**Assignment 3**

## Name : Shrinivas Hatyalikar

## Div: TY-B (B2)

## Roll No: 26

**Write a program for error detection and correction for 7/8 bits ASCII codes using Hamming Codes or CRC.**

CRC

#include <stdio.h>

#include <string.h>

// length of the generator polynomial

#define N strlen(gen\_poly)

// data to be transmitted and received

char data[28];

// CRC value

char check\_value[28];

// generator polynomial

char gen\_poly[10];

int data\_length, i, j;

// function that performs XOR operation

void XOR()

{

    // if both bits are the same, the output is 0  and if they are different then 1

    //

    for (j = 1; j < N; j++)

        check\_value[j] = ((check\_value[j] == gen\_poly[j]) ? '0' : '1');

}

void crc()

{

    for (i = 0; i < N; i++)

        check\_value[i] = data[i];

    do

    {

        if (check\_value[0] == '1')

            XOR(); // if check value is 1 do the XOR

        for (j = 0; j < N - 1; j++)

            check\_value[j] = check\_value[j + 1]; // move by 1 bit

        check\_value[j] = data[i++];

    } while (i <= data\_length + N - 1);

}

void receiver()

{

    printf("Enter the received data: ");

    scanf("%s", data);

    printf("\n-----------------------------\n");

    printf("Data received: %s", data);

    crc();

    //Check if the remainder is zero to find the error

    int errorDetected = 0;

    for (i = 0; i < N - 1; i++)

    {

        if (check\_value[i] == '1')

        {

            errorDetected = 1;

            break;

        }

    }

    if (errorDetected)

    {

        printf("\nError detected\n\n");

    }

    else

    {

        printf("\nNo error detected\n\n");

    }

}

//     for (i = 0; (i < N - 1) && (check\_value[i] != '1'); i++)

//         ;

//     if (i < N - 1)

//         printf("\nError detected\n\n");

//     else

//         printf("\nNo error detected\n\n");

// }

int main()

{

    // get the data to be transmitted

    printf("\nEnter data to be transmitted: ");

    scanf("%s", data);

    printf("\n Enter the Generating polynomial: ");

    // get the generator polynomial

    scanf("%s", gen\_poly);

    // find the length of data

    data\_length = strlen(data);

    // appending n-1 zeros to the data

    for (i = data\_length; i < data\_length + N - 1; i++)

        data[i] = '0';

    printf("\n----------------------------------------");

    // print the data with padded zeros

    printf("\n Data padded with n-1 zeros : %s", data);

    printf("\n----------------------------------------");

    // Cyclic Redundancy Check

    crc();

    // print the computed check value

    printf("\nCRC or Check value is : %s", check\_value);

    // Append data with check\_value(CRC)

    for (i = data\_length; i < data\_length + N - 1; i++)

        data[i] = check\_value[i - data\_length];

    printf("\n----------------------------------------");

    // printing the final data to be sent

    printf("\n Final data to be sent : %s", data);

    printf("\n----------------------------------------\n");

