VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB RECORD

Computer Network Lab (23CS5PCCON)

Submitted by

Ankit Singh Bhatti (1BM22CS353)

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
Academic Year 2024-25 (odd)

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 **Ankit Singh Bhatti (1BM22CS353),** 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.

Prof. Sneha N P Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

Sl. No.	Date	Experiment Title	Page No.
1	4-10-24	Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.	1-4
2	18-10-24	Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply	5-9
3	25-10-24	Configure default route, static route to the Router	10-15
4	8-11-24	Configure DHCP within a LAN and outside LAN.	16-19
5	22-11-24	Configure RIP routing Protocol in Routers	20-24
6	27-11-24	Configure OSPF routing protocol	25-27
7	22-11-24	Demonstrate the TTL/ Life of a Packet	28-31
8	8-11-24	Configure Web Server, DNS within a LAN.	32-34
9	20-12-24	To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)	35-38
10	20-12-24	To understand the operation of TELNET by accessing the router in server room from a PC in IT office.	39-42
11	20-12-24	To construct a VLAN and make the PC's communicate among a VLAN	43-47
12	20-12-24	To construct a WLAN and make the nodes communicate wirelessly	48-52
		CYCLE - 2	
1	15-11-24	Write a program for error detecting code using CRC-CCITT (16-bits).	53-55
2	15-11-24	Write a program for congestion control using Leaky bucket algorithm.	56-58
3	20-12-24	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.	59-61

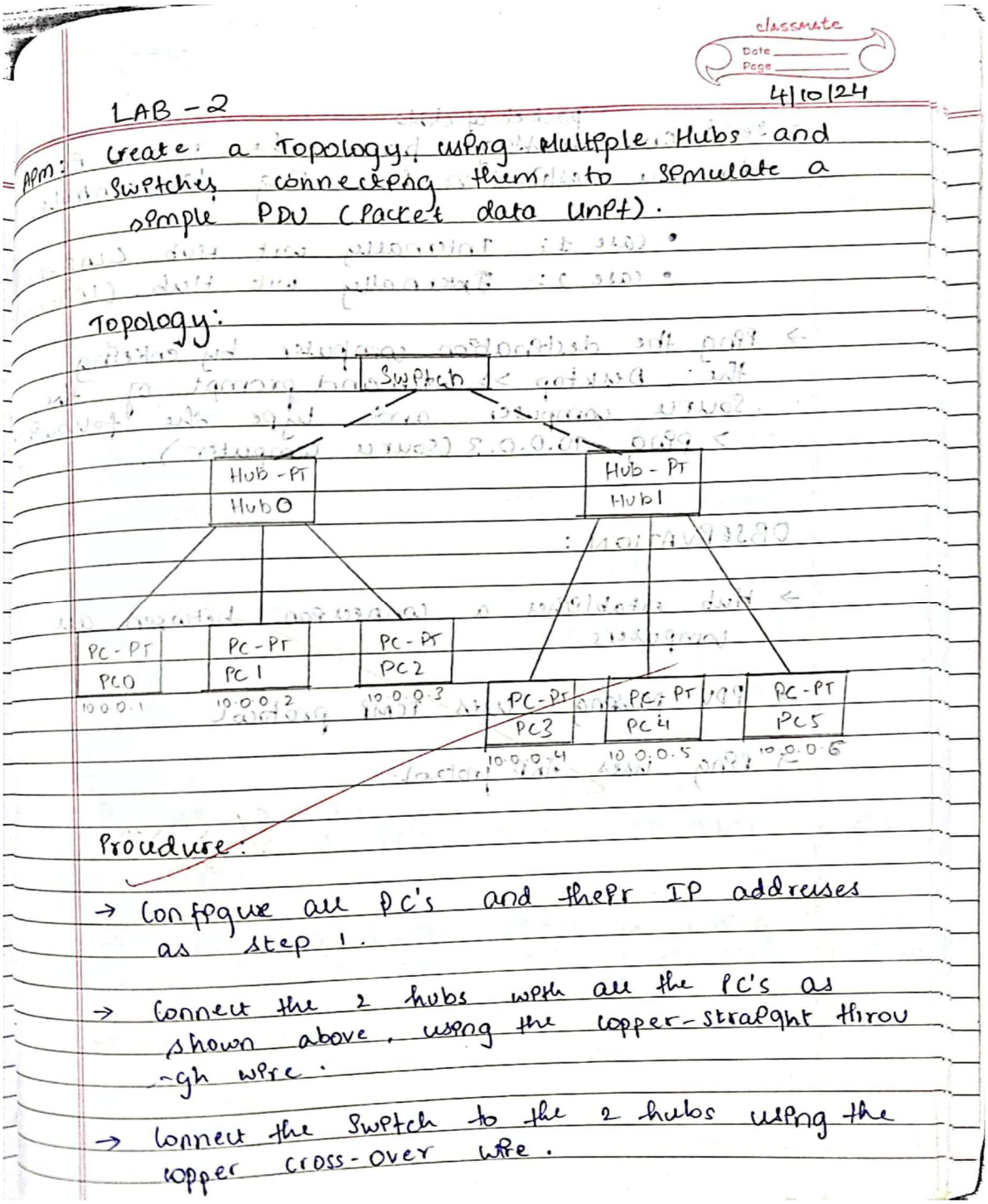
4	20-12-24	Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.	62-64

Github Link: https://github.com/AnkitSB19/CNSLAB

Program 1

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

ii. Procedure along with the topology:



LAB - 2

4/10/24

Aim: Create a Topology using Multiple Hubs and Switches connecting them to simulate a simple PDU (Packet data unit).

