VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

Harbakhshish Singh Arora (1BM23CS104)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

September 2025 – January 2026

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "Computer Network (23CS5PCCON)" carried out by Harbakhshish Singh Arora (1BM23CS104) who is bonafide student of B.M.S. College of Engineering. It is in partial fulfilment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements of the above-mentioned subject and the work prescribed for the said degree.

Rashmi H Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

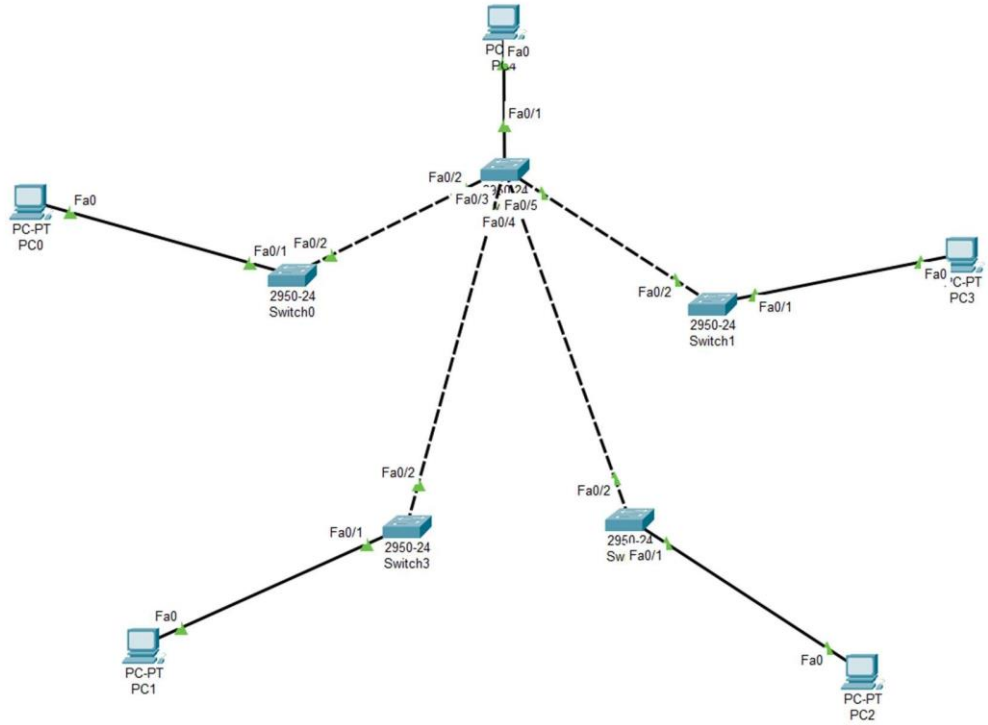
PART-A

Program – 1:

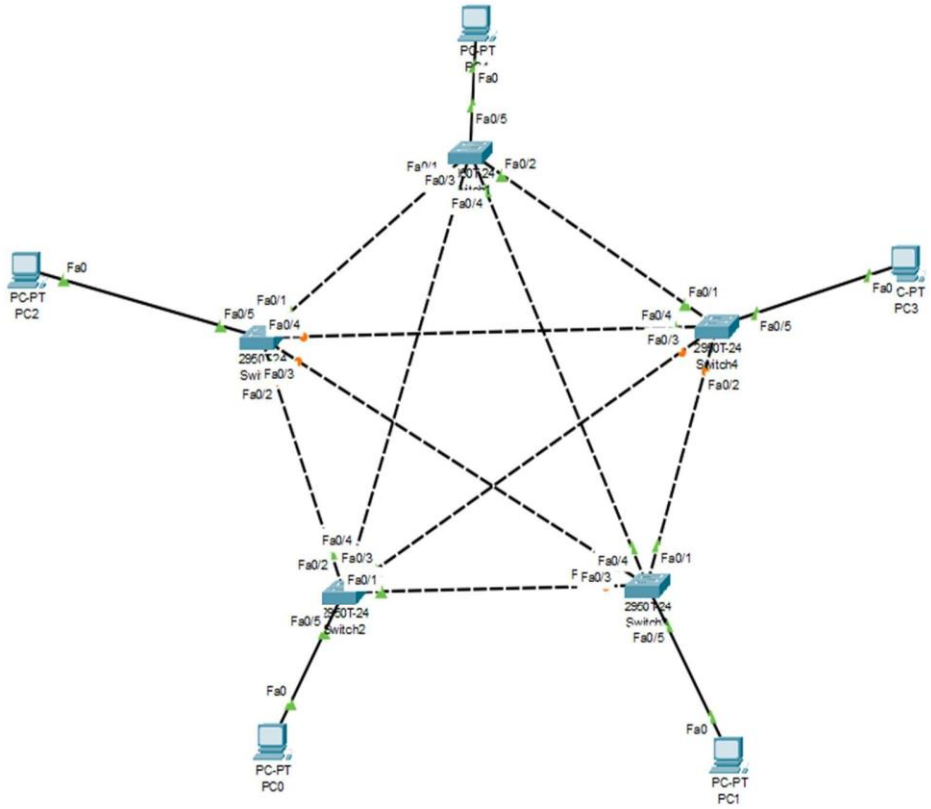
Aim: Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.

Network Diagram:

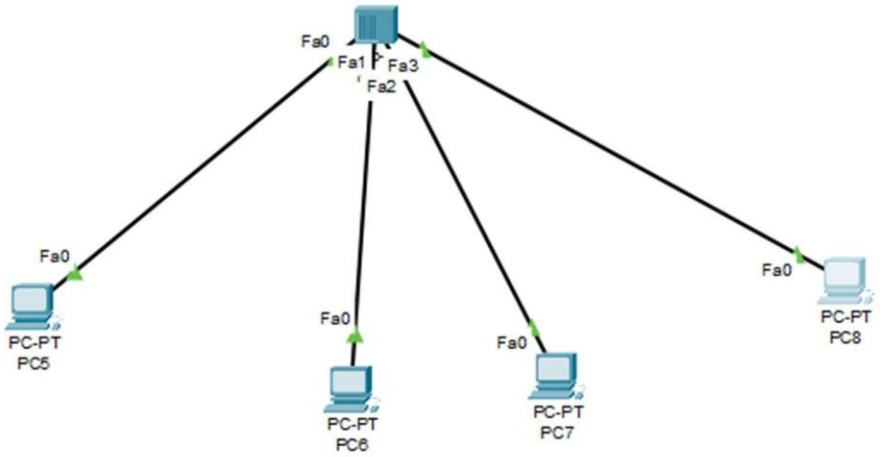
Star Topology:



Mesh Topology:

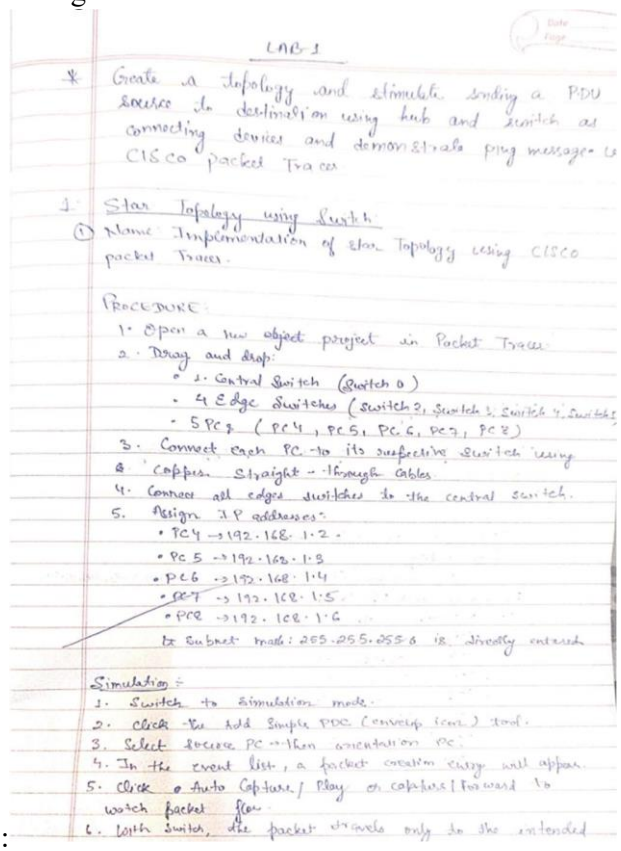


Hub-Based Network Topology:



Configuration

:



LAB 1

* Create a topology and simulate sending a PDU source to destination using hub and switch as connecting devices and demonstrate ping message in CISCO packet Tracer.

1. Star Topology using Switch

① Name: Implementation of star Topology using CISCO packet Tracer.

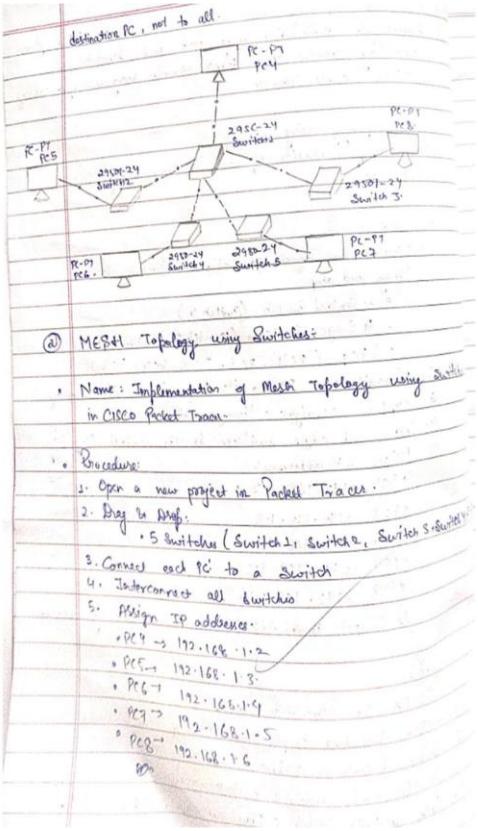
PROCEDURE:

1. Open a new object project in Packet Tracer.
2. Drag and drop:
   - 1 Central Switch (Switch 0)
   - 4 Edge Switches (Switch 1, Switch 2, Switch 4, Switch 5)
   - 5 PCs (PC4, PC5, PC6, PC7, PC8)
3. Connect each PC to its respective switch using
4. Copper Straight-Through Cables
5. Connect all edges switches to the central switch.
6. Assign IP addresses:
   - PC4 → 192.168.1.2
   - PC5 → 192.168.1.3
   - PC6 → 192.168.1.4
   - PC7 → 192.168.1.5
   - PC8 → 192.168.1.6

   & Subnet mask: 255.255.255.0 is already entered.

Simulation:

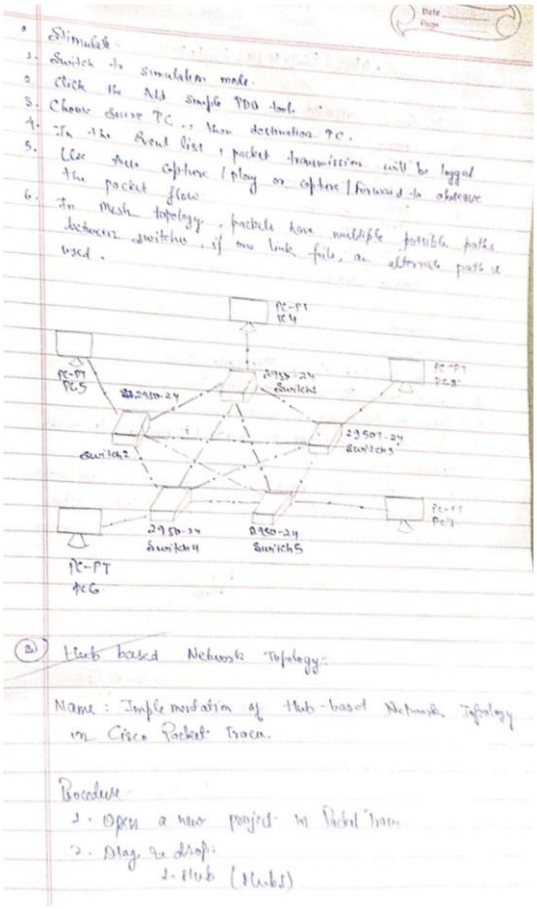
1. Switch to simulation mode.
2. Click the Add Simple PDU (envelop icon) tool.
3. Select source PC then orientation PC.
4. In the event list, a packet creation entry will appear.
5. Click Auto Capture / Play or capture / Forward to watch packet flow.
6. With Switch, the packet travels only to the intended

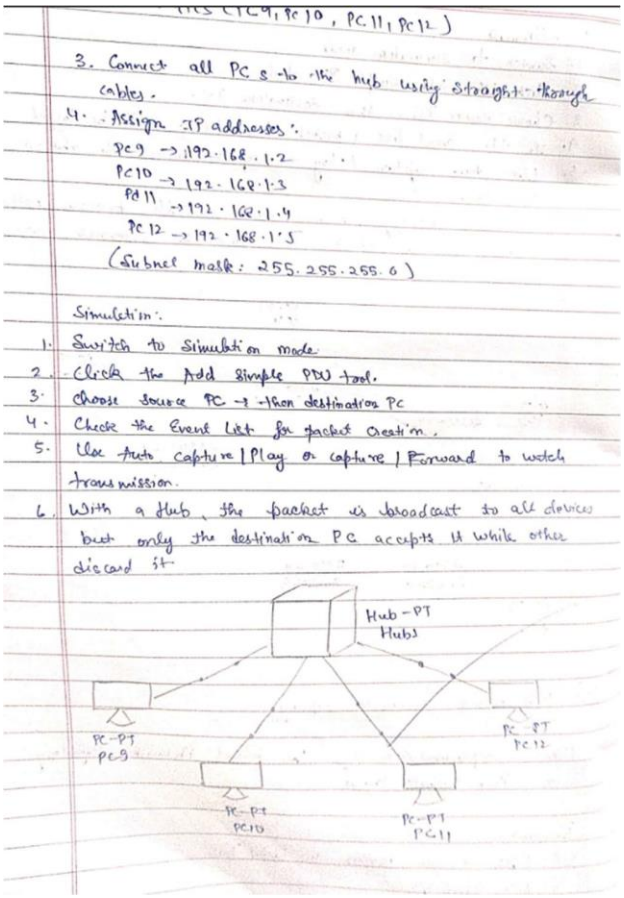


② MESH Topology using Switches:

- Name: Implementation of Mesh Topology using Switches in CISCO Packet Tracer.
- Procedure:

1. Open a new project in Packet Tracer.
2. Drag & drop:
   - 5 Switches (Switch 1, Switch 2, Switch 3, Switch 4, Switch 5)
3. Connect each PC to a Switch
4. Interconnect all Switches
5. Assign IP addresses.
   - PC4 → 192.168.1.2
   - PC5 → 192.168.1.3
   - PC6 → 192.168.1.4
   - PC7 → 192.168.1.5
   - PC8 → 192.168.1.6





Output:

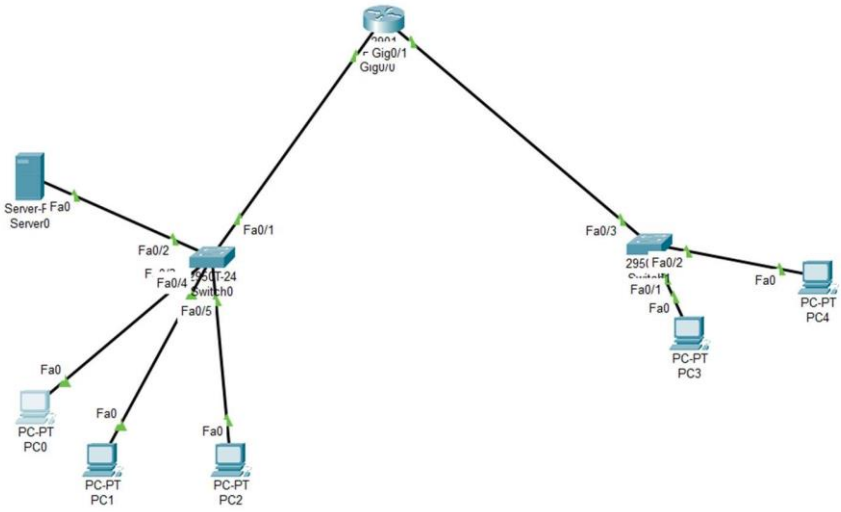
Realtime

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC2	PC4	ICMP		0.000	N	0	(edit)	(delete)
	Successful	PC3	PC2	ICMP		0.000	N	1	(edit)	(delete)
	Successful	PC4	PC3	ICMP		0.000	N	2	(edit)	(delete)

Program 2:

Aim: Configure DHCP within a LAN and outside LAN.

Network diagram:



Configration:

LAB-2.

8) Configure DHCP within a LAN and Outside a LAN.

Steps:

1. Click on DHCP server → Desktop → IP config. → Static → DHCP 192.168.10.2. and Gateway → 192.168.10.1

Services → DHCP → Pool name : Switch One Switch Two

	Switch One	Switch Two
Gateway :	192.168.10.1	192.168.20.1
Start IP address:	192.168.10.1	192.168.20.2
Subnet Mask:	255.255.255.0	255.255.255.0

ADD ADD

Routes:

↳ CLI

↳ no

Router # enable

conf t

info fa0/0

ip address 192.168.10.1 255.255.255.0

ip helper-address 192.168.10.2

no shutdown 192.168.10.2

do write memory

exit

int fa 0/1.

ip address 192.168.20.1 255.255.255.0

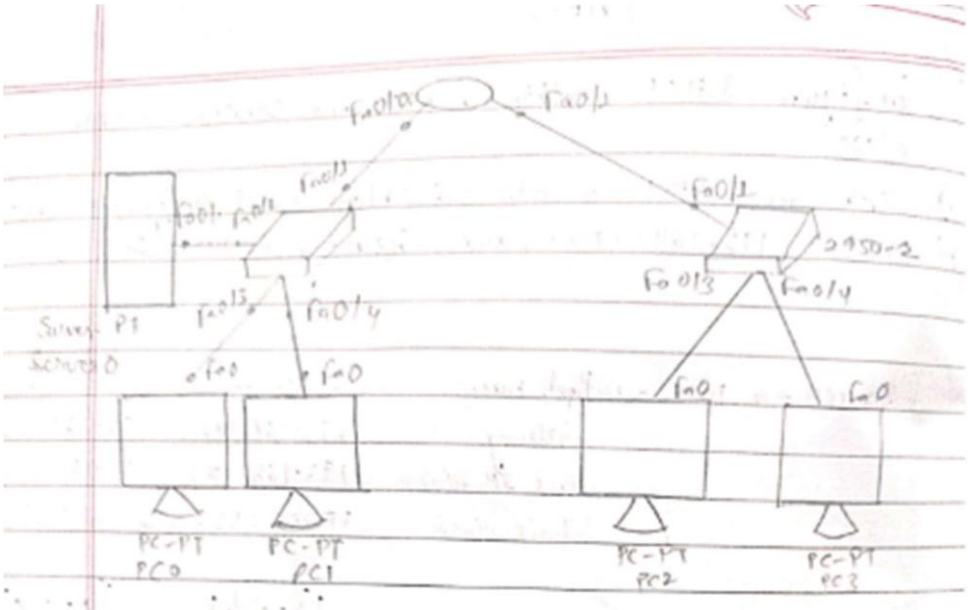
ip helper-address 192.168.10.2

no shutdown

exit

write memory

Note : IP-helper-add is also called Relay Agent



DHCP concepts :

→ DHCP-Server : The device that assigns IP config.

