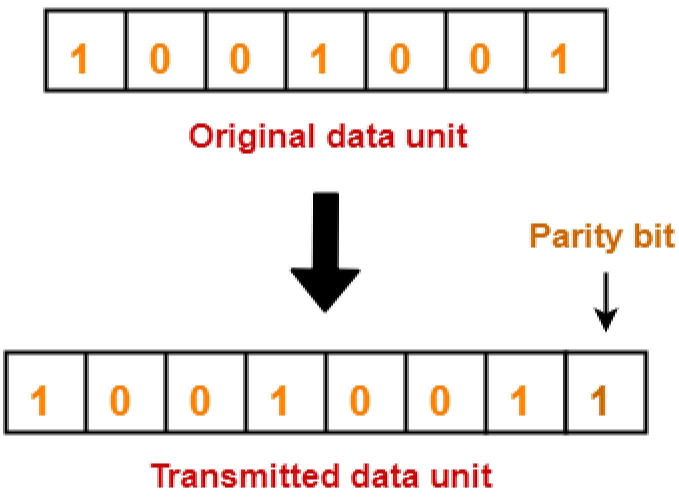
Parity Check Example-

Consider the data unit to be transmitted is 1001001 and even parity is used.

Then,

At Sender Side-

- Total number of 1's in the data unit is counted.
- Total number of 1's in the data unit = 3.
- Clearly, even parity is used and total number of 1's is odd.
- So, parity bit = 1 is added to the data unit to make total number of 1's even.
- Then, the code word 10010011 is transmitted to the receiver.



At Receiver Side-

- After receiving the code word, total number of 1's in the code word is counted.
- Consider receiver receives the correct code word = 10010011.
- Even parity is used and total number of 1's is even.
- So, receiver assumes that no error occurred in the data during the transmission.

Check sum Problem-1

Checksum value of 1001001110010011 and 1001100001001101 of 16 bit segment is-

Solution-

We apply the above discussed algorithm to calculate the checksum.

- 1001001110010011
+ 1001100001001101
  10010101111100000
- Since, the result consists of 17 bits, so 1 bit is wrapped around and added to the result.
- 0010101111100000 + 1 = 0010101111100001
- Now, result consists of 16 bits.
- Now, 1's complement is taken which is
  0010101111100001
  1101010000011110
- Thus, checksum value = 1101010000011110

Problem 2

Consider the data unit to be transmitted is-

10011001111000100010010010000100

Consider 8 bit checksum is used.

Step-01:

At sender side,

The given data unit is divided into segments of 8 bits as-

Now, all the segments are added and the result is obtained as-

10011001 + 11100010 + 00100100 + 10000100 = 1000100011
Since the result consists of 10 bits, so extra 2 bits are wrapped around.
00100011 + 10 = 00100101 (8 bits)
Now, 1's complement is taken which is 11011010.
Thus, checksum value = 11011010

Step-02:
The data along with the checksum value is transmitted to the receiver.

Step-03:
At receiver side,
The received data unit is divided into segments of 8 bits.
All the segments along with the checksum value are added.
Sum of all segments + Checksum value = 00100101 + 11011010 = 11111111
Complemented value = 00000000
Since the result is 0, receiver assumes no error occurred in the data and therefore accepts it.

PRACTICE PROBLEMS BASED ON CYCLIC REDUNDANCY CHECK (CRC)-

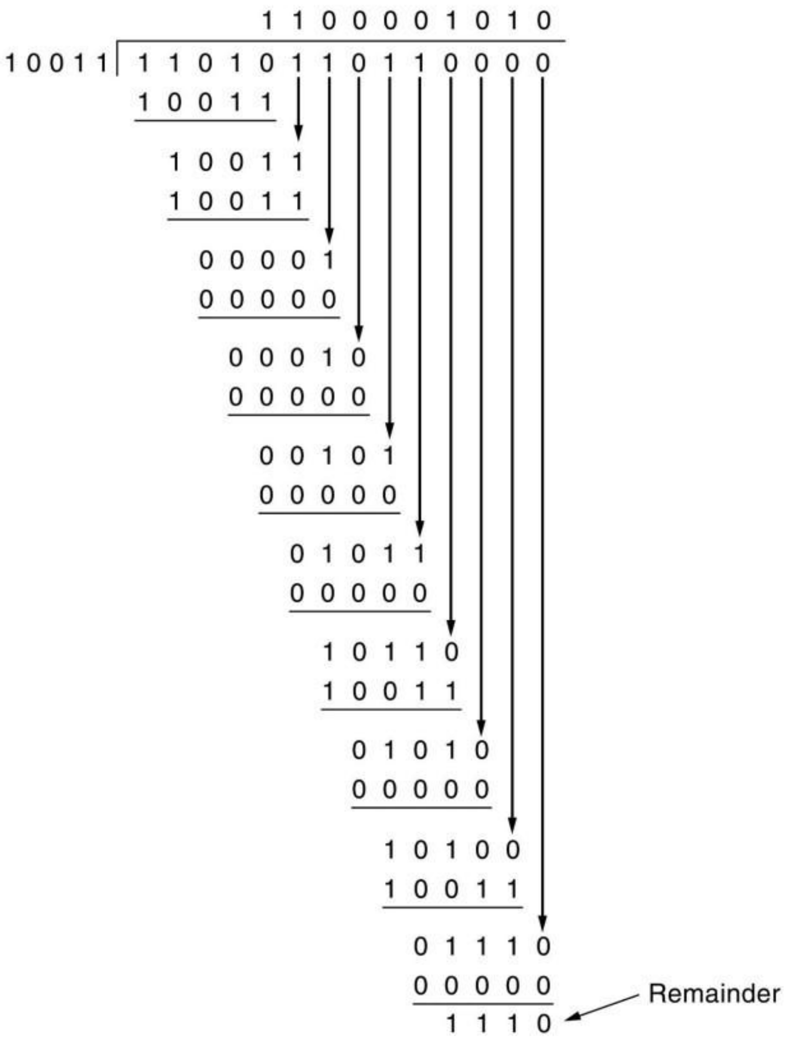
Problem-01:

