PROGRAM-13

Write a program for error detecting code using CRC-CCITT (8-bits).

Code

```
def xor(a, b):
    # XOR operation between two binary strings
    result = []
    for i in range(1, len(b)):
        result.append('0' if a[i] == b[i] else '1') return
    ".join(result)

def mod2div(dividend, divisor): #
    Performs Modulo-2 division
    pick = len(divisor)
    tmp = dividend[:pick]

    while pick < len(dividend): if
        tmp[0] == '1':
            tmp = xor(divisor, tmp) + dividend[pick]
        else:
            tmp = xor('0' * pick, tmp) + dividend[pick] pick
        += 1

    # For the last set of bits if
    tmp[0] == '1':
        tmp = xor(divisor, tmp)
    else:
        tmp = xor('0' * pick, tmp)

    return tmp

def encode_data(data, key): #
    Encode   data   with   CRC
    l_key = len(key)
    padded_data = data + '0' * (l_key - 1)
    remainder = mod2div(padded_data, key)
    codeword = data + remainder
    return codeword, remainder

def check_data(received_data, key): #
    Check received data for errors
    remainder = mod2div(received_data, key)
    return '0' * (len(key) - 1) == remainder

# Main program
if __name__== "__main__":
```

```
    print("Error Detection using CRC-CCITT (8-bits)")

# Transmitter
    data = input("Enter data to be transmitted: ").strip()
    key = input("Enter the Generating polynomial: ").strip()

    print("\n----------------------------------------")
    padded_data = data + '0' * (len(key) - 1) print("Data
    padded with n-1 zeros:", padded_data)

    encoded_data, crc = encode_data(data, key)
    print("CRC or Check value is:", crc)
    print("Final data to be sent:", encoded_data)
    print("----------------------------------------")

    # Receiver
    received_data = input("\nEnter the received data: ").strip()
    print("\n----------------------------------------")
    print("Data received:", received_data)

    if check_data(received_data, key): print("No
        error detected")
    else:
        print("Error detected")
    print("----------------------------------------")
```

Output

```
Enter data to be transmitted: 1001100
Enter the Generating polynomial: 100001011

----------------------------------------
Data padded with n-1 zeros: 100110000000000
CRC or Check value is: 0100010
Final data to be sent: 10011000100010
----------------------------------------

Enter the received data: 10011000100011

------------------------------
Data received: 10011000100011
Error detected
```

```
Error Detection using CRC-CCITT (8-bits)
Enter data to be transmitted: 1001100
cell output actions rating polynomial: 100001011

----------------------------------------
Data padded with n-1 zeros: 100110000000000
CRC or Check value is: 10100010
Final data to be sent: 100110010100010
----------------------------------------

Enter the received data: 100110010100010

------------------------------
Data received: 100110010100010
No error detected
------------------------------
```

Implementation of CRC

```
# Code:
def XOR(a,b):
        result = []
        for i in range(1, len(b)):
            if (a[i] == b[i]):
                result.append('0')
            else:
                result.append('1')
        return ''.join('result')

def modediv(dividend, divisor):
        pick = len(divisor)
        temp = dividend[0: pick]
        while pick < len(dividend):
        if(temp[0] == '1'):
            temp = XOR(divisor, temp) + dividend
                                        [pick]
        else
            temp = XOR('0' * pick, temp) + dividend
                                           [pick]
        pick += 1
        if temp[0] == '1'
            temp = XOR(divisor, temp)
        else:
            temp = XOR(0* pick, temp)
        checkword = temp
        return checkword

def encodeData(data, Key):
        l_key = len(Key)
```

```
append_data = data + '0' * (1 - key - 1)
remainder = mod2div ( append_data, key)
codeword = data + remainder
print ("Remainder", remainder)
print ("Encode data (Data + Remainder)", codeword)

data = "100100"
key = "1101"
encodeData ( data, key)
```

Output

Sender site - -

Remainder : 001

Encode Data(Data + Remainder : 100100001

Receiver Side

correct message Received

3/1/25

PROGRAM-14(A)

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

Code: Client.py

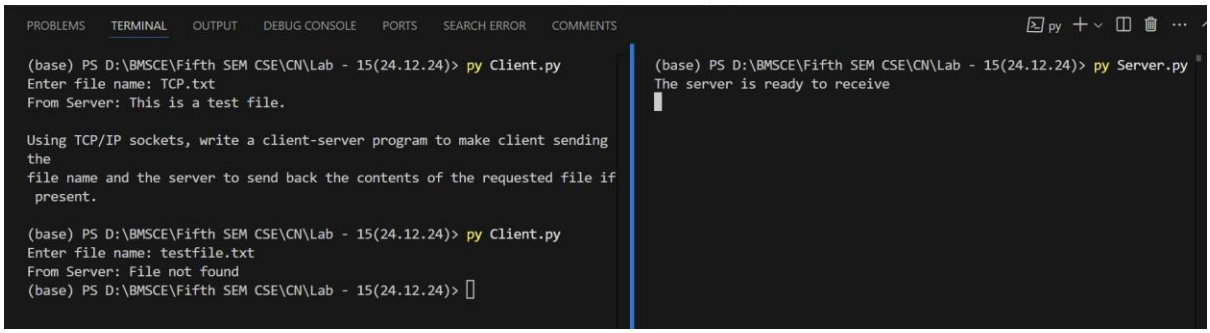
```
from socket import *
serverName = "127.0.0.1"  # Server address (localhost)
serverPort = 12000  # Port number where the server listens

# Create TCP socket
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))  # Connect to server

# Ask user for file name to request
sentence = input("Enter file name: ")

# Send file name to server
clientSocket.send(sentence.encode())

# Receive file contents from server
filecontents = clientSocket.recv(1024).decode()
print('From Server:', filecontents)

# Close the connection
clientSocket.close()
```