→ DHCP Client : A device that requests network setting from DHCP server.

→ IP lease : The temporary assignment of an IP addr to client

→ Scope / pool : Arrays of IP address available for assignment

Output:

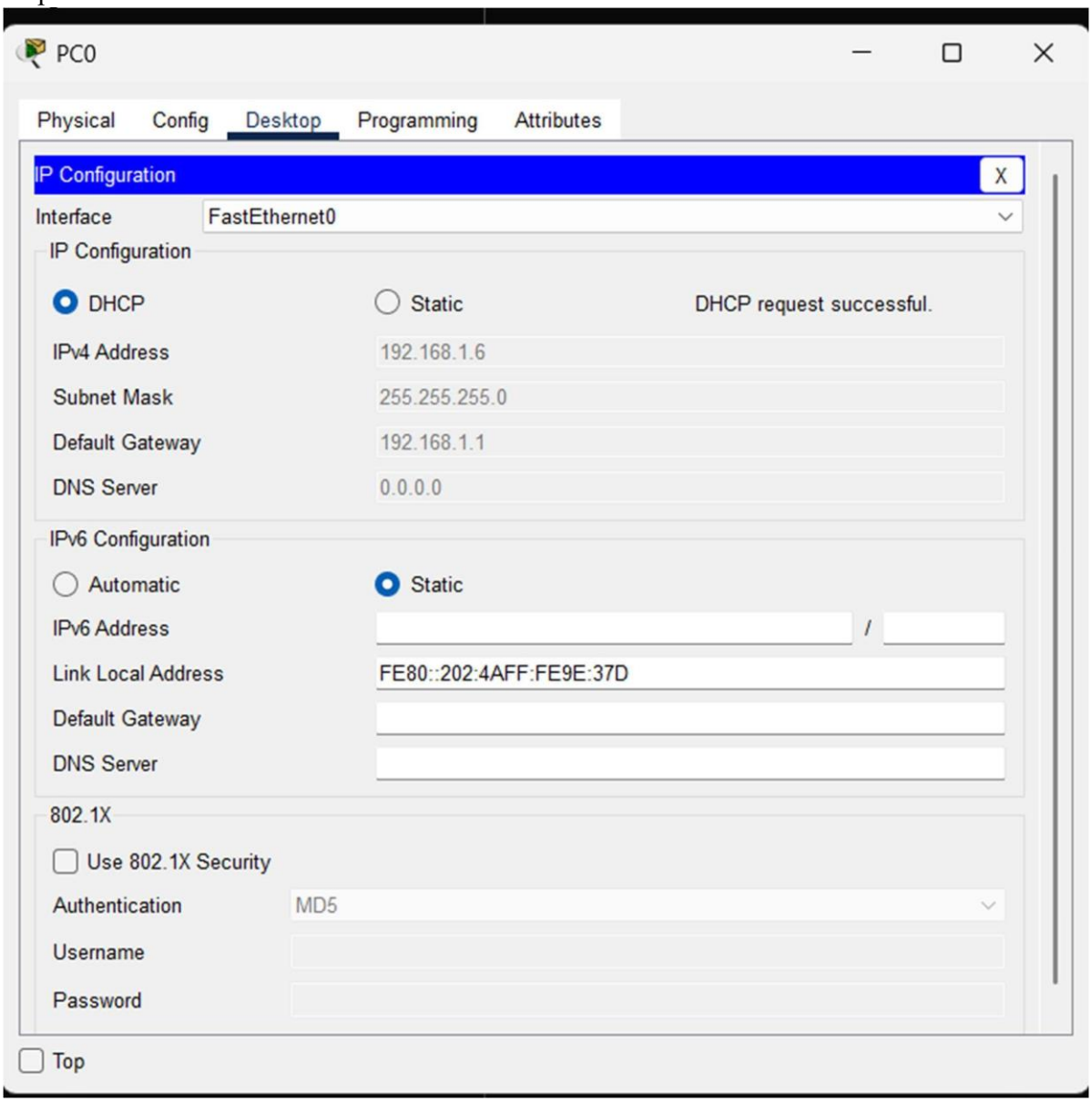


Fig 1. Ip address assigned by DHCP server within LAN (PC1)

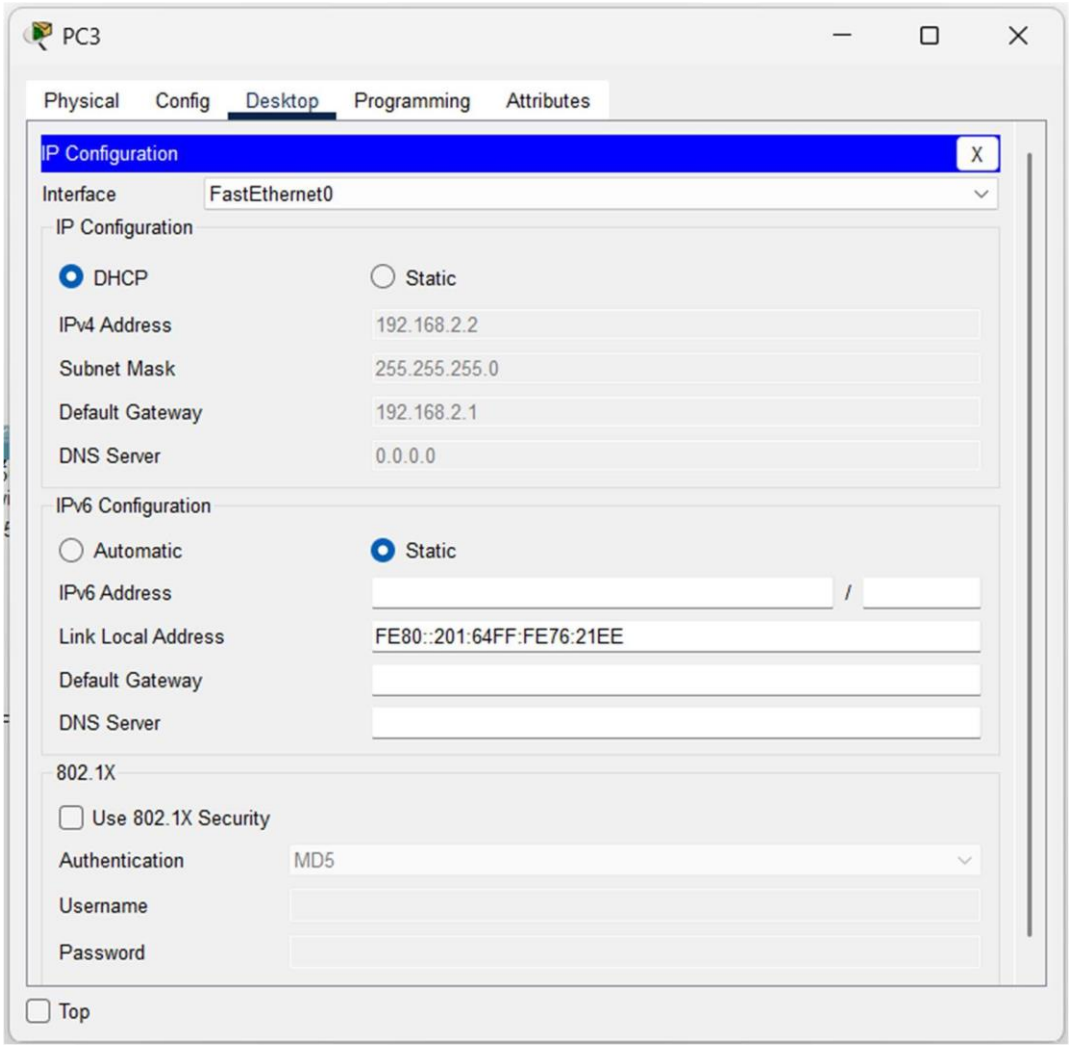
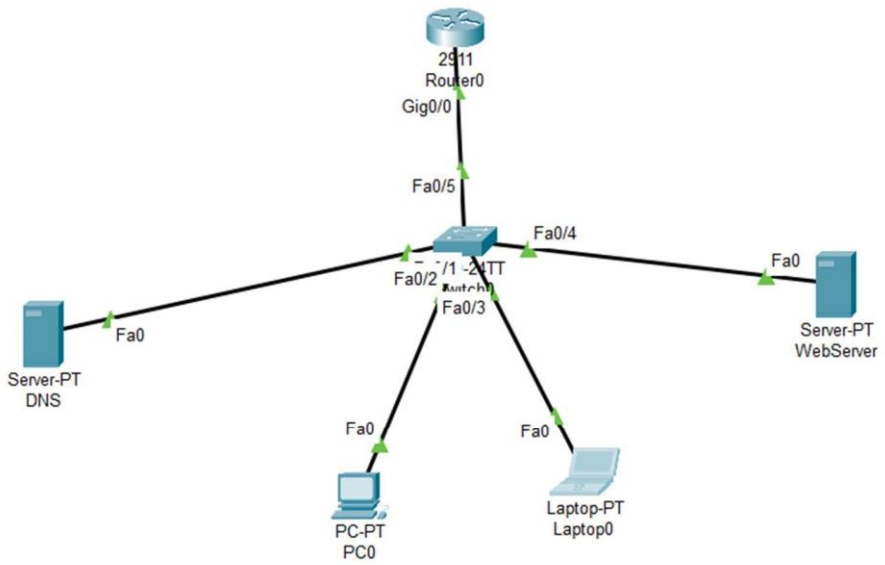


Fig 2. Ip address assigned by DHCP server outside LAN

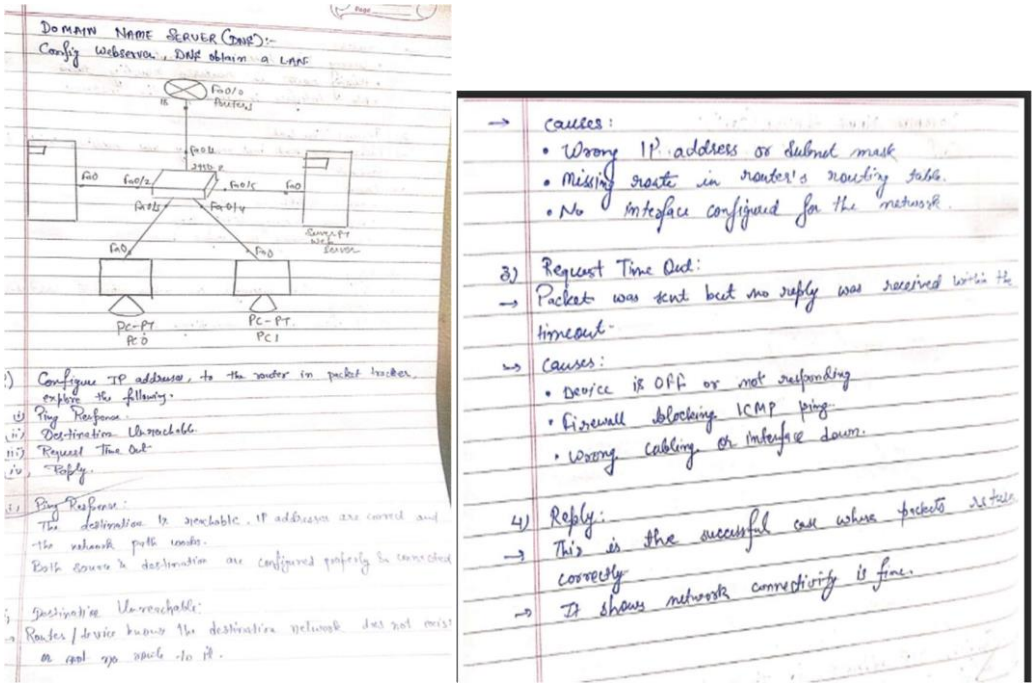
Program 3:

Aim: Configure Web Server, DNS within a LAN.

Network diagram:



Configuration:



Output:

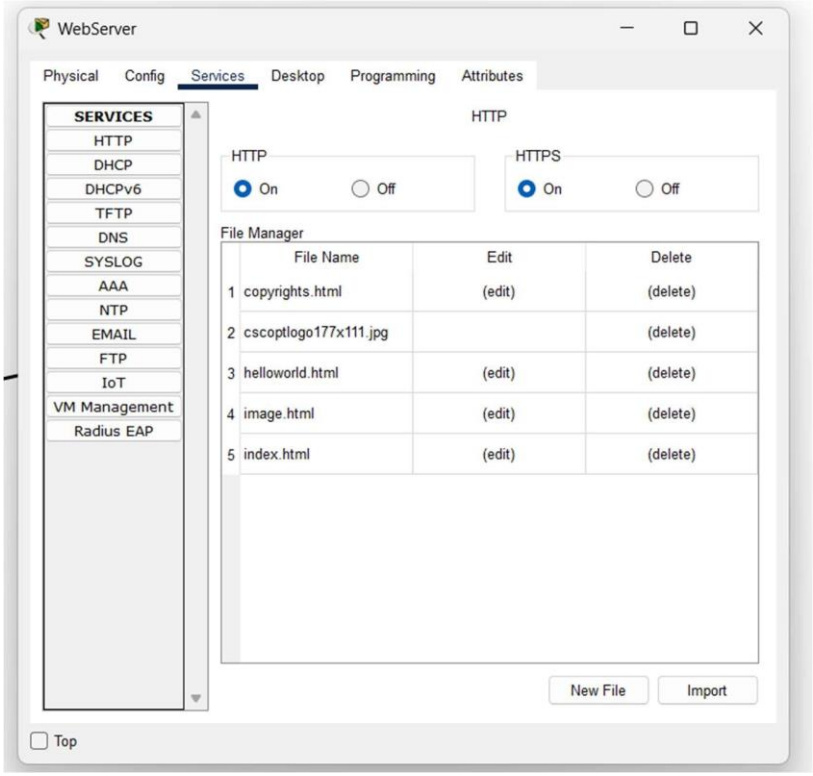


Fig 1. WEB server – HTTP Services

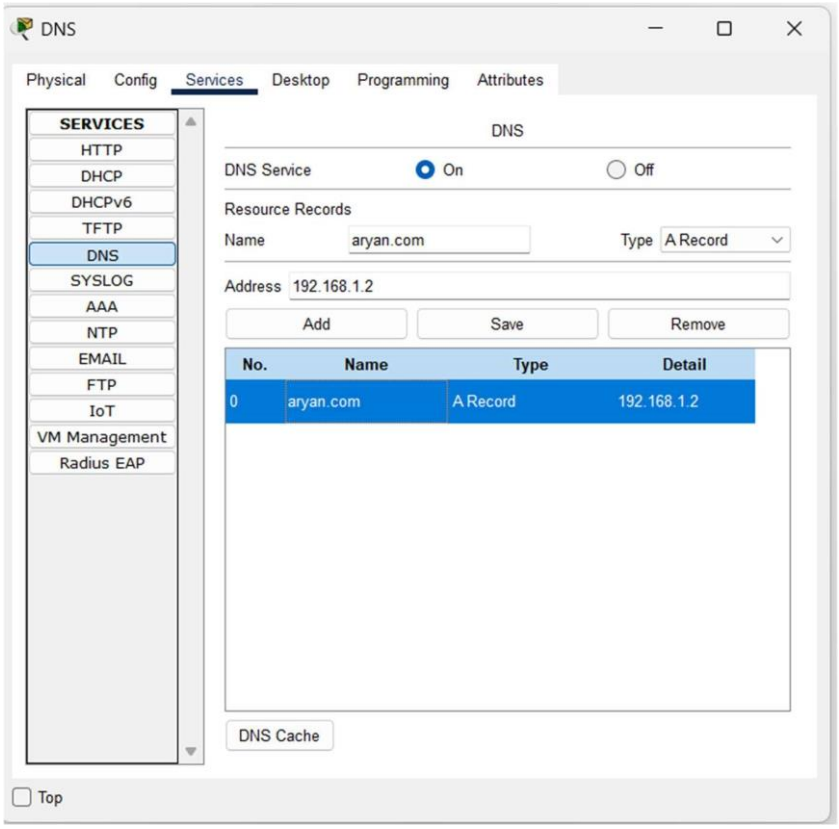


Fig 2. DNS server – DNS Services

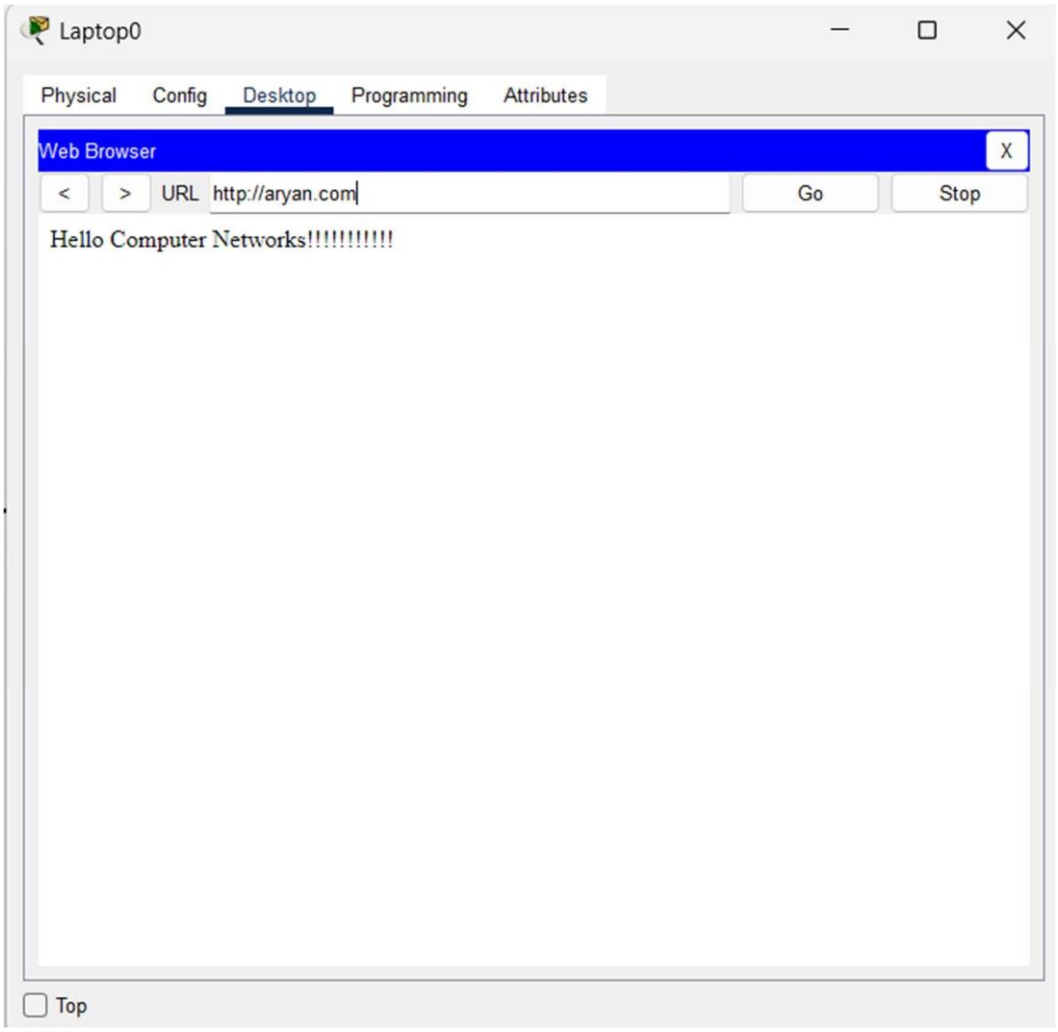
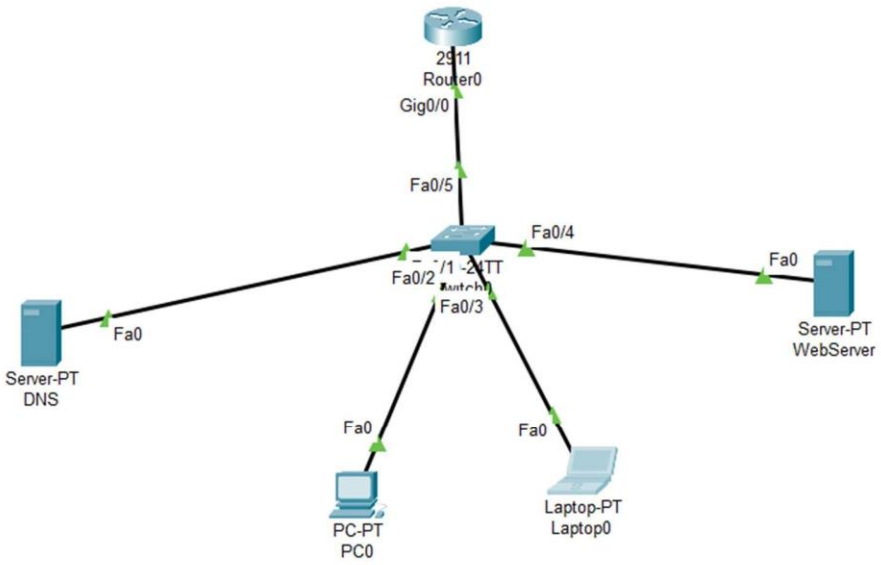


Fig 3. Laptop0 – accessing data from web browser

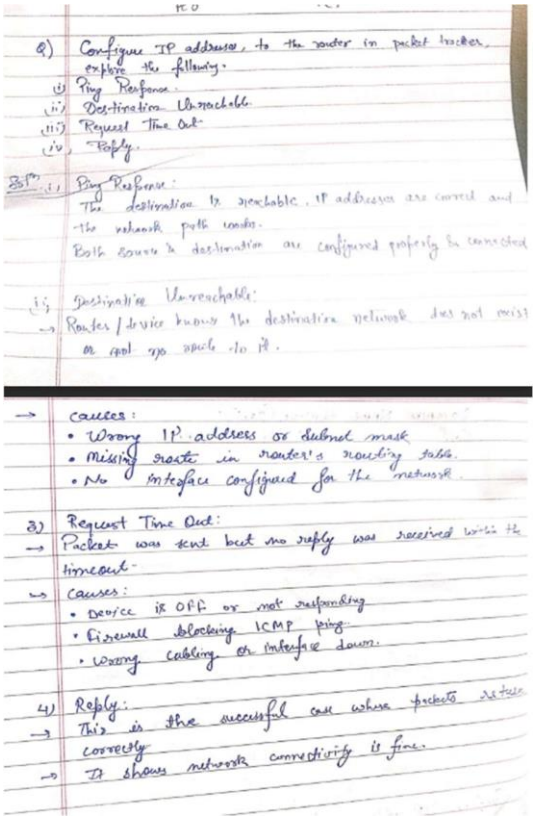
Program 4:

Aim: Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.

Network diagram:



Configuration:



Q) Configure IP addresses to the router in packet tracker, explore the following.
(i) Ping Response
(ii) Destination Unreachable
(iii) Request Time Out
(iv) Reply.

Ans (i) Ping Response:
The destination is reachable. IP addresses are correct and the network path works.
Both source & destination are configured properly & connected

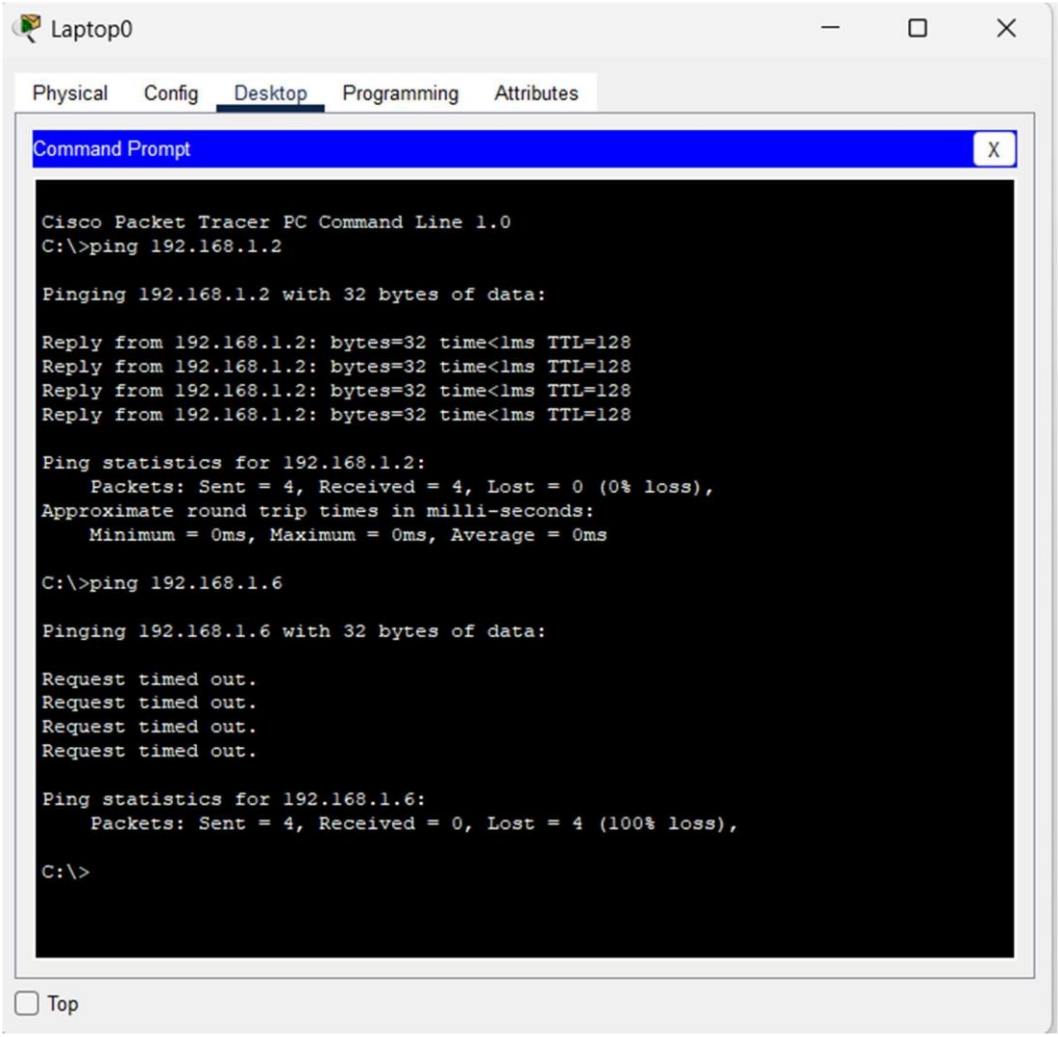
(ii) Destination Unreachable:
→ Router / device knows the destination network does not exist or does not know how to reach it.

→ causes:
- Wrong IP address or subnet mask
- Missing route in router's routing table.
- No interface configured for the network.

3) Request Time Out:
→ Packet was sent but no reply was received within the timeout.
→ causes:
- Device is OFF or not responding
- Firewall blocking ICMP ping
- Wrong cabling or interface down.

4) Reply:
→ This is the successful case where packets return correctly
→ It shows network connectivity is fine.

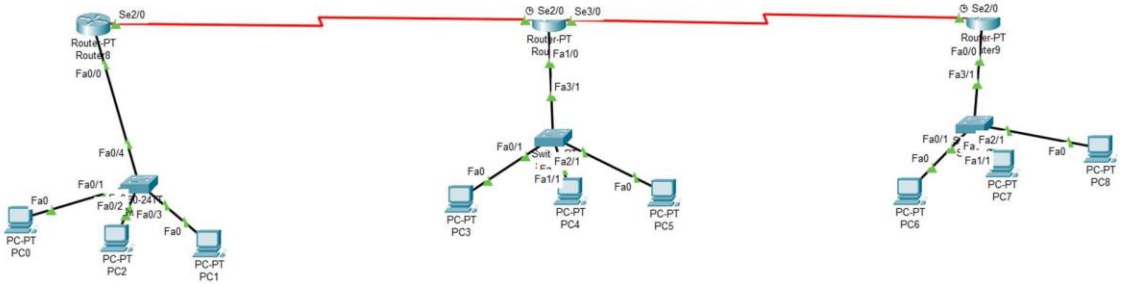
Output:



Program 5:

Aim: Configure default route, static route to the Router.

Network diagram:

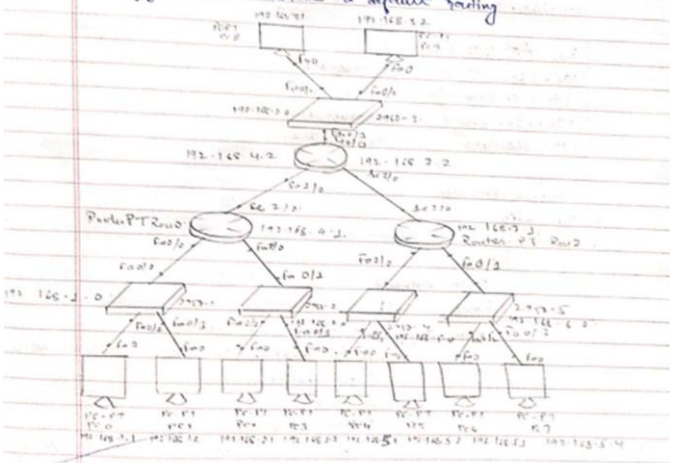


Configuration:

Lab-3

1. Static -- known data
2. Dynamic -- Unknown data

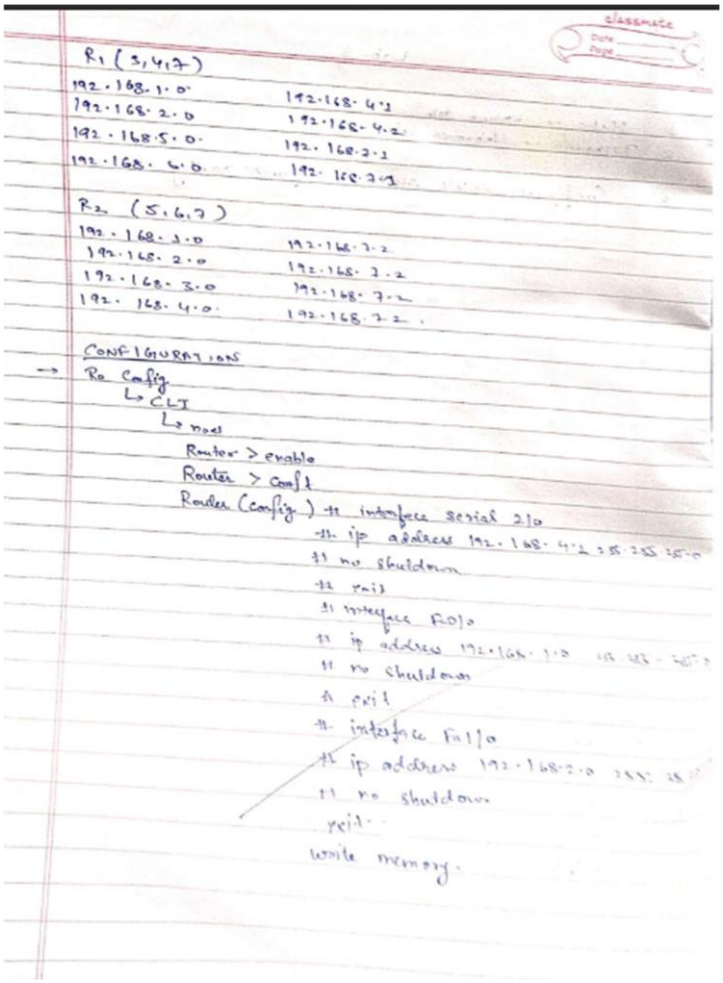
A) Configure IPV4 static & default routing



STATIC ROUTES:

R0 (1,2,4)

192.168.3.0	192.168.4.2
192.168.5.0	192.168.4.2
192.168.6.0	192.168.4.2
192.168.7.0	192.168.4.2



R1 (3,4,7)

192.168.1.0
192.168.2.0
192.168.5.0
192.168.6.0

192.168.4.1
192.168.4.2
192.168.2.1
192.168.3.1

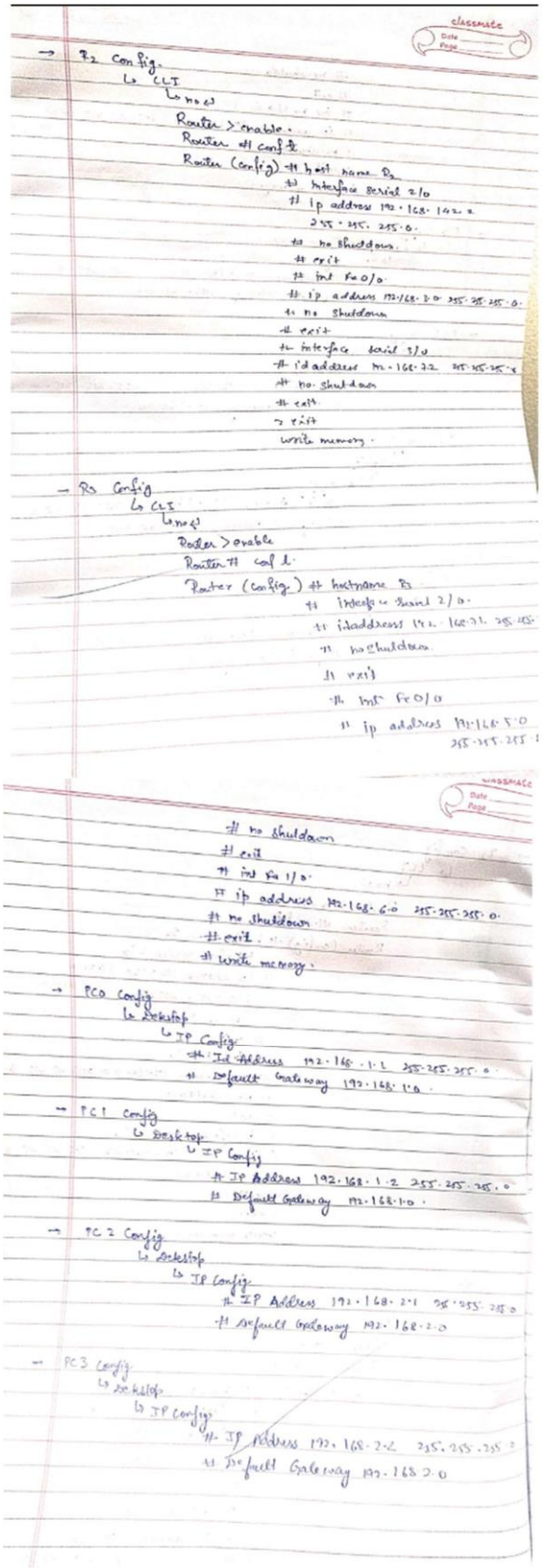
R2 (5,6,7)

192.168.1.0
192.168.2.0
192.168.3.0
192.168.4.0

192.168.7.2
192.168.7.2
192.168.7.2
192.168.7.2

CONFIGURATIONS

→ Ro Config
↳ CLI
↳ root
Router > enable
Router > conf t
Router (config) # interface serial 2/0
ip address 192.168.4.2 255.255.255.0
no shutdown
exit
interface Fa0/0
ip address 192.168.1.0 255.255.255.0
no shutdown
exit
interface Fa1/0
ip address 192.168.2.0 255.255.255.0
no shutdown
exit
write memory



→ R2 Config
↳ CLI
↳ no
Router > enable.
Router # conf t
Router (config) # host name R2
interface serial 2/0
ip address 192.168.192.x
255.255.255.0.
no shutdown.
exit
int fa 0/0.
ip address 192.168.2.0 255.255.255.0.
no shutdown
exit
interface serial 3/0
ip address 192.168.72. 255.255.255.x
no shutdown
exit
exit
write memory.

