William Santosa
wsantosa@ucsc.edu
7 January 2021

CSE13S Winter 2021
Assignment 4 : Hamming Codes
Design Document

PRELAB

1. Calculate Hamming codes for 0000_2–1111_2 using the generator matrix. Show your work.
   Hint: Convert your codes to hex.

Decimal	Binary	Hamming Code
0	0000	0000 0000
1	0001	1000 0111
2	0010	0100 1011
3	0011	1100 1100
4	0100	0010 1101
5	0101	1010 1010
6	0110	0110 0110
7	0111	1110 0001
8	1000	0001 1110
9	1001	1001 1001
10	1010	0101 0101
11	1011	1101 0010
12	1100	0011 0011
13	1101	1011 0100
14	1110	0111 1000
15	1111	1111 1111

2. Decode the following codes. If it contains an error, explain how you can correct it; however,some errors cannot be corrected.
   (a) 1110 0011_2
   (b) 1101 1000_2

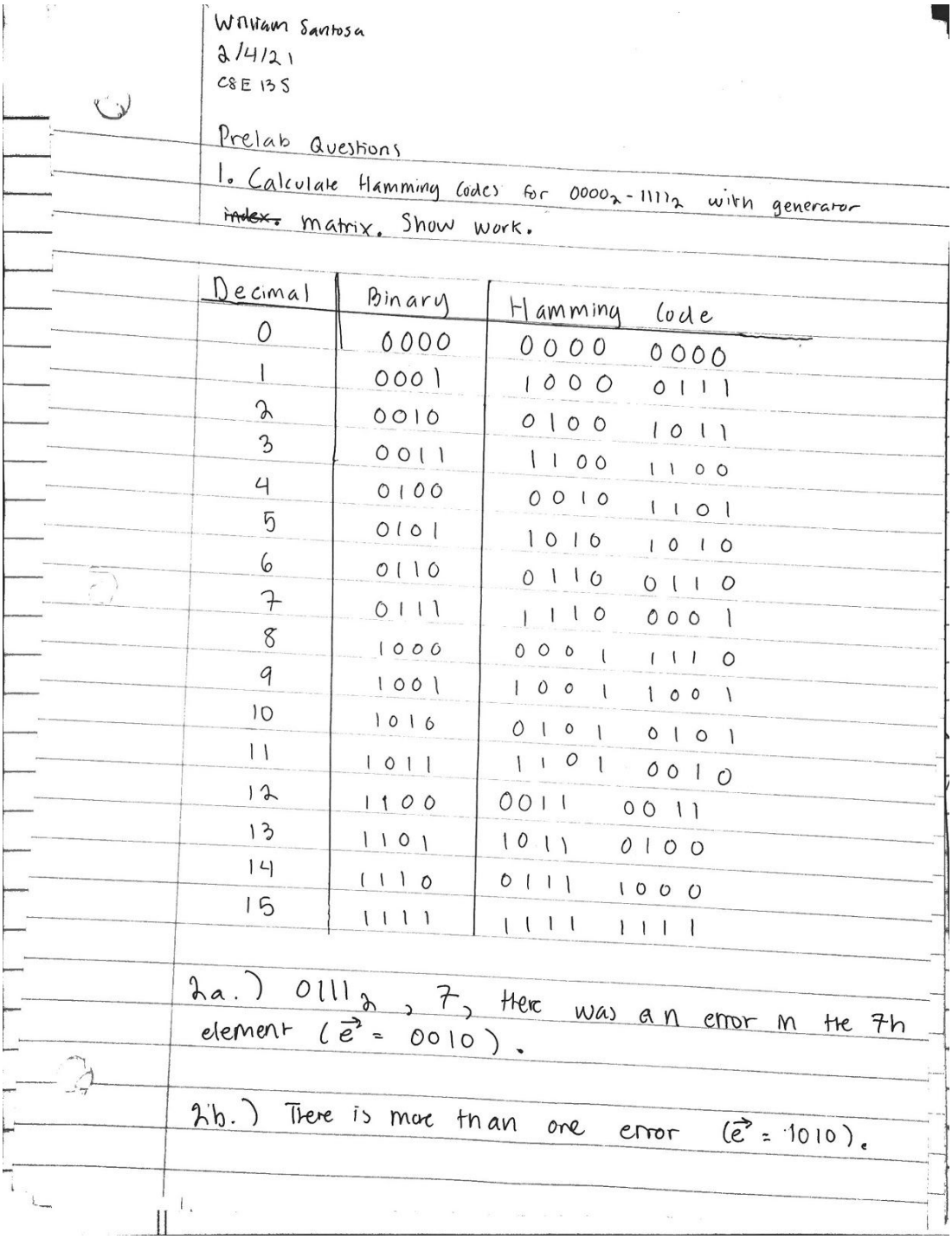
   2a. 0001_2 or 1 in decimal. There was an error in the 2nd element as e arrow = {1011}.

