Assignment 4 Hamming Codes Document

Pre-lab

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

When I using "Factor Tree" methods, I find all sixteen possibilities that are (0000),(0001),(0010),(0011),(0100),(0101),(0110),(0111),(1000),(1001),(1010),(1011),(1100),(1101),(1110),(1111). When I calculate using the generator matrix, (0 0 0 0 0 0 0 0),(0 0 0 1 1 1 1 0),(0 0 1 0 1 1 0 1),(0 0 1 1 0 0 1 1), (0 1 0 0 1 0 1 1),(0 1 0 1 0 1 0 1), (0 1 1 0 0 1 1 0), (0 1 1 1 1 0 0 0),(1 0 0 0 0 1 1 1),(1 0 0 1 1 0 0 1),(1 0 1 0 1 0 1 0),(1 0 1 1 0 1 0 0), (1 1 0 0 1 1 0 0),(1 1 0 1 0 0 1 0),(1 1 1 0 0 0 0 1), (1 1 1 1 1 1 1 1).

(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: When I decoding this code, I got 0010. That corresponds with H^t's sixth column and it contains an error. As a result, we know that the sixth element in c was erroneously flipped. So, flipping the value of the sixth element, we can correct it as 1110 0111.

(b) 1101 1000: When I decoding this code, I got 1010 and we got an error. However, this does not match any H^t's column. Therefore, this error cannot be corrected.

(3) Complete the rest of the look-up table shown below.

0	0
1	5
2	6
3	HAM_ERR
4	7
5	HAM_ERR
6	HAM_ERR
7	4
8	8

9	HAM_ERR
10	HAM_ERR
11	3
12	HAM_ERR
13	2
14	1
15	HAM_ERR

Purpose

The purpose of this document is to explain the details of assignment 4 design. In this assignment we have implemented the ‘Hamming Codes’ application. This has two parts hamming code generator and decoder

Design

Generator

```
Start
        Step 1: Parse the command-line options with getopt().
        Step 2: Initialize the Hamming Code module with ham_init().
        Step 3: Read two byte from the specified file stream or stdin with fgetc(). Step 4:
For each byte pair read, decode the Hamming(8,4) codes for both with ham_decode() to
recover the original upper and lower nibbles of the message. Then, reconstruct the original
byte.
        Step 5: Write the reconstructed byte with fputc().
        Step 6: Repeat steps 3–5 until all data has been read from the file or stdout.
Step 7: Call ham_destroy() to free memory allocated by the Hamming Code module.
        Step 8: Print the following statistics to stderr with fprintf():
                        ● Total bytes processed: The number of bytes read by the decoder.
                        ● Uncorrected errors: The number of Hamming codes that could not
                        be corrected.
                        ● Corrected errors: The number of Hamming codes that
                          experienced an error that was recov-erable.
                        ● Error rate: The rate of uncorrected errors for a given input. This
                          can be calculated by the following formula: e =ut where u is the
                          number of uncorrected errors, and t is the total number of bytes
                        (Hamming codes) processed.
        Step 9: Close both the input and output files with fclose().

End
```

Decoder

```
Start
    Step 1: Parse the command-line options with getopt().
    Step 2: Initialize the Hamming Code module with ham_init().
    Step 3: Read two byte from the specified file stream or stdin with fgetc().
    Step 4: For each byte pair read, decode the Hamming(8,4) codes for both with ham_decode() to recover the original upper and lower nibbles of the message. Then, reconstruct the original byte.number of bytes processed, Hamming codes that required correction, and Hamming codes that could not be corrected.
    Step 5. Write the reconstructed byte with fputc().
    Step 6. Repeat steps 3–5 until all data has been read from the file or stdout.
    Step 7. Call ham_destroy() to free memory allocated by the Hamming Code module.
    Step 8. Print the following statistics to stderr with fprintf():
        ● Total bytes processed: The number of bytes read by the decoder.
        ● Uncorrected errors: The number of Hamming codes that could not be corrected.
        ● Corrected errors: The number of Hamming codes that experienced an error that was recoverable.
        ● Error rate: The rate of uncorrected errors for a given input. This can be calculated by the following formula: $e =ut$ where u is the number of uncorrected errors, and t is the total number of bytes (Hamming codes) processed.
    Step 9: Close both the input and output files with fclose().
End
```