→ R3 Config
↳ CLI
↳ no
Router > enable
Router # conf t.
Router (config) # hostname R3
interface serial 2/0.
ipaddress 192.168.72. 255.255.
no shutdown
exit
int fa 0/0
ip address 192.168.5.0
255.255.255.

no shutdown
exit
int fa 1/0.
ip address 192.168.6.0 255.255.255.0.
no shutdown
exit.
write memory.

→ PC0 Config
↳ Desktop
↳ IP Config
Ip Address 192.168.1.1 255.255.255.0.
Default Gateway 192.168.1.0.

→ PC1 Config
↳ Desktop
↳ IP Config
IP Address 192.168.1.2 255.255.255.0
Default Gateway 192.168.1.0.

→ PC2 Config
↳ Desktop
↳ IP Config
IP Address 192.168.2.1 255.255.255.0
Default Gateway 192.168.2.0

→ PC3 Config
↳ Desktop
↳ IP Config
IP Address 192.168.2.2 255.255.255.0
Default Gateway 192.168.2.0

Output:

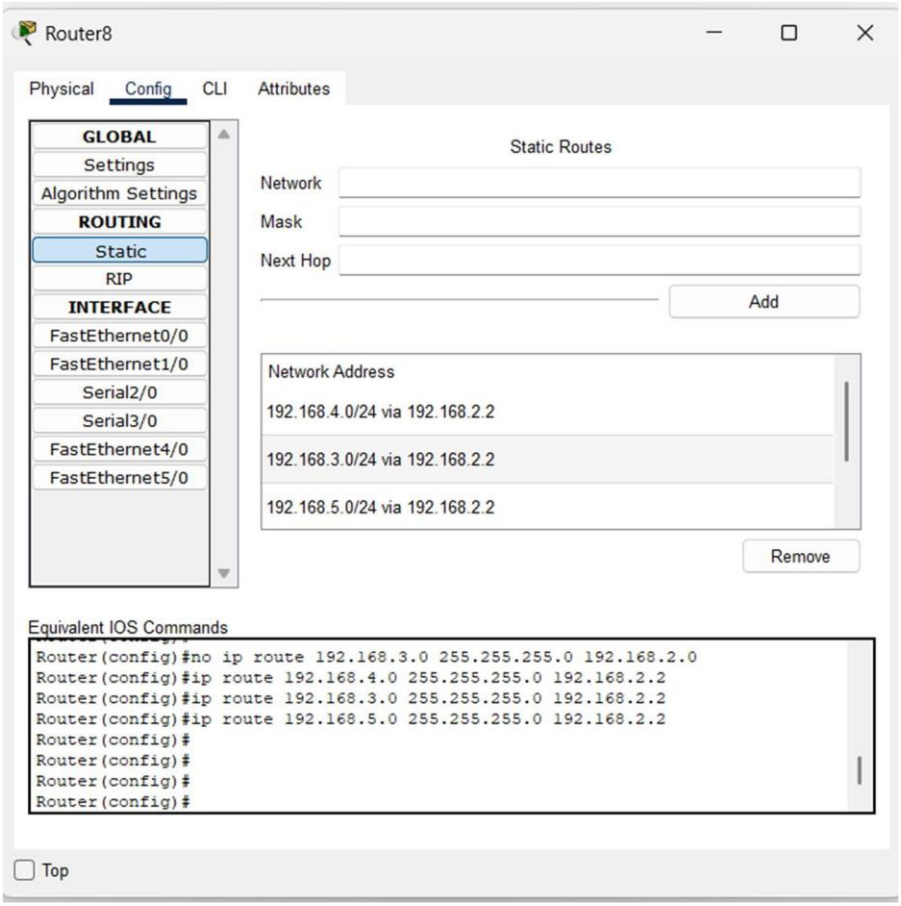


Fig 1. Router– Static routing

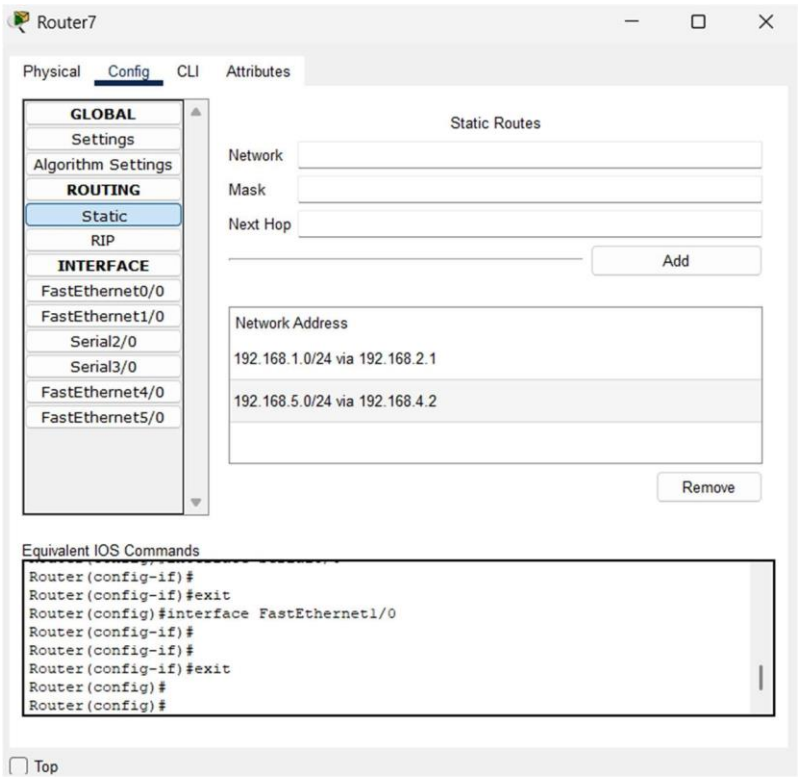


Fig 2. Router– Static routing

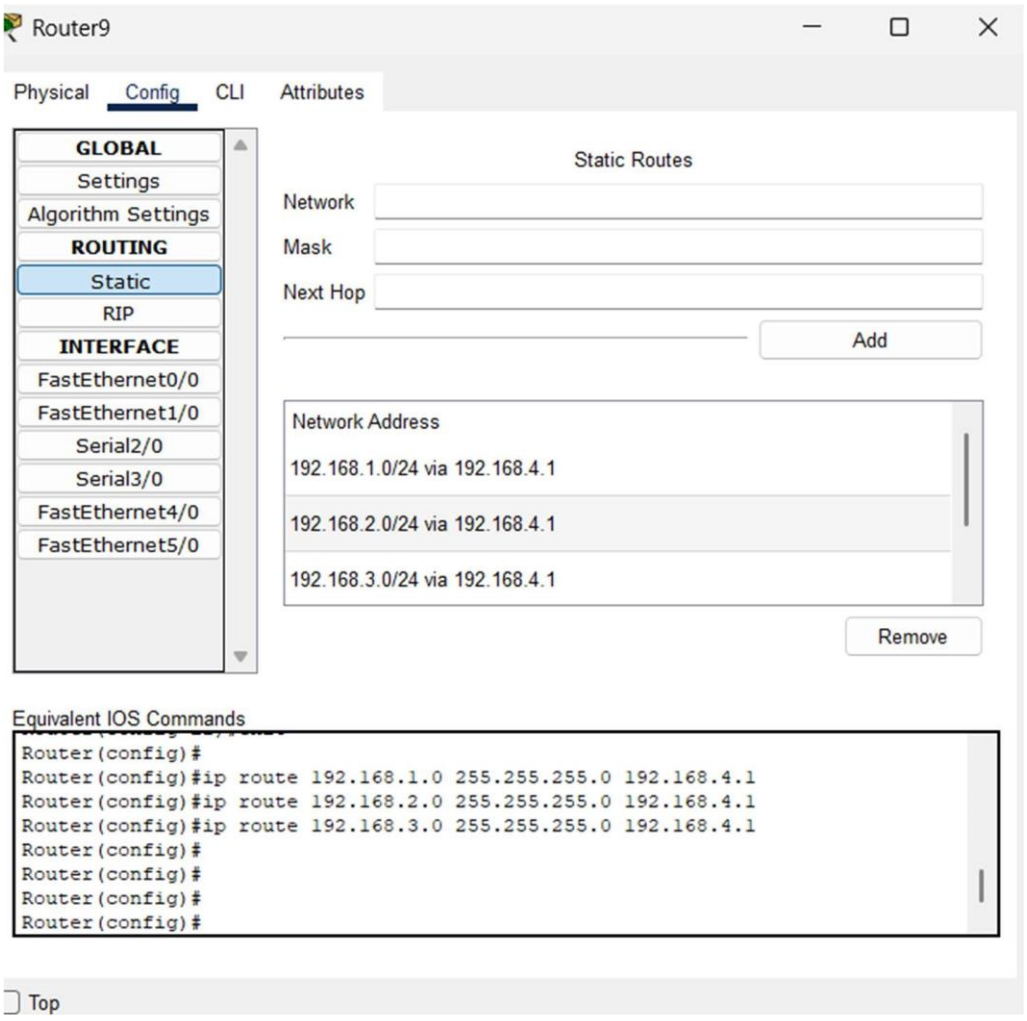
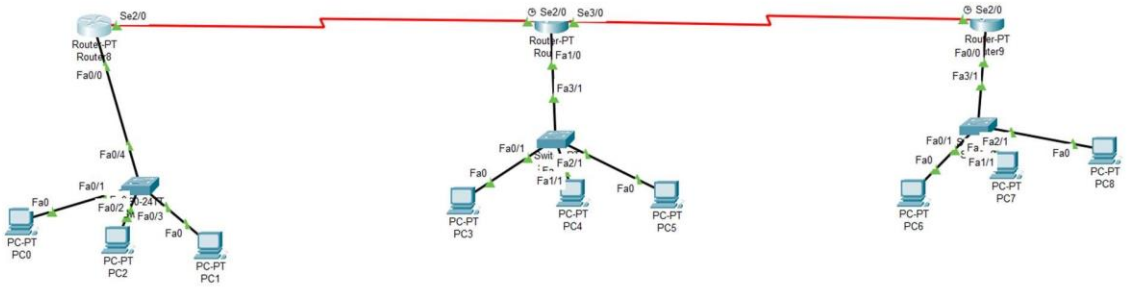


Fig 3. Router– Static routing

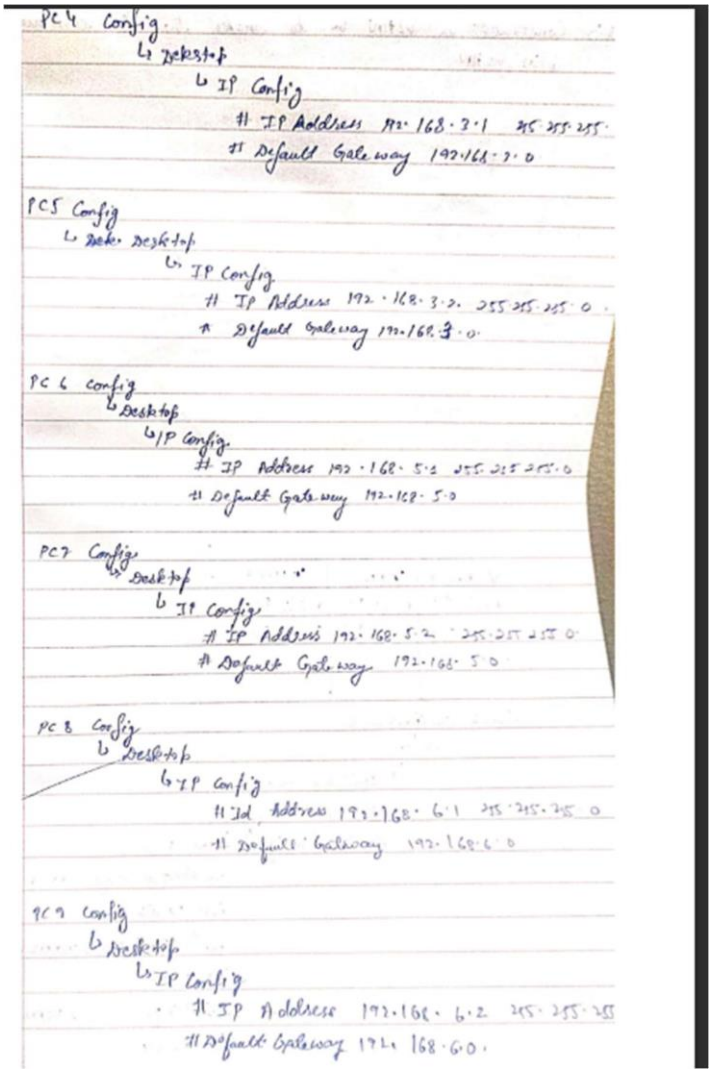
Program 6:

Aim: Configure RIP routing Protocol in Routers.

Network diagram:



Configuration:


PC4 Config
Desktop
IP Config
IP Address 192.168.3.1 255.255.255.
Default Gateway 192.168.3.0
PC5 Config
Desktop
IP Config
IP Address 192.168.3.2 255.255.255.0
Default Gateway 192.168.3.0
PC6 config
Desktop
IP Config
IP Address 192.168.5.2 255.255.255.0
Default Gateway 192.168.5.0
PC7 Config
Desktop
IP Config
IP Address 192.168.5.2 255.255.255.0
Default Gateway 192.168.5.0
PC8 Config
Desktop
IP Config
IP Address 192.168.6.1 255.255.255.0
Default Gateway 192.168.6.0
PC9 Config
Desktop
IP Config
IP Address 192.168.6.2 255.255.255
Default Gateway 192.168.6.0.

Output:

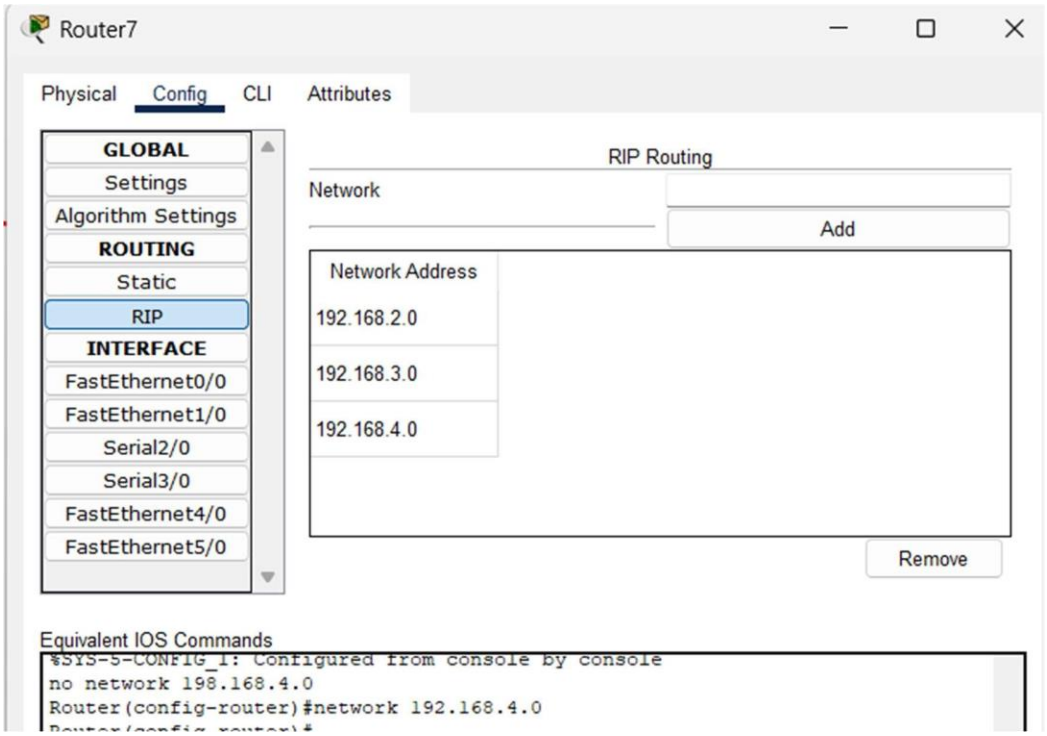


Fig 1. Router – RIP routing

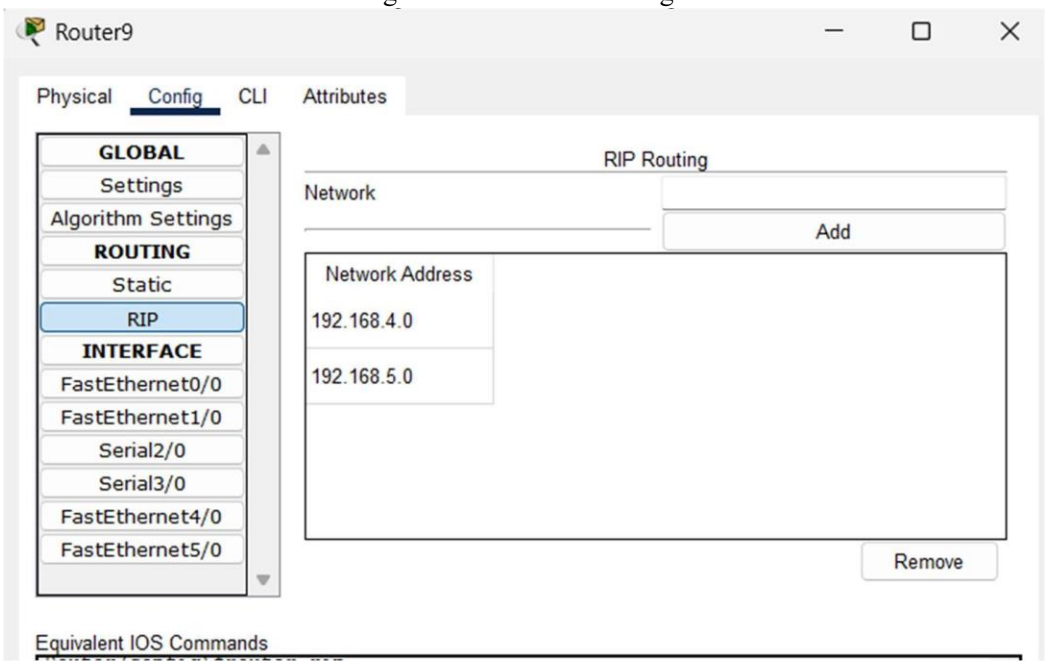


Fig 2. Router – RIP routing

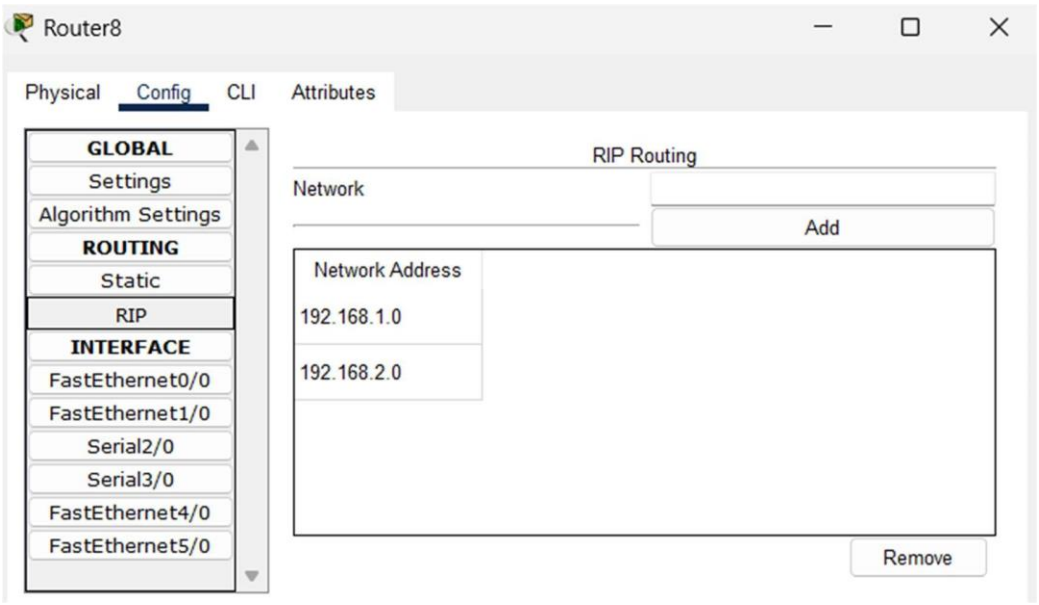
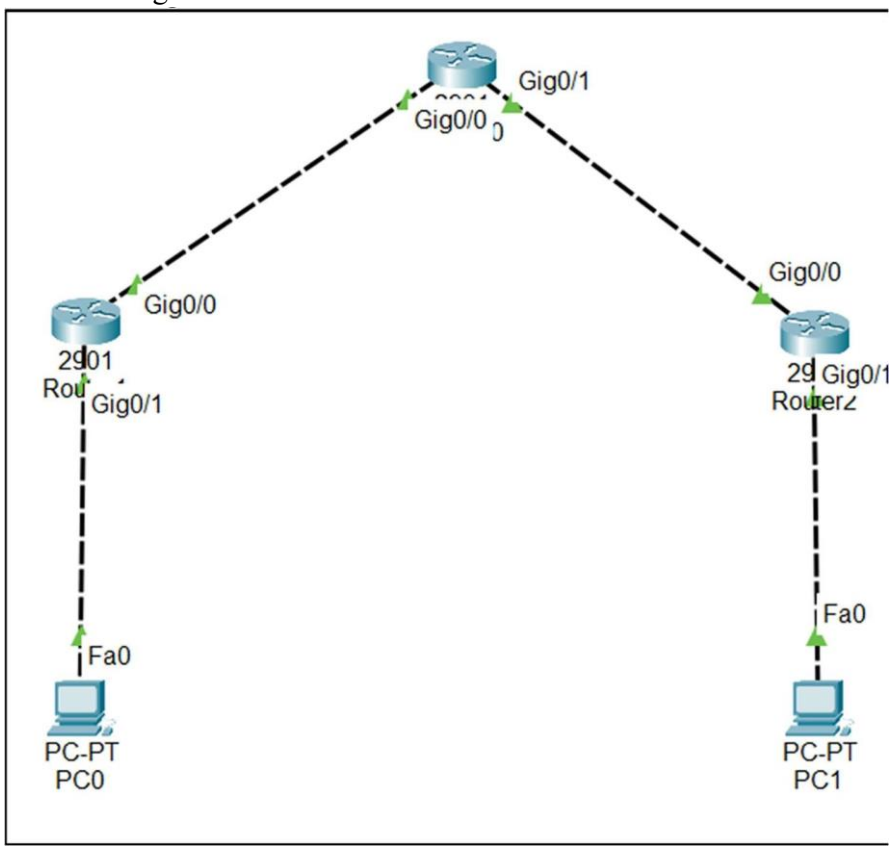


Fig 3. Router – RIP routing

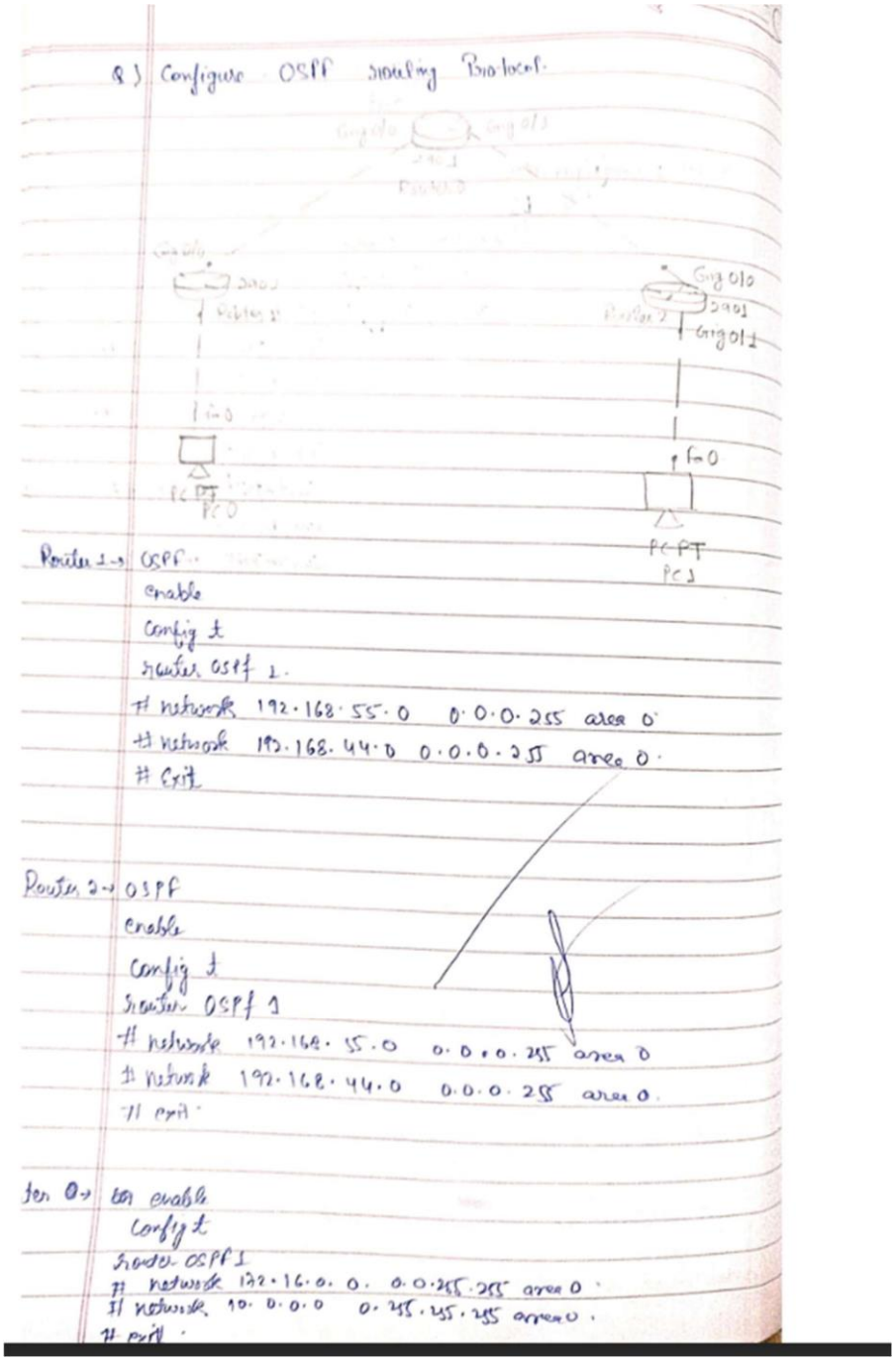
Program 7:

Aim: Configure OSPF routing protocol.

Network diagram:



Configuration:



Output:

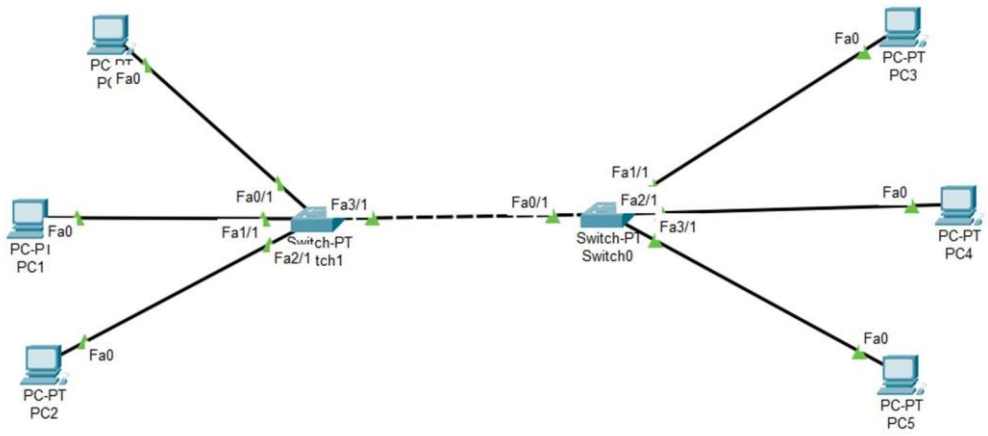
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC2	PC5	ICMP		0.000	N	3	(edit)	(delete)
	Successful	PC1	PC6	ICMP		0.000	N	4	(edit)	(delete)
	Successful	PC1	PC6	ICMP		0.000	N	5	(edit)	(delete)

Checking PDU messages

Program 8:

Aim: To construct a VLAN and make the PC's communicate among a VLAN.

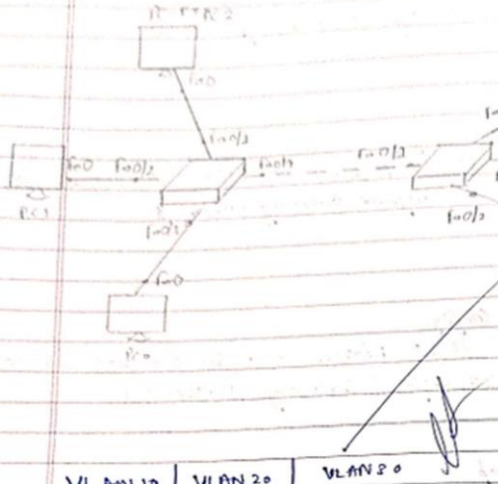
Network diagram:



Configuration:

Week - 4

a> Construct a VLAN to do make the PC communicate among VLAN



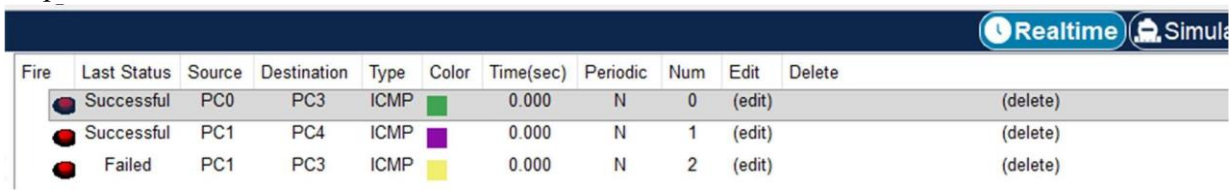
VLAN 10	VLAN 20	VLAN 30
Fa0/1 Pc0	Fa0/2 Pc1	Fa0/3 Pc3
Fa0/1 Pc3	Fa0/2 Pc21	Fa0/3 Pc5

Switch configuration
↳ CLI
↳ Switch> enable
Switch # config t
Switch (config-if) int Fa 0/1.
Switchport access VLAN10
int Fa 0/2
Switchport access VLAN20
int Fa 0/3
Switch port access VLAN30
int Fa 0/4

Switchport mod trunk
exit

Switch 1 configuration
↳ CLI
↳ Switch >enable
Switch # config t
Switch config # int Fa 0/1
Switch port access VLAN 10
int Fa 0/2
switchport access VLAN20
int Fa 0/3.
Switchport access VLAN30
int Fa 0/4.
Switchport mod trunk ...

Output:



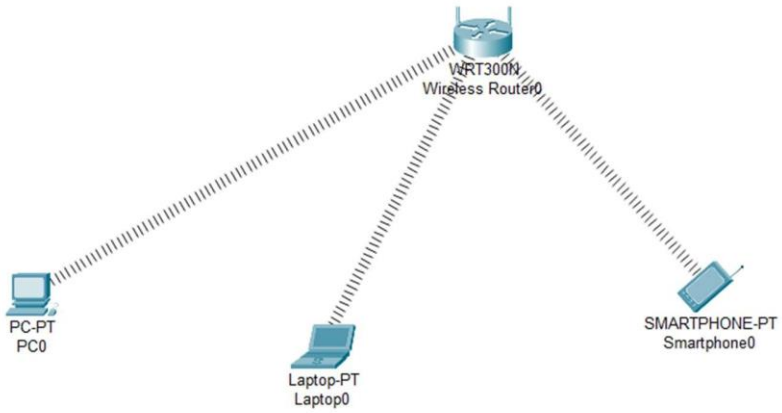
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC3	ICMP		0.000	N	0	(edit)	(delete)
	Successful	PC1	PC4	ICMP		0.000	N	1	(edit)	(delete)
	Failed	PC1	PC3	ICMP		0.000	N	2	(edit)	(delete)

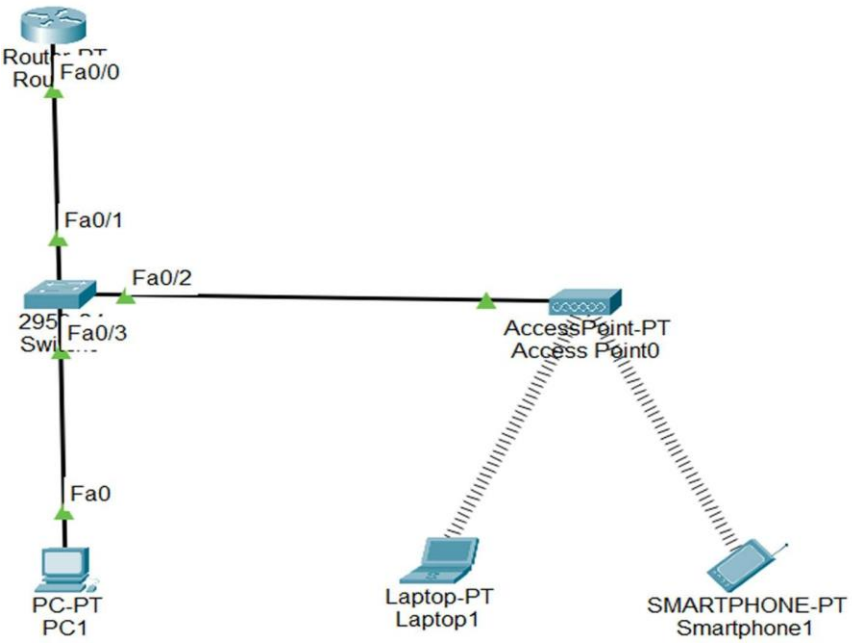
Checking PDU messages

Program 9:

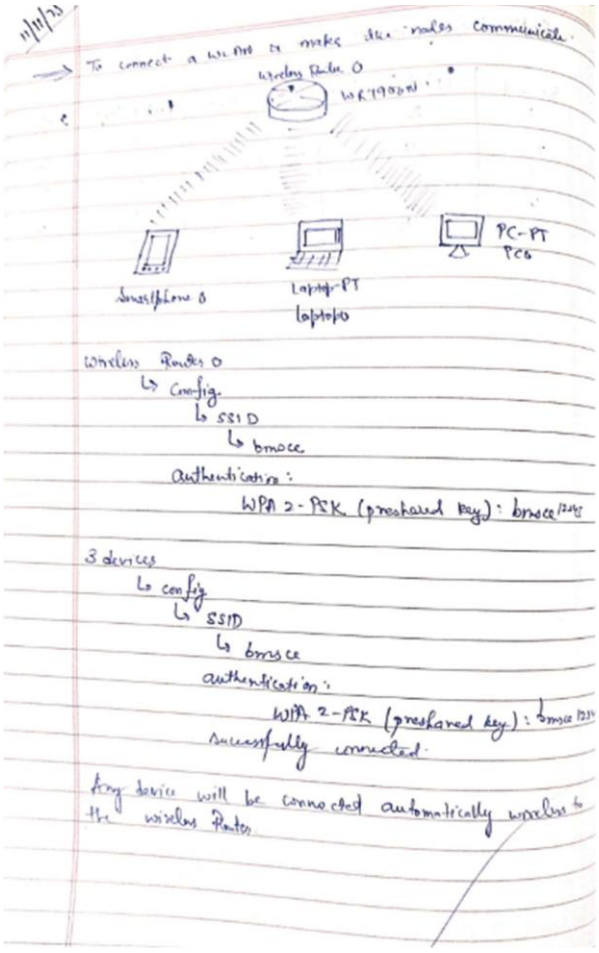
Aim: To construct a WLAN and make the nodes communicate wirelessly.

