KANDRA KSHEERAJ **19BCE0829**

CSE 1004 – Network and Communication

SLOT: L47+L48

Faculty: SRIMATHI C mam

Term End Lab FAT

Lab Date: 02-06-2021

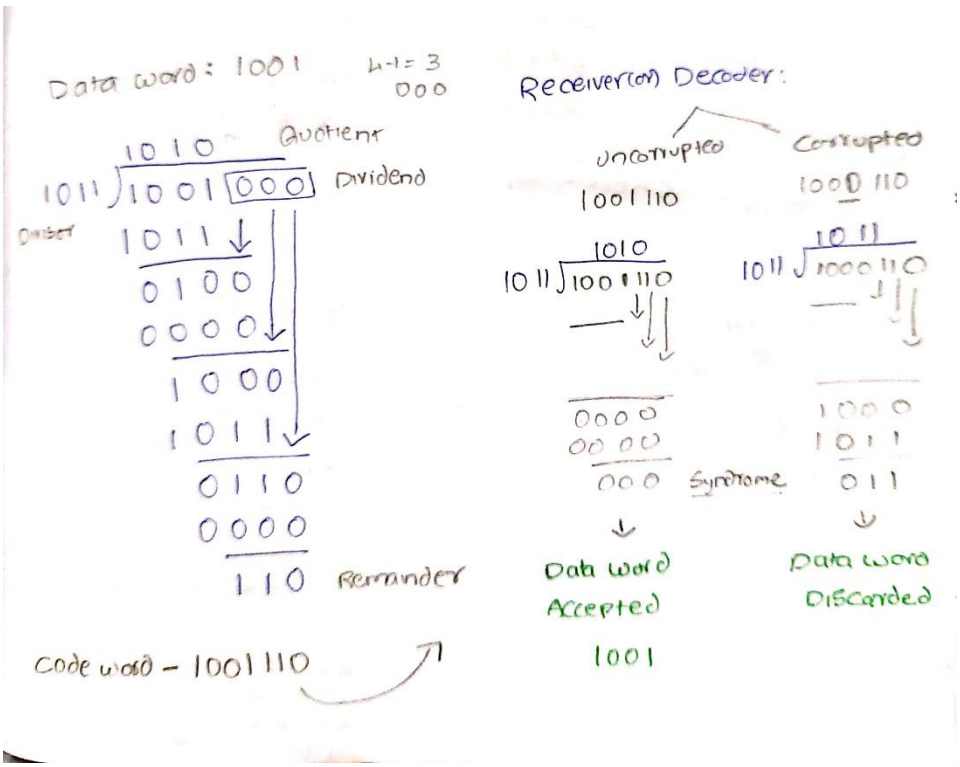
Question 2

a. If the bit sequence 101001100 is received at the host, assume the CRC code polynomial as x 4 + x2 + 1. Implement a program in C to Check, is there an error with this transmission

Concept:

CRC or Cyclic Redundancy Check is a method of detecting accidental changes/errors in the communication channel. It is one of method in Error control mechanism apart from hamming code, checksum methods.

Explanation from notes.



Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
char* xor(char *a, char *b){
int i, length=strlen(a);
char *result;
result = (char *)malloc(sizeof(char)*length);
for (i=0;i<length;i++){
      if(a[i]==b[i])
            {result[i]='0';}
      else
            {result[i]='1';}
      }
      return(result);
}
      char* mod2div(char *dividend,char *divisor){
      int dividend_length = strlen(dividend);
      int divisor_length = strlen(divisor);
      int i;
      char *tmp;
      tmp = (char *)malloc((divisor_length-1)*sizeof(char));
      for(i=0;i<divisor_length;i++){
            tmp[i] = dividend[i];
      }
      int index=divisor_length;
      char* zero;
      while (index < dividend_length)
      {
      if (tmp[0] == '1')
      {
      tmp = xor(divisor, tmp);
      for(i=0;i<divisor_length-1;i++)
            tmp[i]=tmp[i+1];
            tmp[divisor_length-1]=dividend[index];
      }
      else
      {
      zero = (char *)malloc(sizeof(char)*(divisor_length));
      for(i=0;i<divisor_length;i++)
            {zero[i]='0';}
            tmp = xor(zero, tmp);
      for(i=0;i<divisor_length-1;i++)
```

