SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Winter Semester-2020-21/InLab Examination

Programme and Branch: B.Tech – CSE **Max. Marks: 20**

Course: CSE1004-Network and Communication **Duration : 60 Min**

Name: Kulvir Singh
Register No.: 19BCE2074

4.A)In a CRC error-detecting scheme, choose Encode the bits 10010011011.Suppose the channel introduces an error pattern 100010000000000 (i.e., a flipfrom 1 to 0 or from 0 to 1 in position 1 and What is received? Can the error bedetected?Repeat this with error pattern 100110000000001.

Aim:

To write a program to implement the CRC mechanism and calculate the encoded value of the given message bits. Also check the errors for the same message for the 2 different cases given

Algorithm:

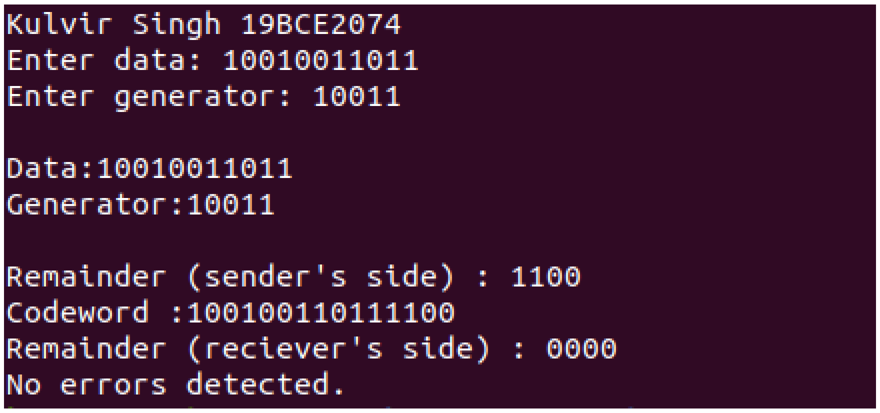
The communicating parties agrees upon the size of message,M(x) and the generator polynomial, G(x). If r is the order of G(x),r, bits are appended to the low order end of M(x). This makes the block size bits, the value of which is xrM(x). The block xrM(x) is divided by G(x) using modulo 2 division. The remainder after division is added to xrM(x) using modulo 2 addition. The result is the frame to be transmitted, T(x). The encoding procedure makes exactly divisible by G(x). The receiver divides the incoming data frame T(x) unit by G(x) using modulo 2 division. Mathematically, if E(x) is the error, then modulo 2 division of [M(x) + E(x)] by G(x) is done. If there is no remainder, then it implies that E(x). The data frame is accepted. A remainder indicates a non-zero value of E(x), or in other words presence of an error. So, the data frame is rejected. The receiver may then send an erroneous acknowledgment back to the sender for retransmission.

Code :

```
#include<iostream>
#include<stdlib.h>
using namespace std;


string xor1(string a, string b)
{
    string result = "";

    int n = b.length();


    for(int i = 1; i < n; i++)
    {
        if (a[i] == b[i])
            result += "0";
        else
            result += "1";
    }
    return result;
}


string mod2div(string divident, string divisor)
{


    int pick = divisor.length();


    string tmp = divident.substr(0, pick);

    int n = divident.length();

    while (pick < n)
    {
        if (tmp[0] == '1')

            tmp = xor1(divisor, tmp) + divident[pick];
        else


            tmp = xor1(std::string(pick, '0'), tmp) +
            divident[pick];
        pick += 1;
    }


    if (tmp[0] == '1')
        tmp = xor1(divisor, tmp);
    else
        tmp = xor1(std::string(pick, '0'), tmp);

    return tmp;
}


string encodeData(string data, string key)
{
    int l_key = key.length();


    string appended_data = (data +
                            std::string(
                                l_key - 1, '0'));

    string remainder = mod2div(appended_data, key);
    cout << "Remainder (sender's side) : "<< remainder << "\n";
    return remainder;


}


int main()
{
    cout<<"Kulvir Singh 19BCE2074\n";
    string data;
    string key;
    cout<<"Enter data: ";
    cin>>data;
    cout<<"Enter generator: ";
    cin>>key;
    cout<<"\nData:"<<data<<"\n";
    cout<<"Generator:"<<key<<"\n\n";
    string rem1 = encodeData(data, key);
    string codeword = data+rem1;
    string codeword1 = "100010000000000";
    string codeword2 = "100110000000001";
    cout << "Codeword :"<< codeword << "\n";

    string rem2 = mod2div(codeword, key);
    cout << "Remainder (reciever's side) : "<< rem2 << "\n";

    int flag=0;
    for(int i=0;i<rem2.length();i++){
        if(rem2[i]!='0'){
            cout<< "Error detected!"<<"\n";
            flag=1;
            break;
        }
    }

    if(flag==0)
        cout<<"No errors detected."<<"\n";
    return 0;
}
```

