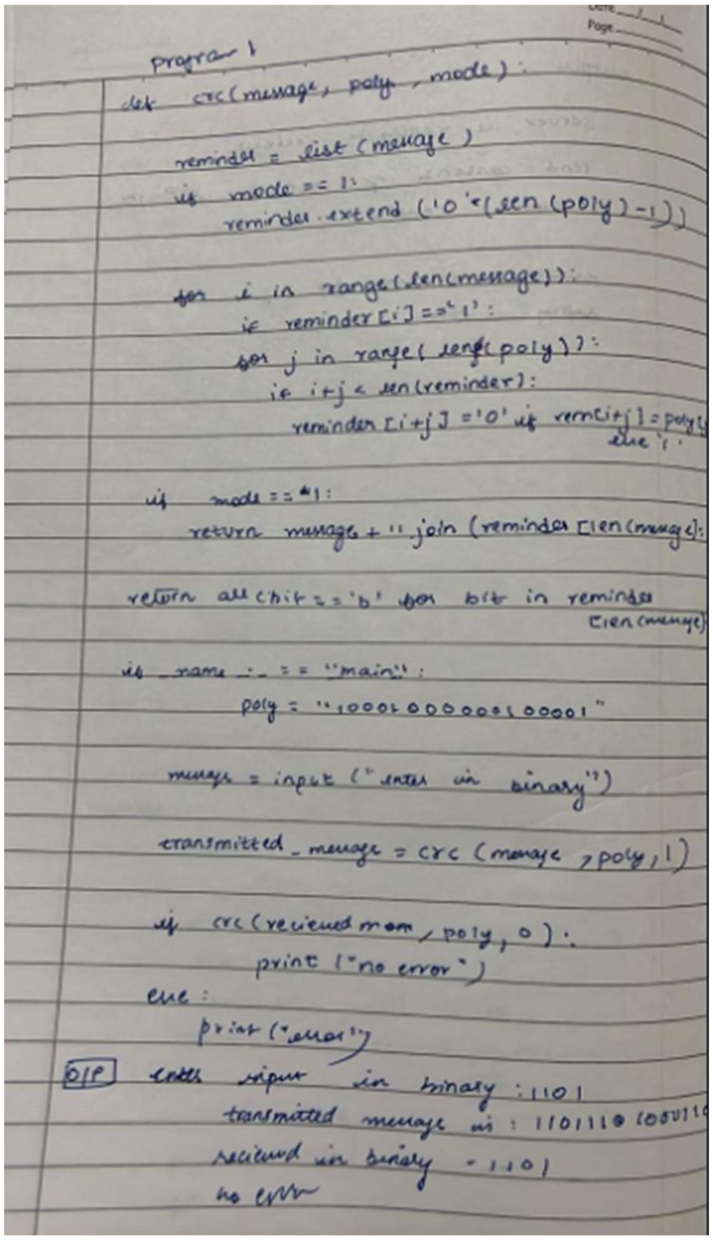
CYCLE 2

EXPERIMENTS



Date ___/___/___
Page ______

program 1

```
def  crc(message, poly , mode):

    reminder = list (message )
    if  mode == 1:
        reminder.extend ('0'*(len (poly)-1))

    for  i in  range(len(message)):
        if  reminder[i]=='1':
        for j in range( len(poly)):
            if i+j < len(reminder):
                reminder [i+j] ='0' if remt[i+j] = poly[j
                                                   else '1'

  if    mode ==*1:
     return  message + '' join (reminder [len(message):

return all (bit == '0' for  bit  in  reminder
                                             [len(message)

if _name_ :_ == ''main'' :
        poly = ''10001000000100001''

    message = input (" enter  in  binary")

    transmitted _message = crc (message ,poly,1)

    if  crc(recieved msm , poly , 0):
            print ("no error")
  else :
        print ("error")
```