```
            tmp[i]=tmp[i+1];
            tmp[divisor_length-1]=dividend[index];
      }
      index += 1;
      }
      if (tmp[0] == '1')
      {
      tmp = xor(divisor, tmp);
      }
      else
      {
            zero = (char *)malloc(sizeof(char)*(divisor_length));
      for(i=0;i<divisor_length;i++)
            {zero[i]='0';}
      tmp = xor(zero, tmp);
      }
      char *crc;
      divisor_length--;
      crc = (char *)malloc((divisor_length)*sizeof(char));
      for(i=0;i<divisor_length;i++)
            crc[i]=tmp[i+1];
            crc[divisor_length]='\0';
            printf("remainder: %s\n", crc);
      return(crc);
}
char* append_data(char *message,char *append_with,int zeros){
 int i;
 int message_length = strlen(message);
 int append_with_length = strlen(append_with);
 char *result;
 append_with_length--;
 result = (char
*)malloc((message_length+append_with_length)*sizeof(char));
 for(i=0;i<strlen(message);i++)
      result[i]=message[i];
 if(zeros==1){
 for(i=0;i<append_with_length;i++)
      result[message_length+i]='0';
 }
 else{
      append_with_length++;
 for(i=0;i<append_with_length;i++)
      result[message_length+i]=append_with[i];
```

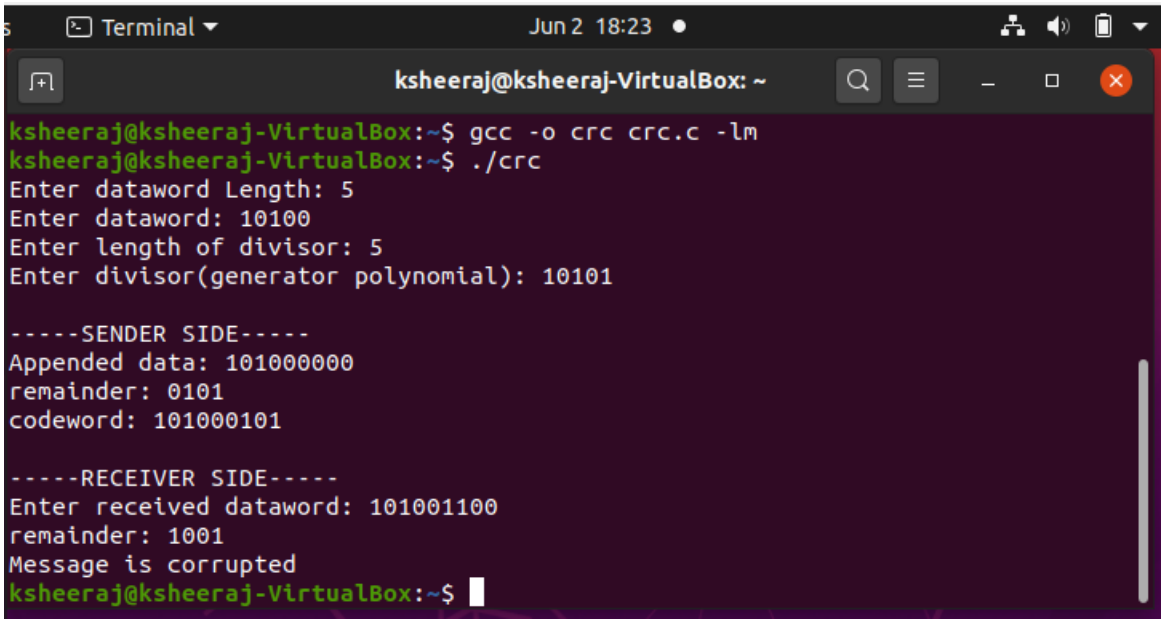
```
 }
 result[message_length+append_with_length]='\0';
return(result);
}
void sender(char dataword[],char divisor[])
{
char *appended_data=append_data(dataword,divisor,1);
printf("Appended data: %s\n", appended_data);
char *crc = mod2div(appended_data,divisor);
 char *codeword = append_data(dataword,crc,0);
 printf("codeword: %s\n",codeword);
}
void receiver(char received_codeword[],char divisor[])
{
 char *received_crc = mod2div(received_codeword,divisor);
 int i;
 for(i=0;i<strlen(divisor)-1;i++){
 if(received_crc[i]!='0'){
 printf("Message is corrupted\n");
 exit(0);
 }
 }
 printf("Message is correct \n");
}
int main(){
 int i;
 int dataword_length,divisor_length;
     printf("Enter dataword Length: ");
 scanf("%d",&dataword_length);
 char dataword[dataword_length];
     printf("Enter dataword: ");
 scanf("%s",dataword);
     printf("Enter length of divisor: ");
 scanf("%d",&divisor_length);
 char divisor[divisor_length];
     printf("Enter divisor(generator polynomial): ");
 scanf("%s",divisor);
     printf("\n-----SENDER SIDE-----\n");
 sender(dataword,divisor);
     printf("\n-----RECEIVER SIDE-----\n");
 char received_codeword[dataword_length+divisor_length-1];
     printf("Enter received dataword: ");
 scanf("%s",received_codeword);
```