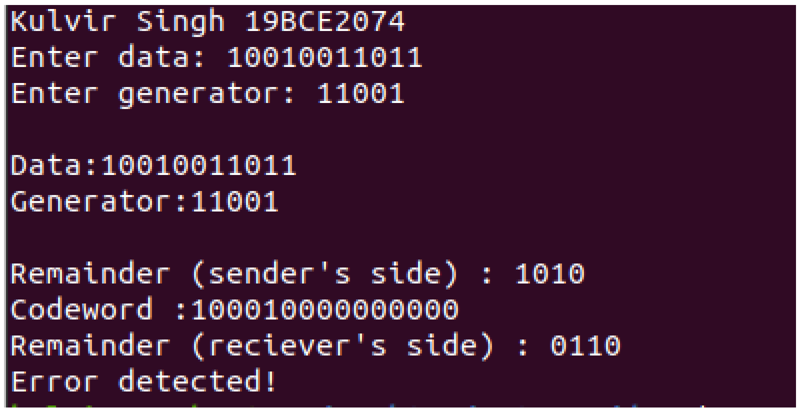
Output Screenshots:

Case 1: Given message bits is encoded and new codeword sent to receiver is correct



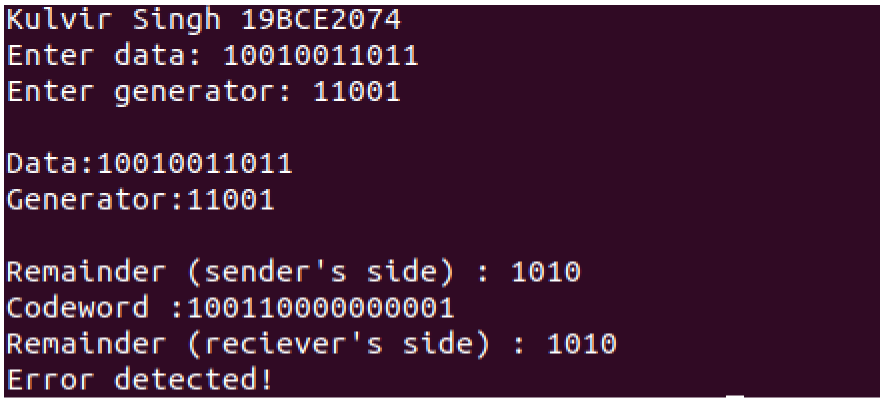
```
Kulvir Singh 19BCE2074
Enter data: 10010011011
Enter generator: 10011

Data:10010011011
Generator:10011

Remainder (sender's side) : 1100
Codeword :100100110111100
Remainder (reciever's side) : 0000
No errors detected.
```

Case 2: Checking error at receiver side when sent codeword is 100010000000000



```
Kulvir Singh 19BCE2074
Enter data: 10010011011
Enter generator: 11001

Data:10010011011
Generator:11001

Remainder (sender's side) : 1010
Codeword :100010000000000
Remainder (reciever's side) : 0110
Error detected!
```

Case 3: Checking error at receiver side when sent codeword is 100110000000001



```
Kulvir Singh 19BCE2074
Enter data: 10010011011
Enter generator: 11001

Data:10010011011
Generator:11001

Remainder (sender's side) : 1010
Codeword :100110000000001
Remainder (reciever's side) : 1010
Error detected!
```

B)Develop a TCP client/server application for transferring a text file from client to server.

Aim:

To create a TCP client server application for transferring a file from client to server and the client receives an acknowledgement from the server once the file is received.