Code: Server.py

```
from socket import *
serverName = "127.0.0.1"  # Server address (localhost)
serverPort = 12000  # Port number to listen on

# Create TCP socket
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))  # Bind socket to the address and port
serverSocket.listen(1)  # Listen for 1 connection
print("The server is ready to receive")

while True:
    # Accept a connection
    connectionSocket, addr = serverSocket.accept()

    # Receive the file name from the client
    sentence = connectionSocket.recv(1024).decode()

    # Try opening the file try:
```

```
file = open(sentence, "r")  # Open file in read mode
        fileContents = file.read(1024)  # Read file content (up to 1024 bytes)
        file.close()
    except FileNotFoundError:
        # Send error message if file not found
        connectionSocket.send("File not found".encode())

    # Close the connection
    connectionSocket.close()
```

Output



Implementation of TCP|IP

Code:

Client.Py

```
from socket import *
Server name = "0.7.0.0.1"
Server Port = 12000
ClientSocket = Socket( AF_INET, SOCK_STREAM)
ClientSocket = connect(( serverName , serverfort)).
sentence = input ("Enter the name")
ClientSocket.send( sentence.encode())
filecontents = ClientSocket.rev( 6024).decode()
print('from server', file contents)
ClientSocket.close()
```

Server.Py

```
from Socket import
Server name = "127.0.0.1"
Serru Port = 12000
Server Socket = Socket( AF_INET , SOCK_STREAM)
Server.Socket.bind(( Server name , Server Port))
Server.Socket.Listen(1)
print (" the server is ready to receive")
while 1:
    connectionSocket , add = ServerSocket.accept()
    Servername = connectionSocket.recv(1024).decode()
    file = open( Seafuce , "r"
    l = file.read(1024)
    connection on socket.send( l.encode())
    file.close()
    connection.socket.close()
```

Output :-

Server side - - -

Server is ready to receive

Client side - - -

Enter file name : hello.txt

from server : Hellow word

3/1/25

PROGRAM-14(B)

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

Code: ClientUdp.py

```
from socket import *

serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)

sentence = input("Enter file name: ")
clientSocket.sendto(sentence.encode(), (serverName, serverPort))

filecontents, serverAddress = clientSocket.recvfrom(2048)
print('From Server:', filecontents.decode())

clientSocket.close()
```

Code: 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 True:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    try:
        with open(sentence.decode(), "r") as file:
            l = file.read(2048)
            serverSocket.sendto(l.encode(), clientAddress)
            print(f"Sent back to client: {l}")
    except FileNotFoundError:
        serverSocket.sendto("File not found.".encode(), clientAddress)
```

Output

```
PROBLEMS   TERMINAL   OUTPUT   DEBUG CONSOLE   PORTS   SEARCH ERROR   COMMENTS                                pwsh

(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py ClientUDP.py
Enter file name: UDP.txt
From Server: This is a test file.

Using UDP sockets, write a client-server program to make client sending
the file
name and the server to send back the contents of the requested file if p
resent.
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py ClientUDP.py
Enter file name: testfile.txt
From Server: File not found
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)>
```

```
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py ServerUDP.py
The server is ready to receive
```

Implement UDP

Q-?

Code:

Client UDP.py

```
from socket import *
Server name = "127.0.0.1"
Server Port = 12000
ClientSocket = Socket(AF_INET, SOCK_DGRAM)
Sentence = input("Enter file name")
ClientSocket.sendto(bytes(Sentence, "utf-8"),
                    "Servername, server port))
fileContents. serverAddress = clientSocket.recvform(2048)
print("from server", fileContents)
clientSocket.close()
```

Server UDP.py

```
from socket import
Server Port = 12000
ServerSockets = Socket(AF_INET, SOCK_DGRAM)
ServerSocket.bind("0.7.0.0.1", serverPort))
print(" the server is ready to receive")
while 1:
    sentence, clientAddress = ServerSockets.recvform
                                        (2048)
    file = open(Sentence, "r")
    l = file.read(2048)
    server socket.sendto(bytes(l, utf-8), clientAddress)
    print(" Sent back to client")
    file.close()
```

Output

Server Side - - -

The server side is ready to receive

Sent back to client : hello world

Client Side - - .

Enter file name : hello.txt

from server : hello world

Wireshark - missing - 3/1/25