Topology:

Procedure:

→ Configure all PC's and their IP addresses as step 1.

→ Connect the 2 hubs with all the PC's as shown above, using the copper-straight through wire.

→ Connect the Switch to the 2 hubs using the copper cross-over wire.

→ Send one packet of data from the source PC to the destination PC using play controls.

- Case 1: Internally wrt Hub (LAB-2-1)
- Case 2: Externally wrt Hub. (LAB-2-2)

→ Ping the destination computer by entering the Desktop >> command prompt of the source computer and type the following:
> ping 10.0.0.3 (source computer).

OBSERVATION:

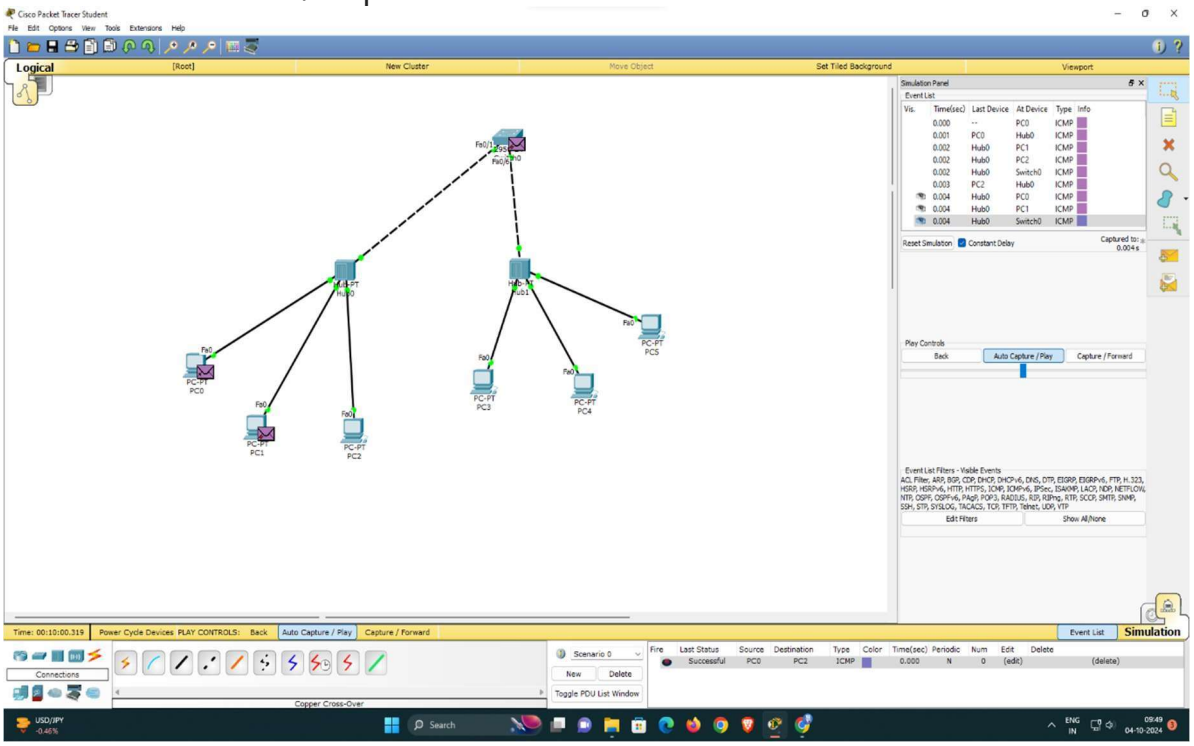
→ Hub establishes a connection between all computers

