1
14	0
15	HAM_ERR

$H^T =$

				index
0	1	1	1	0
1	0	1	1	1
1	1	0	1	2
1	1	1	0	3
1	0	0	0	4
0	1	0	0	5
0	0	1	0	6
0	0	0	1	7

Wendi Tan
wtan37@ucsc.edu
2/4/2021

CSE13s Winter 2021
Assignment 4: Hamming Codes
Design Document

Hamming codes are a family of linear error-correcting codes. Hamming codes can detect up to two-bit errors or correct one-bit errors without detection of uncorrected errors. By contrast, the simple parity code cannot correct errors, and can detect only an odd number of bits in error. Hamming codes are perfect codes, that is, they achieve the highest possible rate for codes with their block length and minimum distance of three.
(cite through https://en.wikipedia.org/wiki/Hamming_code)

In this lab, I suppose to write two programs in C, one for encoding hamming codes, and the other for decoding them. More specifically, the user is asked to enter any input and output files (*stdin* and *stdout* are default files). The encoding program would read bites through the given message in the input file and generate its corresponding hamming code, printed in the output file. Oppositely, the decoding program would decode hamming codes, identify errors, correct certain errors, and generate the original message.