

```

/*
    Computer Networks Laboratory (Lab) 15CSL77
    6. Write a program for error detecting code using CRC-CCITT (16-bits).
*/

/* Layers in networks

    CRC: Cyclic Redundancy Check

    Data Link Layer: CRC for error checking

*/

/* In simple , for information to be sent, say message,
    and if CRC is generated, say crc,
    then data sent = message + crc
    i.e. message followed by crc

    Consider message as number. Then CRC, can be thought of as remainder.
    Divisor used is usually a Polynomial, usually called as Generator Polynomial
    (Operation used is XOR)

    Polynomial

    Length of message(also known as payload) in bits can be - ?

    Depends on Protocol being used.

    crc = remainder of repeated operation of generator polynomial on message

    Specifically, message padded to right with zeros

    Data sent = message . crc
                message concatenated with crc
*/

/* A polynomial of degree n, has atmost n + 1 terms.
    n = highest degree in polynomial

    Atmost vs Atleast

     $x^2 + x + 1$  , is a polynomial of degree 2,  $x^2$  is x squared, x to power 2

    Can be represented as  $1 \quad 1 \quad 1$  , as array of length 3, [ 1 , 1 , 1 ]
                         $x^2 + x + 1$ 

    Array holds 1 , if that term of degree = to index of array is present,
                    0 otherwise
    Consider index from left to right, left being highest i.e. n
    and right being least i.e. 0

     $x^3 + x + 1$  can be represented as  $1 \quad 0 \quad 1 \quad 1$ 
                                       $x^3 + 0.x^2 + x + 1$ 

    To represent polynomial of degree n , array of n + 1 is required
*/

/* Using CRC to detect errors.

    Message preprocessing: Padding message with zeros

    Repeated operations: XOR

    crc = ( message padded with n zeros to right ) repeatedly operated upon
          from left to right by generator polynomial

    n = degree of generator polynomial

```

```

data to be sent = message . crc

        message concatenated with crc

Repeated operation = moving generator polynomial on preprocessed message
                    while
                        Applying XOR operation

End: No more divisor

Result: CRC saved in last n bits (in the location of zero padding)
Send message . crc
*/

/* If received data does not have errors, i.e. no bits flipped, then

        ( message . crc ) operated upon by same generator polynomial = 0
                                will result in zero remainder          ( Why? )
    else
        received data has error[s]
*/

/* Assume message = 11010011      ( Hexadecimal ⇄ Decimal ⇄ Octal ⇄ Binary )

    Let , generator polynomial =  $x^3 + x + 1$ 
                                i.e. 1 0 1 1      , ( 1011 )

    Length of generator polynomial = 4

                                message = 11010011
    message padded with zeros = 110100110000
                                four zeros
                                generator polynomial = 1101

                                data sent = message . crc
*/

/* Calculation CRC , using XOR operation, ^ , at the sender
    XOR is high/enabled only when either of values are high
                                (not both, only one)

    1^1 = 0^0 = 0
    1^0 = 0^1 = 1

    message padded with zero

    110100110000
    ^
    1101
    0000

    000000110000
    ^
    1101      (move, slide generator polynomial, to right)
    1110

    11100000
    ^
    1101
    0011

    00110000
    ^
    1101
    0001

    0100 = crc

    Data sent = 110100110100

```

```

*/
/* At the receiver
   If data received with no error ,

   110100110100
   ^
   1101          (apply XOR, using same generator polynomial)
   0000

       00110100
       ^
       1101
       1110

       11100100
       ^
       1101
       0011

           110100
           ^
           1101
           0000
               00

   No more dividend = no error          ( Why this works? )
*/

/* Now, extend to CRC 16, a polynomial of degree 16 , ( What if its CRC 1 )
   Generator Polynomial(x) =  $x^{16} + x^{12} + x^5 + 1$ 
*/

/* Any guess on run time of Generating CRC?

   Can performance of code be improved?

   Software implementation vs Hardware Implementation of CRC

   Which would be preferred?

   Now, At which network layer hardware component is CRC implemented?

   CRC is used in X.25, V.41, HDLC FCS, XMODEM, Bluetooth, PACTOR, SD, DigRF

   CRCs are specifically designed to protect against burst errors,
   contiguous sequences of erroneous data symbols in messages

   They are not suitable for protecting against intentional alteration of data

   Are there instances where error will go undetected?
*/

/* Textbook: Behrouz Forouzon - Data Communications and Networking,
   McGraw Hill Edition */

```