$\boxed{O/P}$ enter input in binary :1101

transmitted message is : 11011100 100011

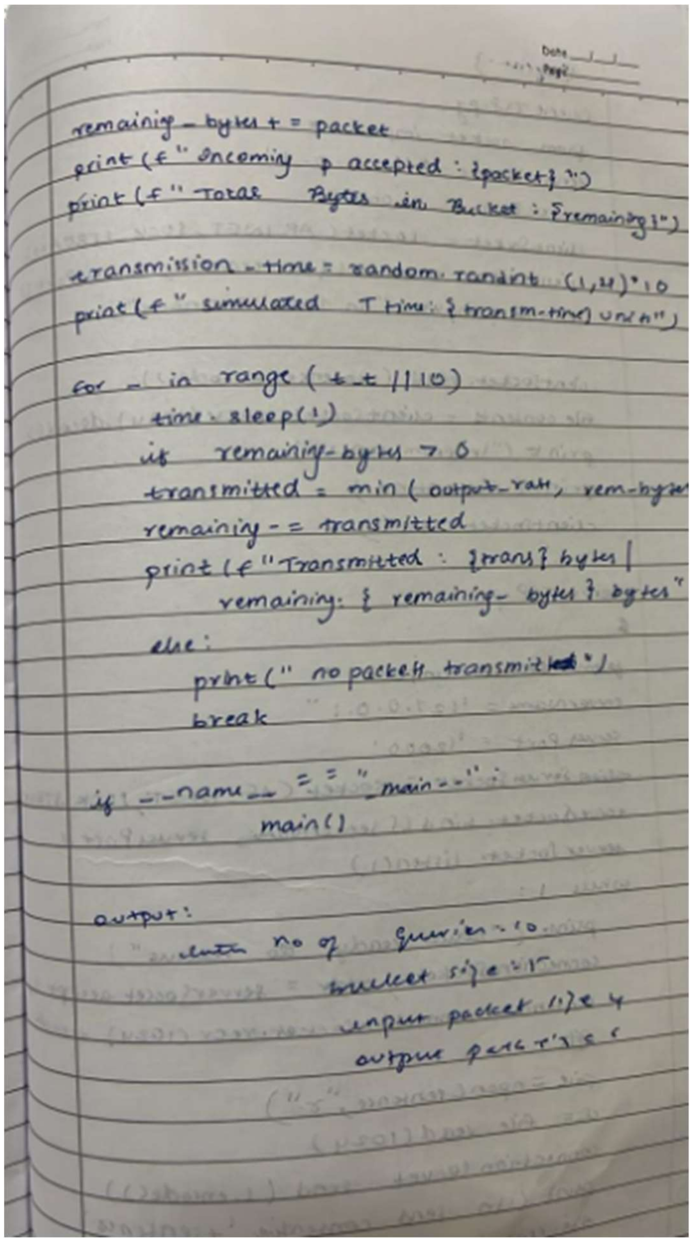
recieved in binary - 1101

no error

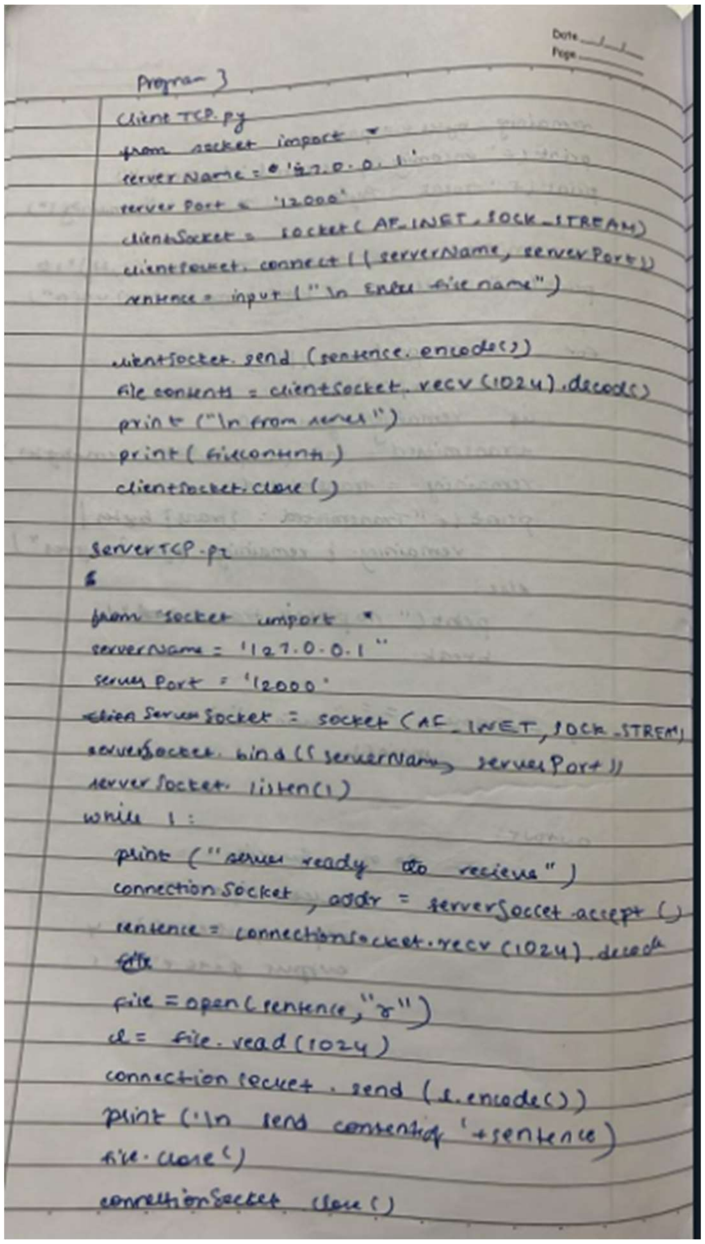
Date / /
Page

Program 2

```
def import time
 import random

 NOF_packets = 10

def  generate_random_packet size( max-size ):
     return random.randint (1, max-size //10 )*10

def main( ):
     packet-sizes =[gen-random -packet-size (80) for
               _ in range (NOF_PACKETS)]
     print ("generated packet sizes:")
     for i, size in enumerate (packet-sizes):
          print (f "Packet[{i}]: {size} bytes")

 output_rate = int (input ("\n Enter output rate:"))
 bucket - size = int (input ( "enter size"))
 remaining bytes = 0

 for i, packet in enumerate (packet -sizes):
      print (f "\n Processing packet [{i}] of size
         {packet} bytes")
      if packet > bucket_size:
      print (f" packet size ( {packet} bytes)
    exceeds capacity {b-size} - Packet rejected")
      continue

      if (remaining_bytes + packet) > b-size:
        print ("bucket_size exceeded - PACKET-REJ")
        continue
```



```
remaining_bytes += packet
print(f" Incoming p accepted : {packet} ")
print(f" Total   Bytes in Bucket : {remaining}")

transmission_time = random.randint (1,4)*10
print(f" simulated  T time: {transm-time} units")

for _ in range ( t_t //10)
    time.sleep(1)
    if  remaining-bytes > 0
    transmitted = min ( output_rate, rem-bytes
    remaining -= transmitted
    print(f"Transmitted : {trans} bytes |
          remaining: { remaining_ bytes } bytes"
    else:
        print(" no packets transmitted")
        break

if __name__ == "__main__":
             main()


output:
      enter no of  queries : 10
                bucket size : 5
                input packet size 4
                output rate : 5
```