   2b. There is more than one error as e arrow = 0101 and cannot be corrected with my current understanding.

3. Complete the rest of the look-up table shown below.
   0 0
   1 4
   ... ...
   15 HAM_ERR

Decimal	Binary	Look-Up
0	0000	0
1	0001	4
2	0010	5
3	0011	HAM_ERR
4	0100	6
5	0101	HAM_ERR
6	0110	HAM_ERR
7	0111	3
8	1000	7
9	1001	HAM_ERR
10	1010	HAM_ERR
11	1011	2
12	1100	HAM_ERR
13	1101	1
14	1110	0
15	1111	HAM_ERR

WORK (Correction for Work #2)



William Santosa
2/4/21
CSE 135

Prelab Questions

1. Calculate Hamming codes for 0000_2 - 1111_2 with generator ~~index.~~ matrix. Show work.

Decimal	Binary	Hamming code
0	0000	0000 0000
1	0001	1000 0111
2	0010	0100 1011
3	0011	1100 1100
4	0100	0010 1101
5	0101	1010 1010
6	0110	0110 0110
7	0111	1110 0001
8	1000	0001 1110
9	1001	1001 1001
10	1010	0101 0101
11	1011	1101 0010
12	1100	0011 0011
13	1101	1011 0100
14	1110	0111 1000
15	1111	1111 1111

2a.) 0111_2, 7, there was an error in the 7th element ($\vec{e}$ = 0010).

2b.) There is more than one error ($\vec{e}$ = 1010).

William Santosa
2/14/21
CSE135

Prelab Questions

3.)

