CN LAB

WEEK 1: 1. ipconfig- It displays the information abt the host computer's TCP/IP confugaration.

2. ipconfig/all- It gives detailed confugaration information abt your TCP/IP connections including router, gateway, DNS, DHCP, and type of Ethernet.

3. nslookup- it is used for diagnosing DNS problems. If you can access a resource by specifying an IP address but not its DNS if you have a DNS problem.

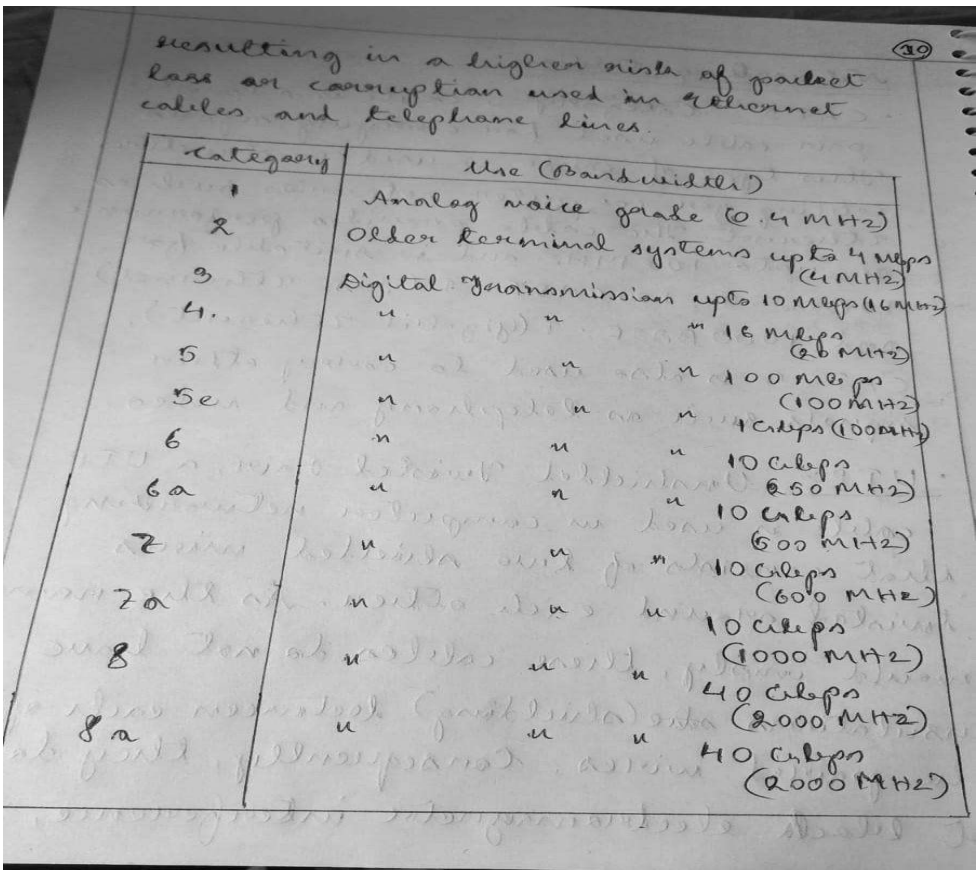
4. ping<ip>- this command sends one datagram per second and prints one line of output for every response received. The ping command calculates round trip time and packet less statistics and displays a brief summary on completion.

5. tracert<ip>- it displays a list of all the routers that a packet has to go through to get from the computer where tracert is run to any other computer on the internet.

6. netstat- it displays a variety of statistics abt a computer TCP/IP connections. This tool is useful when you are having trouble with TCP/IP applications such as HTTP and FTP.

7. CAT5- Category 5 cable is a twisted pair cable used for carrying signals. This type of cable is used in structured cabling in computer networks such as Ethernet. The cable provides performance of upto 100MHz and is suitable for 10 BaseT, 100BaseT(fast Ethernet) and 1000 BaseT(gigabit Ethernet). CAT-5 is also used to carry other signals such as telephony and video.