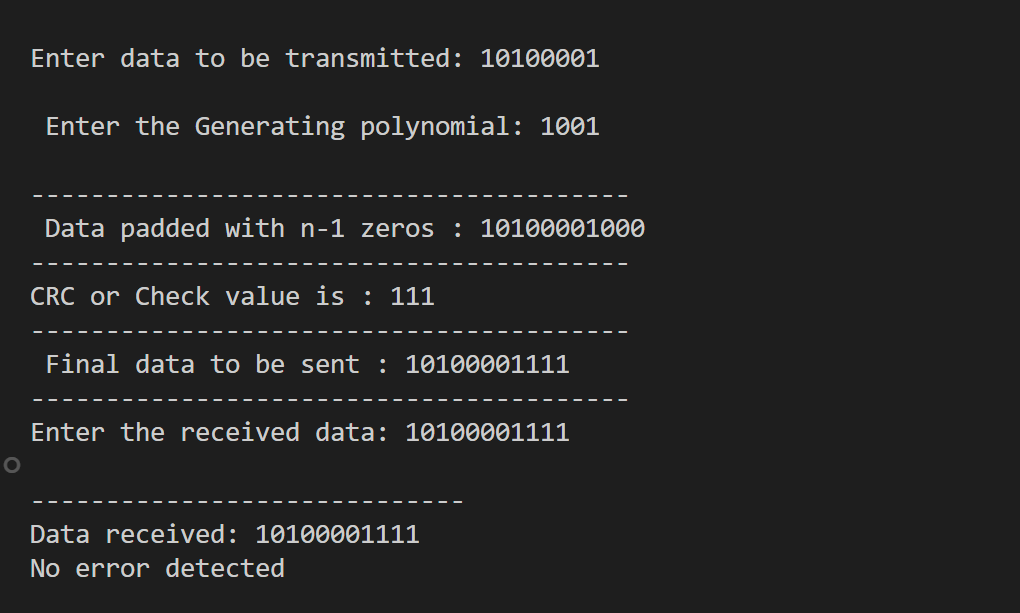
    // Calling the receiver function to check errors

    receiver();

    return 0;

}

Output:



**Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode.**

**Server**

#include <stdio.h>

#include <string.h>

#include <winsock2.h>

#define SERV\_TCP\_PORT 5000

#define N strlen(gen\_poly)

char gen\_poly[10];

char check\_value[28];

char data[30];

int i, j;

void XOR()

{

    for (j = 1; j < N; j++)

        check\_value[j] = ((check\_value[j] == gen\_poly[j]) ? '0' : '1');

}

void crc()

{

    for (i = 0; i < N; i++)

        check\_value[i] = data[i];

    do

    {

        if (check\_value[0] == '1')

            XOR();

        for (j = 0; j < N - 1; j++)

            check\_value[j] = check\_value[j + 1];

        check\_value[j] = data[i++];

    } while (i <= strlen(data) + N - 1);

}

void server()

{

    WSADATA wsa;

    SOCKET sockfd, newsockfd;

    struct sockaddr\_in serv\_addr, cli\_addr;

    int cli\_len = sizeof(cli\_addr);

    // Initialize Winsock

    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)

    {

        printf("Failed to initialize Winsock");

        return;

    }

    // Create a socket

    sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

    if (sockfd == INVALID\_SOCKET)

    {

        printf("Socket creation failed");

        WSACleanup();

        return;

    }

    printf("\nSocket created successfully");

    // Configure the server address

    serv\_addr.sin\_family = AF\_INET;

    serv\_addr.sin\_addr.s\_addr = INADDR\_ANY;

    serv\_addr.sin\_port = htons(SERV\_TCP\_PORT);

    // Bind the socket

    if (bind(sockfd, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) == SOCKET\_ERROR)

    {

        printf("Binding failed");

        closesocket(sockfd);

        WSACleanup();

        return;

    }

    printf("\nBinded successfully");

    // Listen for incoming connections

    if (listen(sockfd, 5) == SOCKET\_ERROR)

    {

        printf("Listening failed");

        closesocket(sockfd);

        WSACleanup();

        return;

    }

    printf("\nServer is listening...\n");

    // Accept incoming connections

    newsockfd = accept(sockfd, (struct sockaddr \*)&cli\_addr, &cli\_len);

    if (newsockfd == INVALID\_SOCKET)

    {

        printf("Acceptance failed");

        closesocket(sockfd);

        WSACleanup();

        return;

    }

    printf("\nClient accepted\n"); // Print when a client connects

    // Receive data from the client

    recv(newsockfd, data, 30, 0);

    recv(newsockfd, gen\_poly, 10, 0);

    printf("\n-----------------------------");

    printf("\nData received: %s", data);

    printf("\nGenerating polynomial: %s", gen\_poly);

    // Calculate CRC

    crc();

    int errorDetected = 0;

    for (i = 0; i < N - 1; i++)

    {

        if (check\_value[i] == '1')

        {

            errorDetected = 1;

            break;

        }

    }

    if (errorDetected)

    {

        printf("\nError detected\n");