```
 receiver(received_codeword,divisor);
 return 0;
}
```

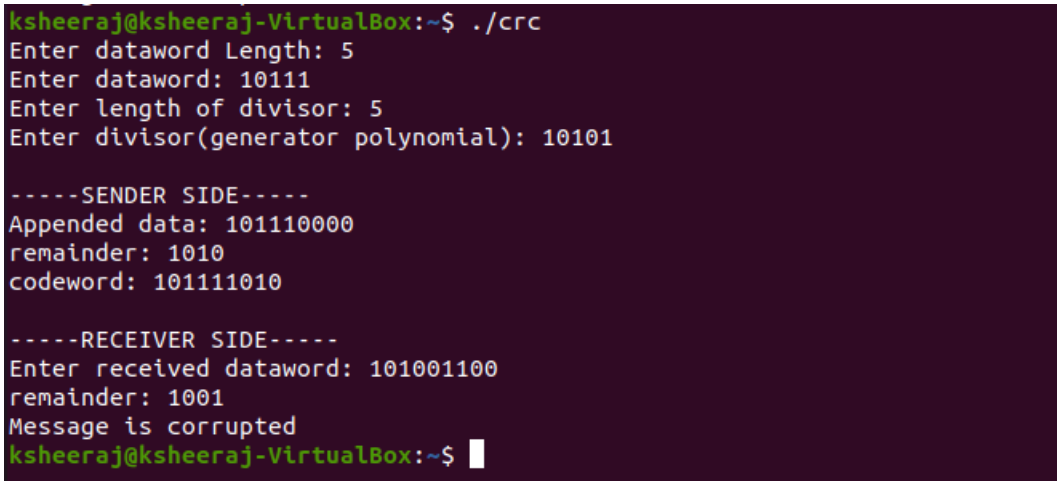
Output Screenshot:

Given generator polynomial: $x^4 + x^2 + 1$ (10101)
Given Codeword input in question: 101001100
Possible data word: 10100

Sample Run 1:



Sample Run 2:



b. Write a socket program (UDP) to extract the substring from the string on the server side when a string is passed from the client side

Concept:

UDP is same as the TCP protocol except this doesn't guarantee the error-checking and data recovery. If you use this protocol, the data will be sent continuously, irrespective of the issues in the receiving end.

Code:

```
   /****************** SERVER CODE ****************/
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <ctype.h>

int main(void){
    int socket_desc, c=0, position, length;
    struct sockaddr_in server_addr, client_addr;
    char server_message[100], client_message[100],
string[1000],sub[1000];
    int client_struct_length = sizeof(client_addr);

    memset(server_message, '\0', sizeof(server_message));
    memset(client_message, '\0', sizeof(client_message));

    socket_desc = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);

    if(socket_desc < 0){
        printf("Error while creating socket\n");
        return -1;
    }
    printf("Socket created successfully\n");