8. UTP- Unshielded Twisted Pair, a UTP cable is used in computer networking that consists of two shielded wires twisted around each other. As the name would imply, these cables do not have insulations(shielding) between each of the paired wires. Consequently, they do not block electromagnetic inference, resulting in a higher risk of packet loss or corruption used in Ethernet cables and telephone lines.


resulting in a higher risk of packet
loss or corruption used in Ethernet
cables and telephone lines.

Category	Use (Bandwidth)
1	Analog voice grade (0.4 MHz)
2	Older terminal systems upto 4 Mbps (4 MHz)
3	Digital transmission upto 10 Mbps (16 MHz)
4	" " " 16 Mbps (20 MHz)
5	" " " 100 Mbps (100 MHz)
5e	" " " 1 Gbps (100 MHz)
6	" " " 10 Gbps (250 MHz)
6a	" " " 10 Gbps (500 MHz)
7	" " " 10 Gbps (600 MHz)
7a	" " " 10 Gbps (1000 MHz)
8	" " " 40 Gbps (2000 MHz)
8a	" " " 40 Gbps (2000 MHz)

9. RJ45- A Registered Jack(RJ) is a standard physical network interface for connecting telecommunication or data equipment. The physical connector that registered jacks uses are mainly of the modular connector and 50 pin

miniature ribbon connector types. The most common twisted pair connector is an 8 position, 8 contact (8P8C) modular plug and jack commonly refered to as an RJ45 connector.

10. T Connector- these are used to split radio frequency power from a cable into two. They can be used to attach a piece of electronic test equipment. T connector were used on IOBASE2 ethernet network.

11. Hubs- a network hub is a node that broadcasts data to every computer or Ethernet based device connected to it. A hub is a less sophisticated than a switch, the later of which can isolate data transmissions to a specific device. Network hubs are best suited for small, simple local area network (lan). They cannot filter the signals.

12. Switches- these are network devices operating at the layer-2 or the data link layer of the OSI model. They connect devices in a network and use packet switching to send receive or forward data packets or data frames over the network. A switch has many ports to which computers are plugged in. It supports unicast, multicast as well as broadcast communications.

WEEK-2:

1. Configure the hostname of the Switch as SW1 .

Switch > en

Switch #config t

Switch(config)#hostname SW1

SW1(config)#

Exit 2 times.

2. Set a message of the day (MOTD) banner for the switch-

********** Only Authorized Users Allowed **********

```
SW1#config t
SW1(config)#banner ?
SW1(config)#banner motd ?
SW1(config)#banner motd $**********Only Authorised Users
Allowed**********$
SW1(config)#exit
exit
```

3. Configure a: i) line console password – India@123 ii)enable secret password – Uem@123

```
SW1>en
SW1#sh run
SW1#config t
SW1(config)#line con 0
SW1(config-line)#password India@123
SW1(config-line)#exit
SW1(config)#enable secret Uem@123
SW1(config)#line con 0
SW1(config-line)#login
SW1(config-line)#
SW1(config-line)#exit
SW1(config)#exit
SW1#
Exit
```

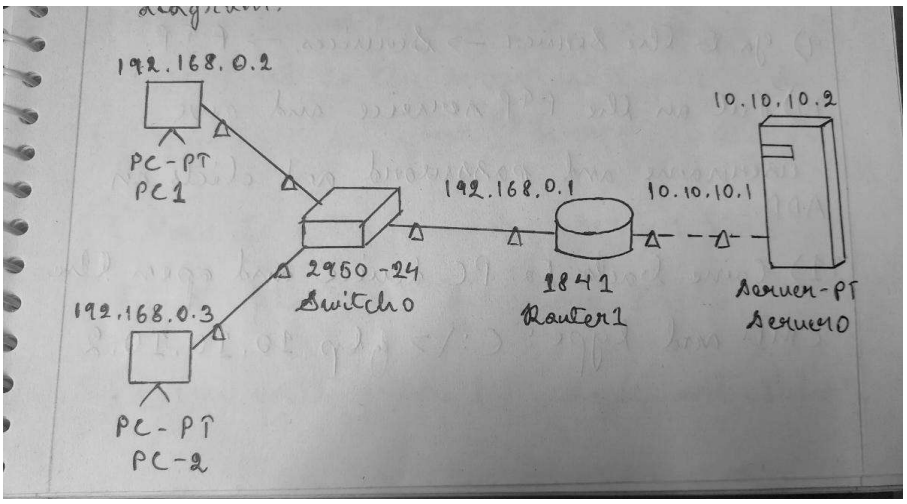
4. FTP, DHCP Server Confirguration

FTP Server Configuration:

1) Open Cisco Packet Tracer and select 2 End Devices (PC device), 1 Switch, 1 Router, 1 Server.

2) Now Connect all the devices using the auto connection.

3) Then configure the IP addresses as per the diagram.



4) Now just wait for some time to let all the connection status turns green.

5) Now we have achieved a connection where a class C IP address is being translated to class A IP Address.

6) Go to one of the PC devices and on Desktop tab select CMD.

7) Now we need to check the connection to the server by typing(C:\> ping 10.10.10.2)

8) If reply is coming then it means the server is properly configured and connected.

9) Go to the Server → Services →FTP.

10) Put on the FTP service and give username and password and click on ADD.

11) Come back to PC device and open the CMD and type (C:\> ftp 10.10.10.2)

12) It will ask for username and password. Provide the username and password configured earlier.

13) Once the connection is established exit rom the CMD and go to Text Editor and make a new text file.

14) Save the new text file and return to cmd and type (ftp>put filename.txt)

15) This will send the text file from the PC device (192.168.0.2) to Server (10.10.10.2).

16) Now to verify that the file has been transferred to the server, so type

17) You will see your Filename in the list.

18) Now to get a file from server to PC type (ftp> get filename.txt)

19) Now exit from FTP type ctrl+C, then type dir to check that the file is there in the PC or not.

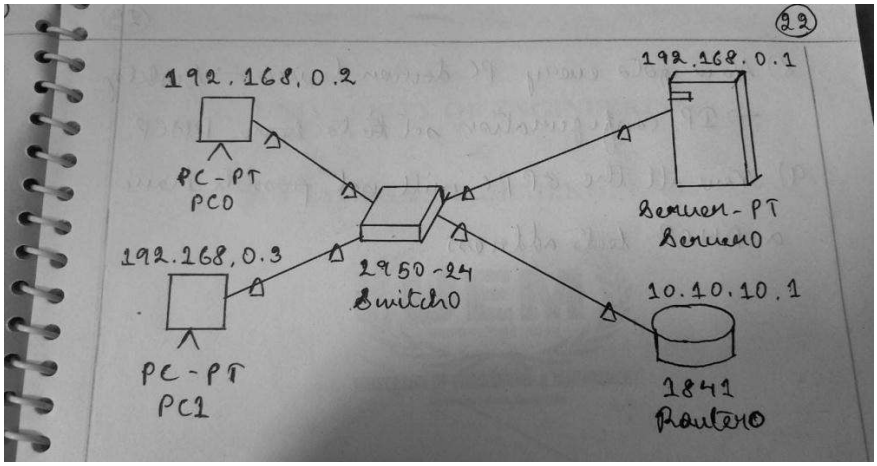
20) So we have successfully send and got a file from a server using FTP protocol.

<u>DHCP Server Configuration:</u>

1) Open Cisco Packet Tracer and select 2 End Devices (PC device), 1 Switch, 1 Router, 1 Server.

2) Now Connect all the devices using the auto connection.

3) Then configure the IP addresses as per the diagram.



4) Now just wait for some time to let all the connection status turns green.

5) Now go to the server → Desktop → IP Configuration and set the IP 192.168.0.1.

6) Go to Services → DHCP and set Default Gateway 192.168.0.1 and DNS Server 10.0.0.1.

7) Set the Start IP 192.168.0.0 and Max Users 256

8) Now go to every PC device → Desktop → IP Configuration and set it to DHCP.