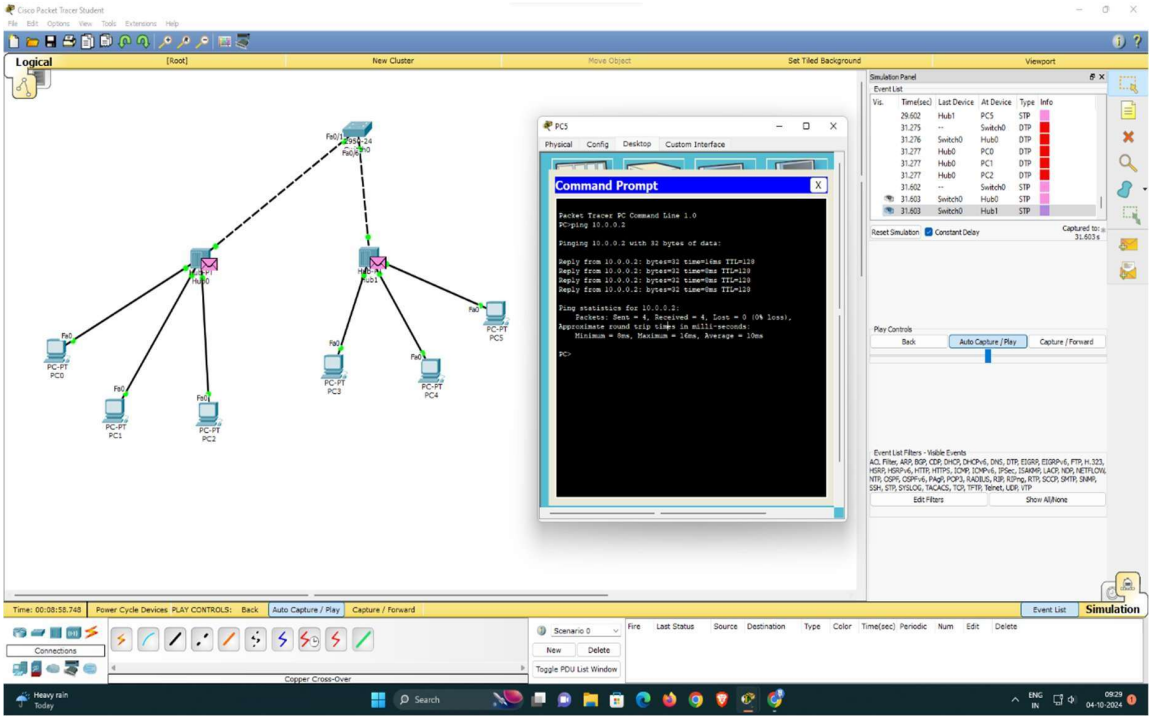
→ PDU passing uses ICMP protocol

→ Ping uses ARP protocol.

31/10/24

iii. Screen shots/ output:





iv. Observation:

→ Send one packet of data from the sourc PC to the destination PC using play controls.

- Case 1: Internally wrt Hub (LAB-2-1)
- Case 2: Externally wrt Hub. (LAB-2-2)

→ Ping the destination computer by entering the Desktop >> Command prompt of the Source computer and type the following:
> ping 10.0.0.3 (Source computer).

OBSERVATION:

→ Hub establishes a connection between all computers

→ PDU passing uses ICMP protocol

→ Ping uses ARP protocol.

Program 2

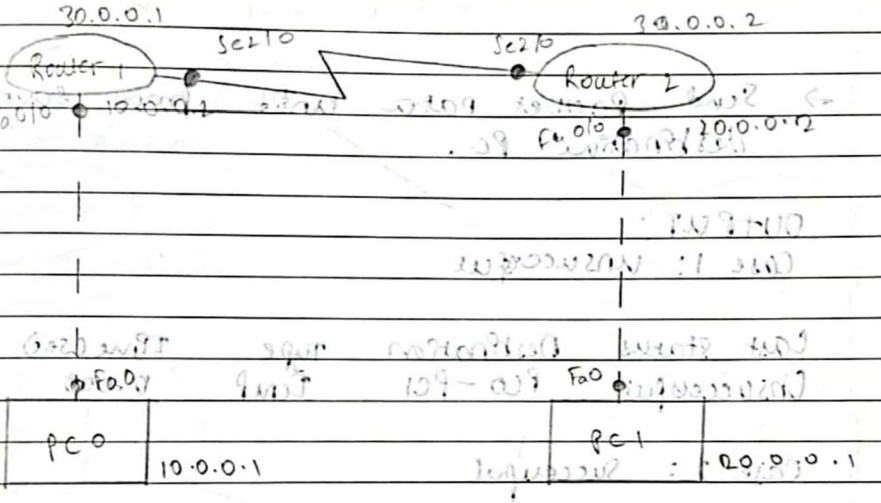
i. Aim of the program: Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.

ii. Procedure along with the topology:

LAB - 3

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

Topology:



Procedure:
→ Configure 2 PC's and their IP addresses as step 1.

→ Configure 2 Routers either by using the CLI or by the config tab for the each of them

→ Click on the 'ON' button on the top right to enable connections.

→ Configure the router's gateway P.e. (10.0.0.2)

→ For both the routers, click on the CLI and type 'show ip route' to check connection.

→ Gateways for end PC should be

→ After 'show ip route', all the different network connections are displayed.

→ To connect one router to the other network, type 'ip route 20.0.0.0 255.0.0.0 30.0.0.2'

20.0.0.0 → Network IP
255.0.0.0 → subnet mask
30.0.0.2 → other networks' IP address

→ Send Packet data unit from source to destination PC.