Algorithm:

TCP Server –

1)using create(), Create TCP socket.
2)using bind(), Bind the socket to server address.
3)using listen(), put the server socket in a passive mode, where it waits for the client to approach the server to make a connection
4)using accept(), At this point, connection is established between client and server, and they are ready to transfer data.
5)Go back to Step 3.
TCP Client –

1)Create TCP socket.
2)connect newly created client socket to server.

Code:

Server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
void error(const char *msg)
{
perror(msg);
exit(1);
}
int main(int argc, char *argv[])
{
   int socket1, newsocket1, portNum;
   socklen_t clilen;
   char buffer[255];
   struct sockaddr_in serv_addr, cli_addr;
   int n;
   if (argc < 2) {
   fprintf(stderr,"ERROR, port not given\n");
   exit(1);
   }
   socket1 = socket(AF_INET, SOCK_STREAM, 0);
   if (socket1 < 0)
   error("ERROR can't open socket");
   bzero((char *) &serv_addr, sizeof(serv_addr));
   portNum = atoi(argv[1]);

   serv_addr.sin_family = AF_INET;
   serv_addr.sin_addr.s_addr = INADDR_ANY;
   serv_addr.sin_port = htons(portNum);
   if (bind(socket1, (struct sockaddr *) &serv_addr,
   sizeof(serv_addr)) < 0)
   error("ERROR can't bind");
   listen(socket1,5);
   clilen = sizeof(cli_addr);
   newsocket1 = accept(socket1,
   (struct sockaddr *) &cli_addr,
   &clilen);
   if (newsocket1 < 0)
   error("ERROR can't accept");
   while(1)
   {
   bzero(buffer,255);
   n = read(newsocket1,buffer,255);
   if (n < 0) error("ERROR ,can't read");
   printf("Client: %s\n",buffer);
   bzero(buffer,255);
   fgets(buffer,255,stdin);
   n = write(newsocket1,buffer,strlen(buffer));
   if (n < 0) error("ERROR ,can't write");
   int i=strncmp("Goodbye" , buffer, 7);
   if(i == 0)
   break;
   }
   close(newsocket1);
   close(socket1);
   return 0;
}
```

__Client.c__

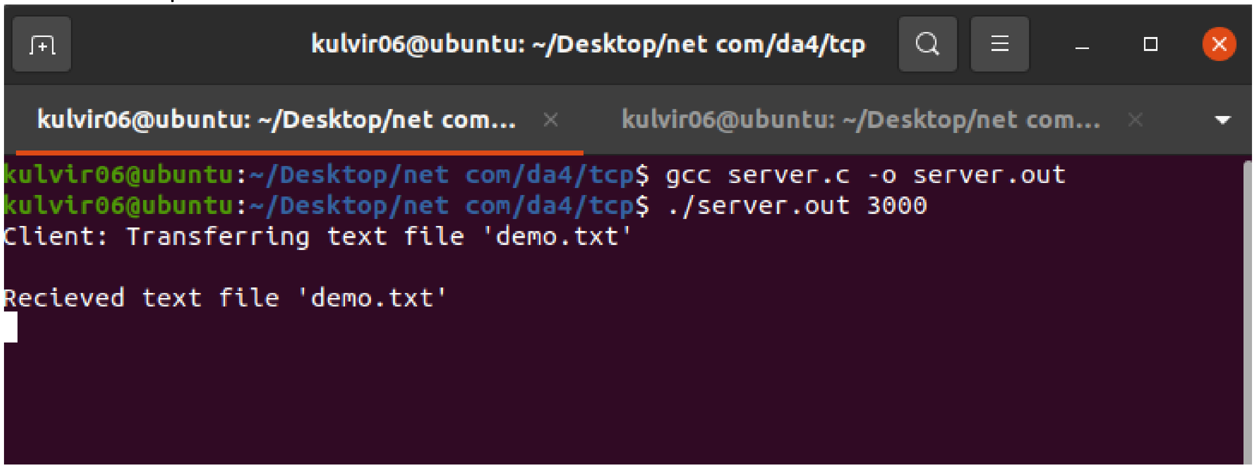
```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
```

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
void error(const char *msg)
{
perror(msg);
exit(0);
}
int main(int argc, char *argv[])
{
int socket1, portnum, n;
struct sockaddr_in serv_addr;
struct hostent *server;
char buffer[256];
if (argc < 3) {
fprintf(stderr,"usage %s hostname port\n", argv[0]);
exit(0);
}
portnum = atoi(argv[2]);
socket1 = socket(AF_INET, SOCK_STREAM, 0);
if (socket1 < 0)
error("ERROR, can't open socket");
server = gethostbyname(argv[1]);
if (server == NULL) {
fprintf(stderr,"ERROR, there exists no such host\n");

exit(0);
}
bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
bcopy((char *)server->h_addr,
(char *)&serv_addr.sin_addr.s_addr,
server->h_length);
serv_addr.sin_port = htons(portnum);
if (connect(socket1,(struct sockaddr *) &serv_addr,sizeof(serv_addr)) < 0)
error("ERROR,can't connect");
printf("Client: ");
while(1)
{
bzero(buffer,256);
fgets(buffer,255,stdin);
n = write(socket1,buffer,strlen(buffer));
if (n < 0)
error("ERROR, writing to socket");
bzero(buffer,256);
n = read(socket1,buffer,255);
if (n < 0)
error("ERROR, reading from socket");
printf("Server : %s\n",buffer);
int i = strncmp("Goodbye" , buffer , 7);
if(i == 0)
break;
}
}
```

