CS2033 Data Communication and Networks – In20

Department of Computer Science and Engineering
University of Moratuwa

Lab Assignment 4 – Transmission Errors

We learned about various transmissions error detection and correction techniques in the class. Information coding schemes based on parity play a major role in transmission error detection and correction. In this lab, you will implement two information coding techniques for error detection and correction.

Lab 4.1 Cyclic Redundancy Checksum

```
void generateCRC(char *sendm, char *poly, char *crc)
```

Write generateCRC function which takes as input sending message **`sendm`** and a generator polynomial **`poly`** as binary sequences, and outputs the resulting CRC binary sequence to the char array **`crc`**.

```
int checkCRC(char *recvm, char *poly)
```

Write checkCRC function which takes as input received message **`recvm`** and the relevant generator polynomial **`poly`** as binary sequences and returns 1 if there has been a transmission error 0 if no transmission error has occurred.

Lab 4.2 (7,4) Single Error Correcting Hamming Code

```
void generateHammingCode(char *sendm, int parity, char
*output)
```

Write generateHammingCode function which takes as input a message as a binary sequence **`sendm`** and the **`parity`** (odd = 1, even = 0), and generates the output message which includes message bits and parity bits. Please follow the example Hamming code scheme discussed in the lecture to place the parity bits in the output. Write the resulting output message to char array **`output`** as a binary sequence.

```
int checkMessage(char *recvm, int parity)
```

Write checkMessage function which takes as input a Hamming Coded message **`recvm`** and the relevant parity (odd = 1, even = 0) and returns the bit position (least significant bit-position is 1) of the erroneous bit. **Assume only a single bit errors will be checked**. For non-erroneous **`recvm`**, the function should return 0.

The void runCRC() and runHammingCode() functions

In addition to the main components, you need to have run functions that do the following. Depending on the input, either runCRC() or runHammingCode() should be invoked.

1. Reads the following input parameters from the **stdin** and print output to **stdout**. You program should be able to handle two types of inputs based on the invocation mode.

    a. Inputs that invoke CRC related functions. (The first input parameter will be set to 0). The first input line will consist of the invoking mode, sendm and poly. The following lines will contain arbitrary number of input lines that contain different recvm bit sequences.

    Eg:

    ```
    0 1101 10
    1101
    1111
    0000
    1100
    ```

    Here the first line contains the invocation mode (0), sendm (1101), poly (10). Lines 2-4 contain three recvm sequences.

    Your program should output a space separated stream of 1/0s indicating the output of checkCRC for each input line (2-4 above).

    Eg:

    ```
    0 1 1 1
    ```

    b. Inputs that invoke Hamming Code related functions. (The first input parameter will be set to 1). The first input line will consist of the invoking mode, sendm and parity. The following lines will contain arbitrary number of input lines that contain different recvm bit sequences.

    Eg:

    ```
    1 1101 1
    1101
    1111
    1100
    0101
    ```

    Here the first line contains the invocation mode (1), sendm (1101), parity (1). Lines 2-4 contain three recvm sequences.

    Your program should output a space separated list of integers indicating the output of the checkMessage function for each input line (2-4 above).

    Eg:

    ```
    0 2 1 4
    ```

2. Perform any other auxiliary tasks, if necessary.

Submission Instructions

You have been provided with a code template that consists of the following files:

lab4.c
transmissionerr.c
transmissionerr.h
Makefile

Please modify transmissionerr.c and lab4.c to complete the program.

You should submit the completed source code into the coderunner autograded quiz on moodle like you did for previous labs.

Note: since this is a brand-new lab designed for your batch, there may be glitches. If you come across any such glitches, kindly inform me (sunimal@cse.mrt.ac.lk).

Important: As you may have noticed, the desired outputs of the test cases could be determined easily by comparing sendm and recvm without following the CRC approach or Hamming Code approach. This is because you are given both sendm and recvm at together. In real world scenarios this is not possible. **Thus, you need to determine the output by properly coding the CRC and Hamming Code approaches without using such shortcuts.** We will perform advanced tests on your submissions and determine non-compliant codes and not award marks for them.

Please note that this is an individual assignment, and those who are caught plagiarizing will be penalized and will be reported for disciplinary action.