LAB 8

CYCLE 2:

Write a program for error detecting code using CRC-CCITT(16-bits).

Code:

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

// CRC-CCITT polynomial: x^16 + x^12 + x^5 + 1 (0x1021)
//#define CRC_POLY 0x1021

// Function to perform bitwise XOR on binary strings
void binaryXOR(char *result, const char *a, const char *b) {
    for (int i = 0; i < 16; i++) {
        result[i] = (a[i] == b[i]) ? '0' : '1';
    }
    result[16] = '\0';
}

// Function to calculate CRC-CCITT checksum
void calculateCRC(const char *data, int length, char *checksum) {
    char crc[17];
    for (int i = 0; i < 16; i++) {
        crc[i] = '0';
    }
```

```
    crc[16] = '\0';

    for (int i = 0; i < length; i++) {
        for (int j = 0; j < 8; j++) {
            char msb = crc[0];
            for (int k = 0; k < 16; k++) {
                crc[k] = crc[k + 1];
            }
            crc[15] = '0';

            if (msb == '1') {
                char temp[17];
                binaryXOR(temp, crc, "10001000000100001"); // CRC_POLY in
binary
                strcpy(crc, temp);
            }
        }
        crc[15] = (data[i] == '1') ? '1' : '0';
    }

    strcpy(checksum, crc);
}

int main() {
    char data[100]; // Replace with your actual data
```

```
    printf("Enter data in binary: ");
    scanf("%s", data);
    int dataLength = strlen(data);
    char checksum[17];
    calculateCRC(data, dataLength, checksum);

    printf("Calculated CRC: %s\n", checksum);
    // Simulating error by changing a bit
    // data[2] ^= 0x01; // Uncomment this line to introduce an error

    // Verify the received data
    char receivedChecksum[17];
    printf("Enter received CRC: ");
    scanf("%s", receivedChecksum);

    if (strcmp(receivedChecksum, checksum) == 0) {
        printf("Data is error-free.\n");
    } else {
        printf("Data contains errors.\n");
    }

    return 0;
```

<u>Output:</u>

```
Enter data in binary: 10001
Calculated CRC: 0111001001000001
Enter received CRC: 0111001001000001
Data is error-free.
```

```
Enter data in binary: 10011
Calculated CRC: 0111001101000001
Enter received CRC: 1011010101010101
Data contains errors.
```

Write a program for congestion control using Leaky bucket algorithm

Code:

```
#include<stdio.h>

int main(){
    int incoming, outgoing, buck_size, n, store = 0;
    printf("Enter bucket size:");
    scanf("%d", &buck_size);
    printf("Enter outgoing rate:");
    scanf("%d", &outgoing);
```

```
    printf("Enter number of inputs:");
    scanf("%d", &n);

    while (n != 0) {
        printf("Enter the incoming packet size: ");
        scanf("%d", &incoming);
        if (incoming <= (buck_size - store)){
            store += incoming;
            printf("Bucket buffer size %d out of %d\n", store, buck_size);
        } else {
            printf("Dropped %d no of packets\n", incoming - (buck_size - store));
            printf("Bucket buffer size %d out of %d\n", store, buck_size);
            store = buck_size;
        }
        store = store - outgoing;
        printf("After outgoing %d packets left out of %d in buffer\n", store,
buck_size);
        n--;
    }
}
```

OUTPUT:

```
Enter bucket size:1000
Enter outgoing rate:100
Enter number of inputs:3
Enter the incoming packet size: 300
Bucket buffer size 300 out of 1000
After outgoing 200 packets left out of 1000 in buffer
Enter the incoming packet size: 400
Bucket buffer size 600 out of 1000
After outgoing 500 packets left out of 1000 in buffer
Enter the incoming packet size: 1100
Dropped 600 no of packets
Bucket buffer size 500 out of 1000
After outgoing 900 packets left out of 1000 in buffer
```

Using TCP/IP sockets, write a client-server program to make the client send the file name and the server to send back the contents of the requested file if present.

Code:

ClientTCP.py

```
from socket import *
serverName = “127.0.0.1”
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = input(“\nEnter file name:”)
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print (“\nFrom Server:\n”)
print(filecontents)
clientSocket.close()
```

ServerTCP.py