9) Now all the PC will have a DHCP address.

WEEK 3:

1. Write a program to find the IP address of the system

```
import socket
print("Host Name:", socket.gethostname(),
    "\nIp Address:", socket.gethostbyname(socket.gethostname()))
```

2. Write a socket program for implementation of echo.

Server side code:

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("127.0.0.1", 9090))
s.listen()
(c, cip) = s.accept()
c.send(c.recv(1024))
s.close()
```

Client side code:

```
import socket
c = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
c.connect(("127.0.0.1", 9090))
data = input()
c.send(data.encode())
dataFromServer = c.recv(1024)
print(dataFromServer.decode())
c.close()
```

3. Write a client-server application for chat using TCP

Server side code:

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("127.0.0.1", 9090))
s.listen()
while True:
   (c, cip) = s.accept()
   data = c.recv(1024).decode()
   print("Client:",data)
   data = input("Enter Text: ")
   c.send(data.encode())
```

Client side code:

```
import socket
c = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
c.connect(("127.0.0.1", 9090))
while True:
   data = input("Enter Text: ")
   c.send(data.encode())
   data = c.recv(1024).decode()
   print("Server:",data)
```

4. Write a program using client server socket programming: Client needs to authenticate itself by entering a server defined string as a password (like OTP) and then to say Hi to server. Server replies with a Hello. Press any key to exit.

Server side code:

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("127.0.0.1", 9090))
s.listen()
(c, cip) = s.accept()
c.send("Enter OTP:".encode())
otp = c.recv(1024).decode()
if otp == '8894':
   c.send("You are Authenticated".encode())
   data = c.recv(1024).decode()
   print("Client:",data)
   data = input("Enter Text: ")
   c.send(data.encode())
else:
   c.send("You are Not Authenticated".encode())
s.close()
```

Client side code:

```
import socket
c = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
c.connect(("127.0.0.1", 9090))
data = c.recv(1024).decode()
print(data, end=" ")
otp = input()
c.send(otp.encode())
data = c.recv(1024).decode()
print(data)
if data == "You are Authenticated":
    data = input("Enter Text: ")
    c.send(data.encode())
    data = c.recv(1024).decode()
    print("Server:",data)
else:
    c.close()
```

WEEK 4:

1. Write a program to Perform File Transfer in Client & Server Using TCP/IP.

Server side code:

```
import socket
import tqdm
import os
BUFFER_SIZE = 1024
s = socket.socket()
s.bind(('127.0.0.1', 5001))
s.listen(1)
print("Listening as 127.0.0.1:5001")
client_socket, address = s.accept()
print(f"Connected to {address[0]}:{str(address[1])}")
filename, filesize = client_socket.recv(BUFFER_SIZE).decode().split('||')
filename = os.path.basename(filename)
filesize = int(filesize)
progress = tqdm.tqdm(range(filesize), f"Receiving{filename}", unit="B", unit_scale=True,unit_divisor=1024)
with open(filename, "wb") as f:
    while True:
        bytes_read = client_socket.recv(BUFFER_SIZE)
        if not bytes_read:
            break
```

```
        f.write(bytes_read)
        progress.update(len(bytes_read))
client_socket.close()
s.close()
```

Client side code:

```
import socket
import tqdm
import os
BUFFER_SIZE = 1024
filename = input("Enter the file path or filename: ")
filesize = os.path.getsize(filename)
s = socket.socket()
print("Connecting to 127.0.0.1:5001")
s.connect(("127.0.0.1", 5001))
print("Connected.")
s.send(f"{filename}||{filesize}".encode())
progress = tqdm.tqdm(range(filesize), f"Sending{filename}", unit="B", unit_scale=True,unit_divisor=1024)
with open(filename, "rb") as f:
    while True:
        bytes_read = f.read(BUFFER_SIZE)
        if not bytes_read:
            break
        s.sendall(bytes_read)
        progress.update(len(bytes_read))
s.close()
```

WEEK 5:

Write a program to implement Remote Command Execution (RCE)

Server side code:

```
import socket
import os
BUFFER_SIZE = 4096
s = socket.socket()
s.bind(('127.0.0.1', 5001))
s.listen(1)
print("Listening as 127.0.0.1:5001")
client_socket, address = s.accept()
print(f"{address[0]}:{str(address[1])} is Connected to terminal")
while True:
    print("\nClient@Server>>", end=" ")
```

```
    command = client_socket.recv(BUFFER_SIZE).decode()
    print(command)
    if command == 'exit':
        break
    os.system(command)
print("Closing remote connection with client")
client_socket.close()
s.close()
```

Client side code:

```
import socket
BUFFER_SIZE = 4096
s = socket.socket()
print("Connecting to 127.0.0.1:5001")
s.connect(("127.0.0.1", 5001))
print("Connected to Server Terminal")
while True:
    command = input("\nServer>> ")
    s.sendall(command.encode())
    if command == 'exit':
        break
print("Closing remote connection with Server")
s.close()
```

WEEK 6:

Write a program to implement simple client-server application using UDP.

Server side code:

```
import socket
localIP= "127.0.0.1"
localPort= 20001
bufferSize= 1024
msg_From_Server= "Hello"
bytes_To_Send = str.encode(msg_From_Server)
Server_Socket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
Server_Socket.bind((localIP, localPort))
print("Server listening...")
while(True):
    bytes_Address_Pair =Server_Socket.recvfrom(bufferSize)
    message = bytes_Address_Pair[0]
    address = bytes_Address_Pair[1]
```

```
    client_Msg = "Message from Client:{}".format(message)
    client_IP  = "Client IP Address:{}".format(address)
    print(client_Msg)
    print(client_IP)
    Server_Socket.sendto(bytes_To_Send, address)
```

Client side code:

```
import socket
msg_From_Client= "Hi"
bytes_To_Send= str.encode(msg_From_Client)
server_Address_Port= ("127.0.0.1", 20001)
bufferSize= 1024
Client_Socket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
Client_Socket.sendto(bytes_To_Send, server_Address_Port)
msg_From_Server = Client_Socket.recvfrom(bufferSize)
msg = "Messagefrom Server {}".format(msg_From_Server[0])
print(msg)
```

WEEK 7:

1. Simulation of ARP

BY PYTHON:

Server side code:

```
import socket
table = {
    '192.168.0.1':'1E.4A.4A.11',
    '192.168.0.2':'5E.6A.4A.11',
    '192.168.0.3':'1E.6A.4A.11',
    '192.168.0.4':'2E.4A.3A.11',
    '192.168.0.5':'3E.4A.2A.11',
    '1E.4A.4A.11':'192.168.0.1',
    '5E.4A.4A.11':'192.168.0.2',
    '1E.6A.4A.11':'192.168.0.3',
    '2E.4A.3A.11':'192.168.0.4',
    '3E.4A.2A.11':'192.168.0.5',
}

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.bind(('',1234))
s.listen()
cls,addr = s.accept()
```

```
print(f"connection from {addr} has Esatablished")
ip = cls.recv(1024)
ip = ip.decode("utf-8")
mac = table.get(ip,'No entry for given address')
cls.send(mac.encode())
```

Client side code:

```
import socket
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(('localhost',1234))
a = input("ARP or RARP")
if (a=="ARP"):
   add = input("Enter IP: ")
elif (a=='RARP'):
   add = input("enter MAC: ")

s.send(add.encode())
mac = s.recv(1024)
mac = mac.decode('utf-8')
if (a=='ARP'):
   print(f"MAC of {add} is : {mac}")
elif (a=='RARP'):
   print(f"IP of {add} is : {mac}")
```

BY C:

Server side code:

```
#include<stdio.h
>
                    #include<sys/types.h>
                    #include<sys/shm.h>
                    #include<string.h>
                    void main() {
                        int shmid, a, i;
                        char* ptr, *shmptr;
                        shmid = shmget(3000, 10, IPC_CREAT|0666);
                        shmptr = shmat(shmid, NULL, 0);
                        ptr = shmptr;
                        for (i=0;i<3;i++) {
                            puts("Enter the name: ");
                            scanf("%s", ptr);
                            a = strlen(ptr);
                            printf("String length: %d", a);
                            ptr[a] = '';
                            puts"Enter IP Address: ";
                            ptr = ptr + a + 1;
                        }

