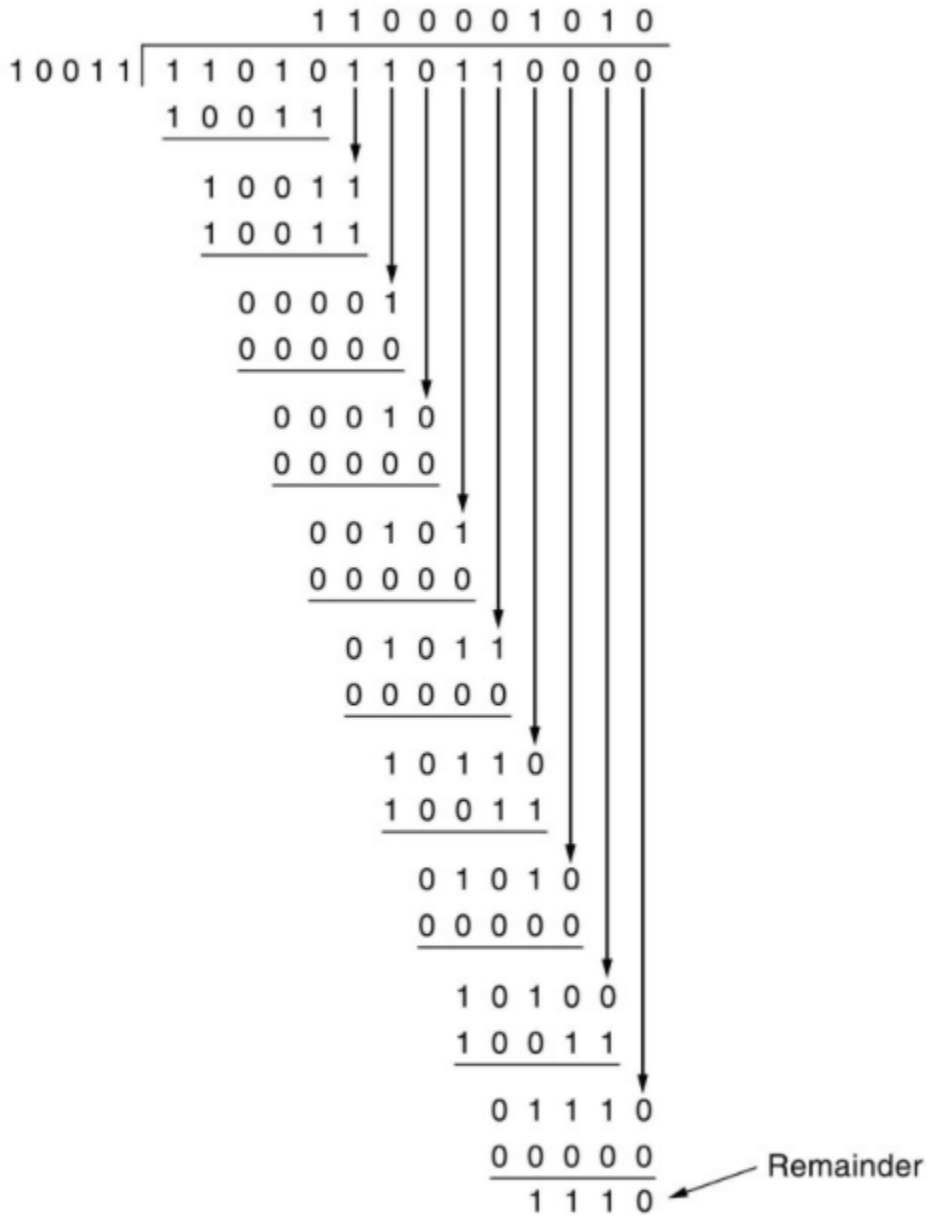
1. Consider the Cyclic Redundancy Check (CRC) algorithm discussed during the lecture. Suppose that the 4-bit generator (G) is 1001, that the data payload (D) is 10011110 and that r = 3. 111
2. Suppose that the 4-bit generator (G) is 1001, that the data payload (D) is 10011011 and that r = 3. 010
3. Suppose that the 4-bit generator (G) is 1001, that the data payload (D) is 10011010 and that r = 3. 011
4. 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?
   X^4+x+1 means 10011.(5 bits) means the actual data was: 1101011011 0000