    }

    else

    {

        printf("\nNo error detected\n");

    }

    printf("\n-----------------------------\n");

    // Close sockets and clean up

    closesocket(newsockfd);

    closesocket(sockfd);

    WSACleanup();

}

int main()

{

    server();

    return 0;

}

**Client**

#include <stdio.h>

#include <string.h>

#include <winsock2.h>

#define SERV\_TCP\_PORT 5000

#define N strlen(gen\_poly)

char gen\_poly[10];

char data[30];

char check\_value[28];

int data\_length, i, j;

void XOR() {

    for (j = 1; j < N; j++)

        check\_value[j] = ((check\_value[j] == gen\_poly[j]) ? '0' : '1');

}

void crc() {

    for (i = 0; i < N; i++)

        check\_value[i] = data[i];

    do {

        if (check\_value[0] == '1')

            XOR();

        for (j = 0; j < N - 1; j++)

            check\_value[j] = check\_value[j + 1];

        check\_value[j] = data[i++];

    } while (i <= data\_length + N - 1);

}

void sender() {

    WSADATA wsa;

    SOCKET sockfd;

    struct sockaddr\_in serv\_addr;

    // Initialize Winsock

    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0) {

        printf("Failed to initialize Winsock");

        return;

    }

    // Create a socket

    sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

    if (sockfd == INVALID\_SOCKET) {

        printf("Socket creation failed");

        WSACleanup();

        return;

    }

    printf("\nSocket created successfully");

    // Configure the server address

    serv\_addr.sin\_family = AF\_INET;

    serv\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1"); // Use the server's IP address

    serv\_addr.sin\_port = htons(SERV\_TCP\_PORT);

    // Connect to the server

    if (connect(sockfd, (struct sockaddr\*)&serv\_addr, sizeof(serv\_addr)) == SOCKET\_ERROR) {

        printf("Connection to server failed\n");

        closesocket(sockfd);

        WSACleanup();

        return;

    }

    printf("\nConnected to server\n"); // Print when the connection is successful

    printf("\nEnter data to be transmitted: ");

    scanf("%s", data);

    printf("\nEnter the Generating polynomial: ");

    scanf("%s", gen\_poly);

    data\_length = strlen(data);

    for (i = data\_length; i < data\_length + N - 1; i++)

        data[i] = '0';

    crc();

    printf("\n-----------------------------");

    printf("\nData padded with n-1 zeros: %s", data);

    printf("\nCRC or Check value is: %s", check\_value);

    printf("\nFinal data to be sent: %s", strcat(data, check\_value));

    printf("\n-----------------------------\n");

    // Send data to the server

    send(sockfd, data, 30, 0);

    send(sockfd, gen\_poly, 10, 0);

    // Close the socket and clean up

    closesocket(sockfd);

    WSACleanup();

