Assignment 7

COMPUTER NETWORK
NAME = ANIK BARURY
ROLL = CSE22017
REG NO = 871

Q. Write a program (in C/C++/Java/Python) to implement a client-server program using TCP/UDP sockets. The client will send a message to the server, and the server will perform a cyclic redundancy check (CRC) on the message to detect errors. The server will then send the result back to the client. Display appropriate messages to the user indicating the status of the connection and the result of the CRC check.

Tcp _*Server.py* ==>

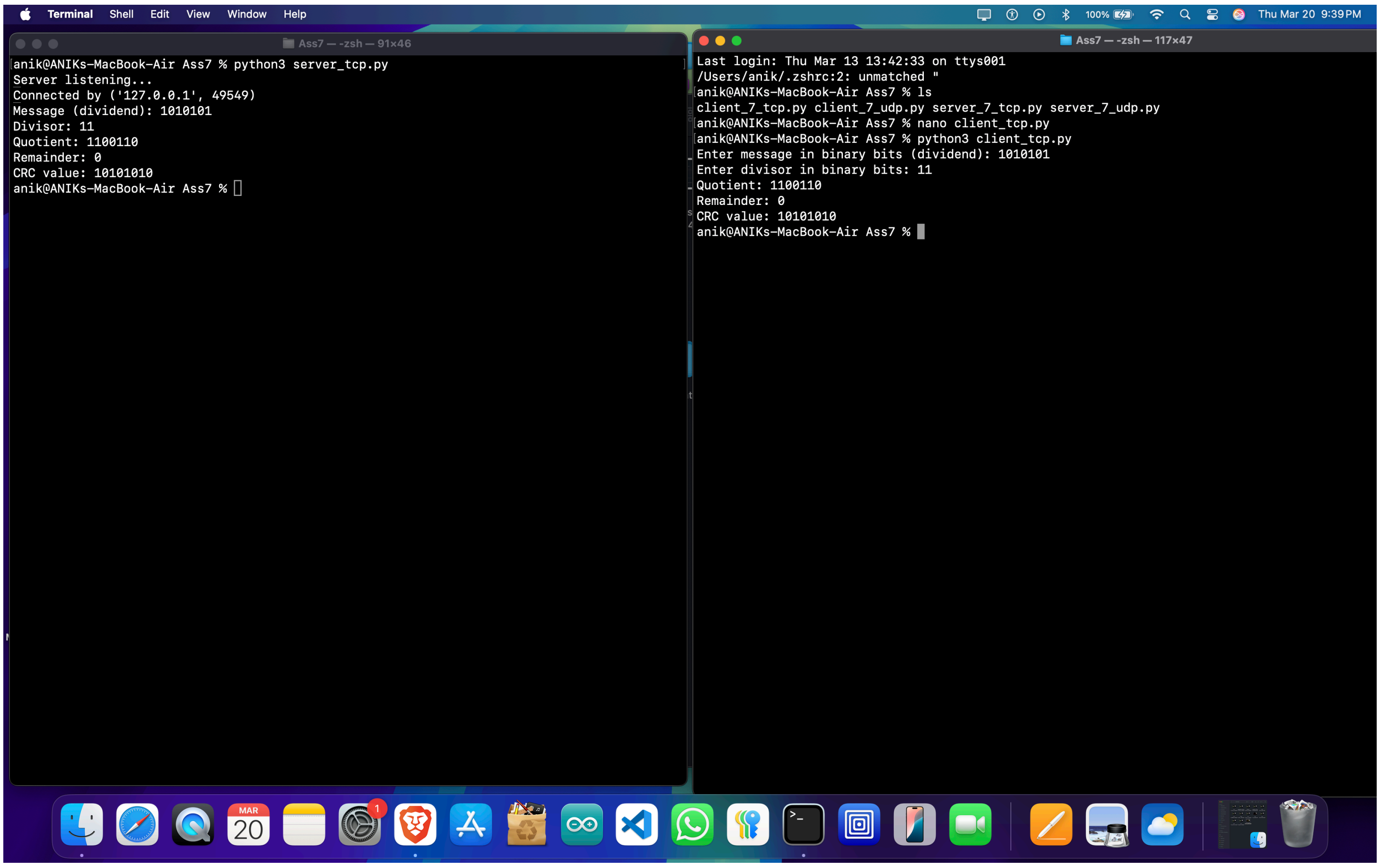
```
import socket

def crc(dividend, divisor):
    m = len(divisor)
    appended = dividend + "0"*(m-1)
    dividend_list = list(appended)
    quotient = ""
    for i in range(len(dividend)):
        if dividend_list[i] == '1':
            quotient += '1'
            for j in range(m):
                dividend_list[i+j] = str(int(dividend_list[i+j] != divisor[j]))
        else:
            quotient += '0'
    remainder = "".join(dividend_list[-(m-1):])
    crc_value = dividend + remainder
    return quotient, remainder, crc_value

host = 'localhost'
port = 12345

# Create socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((host, port))
s.listen(1)

print("Server listening...")

conn, addr = s.accept()
print(f"Connected by {addr}")

# Receive message and divisor
data = conn.recv(1024).decode()
message, divisor = data.split(',')
print("Message (dividend):", message)
print("Divisor:", divisor)
```

```
# Perform CRC
q, r, crc_value = crc(message, divisor)
print("Quotient:", q)
print("Remainder:", r)
print("CRC value:", crc_value)

# Send result back
conn.send(f"{q},{r},{crc_value}".encode())

conn.close()
s.close()
```