```
from socket import *
serverName="127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
```

```
print ("The server is ready to receive")
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()
file=open(sentence,"r")
l=file.read(1024)
connectionSocket.send(l.encode())
print ("\nSent contents of "+ sentence)
file.close()
connectionSocket.close()
```

Output:

Server.tcp.py - C:/Users/dhiks/Desktop/TFCS Notes/Server.tcp.py (3.11.0)

File Edit Format Run Options Window Help

```
from socket import *
serverName="127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
print ("The server is ready to receive")
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()
file=open(sentence,"r")
l=file.read(1024)
connectionSocket.send(l.encode())
print ("\nSent contents of "+ sentence)
file.close()
connectionSocket.close()
```

Client.tcp.py - C:/Users/dhiks/Desktop/TFCS Notes/Client.tcp.py (3.11.0)

File Edit Format Run Options Window Help

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = input("\nEnter file name:")
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print ("\nFrom Server:\n")
print(filecontents)
clientSocket.close()
```

```
Enter file name:Server.tcp.py

From Server:

from socket import *
serverName="127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
    print ("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    file=open(sentence,"r")
    l=file.read(1024)
    connectionSocket.send(l.encode())
    print ("\nSent contents of "+ sentence)
    file.close()
    connectionSocket.close()
```

```
--------
The server is ready to receive

Sent contents of Server.tcp.py
The server is ready to receive
```

Using UDP sockets, write a client-server program to make the client send the file name and the server to send back the contents of the requested file if present.

<u>Code:</u>

ClientUDP.py

```
from socket import *

serverName = '127.0.0.1'

serverPort = 12000

clientSocket = socket(AF_INET, SOCK_DGRAM)

sentence = input('\nEnter file name')

clientSocket.sendto(bytes(sentence,'utf-8'),(serverName, serverPort))
```

```
filecontents,serverAddress = clientSocket.recvfrom(2048)
print ('\nReply from Server:\n')
print (filecontents.decode(&quot;utf-8&quot;))
clientSocket.close()
clientSocket.close()
```

ServerUDP.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('127.0.0.1', serverPort))
print ('The server is ready to receive')
while 1:
sentence, clientAddress = serverSocket.recvfrom(2048)
sentence = sentence.decode(‘utf-8’)
file=open(sentence,'r')
con=file.read(2048)
serverSocket.sendto(bytes(con,’utf=8’),clientAddress)
print ('\nSent contents of', end = ' ')
print (sentence)
file.close()
```

Output:

Client.udp.py - C:/Users/dhiks/Desktop/TFCS Notes/Client.udp.py (3.11.0)

File Edit Format Run Options Window Help

```
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input('\nEnter file name')
clientSocket.sendto(bytes(sentence,'utf-8'),(serverName, serverPort))
filecontents,serverAddress = clientSocket.recvfrom(2048)
print ('\nReply from Server:\n')
print (filecontents.decode('utf-8'))
clientSocket.close()
clientSocket.close()
```

Server.udp.py - C:/Users/dhiks/Desktop/TFCS Notes/Server.udp.py (3.11.0)

File Edit Format Run Options Window Help

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('127.0.0.1', serverPort))
print ('The server is ready to receive')
while 1:
        sentence, clientAddress = serverSocket.recvfrom(2048)
        sentence = sentence.decode('utf-8')
        file=open(sentence,'r')
        con=file.read(2048)
        serverSocket.sendto(bytes(con,'utf-8'),clientAddress)
        print ('\nSent contents of', end = ' ')
        print (sentence)
        file.close()
```

```
Enter file nameServer.udp.py

Reply from Server:

from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('127.0.0.1', serverPort))
print ('The server is ready to receive')
while 1:
        sentence, clientAddress = serverSocket.recvfrom(2048)
        sentence = sentence.decode('utf-8')
        file=open(sentence,'r')
        con=file.read(2048)
        serverSocket.sendto(bytes(con,'utf-8'),clientAddress)
        print ('\nSent contents of', end = ' ')
        print (sentence)
        file.close()
```

```
The server is ready to receive

Sent contents of Server.udp.py
```