OUTPUT:

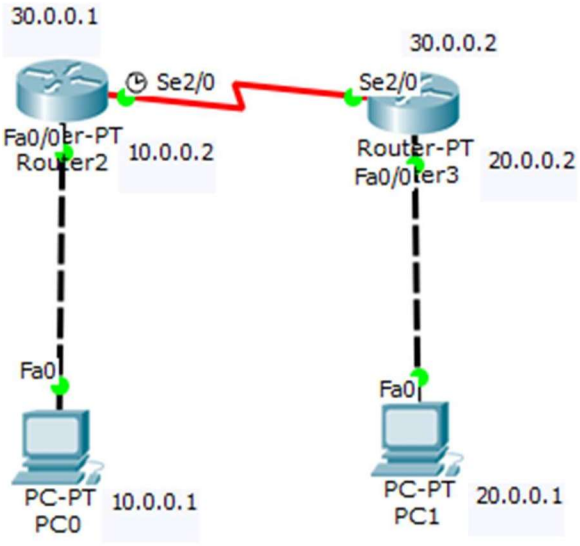
Case 1: Unsuccessful

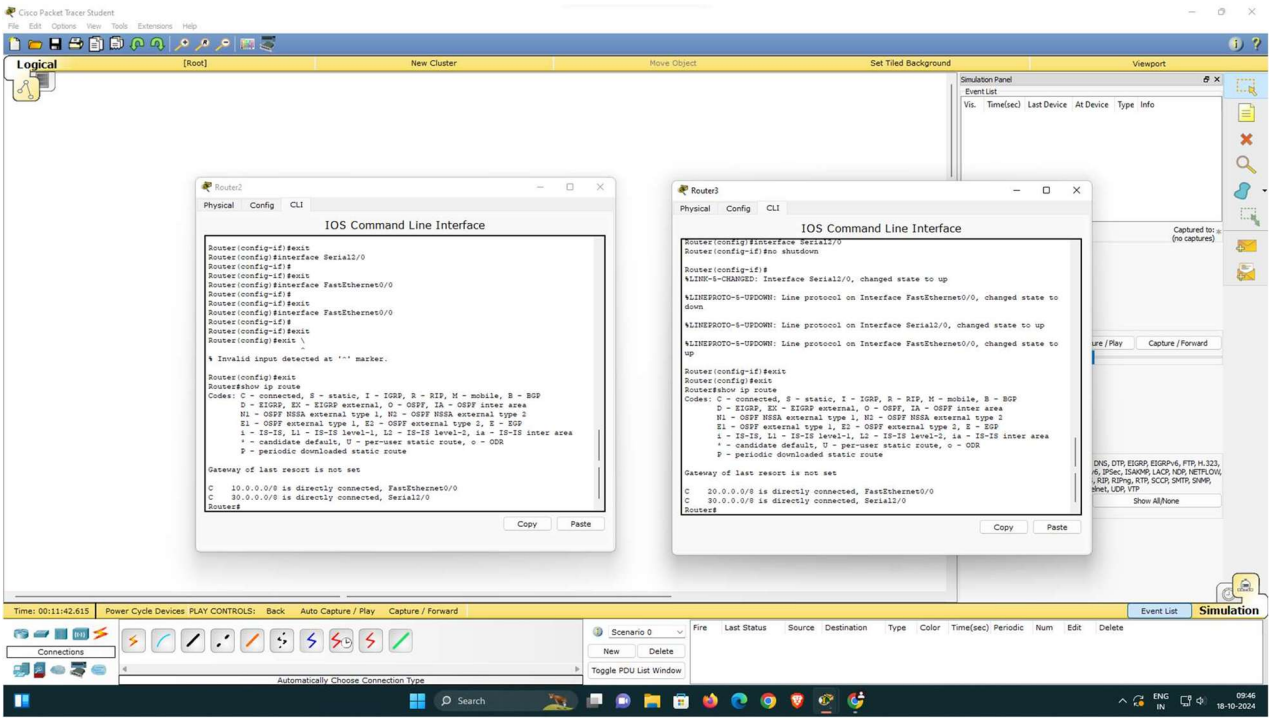
Last Status	Destination	Type	Time(sec)	Periodic
Unsuccessful	PC0 - PC1	ICMP	0.000	N

Case 2: Successful

Last Status	Destination	Type	Time(sec)	Periodic
Successful	PC0 - PC1	ICMP	0.000	N

iii. Screen shots/ output:





```
Router(config-if)#exit
Router(config)#exit
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

C    20.0.0.0/8 is directly connected, FastEthernet0/0
C    30.0.0.0/8 is directly connected, Serial2/0
Router#cocconfig t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#ip route 10.0.0.0 255.0.0.0 30.0.0.1
Router(config)#
Router(config)#
%SYS-5-CONFIG_I: Configured from console by console

Router(config)#exit
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

S    10.0.0.0/8 [1/0] via 30.0.0.1
C    20.0.0.0/8 is directly connected, FastEthernet0/0
C    30.0.0.0/8 is directly connected, Serial2/0
Router#
%SYS-5-CONFIG_I: Configured from console by console
```

iv. Observation:

→ After 'show ip route', all the different network connections are displayed.

→ To connect one router to the other network, type 'ip route 20.0.0.0 255.0.0.0 30.0.0.2'

20.0.0.0 — Network IP
255.0.0.0 — Subnet mask
30.0.0.2 — other networks' IP address

→ Send Packet data unit from source to destination PC.