}

int main() {

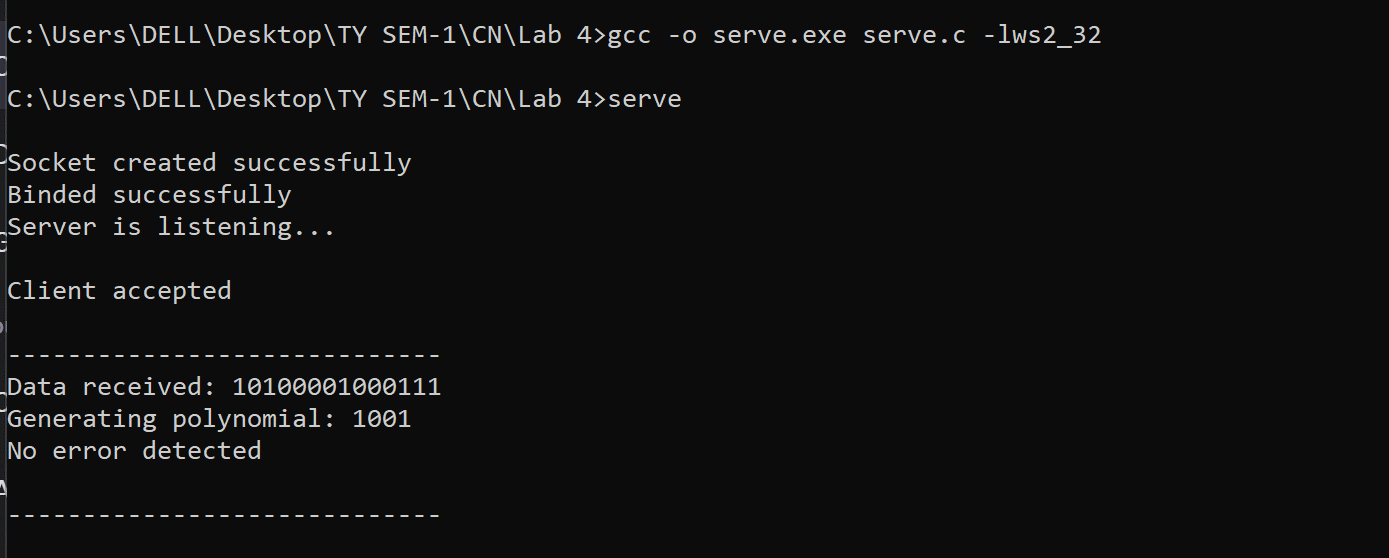
    sender();

    return 0;

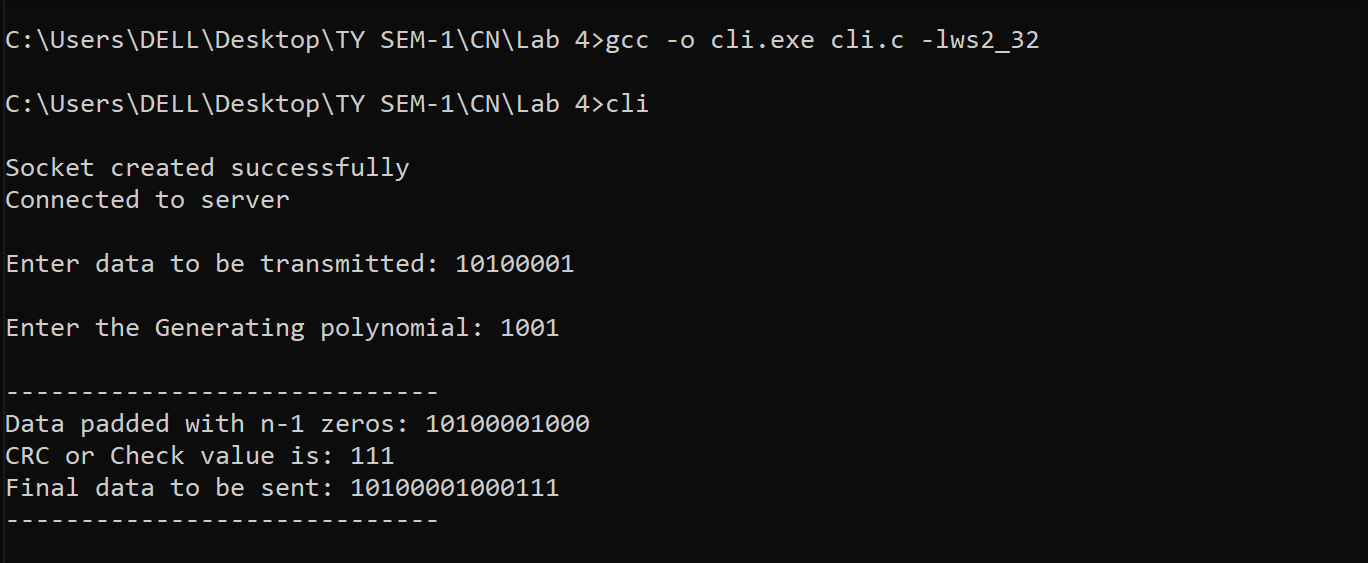
}

**Output:**

**Server**



**Client**



**Packets captured traces using Wireshark Packet Analyzer Tool**