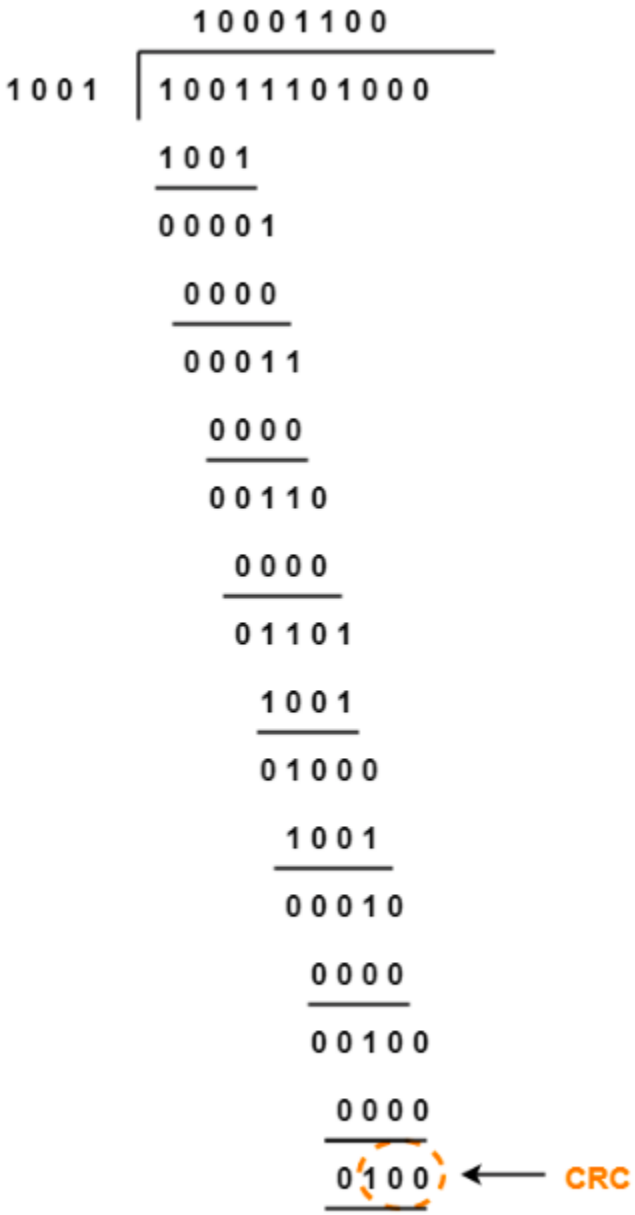
5. Calculate the CRC of above example



CRC = 1110.

6. A bit stream 10011101 is transmitted using the standard CRC method. The generator polynomial is x3+1.
    - What is the actual bit string transmitted?
    - Suppose the third bit from the left is inverted during transmission. How will receiver detect this error?

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 1001101**000**.



From here, CRC = 100.

Now,

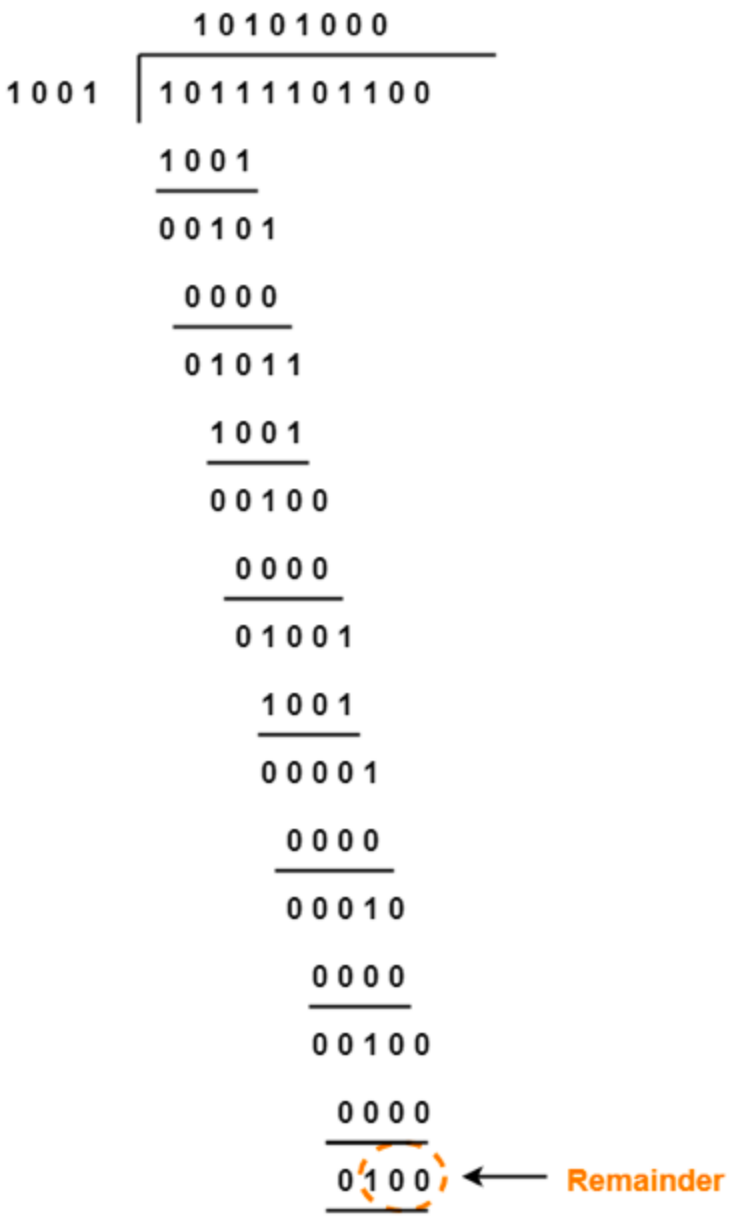
The code word to be transmitted is obtained by replacing the last 3 zeroes of 10011101000 with the CRC. Thus, the code word transmitted to the receiver = 10011101100.