Date __/__/__
Page ______

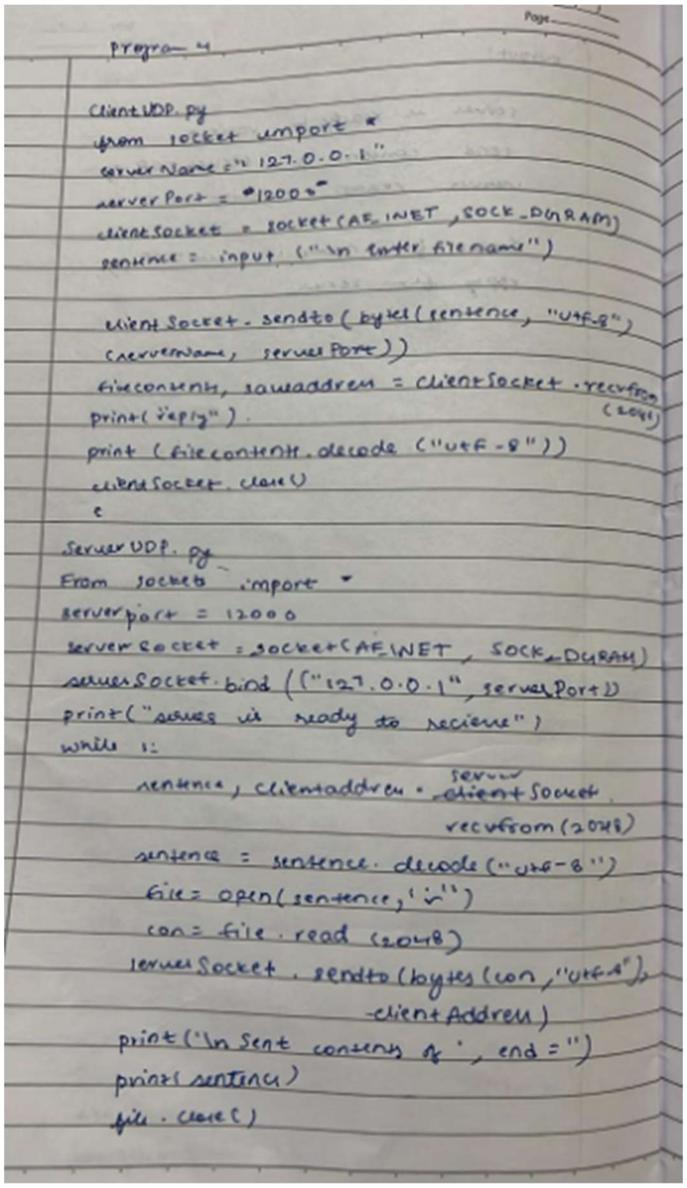
Program 3

clientTCP.py

```
from socket import *
serverName = '127.0.0.1'
serverPort = '12000'
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input("\n Enter file name")

clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print("\n from server")
print(filecontents)
clientSocket.close()
```

serverTCP.py

```
from socket import *
serverName = '127.0.0.1''
serverPort = '12000'
~~clien~~ serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while 1:
   print("server ready to recieve")
   connectionSocket, addr = serverSocket.accept()
   sentence = connectionSocket.recv(1024).decode
   file = open(sentence, "r")
   l = file.read(1024)
   connectionSocket.send(l.encode())
   print('\n send contentof '+sentence)
   file.close()
   connectionSocket.close()
```



Page

program 4

```
Client UDP. py
from socket import *
serverName = "127.0.0.1"
serverPort = *12000"
clientSocket = socket(AF_INET ,SOCK_DGRAM)
sentence = input ("\n Enter filename")

 clientSocket . sendto( bytes( sentence, "utf-8")
 (serverName, serverPort))
 filecontents, serveraddress = clientSocket . recvfrom
print( "reply" ).                                        (2048)
print (filecontents . decode ("utf-8"))
clientSocket . close()
 c

Server UDP. py
From socket import *
serverport = 12000
server Socket = socket(AF_INET , SOCK_DGRAM)
serverSocket. bind (("127.0.0.1", serverPort))
print("server is ready to recieve")
while 1:
                                         server
       sentence, clientaddress = ~~client~~ Socket.
                                         recvfrom (2048)
       sentence = sentence. decode ("utf-8")
        file= open( sentence, 'r')
        con= file . read (2048)
      serverSocket . sendto (bytes (con, "utf-8"),
                                  clientAddress)
    print ('\n Sent contents of ', end ='')
    print( sentence)
    file . close()
```

Exp-12

Date __/__/__
Page ____

Wireshark

key features:

1. packet capture
2. protocol analysis
3. following offers
4. visualization

[illegible]

use cases

1. Network trouble shooting
   diagnosing slow network speed
   or other misconfiguration
2. security analysis:
   detecting malicious traffic or
   intentions
3. Protocol study
   understanding packet structures &
   commercial flow

O/P:

1. http : show only http traffic
2. tcp : 80: show traffic on TCP port 8
3. ip : 192.168.11.1 : show info to & fr
3. udp : show only UDP traffic