5 Error Coding

Modulo-2 multiplication and division

1. Implement the following Python code and complete Table 1

http://asecuritysite.com/calculators/mod2

Table 1: Multiply table

Value 1	Value 2	Modulo-2 multiplication
101010111	101101	
101010111	110011	
101010111	100110	
101010111	111010	
101010111	100001	
101010111	100000	
101010111	100010	

2. Prove these results using the following link:

http://www.ee.unb.ca/cgi-bin/tervo/calc.pl

3. Implement the following Python code and complete Table 2:

http://asecuritysite.com/calculators/mod2_div

Table 2: Division table

Value 1	Value 2	Modulo-2 result	Modulo-2 remainder
101010111	101101		
101010111	110011		
101010111	100110		
101010111	111010		
101010111	100001		
101010111	100000		
101010111	100010		

4. You have been asked to produce a sending Python program in which you have 8-bit input values. In order to detect errors you have been asked to multiply the sending byte by “10101”.Now generate each possible value, and outline the first eight values:

```
0000 0000:
0000 0001:
0000 0010:
0000 0011:
0000 0100:
0000 0101:
0000 0110:
0000 0111:
```

A sample run with a range of 0 to 255 and with a multiplier of ‘11011’ is:

```
$ python ex05_01.py 255 11011
00000000        00000
00000001        11011
00000010        110110
00000011        101101
00000100        1101100
00000101        1110111
00000110        1011010
00000111        1000001
00001000        11011000
..
```

5. Now produce the receiving program, and input the sending values given above, and provide that the remainder is zero:

Value sent	Value received	Value received correctly
0000 0000		Y / N
0000 0001		Y / N
0000 0010		Y / N
0000 0011		Y / N
0000 0100		Y / N
0000 0101		Y / N
0000 0110		Y / N
0000 0111		Y / N

6. Now produce the receiving program, and input the sending values given above, and provide that the remainder is zero:

Value sent	Value received (with one error)	Error detected
0000 0000		Y / N
0000 0001		Y / N
0000 0010		Y / N
0000 0011		Y / N
0000 0100		Y / N
0000 0101		Y / N
0000 0110		Y / N
0000 0111		Y / N

Block parity

1. Block parity is a block code which adds a parity symbol to the end of a block of code. For example a typical method is to transmit the one's complement (or sometimes the two's complement) of the modulo-2 sum of the transmitted values. Implement a Python program which determines the block parity value for:

(i) {00000001, 00000100, 00001100, 11111111, 10011010, 00010001, 01000000, 01110110}
(ii) {00000111, 11100100, 00011100, 01111111, 10011010, 00010001, 01011000, 01111111}
(iii) { 00011111, 11111100, 10011100, 01111111, 10011010, 11010001, 01011000, 01111111}

A coding example is here:

http://asecuritysite.com/comms/block

2. For the following received values (with the last one being the received block parity check, determine if there is an error in the transmitted data, and, if so, where the error is:

(i) { 00011011, 11111100, 10011100, 01111111, 11111010, 00010001, 01011000, 01001111} Block parity value {11111000}

(ii) { 11011011, 11111100, 10011100, 01110011, 10110010, 00010001, 01011000, 01001111} Block parity value {01111000}

(iii) { 11011011, 11111100, 10011100, 01110011, 10110011, 10010001, 01011000, 01001100} Block parity value { 11111111}

3. Update the program so that it generates a 16-bit value checksum from eight 16-bit values.

4. Update the program so that the it takes eight values and the parity block value, and identifies if there are errors and the position of the error (assuming a single bit in error).

5. The program uses **even parity**, where the number of 1s is even. Update the program so that it also calculates the block code with odd parity.

CRC

1. Using the following and the Python program outlined at:

http://asecuritysite.com/comms/crc_div

Determine the transmitted bits for the following:

(i) $P(x)=x^3+x^2+x^0$ (1101)
$G(x)=x_6+x_3+x_2$ (1001100)

(ii) $P(x)=x^3+x^2+x^0$ (1101)
$G(x)=x_5+x_3+x_2$ (0101100)

Check your answer by using:

http://asecuritysite.com/comms/mod_div

and show there is a zero remainder in each case.

LRC

1. Implement an LRC/VRC calculator from:

http://asecuritysite.com/calculators/lrc

Use this program to complete the following table:

String	LRC	VRC
captain	0x01	0,0,0,1,1,1,0
jelly999		
Fishoil		
1234567		

Hamming distance

1. Implement a Hamming distance checker using the Python program defined on this page:

http://asecuritysite.com/calculators/ham

Use this program to complete the following table:

Value 1	Value 2	Hamming distance
0000 0000	**0101 1010**	
0000 0001	**0101 1010**	
0000 0010	**0101 1010**	
0000 0011	**0101 1010**	
0000 0100	**0101 1010**	
0000 0101	**0101 1010**	
0000 0110	**0101 1010**	
0000 0111	**0101 1010**	

2. With codes we might need to find the minimum Hamming distance for a range of codes. Implement the code here:

http://asecuritysite.com/comms/ham2

3. Now determine the minimum Hamming distance for the following codes:

- {0110,1111,0011,1010}
- {00001100,0110 0101, 1011 0011,11111111}

Bit errors

If we have n bits, we can determine the probability that there will be an error with:

Probability of no errors = $(1-p)_n$

The probability of an error will thus be:

Probability of an error = $1-(1-p)_n$

If we want to determine the number of m-bit errors combinations, we can use:

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

to determine the number of error. The probability of *m* bits being in error is thus:

$$P_e(m) = \binom{n}{m} . p^m . (1-p)^{n-m}$$

Using the Python program defined in:

http://asecuritysite.com/calculators/bit_error

and use it to complete the follow (the first one has been completed):

Bits in error	P(error)	No of errors
1	0.011868658	12
2		
3		
4		
5		
6		
7		
8		
9		

10		
11		
12		
Totals:		

Answers

1.5. A sample program is here:

https://dl.dropboxusercontent.com/u/40355863/ex03_01.zip