                        ptr[strlen(ptr)] = '\0';
                        printf("\n ARP Table at serverside: \n%s", shmptr);
                        shmdt(shmptr);
                    }
```

Client side code:

```
#include<stdio.h
>
                    #include<string.h>
                    #include<sys/types.h>
                    #include<sys/shm.h>

                    void main() {
                        int shmid, a;
                        char* ptr, *shmptr;
                        char ptr2[51], ip[12], mac[26];
                        shmid = shmget(3000, 10, 0666);
                        shmptr = shmat(shmid, NULL, 0);
                        puts("The ARP Table is: ");
                        printf("%s", shmptr);
                        printf("\n1. ARP \n2. RARP \n3. EXIT\n");
                        scanf("%d", %a);
                        switch(a) {
                            case 1: puts("Enter IP Address: ");
                                    scanf("%s", &ip);
                                    ptr = strstr(shmptr, ip);
                                    ptr-=8;
                                    sscanf(ptr, "%s%s", ptr2);
                                    printf("MAC Address is: %s", ptr2);
                                    break;

                            case 2: puts("Enter MAC Address: ");
                                    scanf("%s", &mac);
                                    ptr = strstr(shmptr, mac);
                                    sscanf(ptr, "%s%s", ptr2);
                                    printf("%s",ptr2);
                                    break;

                            case 3: exit(1);
                                    break;
                        }
                    }
```

2. TCP Module Implementation

Server side code:

```
import socket

import socket

ip = '127.0.0.1'

port = 1234

server = socket.socket(socket.AF_INET,socket.SOCK_STREAM)

server.bind((ip,port))

server.listen(5)


while True:
   cli,add = server.accept()
   print(f"Connection Established with {add}")
   st = cli.recv(1024).decode('utf-8')
   print(st)
   cli.close()
```

Client side code:

```
import socket
ip = '127.0.0.1'
port = 1234
cli = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
cli.connect((ip,port))
str = input("Enter: ")
st = bytes(str,'utf-8')
cli.send(st)
```

WEEK 8:

Implementation of RMI

Server side code:

```
import
java.rmi.*;
              import java.rmi.server.*;
              public class remoServer extends UnicastRemoteObject
              implements remointer
              {
                  public remoServer() throws Remote Exception {
                      return "Hi";
                  }
                  public static void main (String args[]) {
                      try {
                          remoServer r = new remoServer();
                          Naming.rebind("Test Server", r);
                          System.out.println("The server is
              ready");
                      }
                      catch (Exception e) {

                      }
                  }
              }
```

Client side code:

```
import
java.rmi.*;
              import java.rmi.registry.*;
              public class remoClient
              {
                  public static void main(String args[])
              {
                      try {
                          remointer s = (remointer);
                          Naming.lookup("Test Server");

              System.out.println(s.message[]);
                      }
                      catch (Exception e) {

                      }
                  }
              }
```

WEEK 10:

1. Character Count:

Server side code:

```
n=int(input("Enter the no. of frames: "))
l=[]
l2=[]
for i in range(0,n):
    x=input()
    l.append(x)
    l2.append(len(x)+1)
sender=''
for i in range(0,n):
    sender+= str(l2[i])+l[i]
print(sender)
```

Client side code:

```
data=input("Enter The message: ")
i=0
inv=0
while(i<len(data)):
    if(data[i].isnumeric()==True):
        print(data[i+1:int(data[i])+i])
        i+=int(data[i])
```

2. Bit Stuffing

Server side code:

```
def bitStuffing(N, arr):
    # Stores the stuffed array
    brr = [0 for _ in range(30)]
    # Variables to traverse arrays
    k = 0
    i = 0
    j = 0
    # Stores the count of consecutive ones
    count = 1