Decimal	Look-Up
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

William Santosa
2/4/21
CSE135

Work Q#1

~~Deco~~ Convert to Hamming code

Let $G = \begin{Bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{Bmatrix}$

0.) 0000

$\{0000\} \cdot G \ \%_2 = \vec{c}$

$\{00000000\} \%_2 =$

$\{00000000\} =$

1.) 0001

$\{1000\} \cdot G \ \%_2 = \vec{c}$

$\{10000111\} \%_2 =$

$\{10000111\} =$

William Santosa

2/24/21

CSE 135

Work Q#1

2.) 0010

$\{0^{10}0\} \cdot 6 \;\%\; 2 = \vec{c}$

$\{0100\ 1011\} \;\%\; 2 = \vec{c}$

$\{0100\ 1011\} =$

3.) 0011

$\{1100\} \cdot 6 \;\%\; 2 = \vec{c}$

$\{1100\ 11^{22}00\} \;\%\; 2 =$

$\{1100\ 1100\} =$

4.) 0100

$\{0010\} \cdot 6 \;\%\; 2 = \vec{c}$

$\{0010\ 1101\} \;\%\; 2 =$

$\{0010\ 1101\} =$

5.) 0101

$\{1010\} \cdot 6 \;\%\; 2 = \vec{c}$

$\{1010\ 1212\} \;\%\; 2 =$

$\{1010\ 1010\} =$

6.) 0110

$\{0110\} \cdot 6 \;\%\; 2 = \vec{c}$

$\{0110\ 2112\} \;\%\; 2 =$

$\{0110\ 0110\} =$

7.) 0111

$\{1110\} \cdot 6 \;\%\; 2 = \vec{c}$

$\{1110\ ^{222}000\ 1\} \;\%\; 2 =$

$\{1110\ 0001\} =$

8.) 1000

$\{0001\} \cdot 6 \;\%\; 2 = \vec{c}$

$\{0001\ 1110\} \;\%\; 2 =$

$\{0001\ 1110\} =$

9.) 1001

$\{1001\} \cdot 6 \;\%\; 2 = \vec{c}$

$\{1001\ 1221\} \;\%\; 2 =$

$\{1001\ 1001\} =$

William Santosa
2/4/21
CSE13S

Work Q#1

10.) 1010

$\{0101\} \cdot G \;\%.\; 2 = \vec{c}$

$\{0101\ 2121\} \;\%.\; 2 =$

$\{0101\ 0101\} =$

11.) 1011

$\{1101\} \cdot G \;\%.\; 2 = \vec{c}$

$\{1101\ 2322\} \;\%.\; 2 =$

$\{1101\ 0010\} =$

12.) 1100

$\{0011\} \cdot G \;\%.\; 2 = \vec{c}$

$\{0011\ 2211\} \;\%.\; 2 =$

$\{0011\ 0011\} =$

13.) 1101

$\{1011\} \cdot G \;\%.\; 2 = \vec{c}$

$\{1011\ 2322\} \;\%.\; 2 =$

$\{1011\ 0100\} =$

14.) 1110

$\{0111\} \cdot G \;\%.\; 2 = \vec{c}$

$\{0111\ 3222\} \;\%.\; 2 =$

$\{0111\ 1000\} =$

15.) 1111

$\{1111\} \cdot G \;\%.\; 2 = \vec{c}$

$\{1111\ 3333\} \;\%.\; 2 =$

$\{1111\ 1111\} =$

William Santos
2/4/21
CSE 135

Work QHw

Let $H =$

$$\begin{Bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{Bmatrix}$$

2a.) $\{1110 \quad 0011\}$

$\{1110 \quad 0011\} \cdot H \;\%\; 2 = \vec{e}$

$\{2234\} \;\%\; 2 = \vec{e}$

$\{0010\} = \vec{e}$

There is an error in the 7th row as $\vec{e}$ matches it. Flip 7th element.

$\{1110 \quad 0001\} \cdot H \;\%\; 2 = \vec{e}$

$\{2224\} \;\%\; 2 = \vec{e}$

$\{0000\} = \vec{e}$

Double check

The original number is 7, or 0111_2, since $\vec{m} = \{1110\}$

William Santosa
2/4/21
CSE 135

Work Q#2

2b.) {1101 1000}

{1101 1000} • H %. 2 = $\vec{e}$

{3 2 3 2} %. 2 = $\vec{e}$

{1 0 1 0} = $\vec{e}$

The ~~ta~~ matrix $\vec{e}$ doesn't match any rows, ~~there~~ there is more 1 error in ~~the~~ $\vec{c}$.

William Santosa
2/4/21
CSE13S

Work Q#3

3. Fill out look up table

0	0
1	4
...	...
15	HAM_ERR

$$H = \begin{Bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{Bmatrix}$$

For 0 - 15, just check if the binary matches one of the rows in H. If it matches, the look up value is the row # - 1, else, it's a HAM_ERR.

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.) HAMERR
11.) 2
12.) HAM-ERR
13.) 1
14.) 0
15.) HAM-ERR

William Santosa
2/4/21
CSE135

Work #2 (corrected

2a.) {1110 0011}

{1100 0111} · H %. 2 · $\vec{e}$

{1233} %. 2 = $\vec{e}$

{1011} = $\vec{e}$

(1st row)

There is an error in the 2nd element as $\vec{e}$ = 1011.

{ [redacted] 1000 0111} · H %. 2 = $\vec{e}$

{0222} %. 2 = $\vec{e}$

{0000} = $\vec{e}$

Double check

[redacted] $\vec{m}$ = {1000} Original Number = 0001_2

Decimal = 1

2b.) {1101 1000}

{0001 1011} · H %. 2 = $\vec{e}$

{2121} %. 2 = $\vec{e}$

{0101} = $\vec{e}$

There is at least 1 error in the Hamming code as $\vec{e}$ = {0101}

DESCRIPTION

There are many different ways to correct errors (known as noise) in data and one such way is with a Hamming Code. A Hamming Code is a simple encoding and decoding technique that allows receivers and senders to transmit extra bits of data in order to detect and correct these errors. In this lab, we implement our own Hamming Code within C using the following files.