Network diagram:





Configuration:



Output:

Fig 1.1 Step1: Turn off light / Power off laptop

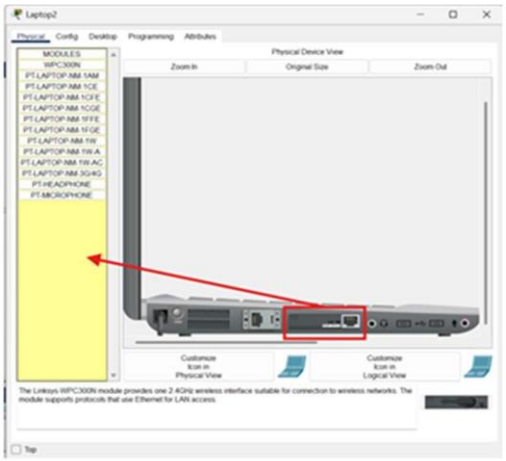


Fig 1.2 Step2: Drag and Drop the Ethernet into pointed location

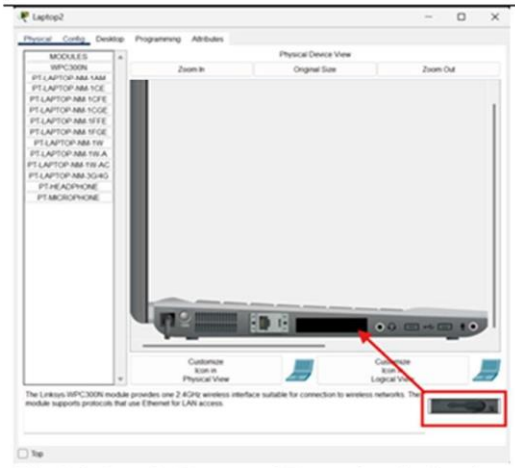


Fig 1.3 Step3: Drag and Drop the device into pointed location and Turn on light/Laptop

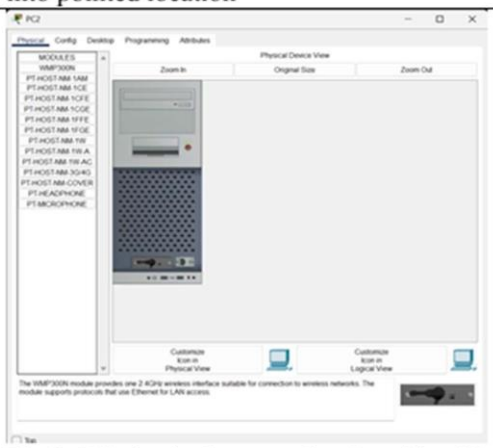


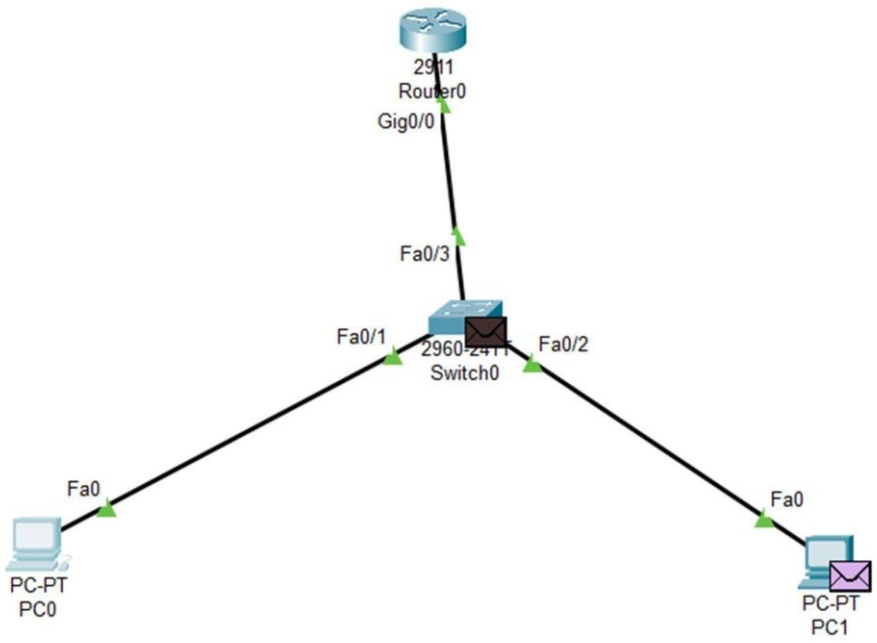
Fig 2. PC physical connection (combined 3 steps)

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	Wirel...	Smartpho...	ICMP		0.000	N	0	(edit)	(delete)
	Successful	PC0	Wireless ...	ICMP		0.000	N	1	(edit)	(delete)
	Successful	Laptop0	Smartpho...	ICMP		0.000	N	2	(edit)	(delete)

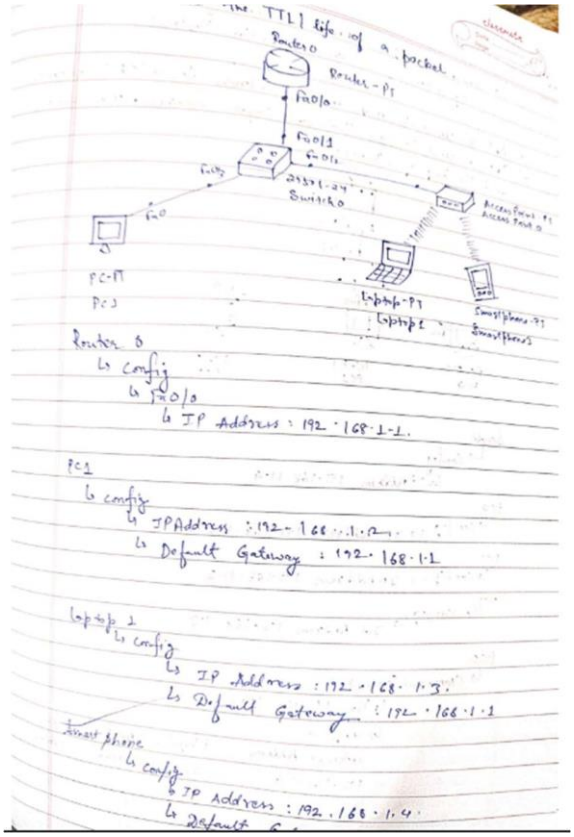
Program 10:

Aim: Demonstrate the TTL/ Life of a Packet.

Network diagram:



Configuration:


TTL life of a packet
Router0
Router-PT
Fa0/0
Fa0/1
Fa0/2
Fa0/3
2950T-24
Switch0
Fa0
PC-PT
PC3
Laptop-PT
Laptop1
Smartphone-PT
Smartphone3
Router 0
config
Fa0/0
IP Address : 192.168.1.1.
PC1
config
IPAddress : 192.168.1.2
Default Gateway : 192.168.1.1
Laptop 1
config
IP Address : 192.168.1.3.
Default Gateway : 192.168.1.1
Smart phone
config
IP Address : 192.168.1.4.

Output:

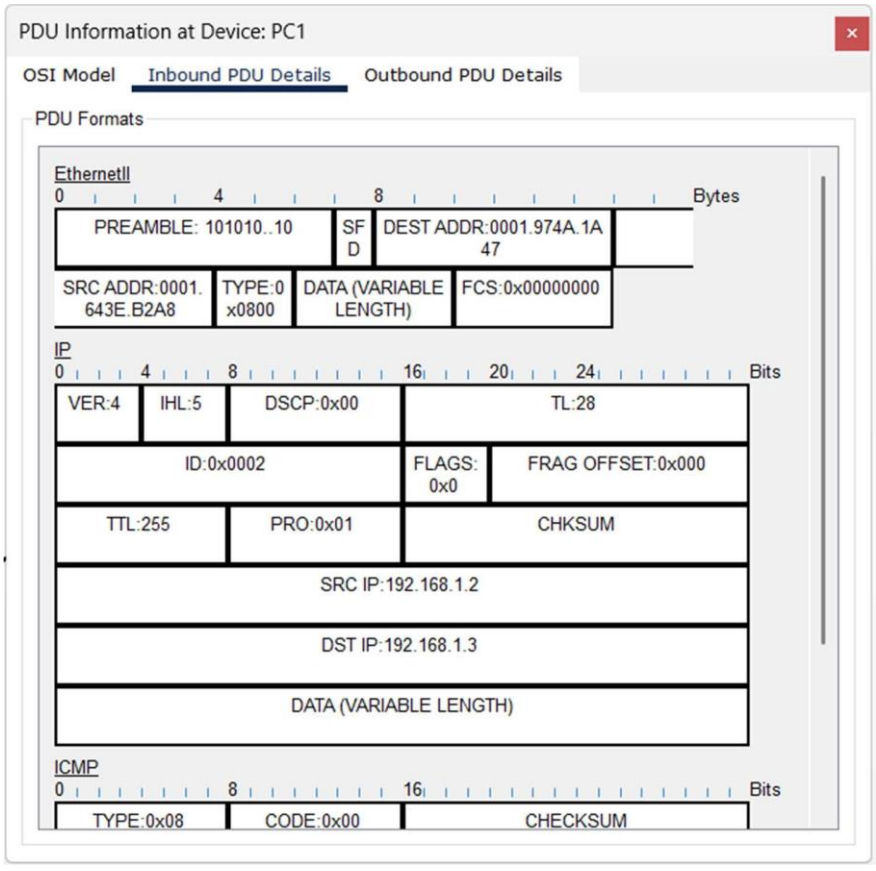

PDU Information at Device: PC1
OSI Model
Inbound PDU Details
Outbound PDU Details
PDU Formats
EthernetII
0
4
8
Bytes
PREAMBLE: 101010..10
SFD
DEST ADDR:0001.974A.1A47
SRC ADDR:0001.643E.B2A8
TYPE:0x0800
DATA (VARIABLE LENGTH)
FCS:0x00000000
IP
0
4
8
16
20
24
Bits
VER:4
IHL:5
DSCP:0x00
TL:28
ID:0x0002
FLAGS:0x0
FRAG OFFSET:0x000
TTL:255
PRO:0x01
CHKSUM
SRC IP:192.168.1.2
DST IP:192.168.1.3
DATA (VARIABLE LENGTH)
ICMP
0
8
16
Bits
TYPE:0x08
CODE:0x00
CHECKSUM

Fig 1. Inbound PDU Details at Device PC1

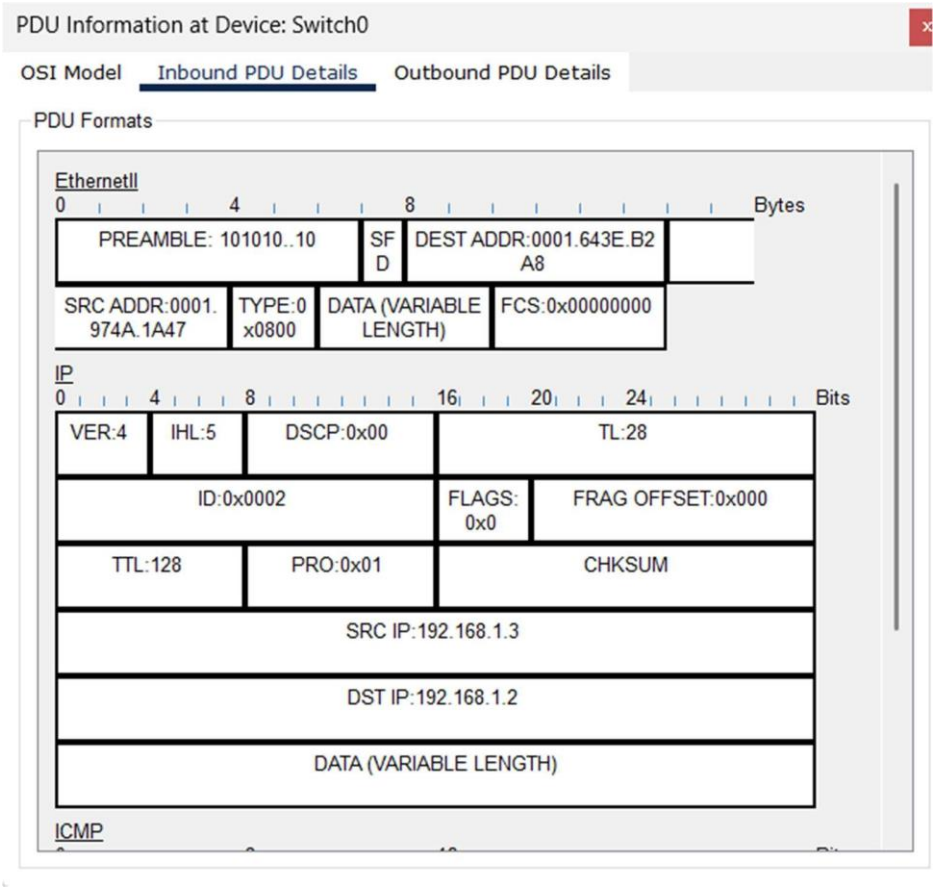
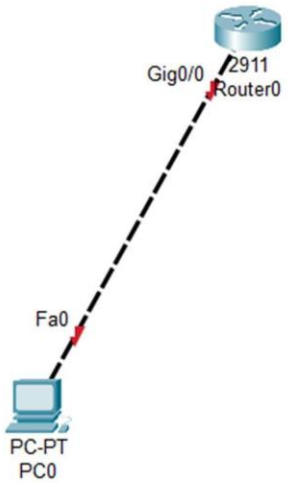


Fig 1. Inbound PDU Details at Device Switch0

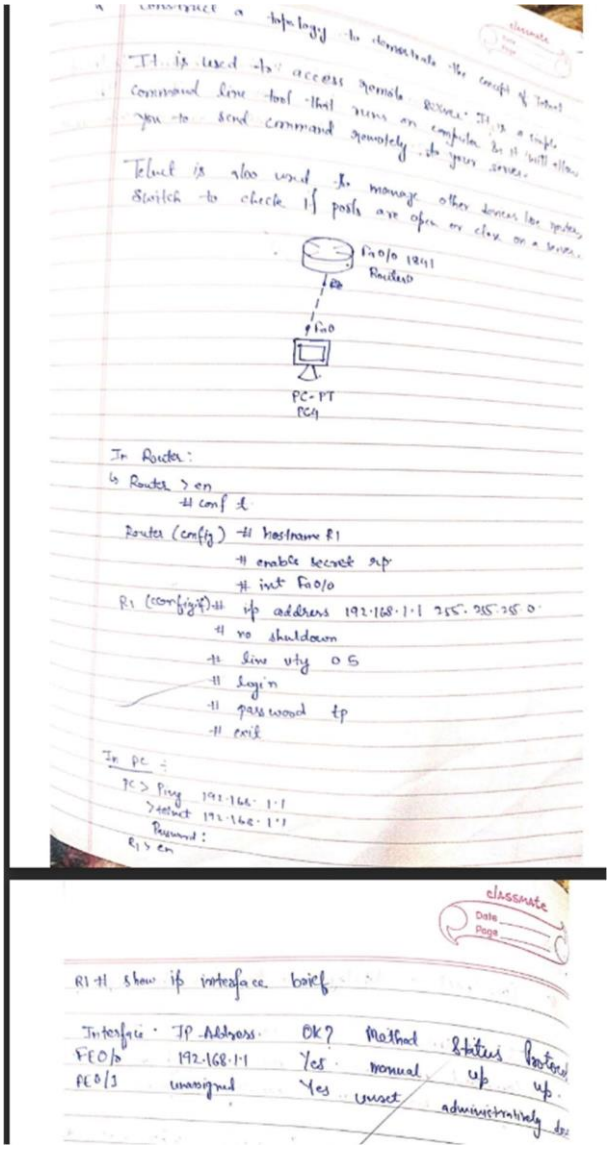
Program 11:

Aim: To understand the operation of TELNET by accessing the router in server room from a PC in IT office.

Network diagram:



Configuration:


construct a topology to demonstrate the concept of Telnet
Telnet is used to access remote device. It is a simple command line tool that runs on computer. It will allow you to send command remotely to your server.
Telnet is also used to manage other devices like routers, switch to check if ports are open or close on a server.
Fa0/0 192.168.1.1
Router0
Fa0
PC-PT
PC4
In Router:
Router > en
conf t
Router (config) # hostname R1
enable secret cp
int Fa0/0
R1 (config-if) # ip address 192.168.1.1 255.255.255.0
no shutdown
line vty 0 5
login
password tp
exit
In PC :
PC > Ping 192.168.1.1
> telnet 192.168.1.1
Password:
R1 > en
classmate
Date
Page
R1 # show ip interface brief
Interface IP Address OK? Method Status Protocol
FE0/0 192.168.1.1 Yes manual up up
FE0/1 unassigned Yes unset administratively down

Output:

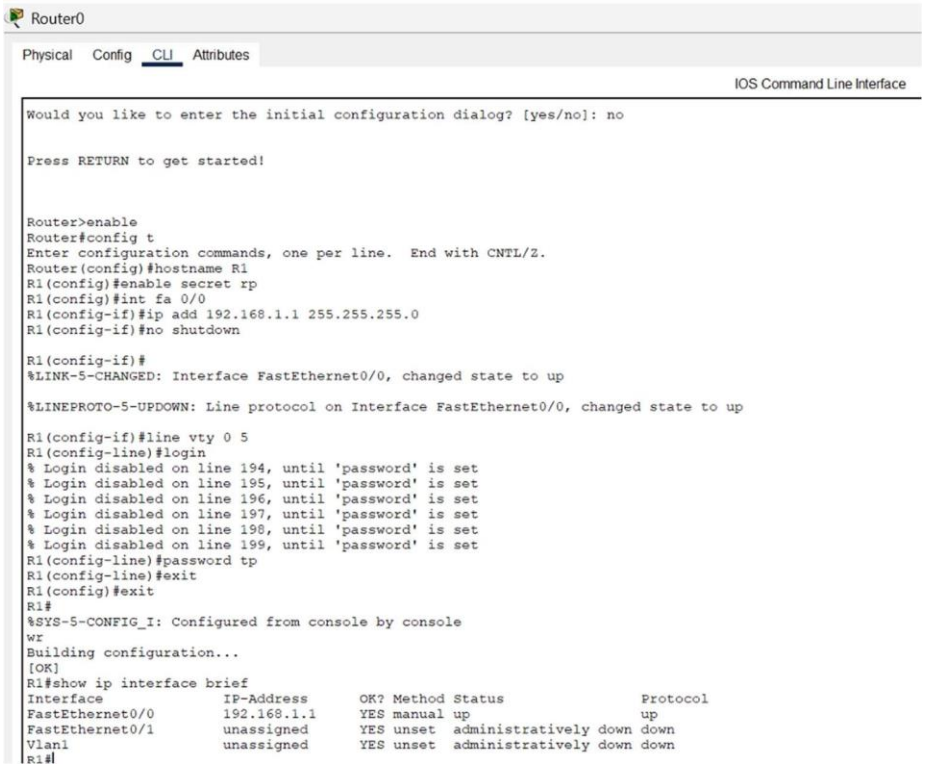


Fig 1. Router0 – CLI commands

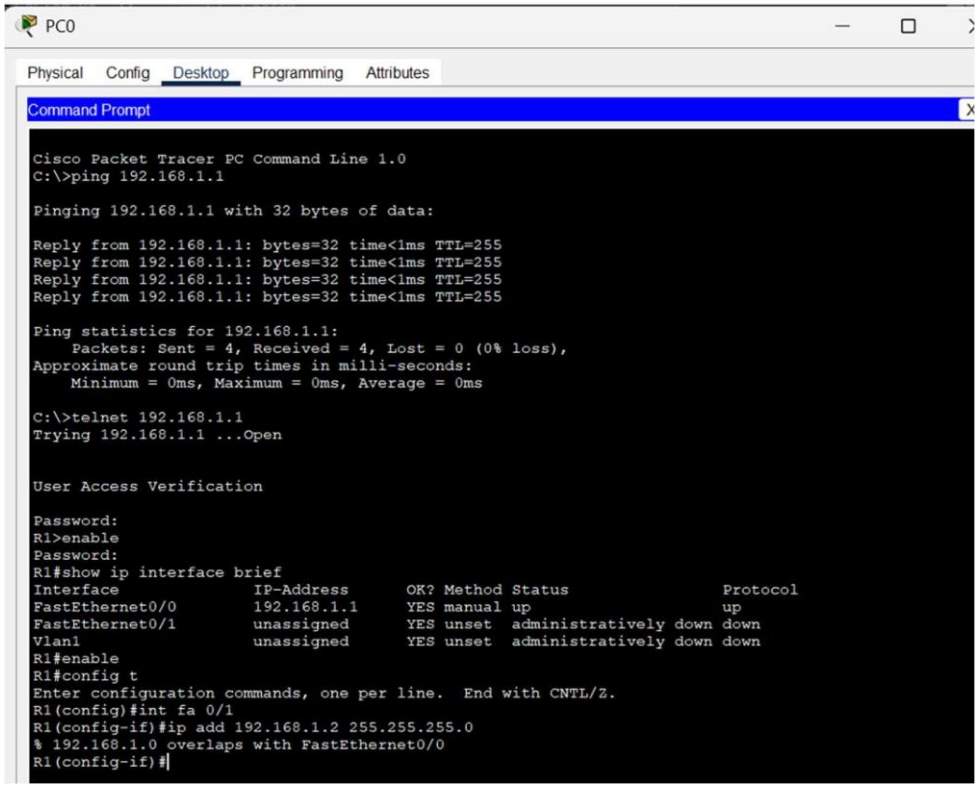


Fig2. PC Command Line Prompt

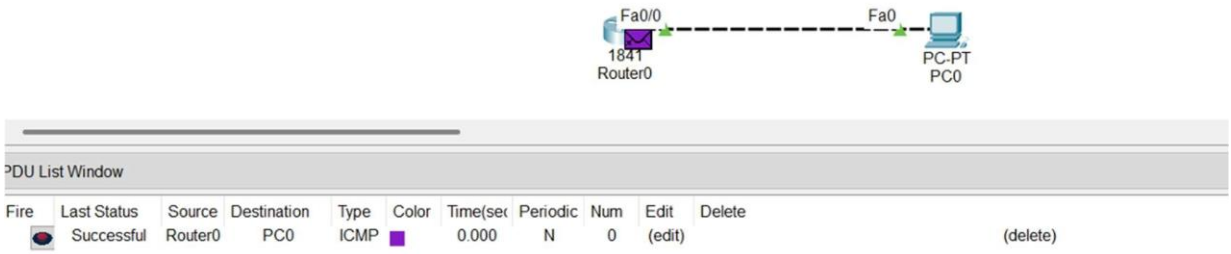
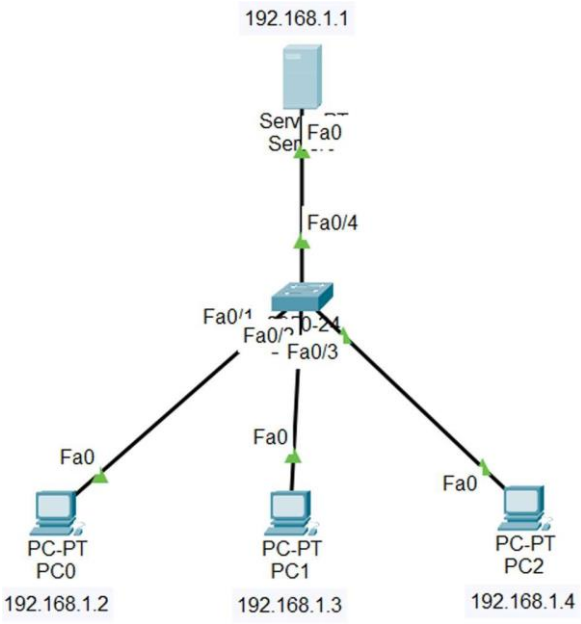


Fig3. PDU Message successful

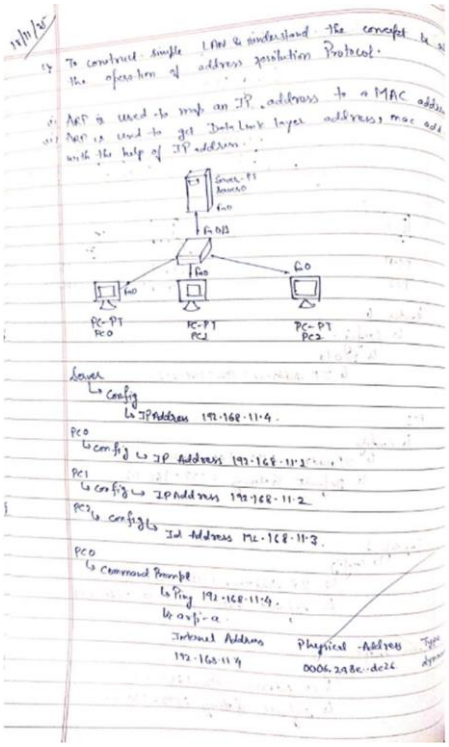
Program 12:

Aim: To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP).

Network diagram:



Configuration:



:

ARP Table for Server0

IP Address	Hardware Address	Interface
192.168.1.2	00E0.F736.0126	FastEthernet0
192.168.1.3	0090.0C24.1CCC	FastEthernet0
192.168.1.4	00D0.D396.D2B5	FastEthernet0

Fig 1.1 ARP table at Server0

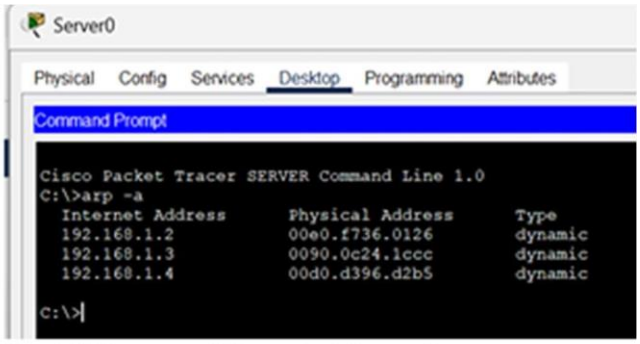


Fig 1.2 Command Prompt at Server0

ARP Table for PC0

IP Address	Hardware Address	Interface
192.168.1.1	00E0.F7C6.AC93	FastEthernet0

Fig 2.1 ARP table at PC0

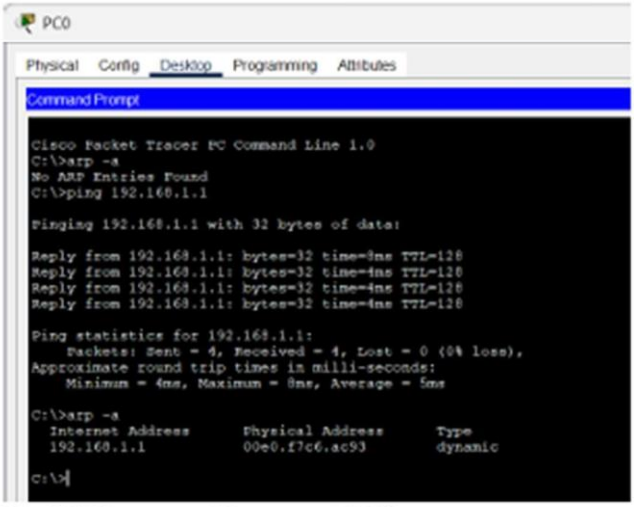


Fig 2.2 Command Prompt at PC0

ARP Table for PC1

IP Address	Hardware Address	Interface
192.168.1.1	00E0.F7C6.AC93	FastEthernet0

Fig 3.1 ARP table at PC1

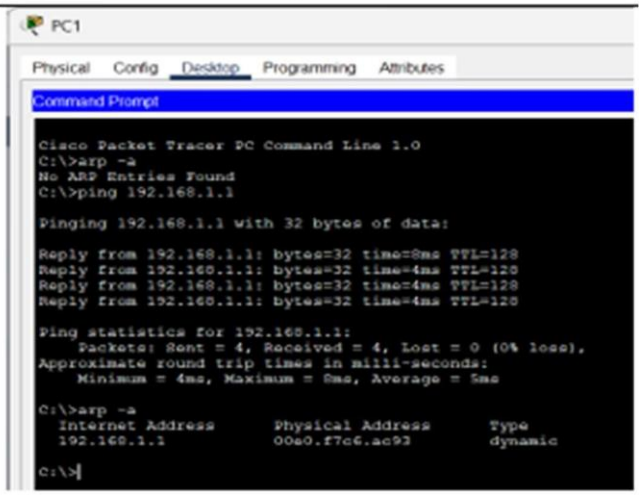


Fig 3.2 Command Prompt at PC1

PART – B Program

1:

Aim: Write a program for congestion control using Leaky bucket algorithm.

Code:

PART - B

Program 1:

Aim: Write a program for congestion control using Leaky bucket algorithm.

Code:

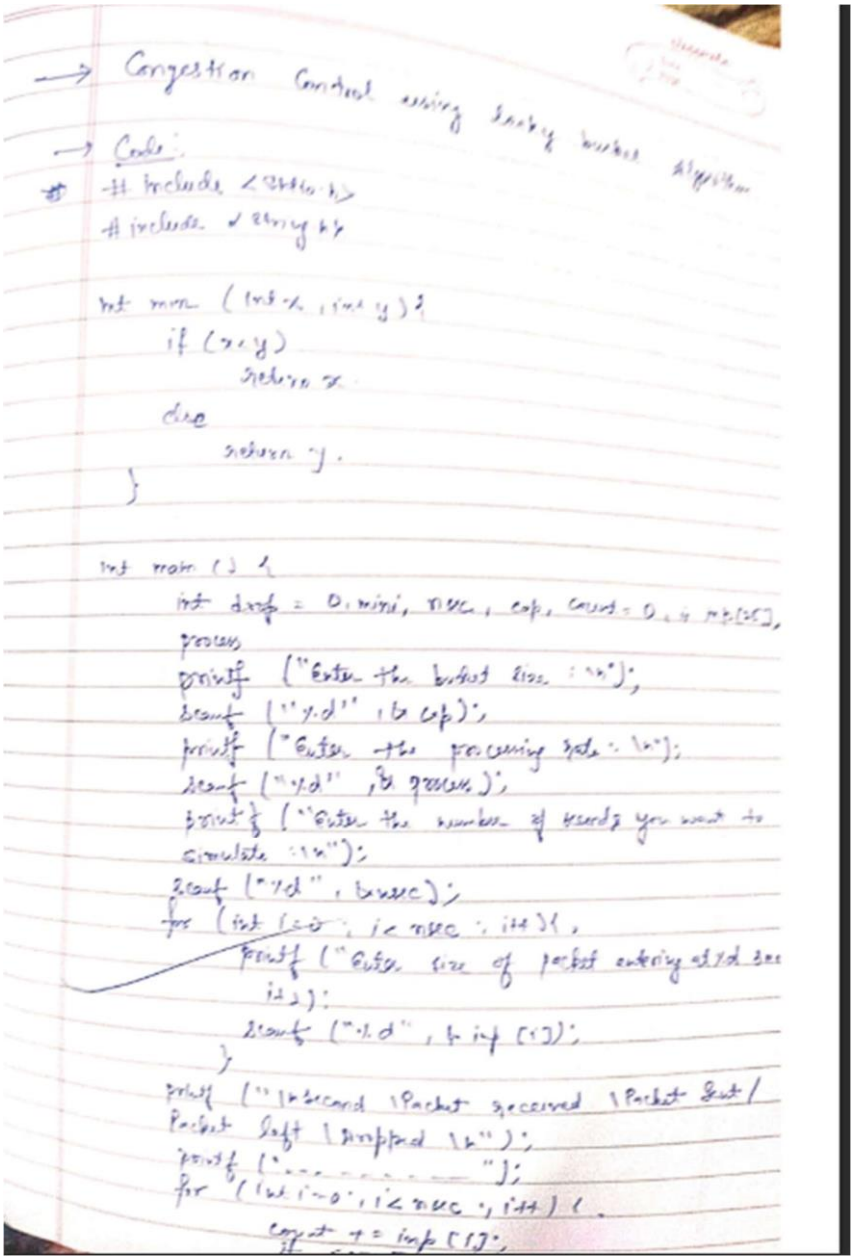
```
#include <stdio.h>

int min(int x, int y)
{     if (x < y)
return x;     else
return y;
}

int main() {     int drop = 0, mini, nsec, cap, count
= 0, i, inp[25],  process;

   printf("Enter the bucket size:\n");
scanf("%d", &cap);

   printf("Enter the processing rate:\n");
scanf("%d", &process);

   printf("Enter the number of seconds you want to
simulate:\n");
   scanf("%d", &nsec);

   for (i = 0; i < nsec; i++) {          printf("Enter the
size of the packet entering at %d  sec:\n", i + 1);
32

      scanf("%d", &inp[i]);
   }

   printf("\nSecond | Packet Received | Packet Sent | Packet
Left | Dropped\n");
   printf("-------------------------------------------------------------------\n");

   for (i = 0; i < nsec; i++) {
      count += inp[i];

      if (count > cap) {
drop = count - cap;
```

```
            count = cap;
        }

        printf("%d\t  %d\t\t", i + 1, inp[i]);

        mini = min(count, process);
        printf("%d\t\t", mini);

        count = count - mini;
        printf("%d\t\t %d\n", count, drop);

        drop = 0;
    }

    // Remaining packets after time
ends      for (; count != 0; i++) {
if (count > cap) {
33

            drop = count - cap;
            count = cap;
        }

        printf("%d\t  0\t\t", i + 1);

        mini = min(count, process);
        printf("%d\t\t", mini);

        count = count - mini;
        printf("%d\t\t %d\n", count, drop);

        drop = 0;
    }

    return 0;
}
```

OUTPUT:

```
Enter the bucket size:
8
Enter the processing rate:
2
Enter the number of seconds you want to simulate:
5
Enter the size of the packet entering at 1 sec:
3
Enter the size of the packet entering at 2 sec:
1
Enter the size of the packet entering at 3 sec:
4
Enter the size of the packet entering at 4 sec:
3
Enter the size of the packet entering at 5 sec:
5

Second | Packet Received | Packet Sent | Packet Left | Dropped
-----------------------------------------------------------------
1      3      2       1       0
2      1      2       0       0
3      4      2       2       0
4      3      2       3       0
5      5      2       6       0
6      0      2       4       0
7      0      2       2       0
```

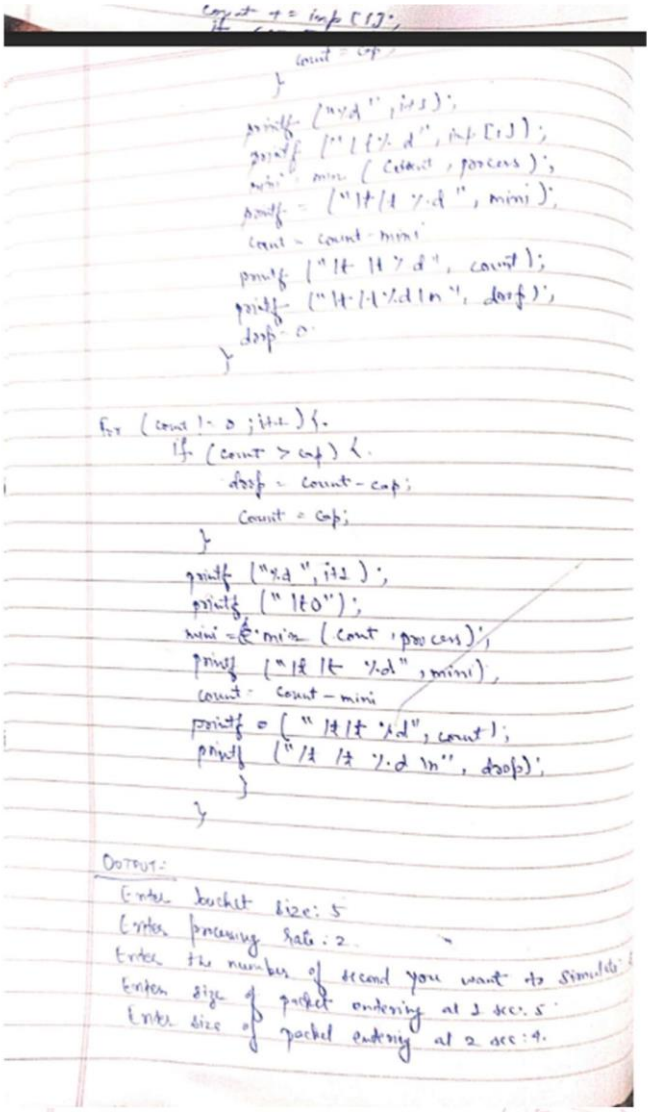
Observation:



→ Congestion Control using Leaky bucket Algorithm

→ Code:

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

int min (int x, int y) {
    if (x < y)
        return x;
    else
        return y;
}

int main () {
    int drop = 0, mini, nsec, cap, count = 0, i, inp[25],
    process;
    printf ("Enter the bucket size : \n");
    scanf ("%d", &cap);
    printf ("Enter the processing rate : \n");
    scanf ("%d", &process);
    printf ("Enter the number of seconds you want to
    simulate : \n");
    scanf ("%d", &nsec);
    for (int i = 0; i < nsec; i++) {
        printf ("Enter size of packet entering at %d sec
        i+1);
        scanf ("%d", &inp[i]);
    }
    printf ("\nSecond | Packet received | Packet sent |
    Packet left | Dropped \n");
    printf ("-----------");
    for (int i = 0; i < nsec; i++) {
        count += inp[i];
```



```
        count = cap;
    }
    printf("%d", i+1);
    printf("\t%d", inp[i]);
    mini = min(count, process);
    printf("\t\t%d", mini);
    count = count - mini;
    printf("\t\t%d", count);
    printf("\t\t%d\n", drop);
    drop = 0;
}

for (count = 0; i++;) {
    if (count > cap) {
        drop = count - cap;
        count = cap;
    }
    printf("%d", i+1);
    printf("\t0");
    mini = min(count, process);
    printf("\t\t%d", mini);
    count = count - mini;
    printf("\t\t%d", count);
    printf("\t\t%d\n", drop);
    }
}
```

OUTPUT:

```
Enter bucket size: 5
Enter processing rate: 2
Enter the number of second you want to simulate:
Enter size of packet entering at 1 sec: 5
Enter size of packet entering at 2 sec: 4
Enter size of packet entering at 3 sec: 3
```

Second	Packet Received	Packet Sent	Packet left	Dropped
1	5	2	3	0
2	4	2	3	2
3	3	2	3	1
4	0	2	1	0
5	0	1	0	0

Program 2:

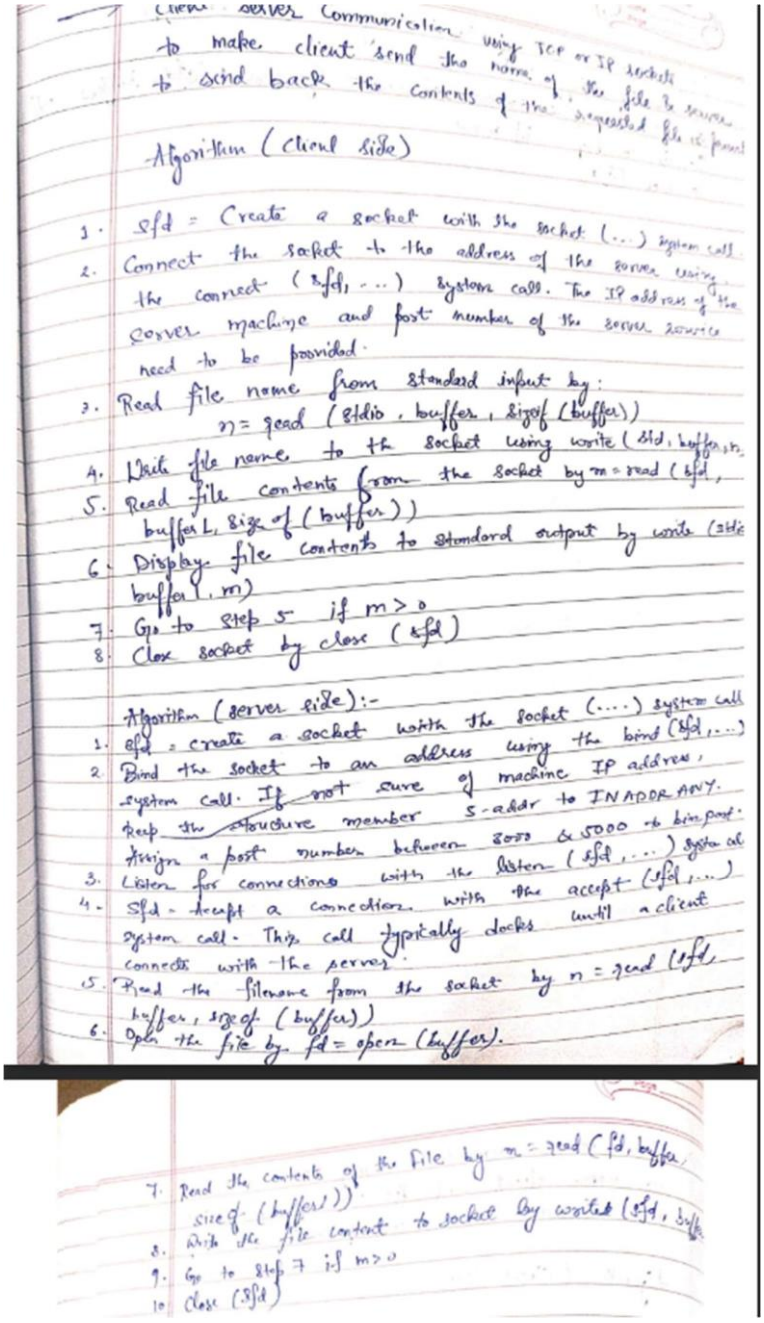
Aim: Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

Code:

```
# tcp_client.py


import socket


# Step 1: Create TCP socket
client_socket =
socket.socket(socket.AF_INET,
socket.SOCK_STREAM)


# Step 2: Connect to server
client_socket.connect(('localhost',
8080))


# Step 3: Send filename
filename = input("Enter filename to
request: ")


client_socket.send(filename.encode())


# Step 4: Receive file contents
data =
client_socket.recv(4096).decode()


print("\n--- File Content ---\n")
print(data)


# Step 5: Close connection
client_socket.close()
```

```
# tcp_server.py

import socket

# Step 1: Create a TCP socket
server_socket =
socket.socket(socket.AF_INET,
socket.SOCK_STREAM)

# Step 2: Bind to address and port
server_socket.bind(('localhost',
8080))

# Step 3: Listen for client
connections
server_socket.listen(1)
print("Server is listening on port
8080...")

# Step 4: Accept connection
conn, addr = server_socket.accept()
print("Connected by:", addr)

# Step 5: Receive file name
filename =
conn.recv(1024).decode().strip()

try:
    # Step 6: Open and read file
    with open(filename, 'r') as f:
        data = f.read()

    conn.send(data.encode())  # Send
file contents

except FileNotFoundError:
    conn.send(b"File not found on
server.")

# Step 7: Close connection
conn.close()
server_socket.close()
```

Observation:



Program 3:

Aim: 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:

```
# udp_client.py


import socket


# Step 1: Create UDP socket

client_socket =
socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)


server_address = ('localhost',
8081)


filename = input("Enter filename
to request: ")


# Step 2: Send filename to
server

client_socket.sendto(filename.en
code(), server_address)


# Step 3: Receive response

data, addr =
client_socket.recvfrom(4096)


print("\n--- File Content ---
\n")

print(data.decode())


# Step 4: Close socket

client_socket.close()
```

```
# udp_server.py

import socket

# Step 1: Create UDP socket
server_socket =
socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)

# Step 2: Bind to address and port
server_socket.bind(('localhost',
8081))

print("UDP Server is ready...")

while True:
    # Step 3: Receive filename
from client
    filename, addr =
server_socket.recvfrom(1024)
    filename =
filename.decode().strip()

    print(f"Requested file:
{filename}")

    try:
        # Step 4: Open file and
send content
        with open(filename, 'r')
as f:
            data = f.read()

        server_socket.sendto(data.
encode(), addr)

    except FileNotFoundError:
        server_socket.sendto(b"Fil
e not found on server.", addr)
```

Program 4:

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

Code:

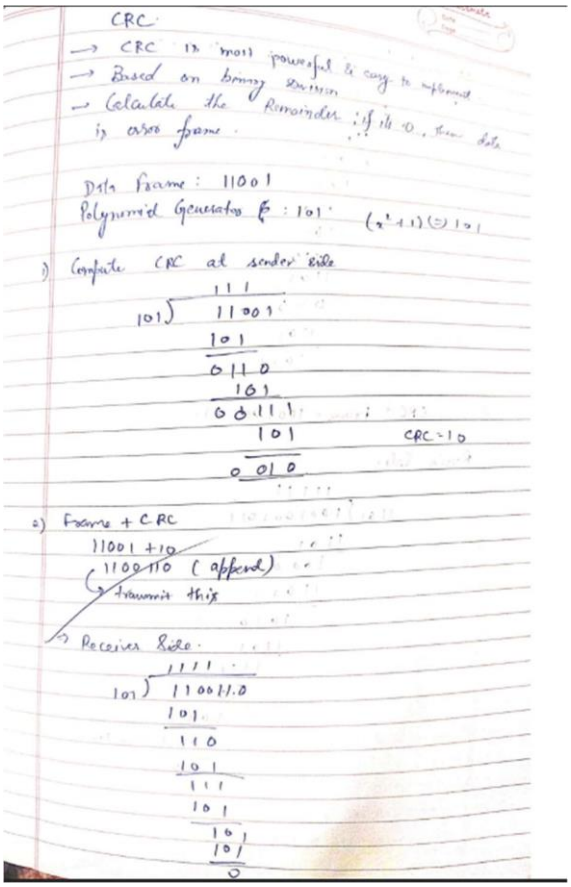
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>


int main() {      char rem[50], a[50], s[50], c,
msj[50], gen[30];
   int i, genlen, t, j, flag = 0, k, n;

   printf("Enter the generation polynomial:\n");
   gets(gen);      printf("Generator polynomial is CRC-
CCITT: %s\n", gen);

   genlen = strlen(gen);
   k = genlen - 1;

   printf("Enter                    the
message:\n");      n = 0;      while
((c = getchar()) != '\n') {
       msj[n] =
c;          n++;
}      msj[n] =
'\0';

   for (i = 0; i < n; i++)
      a[i] = msj[i];

   for (i = 0; i < k; i++)
a[n + i] = '0';
   a[n + k] = '\0';

   printf("\nMessage polynomial appended with zeros:\n");
puts(a);

   for (i = 0; i < n; i++) {
if (a[i] == '1') {
          t = i;             for (j = 0;
j <= k; j++) {                if
(a[t] == gen[j])
a[t] = '0';               else
a[t] = '1';               t++;
          }
       }
   }

   for (i = 0; i < k; i++)
rem[i] = a[n + i];
   rem[k] = '\0';

   printf("Checksum (remainder):\n");
   puts(rem);
```

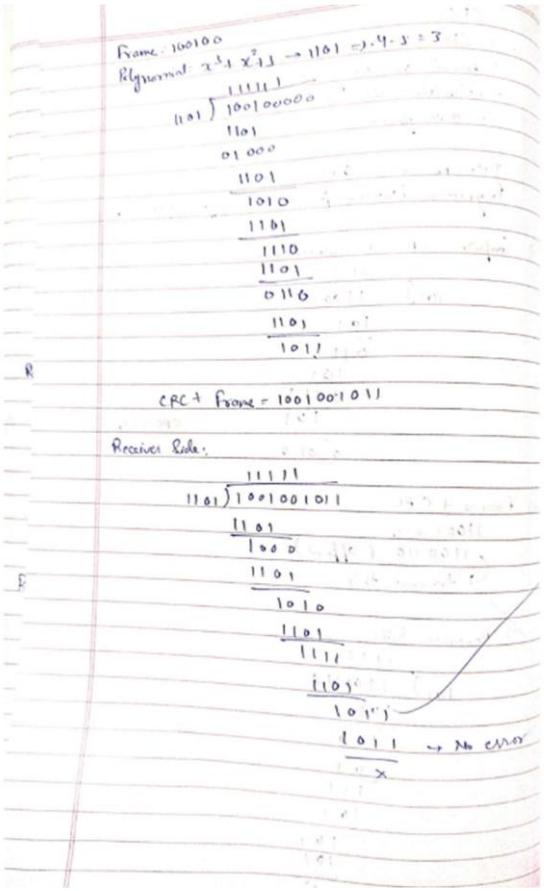
```
   printf("\nMessage with checksum
appended:\n");      for (i = 0; i < n; i++)          a[i] =
msj[i];
=
   for (i = 0; i < k; i++)
a[n + i] = rem[i];      a[n +
k] = '\0';
   puts(a);

   n = 0;
   printf("Enter the received message:\n");
while ((c = getchar()) != '\n') {
       s[n] =
c;          n++;
}      s[n] =
'\0';

   for (i = 0; i < n; i++) {
if (s[i] == '1') {
           t = i;              for (j = 0; j <=
k; j++, t++) {                  if (s[t] ==
gen[j])                     s[t] = '0';
else                    s[t] = '1';
           }
       }
   }

   for (i = 0; i < k; i++)
rem[i] = s[n + i];
   rem[k] = '\0';

   for (i = 0; i < k; i++) {
if (rem[i] == '1')
flag = 1;
   }

   if (flag == 0)          printf("Received polynomial
is error-free \n");      else          printf("Received
polynomial contains error \n");

   return 0;
}
```

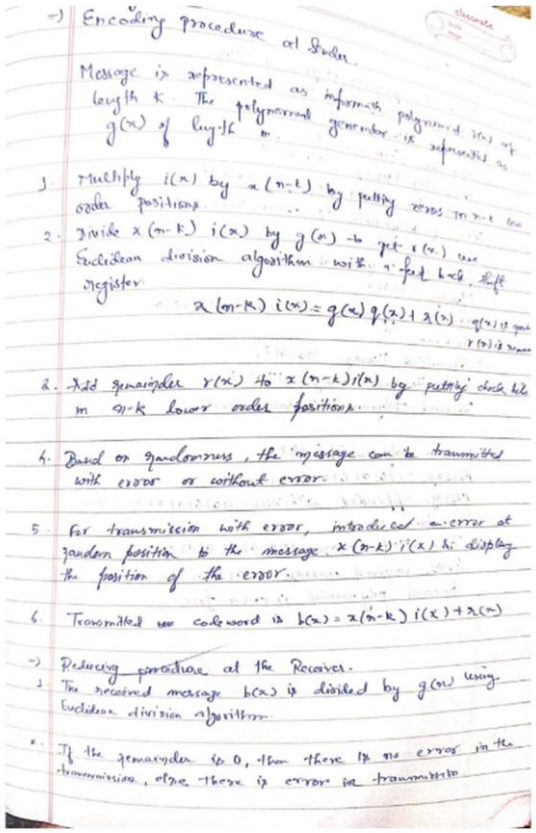
OUTPUT:

```
Enter the generation polynomial:
101
Generator polynomial is CRC-CCITT: 101
Enter the message:
1101010101010100

Message polynomial appended with zeros:
1101010101010100 00
Checksum (remainder):
11

Message with checksum appended:
1101010101010100 11
Enter the received message:
1101010101010100 11
Received polynomial is error-free

Process returned 0 (0x0)   execution time : 33.192 s
Press any key to continue.
```

Observation:







→ Encoding procedure at Sender.

Message is represented as information polynomial $i(x)$ of length k. The polynomial generator is represented as $g(x)$ of length n.

1. Multiply $i(x)$ by $x^{(n-k)}$ by putting zeros in $n-k$ low order positions.
2. Divide $x^{(n-k)} i(x)$ by $g(x)$ to get $r(x)$ using Euclidean division algorithm with $n-k$ bit shift register.

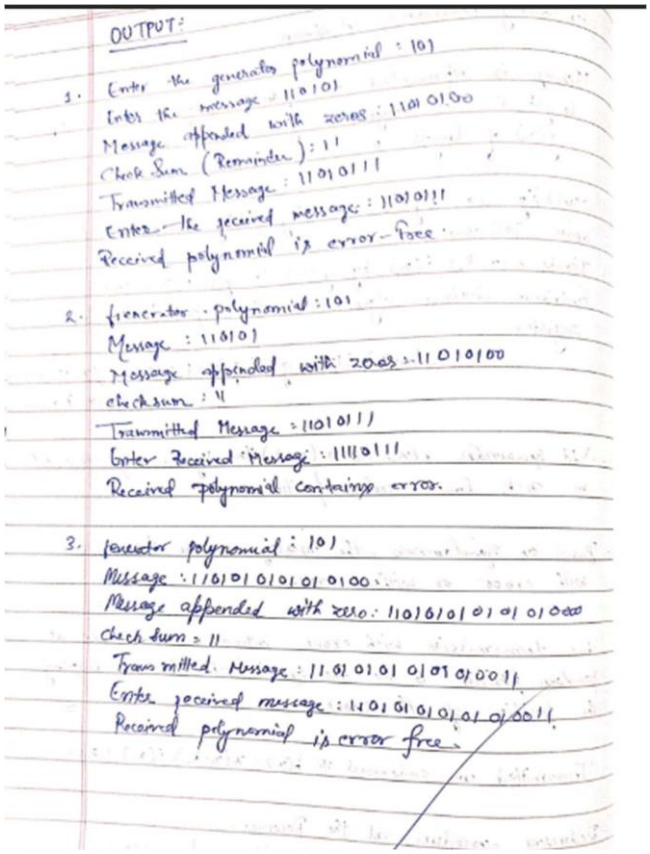
$$x^{(n-k)} i(x) = g(x) q(x) + r(x)$$

$q(x)$ is quotient, $r(x)$ is remainder

3. Add remainder $r(x)$ to $x^{(n-k)} i(x)$ by putting check bits in $n-k$ lower order positions.
4. Based on randomness, the message can be transmitted with error or without error.
5. For transmission with error, introduced an error at random position to the message $x^{(n-k)} i(x)$ & display the position of the error.
6. Transmitted use codeword is $b(x) = x^{(n-k)} i(x) + r(x)$

→ Reducing procedure at the Receiver.

1. The received message $b(x)$ is divided by $g(x)$ using Euclidean division algorithm.
2. If the remainder is 0, then there is no errors in the transmission, else there is error in transmitted



OUTPUT:

1. Enter the generator polynomial : 101
   Enter the message : 110101
   Message appended with zeros : 11010100
   Check Sum (Remainder) : 11
   Transmitted Message : 11010111
   Enter the received message : 11010111
   Received polynomial is error-free.

2. Generator polynomial : 101
   Message : 110101
   Message appended with zeros : 11010100
   checksum : 11
   Transmitted Message : 11010111
   Enter Received Message : 11110111
   Received polynomial contains error.

3. Generator polynomial : 101
   Message : 1101010101010100
   Message appended with zero : 11010101010101000
   check sum : 11
   Transmitted Message : 1101010101010011
   Enter received message : 1101010101010011
   Received polynomial is error free.