    # Loop to traverse in the range [0, N)
    while (i < N):
        # If the current bit is a set bit
        if (arr[i] == 1):
            # Insert into array brr[]
            brr[j] = arr[i]
```

```
            # Loop to check for
            # next 5 bits
            k = i + 1
            while True:
                if not (k < N and arr[k] == 1 and
                        count < 5):
                    break
                j += 1
                brr[j] = arr[k]
                count += 1
                # If 5 consecutive set bits
                # are found insert a 0 bit
                if (count == 5):
                    j += 1
                    brr[j] = 0
                i = k
                k += 1
        # Otherwise insert arr[i] into
        # the array brr[]
        else:
            brr[j] = arr[i]
        i += 1
        j += 1
    # Print Answer
    for i in range(0, j):
        print(brr[i], end = "")
# Driver Code
if _name_ == "_main_":
    N = 6
    arr = [ 1, 1, 1, 1, 1, 1 ]
    bitStuffing(N, arr)
```

Client side code:

```
import socket
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 mod2div(dividend, divisor):
    pich = len(divisor)
    temp = dividend[0:pich]
    while pich < len(dividend):
        if temp[0] == '1':
            temp = xor(divisor, temp) + dividend[pich]
        else:
            temp = xor('0'*pich, temp) + dividend[pich]
        pich+=1
        if temp[0] == '1':
            temp = xor(divisor, temp)
        else:
            temp = xor('0'*pich, temp)
        checkword = temp
    return checkword
def decodeData (data, key):
    l_key = len(key)
    appended_data = data.decode() + '0'*(l_key-1)
    remainder = mod2div(appended_data, key)
    return remainder
c = socket.socket()
print("Socket successfully created")
port = 12345
c.bind(('127.0.0.1', port))
print("Socket binded to %s" %(port))
c.listen(0)
print("Socket is listening")
while True:
    c.addr = c.accept()
    print("Connected to ", c.addr)
    data = c.recv(1024)
    print("received: ", data.decode())
    if not data:
        break
key = '1001'
ans = decodeData(data, key)
print("Remainder aftre decoding is: " + ans)
temp = '0'* (len(key) - 1)
if ans==temp:
    c.sendto(("Thank You" + data.decode() + "Received").encode(), (('127.0.0.1', 12345)))
else:
    c.sendto(("error").encode(), ('127.0.0.1',12345))
```

```
    c.close()
```

WEEK 11:

Sender side code:

```
data=input('enter input: ')
divisor = "1101"
p = len(divisor)
divident = data + '0'*(p-1)
tmp = divident[0 : p]
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)
while p < len(divident):
    if tmp[0] == '1':
        tmp = xor(divisor , tmp) + divident[p]
    else:
        tmp = xor('0'*p , tmp) + divident[p]
    p += 1
if tmp[0] == '1':
    tmp = xor(divisor , tmp)
else:
    tmp = xor('0'*p , tmp)
checkword = tmp
print('enter the checkword: ',checkword)
codeword = data + checkword
print('enter the codeddata: ',codeword)
```

Receiver side code:

```
data=input('enter input')
divisor = "1101"
p = len(divisor)
divident = data
tmp = divident[0 : p]
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)
while p < len(divident):
    if tmp[0] == '1':
        tmp = xor(divisor , tmp) + divident[p]
    else:
        tmp = xor('0'*p , tmp) + divident[p]
    p += 1
if tmp[0] == '1':
    tmp = xor(divisor , tmp)
else:
    tmp = xor('0'*p , tmp)
checkword = tmp
print('the checkword:',checkword )
if(int(checkword,2)==0):
    print('The data is error free.')
else:
    print('The data is error added.')
```

GITHUB LINK BY DIBYASANKHA:
https://github.com/palsuryaofficio/palsuryaofficio-Computer-Networking-Practical-Material-Ref SHORT CUT LINK: http://bit.ly/3hmGjDY

GITHUB LINK BY ARCHISMAN:
https://github.com/ArchishmanSengupta/Competitive-Coding/tree/main/0-CollegePractical/5th_Sem_Lab_Code