CS3210 – Computer Networks Lab
Jan-May 2017, Prof. Krishna Sivalingam
Lab 5 – Hamming Error Correction Code
Language: C/C++/Java

No Network Access during the Lab
Due date: Feb. 14, 2017, 5 PM (Not Extensible), On Moodle
This is not available for those who are not able to attend Lab in person

February 14, 2017

1 Description

The purpose of this lab is to understand and implement the Hamming Code, as explained in class. A PDF of the Wikipedia page is also available on the CSE Moodle.

Input: The input to your program will be as follows:

```
./hc infile outfile
```

The input file (infile) contains a set of binary strings (each of length 26 bits). The number of parity bits is 5, for a total message length of 31 bits. This is referred to as the $\mathbf{Hamming}(31, 26)$ code. In general, for k parity bits, the code is $\mathbf{Hamming}(2^k - 1, 2^k - k - 1)$. For each string, you have to compute the checksum, as explained in class.

Output: The output for each input string will be as shown in the example:

```
Input: 0101......01
Ouput: 0101......01110..11
<Other Output> - See below
```

Note that each input string is 26 bits and the checksums are 5 bits each. All output will be stored in the 'outfile'.

1.1 Error Detection Properties

Also, for each entry in the input file, report the error detection properties of the mechanism as follows:

- Generate an error string (from a given 31-bit message string) with one random bit error (for each string) and check if the error is corrected. The output format is as follows:

```
Original String: 0101......01110
Original String with Parity: 0101......01110..11
Corrupted String: 1011......01110..11
```

```
Error Location(s) is/are: ..
Number of Errors Introduced: 1
Error Location computed by receiver algorithm is: 5
```

- Generate 10 random two-bit errors (for each string) and check if the error is detected. The output format is shown below.

```
Original String: 0101......01110
Original String with Parity: 0101......01110..11
Corrupted String: 1011......01110..11
Error Locations is/are: ...
Number of Errors Introduced: 2
Error Detected (Yes/No):
Parity bits that failed are at locations: 3 5
```

This output should also be appended to the output file mentioned earlier.

2 What to Submit

- Source code files
- One sample input file used for your testing, with at least 50 entries
- Corresponding Output file, showing both parity bit computation and error detection tests

Random generators such as *random* and *drand48* from `stdlib.h` may be useful.

3 Grading

- Hamming Code: 35 points
- Error Verification: 15 points

Note: This is an INDIVIDUAL assignment. If you are having difficulty with the assignment, please talk to the TAs/the instructor. Downloaded Code from the Web will NOT be considered for grading and such action will lead to academic penalties.