Tcp_Client.py ==>

```
import socket

host = 'localhost'
port = 12345

# Create socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))

# Get user input
message = input("Enter message in binary bits (dividend): ")
divisor = input("Enter divisor in binary bits: ")

# Send both message and divisor, separated by comma
s.send(f"{message},{divisor}".encode())

# Receive result
data = s.recv(1024).decode()
q, r, crc_value = data.split(',')
print("Quotient:", q)
print("Remainder:", r)
print("CRC value:", crc_value)

s.close()
```

Tcp_Output ==>



```
anik@ANIKs-MacBook-Air Ass7 % python3 server_tcp.py
Server listening...
Connected by ('127.0.0.1', 49549)
Message (dividend): 1010101
Divisor: 11
Quotient: 1100110
Remainder: 0
CRC value: 10101010
anik@ANIKs-MacBook-Air Ass7 %
```

```
Last login: Thu Mar 13 13:42:33 on ttys001
/Users/anik/.zshrc:2: unmatched "
anik@ANIKs-MacBook-Air Ass7 % ls
client_7_tcp.py client_7_udp.py server_7_tcp.py server_7_udp.py
anik@ANIKs-MacBook-Air Ass7 % nano client_tcp.py
anik@ANIKs-MacBook-Air Ass7 % python3 client_tcp.py
Enter message in binary bits (dividend): 1010101
Enter divisor in binary bits: 11
Quotient: 1100110
Remainder: 0
CRC value: 10101010
anik@ANIKs-MacBook-Air Ass7 %
```