OUTPUT:

Case 1: Unsuccessful

Last status	Destination	Type	Time(sec)	Periodic
Unsuccessful	PC0 - PC1	ICMP	0.000	N

Case 2: Successful

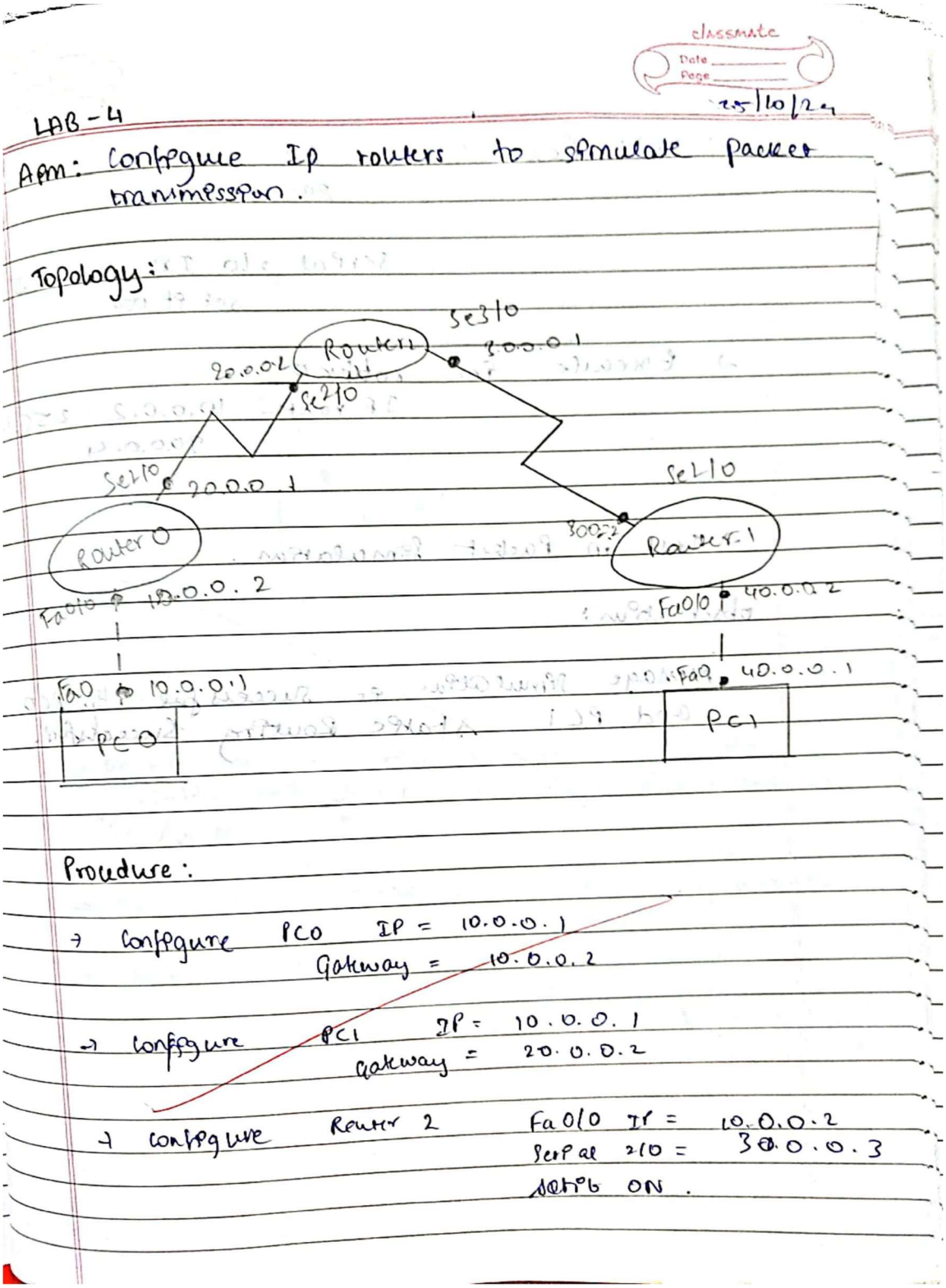
Last status	Destination	Type	Time(sec)	Periodic
Successful	PC0 - PC1	ICMP	0.0.0.1	N