Files:

- generator.c
  - Contains my implementation of the Hamming Code generator
- decoder.c
  - Contains my implementation of the Hamming Code decoder
- bm.h
  - Contains the BitMatrix ADT interface
- bm.c
  - Contains my implementation of the BitMatrix ADT interface
- hamming.h
  - File provided in resources repository, contains the Hamming Code module
- hamming.c
  - Contains my implementation of Hamming Code module
- error.c
  - File provided in resources repository
- Makefile
  - Runs program (All, gen, dec, err, clean, format)
- README.md
  - Information about building, running, and options of the program
- DESIGN.pdf
  - Describes purpose, covers the layout, clear description of program parts, pseudo code, and contains the pre lab questions.

TOP LEVEL DESIGN / PSEUDOCODE

typedef enum ham_rc { HAM_ERR = -1, HAM_OK = 0, HAM_ERR_OK = 1 } ham_rc;

ham_rc ham_init(void){
    Create G and H BitMatrix
    G has 4 rows 8 columns
    H has 8 rows 4 columns
    For loop to set the G and H matrix
    Return HAM_ERR if fails
    Return 0 if no errors
    Return HAM_ERR_OK if 1 error and is fixed

```
}

void ham_destroy(void){
        Free G and H columns
        Free G and H rows
        Free G and H
        Return nothing
}

ham_rc ham_encode(uint8_t data, uint8_t *code){
        BitMatrix C initialize 1 row 8 columns
        Check if data and code is valid
        Multiply data by code using for loop
        Check for HAM_OK/ERR/ERR_OK
        Return HAM_OK if successful generate and valid
        Return HAM_ERR if not valid and not fixable
        Return HAM_ERR_OK if not valid but fixable
}

ham_rc ham_decode(uint8_t code, uint8_t *data){
        BitMatrix C initialize 1 row 4 columns
        Check if data and code is valid
        Multiply data by code using for loop
        Check for HAM_OK/ERR/ERR_OK
        Return HAM_OK if successful generate and valid
        Return HAM_ERR if not valid and not fixable
        Return HAM_ERR_OK if not valid but fixable
}

struct BitMatrix BitMatrix{
        uint32_t rows;
        uint32_t cols;
        uint8_t ** mat;
}

BitMatrix *bm_create(uint32_t rows, uint32_t cols){ // SIMILAR TO UNIVERSE.C
        Create m pointer to (BitMatrix *) calloc(1, sizeof(BitMatrix))
        Set rows and cols
        Make a mat point toward (uint8_t **) calloc(rows, sizeof(uint8_t**)
        For loop to make all rows contain cols amount of column
        For loop to set all elements to 0
        Return m
}
```

```
void bm_delete(BitMatrix **m){ // SIMILAR TO UNIVERSE.C
        Free the columns first
        Free rows
        Free BitMatrix m
        Return nothing
}

uint32_t bm_rows(BitMatrix *m){
        Return m pointer to rows
}

uint32_t bm_cols(BitMatrix *m){
        Return m pointer to cols
}

void bm_set_bit(BitMatrix *m, uint32_t r, uint32_t c){
        M pointer to mat[r][c/8] set 1
        Return nothing
}

void bm_clr_bit(BitMatrix *m, uint32_t r, uint32_t c){
        M pointer to mat[r][c/8] set 0
        Return nothing
}

uint8_t bm_get_bit(BitMatrix *m, uint32_t r, uint32_t c){
        Return m pointer mat[r][c/8]
}

void bm_print(BitMatrix *m){ // DO THIS FIRST
        For loop to access every element and print out the bitmatrix
        Return nothing
}
```

NOTE : Will definitely be altered (Listed below)

DESIGN PROCESS / MODIFICATIONS

- Using pointers for shifting bits in the BitMatrix stuff
- Using pointers for shifting bits in the hamming code
    - Checking with the Q#1 and Q#3 prelab question using shift bits
- Generator.c uses one input, decoder.c needs two inputs
    - fgetc() twice, double check within while loop and make a label + goto to get out
- Encode and decode needs to be double checked
    - Don't need to flip the code bit
- Decode needs if this then change this bit, else return error or return ok

- Makefile needs format
- Counters to keep track of iterations