A bit stream 1101011011 is transmitted using the standard CRC method. The generator polynomial is x^4+x+1. What is the actual bit string transmitted?

Solution-

- The generator polynomial $G(x) = x^4 + x + 1$ as $1.x^4 + 0.x^3 + 0.x^2 + 1.\ x^1 + 1.\ x^0$ is encoded as 10011.
- Clearly, the generator polynomial consists of 5 bits.
- So, a string of 4 zeroes is appended to the bit stream to be transmitted.
- The resulting bit stream is 11010110110000.

Now, the binary division is performed as-



From here, CRC = 1110.

Now,

- The code word to be transmitted is obtained by replacing the last 4 zeroes of 1101011011**0000** with the CRC.
- Thus, the code word transmitted to the receiver = 1101011011**1110**.

Problem-02:

A bit stream 10011101 is transmitted using the standard CRC method. The generator polynomial is x^3+1.

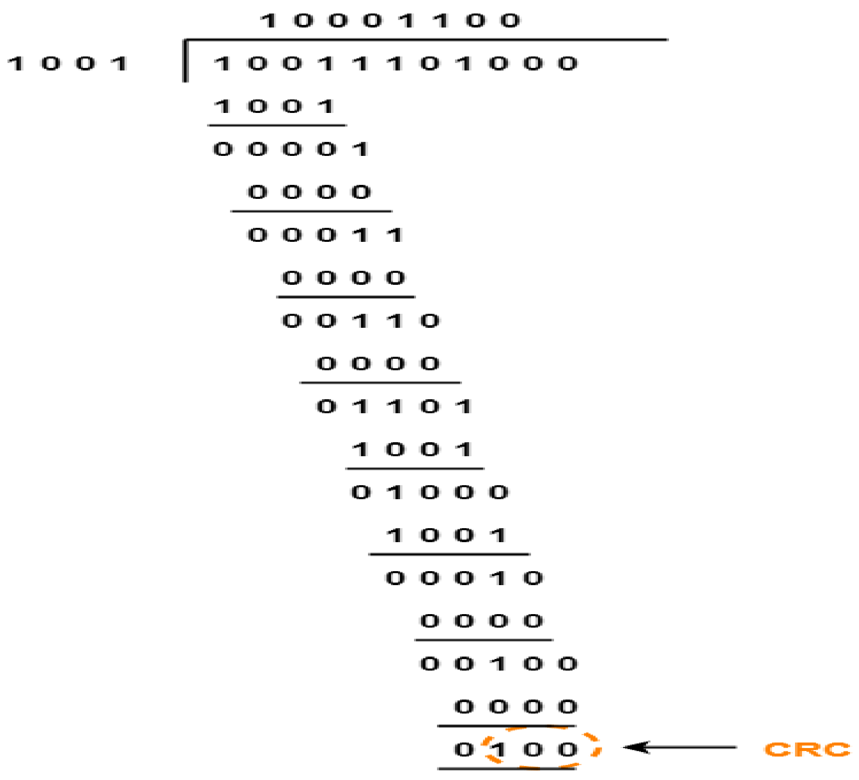
1. What is the actual bit string transmitted?
2. Suppose the third bit from the left is inverted during transmission. How will receiver detect this error?

Solution-

Part-01:

- The generator polynomial $G(x) = x^3 + 1$ is encoded as 1001.
- Clearly, the generator polynomial consists of 4 bits.
- So, a string of 3 zeroes is appended to the bit stream to be transmitted.
- The resulting bit stream is 10011101**000**.

Now, the binary division is performed as-



From here, CRC = 100.

Now,

- The code word to be transmitted is obtained by replacing the last 3 zeroes of 10011101**000** with the CRC.
- Thus, the code word transmitted to the receiver = 10011101**100**.

<u>Part-02:</u>

According to the question,

- Third bit from the left gets inverted during transmission.
- So, the bit stream received by the receiver = 10111101100.

Now,

- Receiver receives the bit stream = 10111101100.
- Receiver performs the binary division with the same generator polynomial as-

```
         10101000
1001 | 10111101100
       1001
       00101
        0000
        01011
         1001
         00100
          0000
          01001
           1001
           00001
            0000
            00010
             0000
             00100
              0000
              0100   <-- Remainder
```