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



From here,

- The remainder obtained on division is a non-zero value.

The remainder obtained on division is a non-zero value.

- This indicates to the receiver that an error occurred in the data during the transmission.
- Therefore, the receiver rejects the data and asks the sender for retransmission.

Socket programming:

Create three programs, two of which are clients to a single server using datagram socket. Client1 will send a character to the server process. The server will decrement the letter to the next letter in the alphabet and send the result to client2. Client2 prints the letter it receives and then all the processes terminate. Compile and run this exercise.

Client1:

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<unistd.h>

int main()
{
   int client_socket;
   int error_check;
   struct sockaddr_in client_address;
   socklen_t server_len;
   char client_message[2];

   server_len = sizeof(client_address);

   client_socket = socket(AF_INET, SOCK_DGRAM, 0);
   if(client_socket == -1)
   {
      printf("Error creating socket\n");
      exit(1);
   }

   client_address.sin_family = AF_INET;
   client_address.sin_port = htons(9002);
   client_address.sin_addr.s_addr = INADDR_ANY;

   printf("Enter character to send\n");
   fgets(client_message, sizeof(client_message), stdin);

   error_check = sendto(client_socket, client_message, sizeof(client_message),0, (struct sockaddr *)
&client_address, server_len);
   if(error_check == -1)
   {
      printf("Error sending message\n");
      exit(1);
   }

   printf("message sent by client\n");

   close(client_socket);
```

```
    return 0;
}
```

Client2:

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<unistd.h>

int main()
{
    int client_socket;
    int error_check;
    struct sockaddr_in client_address;
    socklen_t server_len;
    char client_message[1024];

    server_len = sizeof(client_address);

    client_socket = socket(AF_INET, SOCK_DGRAM, 0);
    if(client_socket == -1)
    {
        printf("Error creating socket\n");
        exit(1);
    }

    client_address.sin_family = AF_INET;
    client_address.sin_port = htons(9002);
    client_address.sin_addr.s_addr = INADDR_ANY;

    error_check = sendto(client_socket, client_message, sizeof(client_message),0, (struct sockaddr *)
&client_address, server_len);
    if(error_check == -1)
    {
        printf("Error sending message\n");
        exit(1);
    }

    error_check = recvfrom(client_socket, client_message, sizeof(client_message), 0, (struct
sockaddr *) &client_address, &server_len);
    if(error_check == -1)
    {
        printf("Error receiving message\n");
        exit(1);
    }
```

```
    printf("Server sent following character:\n");
    printf("%s\n", client_message);

    close(client_socket);

    return 0;
}
```

Server:

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>

int main()
{
    int server_socket;
    int error_check;
    char server_message[256];
    char ping[10];
    struct sockaddr_in client_address;
    socklen_t client_len;

    client_len = sizeof(client_address);

    server_socket = socket(AF_INET, SOCK_DGRAM, 0);

    struct sockaddr_in server_address;
    server_address.sin_family = AF_INET;
    server_address.sin_port = htons(9002);
    server_address.sin_addr.s_addr = INADDR_ANY;

    error_check = bind(server_socket, (struct sockaddr *) &server_address, sizeof(server_address));
    if(error_check == -1)
    {
        printf("Error while binding\n");
        exit(1);
    }

    // get message from client 1
    error_check = recvfrom(server_socket,server_message,sizeof(server_message),0, (struct
sockaddr *) &client_address, &client_len);
    if(error_check == -1)
    {
```

```
        printf("error receiving message from client.\n");
        exit(1);
    }

    printf("Client sent character: %s\n", server_message);

    // process the message and decrement the character sent by client 1
    if(server_message[0] == 'a')
    {
        server_message[0] = 'z';
    }
    else if (server_message[0] == 'A')
    {
        server_message[0] == 'Z';
    }
    else
    {
        server_message[0] = server_message[0] - 1;
    }

    // client 2 pings so that its ip address and port can be known and response can be sent.
    error_check = recvfrom(server_socket,ping,sizeof(ping),0, (struct sockaddr *) &client_address,
&client_len);
    if(error_check == -1)
    {
        printf("error receiving message from client.\n");
        exit(1);
    }

    error_check = sendto(server_socket, server_message, sizeof(server_message), 0, (struct
sockaddr *) &client_address, client_len);
    if(error_check == -1)
    {
        printf("Error while sending message.\n");
        exit(1);
    }

    printf("Server says I am done.\n");

    close(server_socket);
    return 0;
}
```