Udp_Server.py ==>

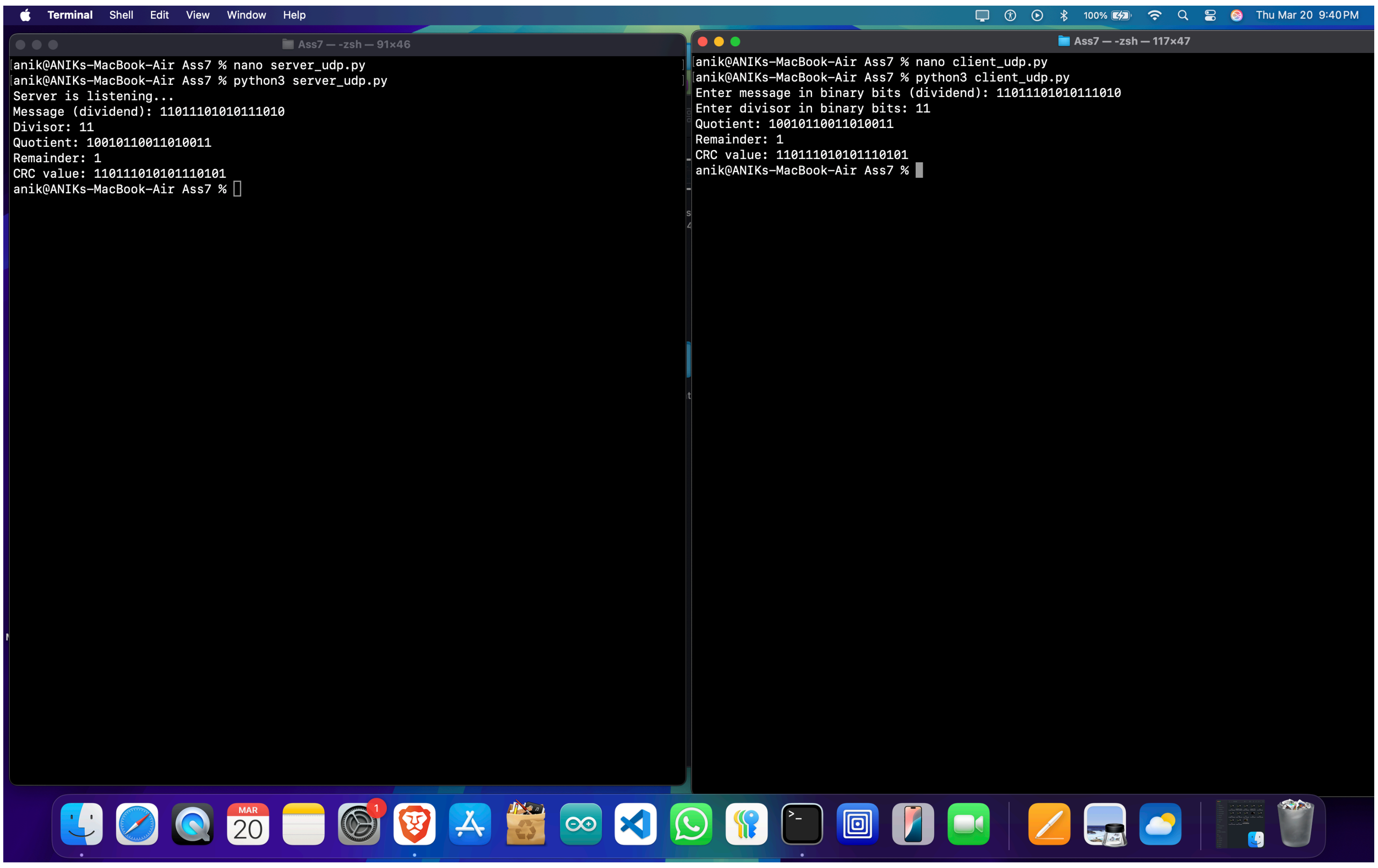
```
import socket

def crc(dividend, divisor):
    m = len(divisor)
    appended = dividend + "0"*(m-1)
    dividend_list = list(appended)
    quotient = ""
    for i in range(len(dividend)):
        if dividend_list[i] == '1':
            quotient += '1'
            for j in range(m):
                dividend_list[i+j] = str(int(dividend_list[i+j] != divisor[j]))
        else:
            quotient += '0'
    remainder = "".join(dividend_list[-(m-1):])
    crc_value = dividend + remainder
    return quotient, remainder, crc_value

host = 'localhost'
port = 12345

# Create UDP socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind((host, port))

print("Server is listening...")

# Receive data
data, client_addr = s.recvfrom(1024)
received_data = data.decode()
message, divisor = received_data.split(',')

print("Message (dividend):", message)
print("Divisor:", divisor)

# Perform CRC calculation
q, r, crc_value = crc(message, divisor)
print("Quotient:", q)
print("Remainder:", r)
print("CRC value:", crc_value)

# Send result back
s.sendto(f"{q},{r},{crc_value}".encode(), client_addr)

s.close()
```

Udp_Client.py ==>

```
import socket

host = 'localhost'
port = 12345
```

```
# Create UDP socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Get user inputs
message = input("Enter message in binary bits (dividend): ")
divisor = input("Enter divisor in binary bits: ")

# Send both message and divisor separated by comma
s.sendto(f"{message},{divisor}".encode(), (host, port))

# Receive response
data, addr = s.recvfrom(1024)
q, r, crc_value = data.decode().split(',')
print("Quotient:", q)
print("Remainder:", r)
print("CRC value:", crc_value)

s.close()
```

__Udp_Output ==>__



```
anik@ANIKs-MacBook-Air Ass7 % nano server_udp.py
anik@ANIKs-MacBook-Air Ass7 % python3 server_udp.py
Server is listening...
Message (dividend): 11011101010111010
Divisor: 11
Quotient: 10010110011010011
Remainder: 1
CRC value: 11011101010111010 1
anik@ANIKs-MacBook-Air Ass7 %
```

```
anik@ANIKs-MacBook-Air Ass7 % nano client_udp.py
anik@ANIKs-MacBook-Air Ass7 % python3 client_udp.py
Enter message in binary bits (dividend): 11011101010111010
Enter divisor in binary bits: 11
Quotient: 10010110011010011
Remainder: 1
CRC value: 110111010101110101
anik@ANIKs-MacBook-Air Ass7 %
```