13/10/24

Program 3:

i. Aim of the program: Configure default route, static route to the Router

ii. Procedure along with the topology



→ Config. Router 3 IP
20.0.0.2

Serial 2/0 IP. = 40.0.0.5
set pt on

→ Execute on router 0
IP route 10.0.0.2 255.0.0.0
30.0.0.4

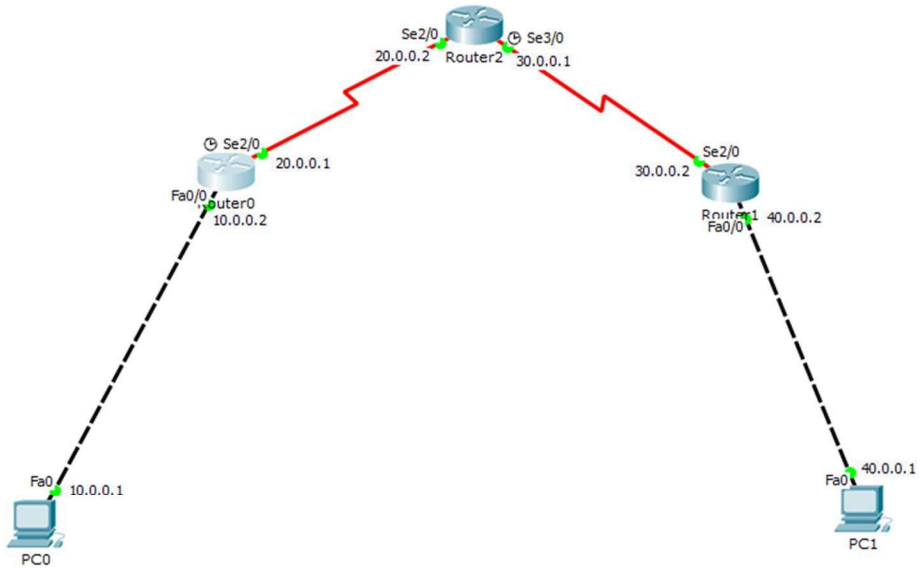
→ Test on Packet Simulation.

observation:

Message simulation is successful by PC0 and PC1 Static Routing Successful.

8/11/24

iii. Screen shots/ output:



Router1

Physical | Config | CLI

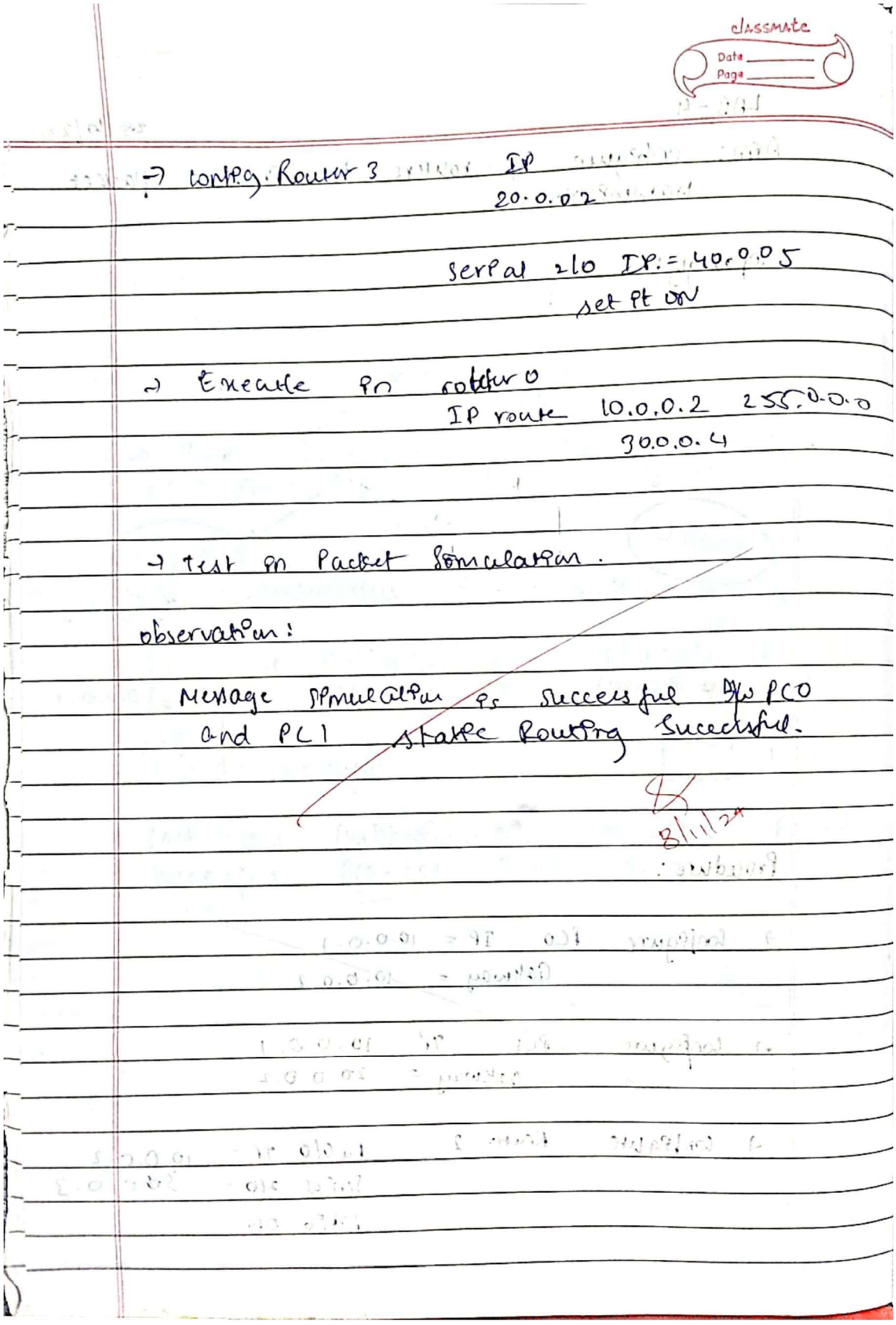
```
Router(config-if)#
%LINK-5-CHANGED: Interface Serial2/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to up

Router(config-if)#exit
Router(config)#interface FastEthernet1/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/0
Router(config-if)#ip address 40.0.0.2 255.0.0.0
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up

Router(config-if)#exit
Router(config)#interface Serial2/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial2/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial2/0
Router(config-if)#exit
Router(config)#exit
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

C    30.0.0.0/8 is directly connected, Serial2/0
C    40.0.0.0/8 is directly connected, FastEthernet0/0
Router#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#ip route 20.0.0.0 255.0.0.0 30.0.0.1
Router(config)#ip route 10.0.0.0 255.0.0.0 30.0.0.1
Router(config)#exit
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

S    10.0.0.0/8 [1/0] via 30.0.0.1
S    20.0.0.0/8 [1/0] via 30.0.0.1
C    30.0.0.0/8 is directly connected, Serial2/0
C    40.0.0.0/8 is directly connected, FastEthernet0/0
Router#
```