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(2000);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    // Bind to the set port and IP:
    if(bind(socket_desc, (struct sockaddr*)&server_addr,
sizeof(server_addr)) < 0){
```

```
        printf("Couldn't bind to the port\n");
        return -1;
    }
    printf("Done with binding\n");

    printf("Listening ..\n\n");

    // Receive client's message:
    if (recvfrom(socket_desc, client_message,
sizeof(client_message), 0,
         (struct sockaddr*)&client_addr, &client_struct_length) <0){
        printf("Couldn't receive\n");
        return -1;
    }
    printf("Received message from IP: %s and port: %i\n",
           inet_ntoa(client_addr.sin_addr),
ntohs(client_addr.sin_port));

    printf("Msg from client: %s\n", client_message);


    printf("Enter the position and length of substring\n");
    scanf("%d%d", &position, &length);
    while (c < length) {
      sub[c] = client_message[position+c-1];
      c++;
      }

    strcpy(server_message, sub);

    if (sendto(socket_desc, server_message, strlen(server_message),
0,
         (struct sockaddr*)&client_addr, client_struct_length) < 0){
        printf("Can't send\n");
        return -1;
    }
}
```

/**************** CLIENT CODE ****************/**

```
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
```

```
#include <ctype.h>
int main(void){
    int socket_desc;
    struct sockaddr_in server_addr;
    char server_message[100], client_message[100];
    int server_struct_length = sizeof(server_addr);

    // Clean buffers:
    memset(server_message, '\0', sizeof(server_message));
    memset(client_message, '\0', sizeof(client_message));

    // Create socket:
    socket_desc = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);

    if(socket_desc < 0){
        printf("Error while creating socket\n");
        return -1;
    }
    printf("Socket created successfully\n");
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(2000);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    printf("Enter message: ");
    fgets(client_message, 20, stdin);
    // Send the message to server:
    if(sendto(socket_desc, client_message, strlen(client_message),0,
(struct sockaddr*)&server_addr, server_struct_length) < 0){
        printf("Unable to send message\n");
        return -1;
    }

    // Receive the server's response:
    if(recvfrom(socket_desc, server_message,
sizeof(server_message),0, (struct sockaddr*)&server_addr,
&server_struct_length) < 0){
        printf("Error while receiving server's msg\n");
        return -1;
    }

    printf("Data received: %s\n", server_message);

    return 0;
}
```

KANDRA KSHEERAJ
19BCE0829

Output Screenshot:

Sample Run 1:
String input: ksheeraj

```
ksheeraj@ksheeraj-VirtualBox:~$ cc udpserver1.c
ksheeraj@ksheeraj-VirtualBox:~$ ./a.out
Socket created successfully
Done with binding
Listening ..

Received message from IP: 127.0.0.1 and port: 34546
Msg from client: ksheeraj

Enter the position and length of substring
2
5
ksheeraj@ksheeraj-VirtualBox:~$
```

Extracted Substring: sheer

```
ksheeraj@ksheeraj-VirtualBox:~$ cc udpclient.c
ksheeraj@ksheeraj-VirtualBox:~$ ./a.out
Socket created successfully
Enter message: ksheeraj
Data received: sheer
```

Sample Run 2:
String input: my name is ksheeraj
Extracted Substring: name

```
ksheeraj@ksheeraj-VirtualBox:~$ cc udpserver1.c
ksheeraj@ksheeraj-VirtualBox:~$ ./a.out
Socket created successfully
Done with binding
Listening ..

Received message from IP: 127.0.0.1 and port: 60842
Msg from client: my name is ksheeraj
Enter the position and length of substring
4
4
ksheeraj@ksheeraj-VirtualBox:~$
```

```
ksheeraj@ksheeraj-VirtualBox:~$ cc udpclient.c
ksheeraj@ksheeraj-VirtualBox:~$ ./a.out
Socket created successfully
Enter message: ksheeraj
Data received: sheer
ksheeraj@ksheeraj-VirtualBox:~$ cc udpclient.c
ksheeraj@ksheeraj-VirtualBox:~$ ./a.out
Socket created successfully
Enter message: my name is ksheeraj
Data received: name
```