Output Screenshots:

Server side output :

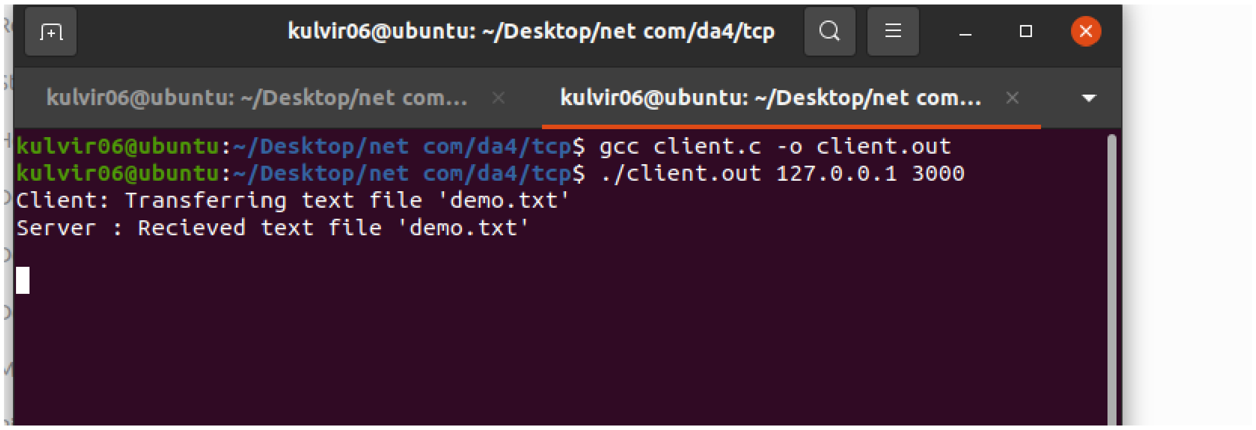
Run server at port 3000


kulvir06@ubuntu: ~/Desktop/net com/da4/tcp
kulvir06@ubuntu: ~/Desktop/net com...
kulvir06@ubuntu: ~/Desktop/net com...

```
kulvir06@ubuntu:~/Desktop/net com/da4/tcp$ gcc server.c -o server.out
kulvir06@ubuntu:~/Desktop/net com/da4/tcp$ ./server.out 3000
Client: Transferring text file 'demo.txt'

Recieved text file 'demo.txt'
```

Client side output:

Run client at 127.0.0.1, port 3000 and send file demo.txt to server


kulvir06@ubuntu: ~/Desktop/net com/da4/tcp
kulvir06@ubuntu: ~/Desktop/net com...
kulvir06@ubuntu: ~/Desktop/net com...

```
kulvir06@ubuntu:~/Desktop/net com/da4/tcp$ gcc client.c -o client.out
kulvir06@ubuntu:~/Desktop/net com/da4/tcp$ ./client.out 127.0.0.1 3000
Client: Transferring text file 'demo.txt'
Server : Recieved text file 'demo.txt'
```