Router2

Physical | Config | CLI

```
                ^
% Invalid input detected at '^' marker.

Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
show ip oute
                ^
% Invalid input detected at '^' marker.

Router#show ip route '
Translating "'"...domain server (255.255.255.255)
% Invalid input detected
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

C    20.0.0.0/8 is directly connected, Serial2/0
C    30.0.0.0/8 is directly connected, Serial3/0
Router#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#ip route 10.0.0.0 255.0.0.0 20.0.0.1
Router(config)#exit
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

S    10.0.0.0/8 [1/0] via 20.0.0.1
C    20.0.0.0/8 is directly connected, Serial2/0
C    30.0.0.0/8 is directly connected, Serial3/0
Router#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#ip route 40.0.0.0 255.0.0.0 30.0.0.2
Router(config)#exit
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

S    10.0.0.0/8 [1/0] via 20.0.0.1
C    20.0.0.0/8 is directly connected, Serial2/0
C    30.0.0.0/8 is directly connected, Serial3/0
S    40.0.0.0/8 [1/0] via 30.0.0.2
```

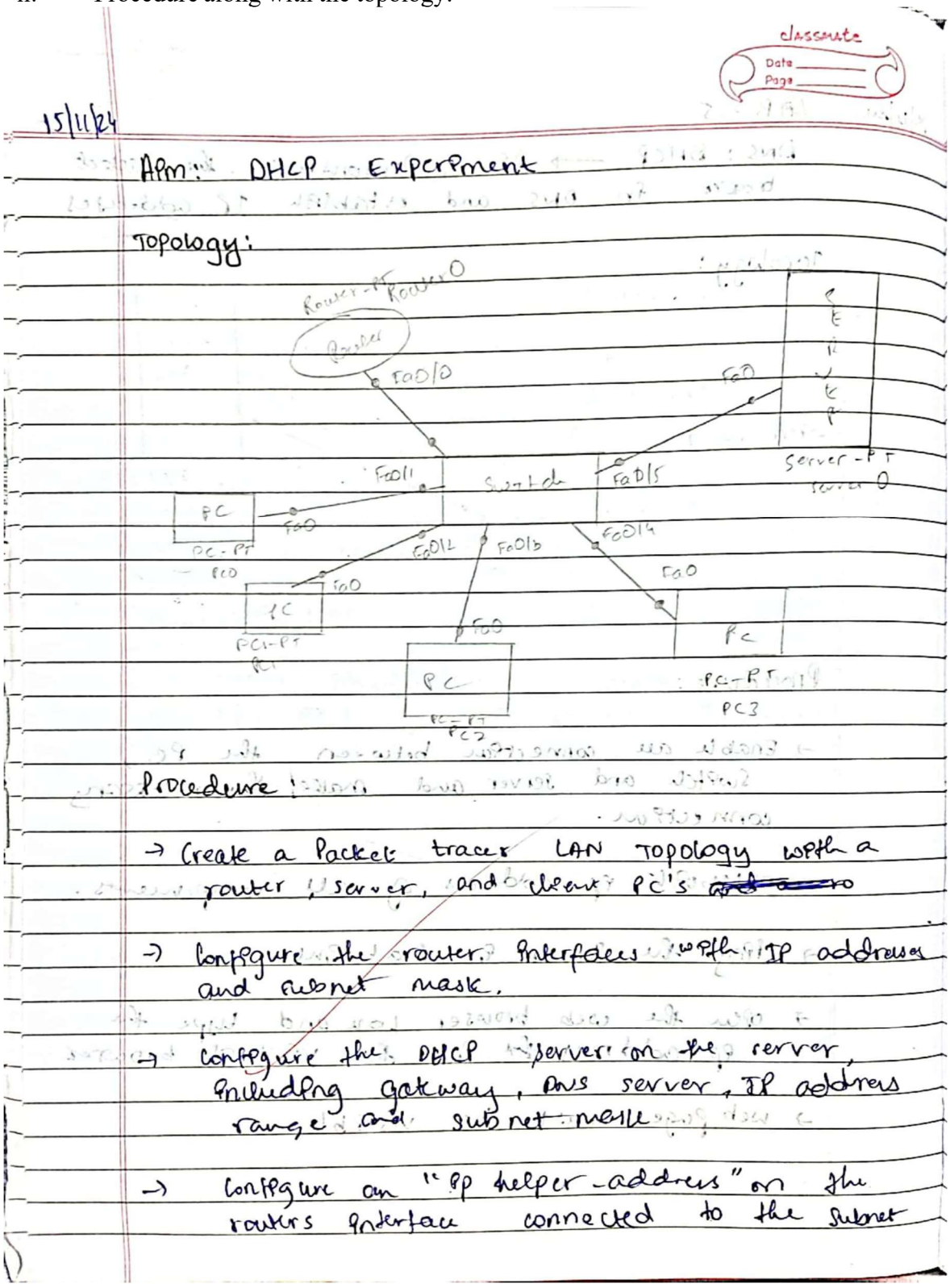
iv. Observation:



Program 4

i. Aim of the program: Configure DHCP within a LAN and outside LAN.

ii. Procedure along with the topology:



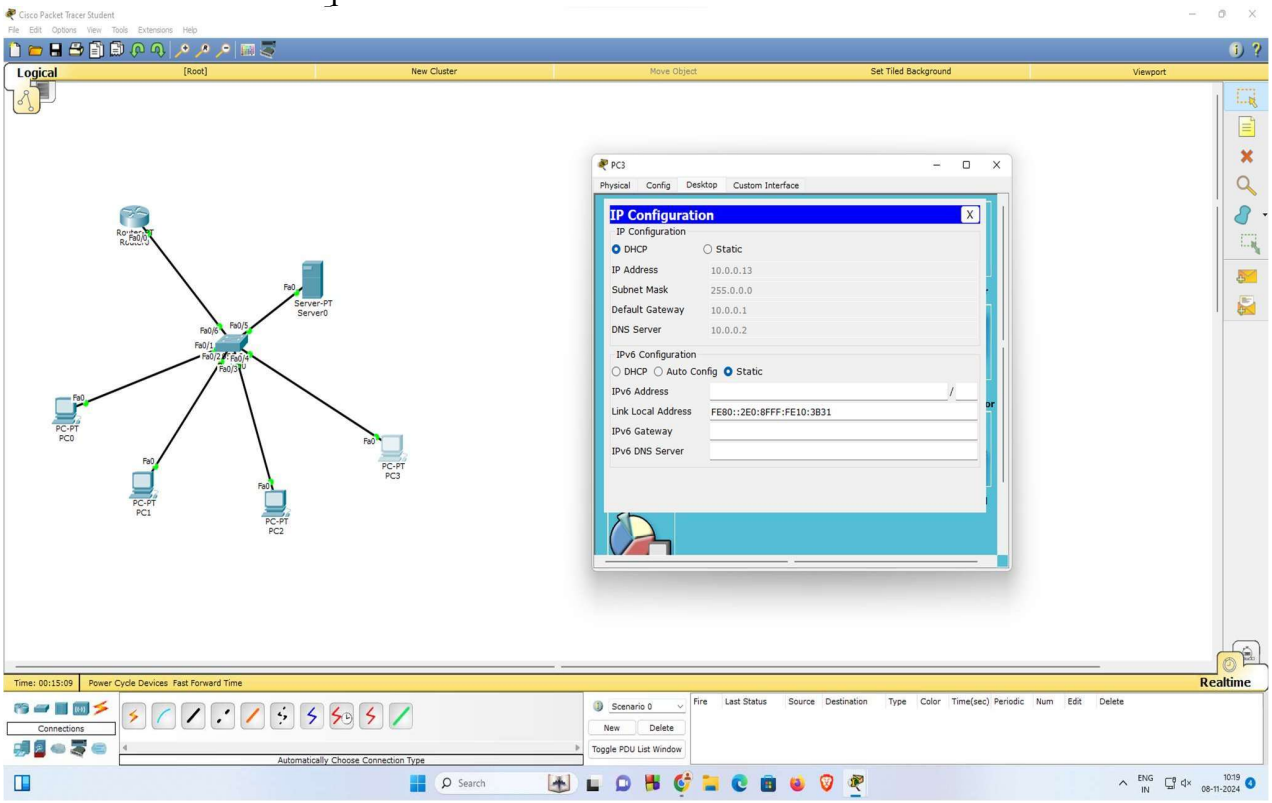
without a DHCP server, pointing to the DHCP server's IP address.

→ On client PCs, select "DHCP" on IP configuration to obtain IP addresses automatically.

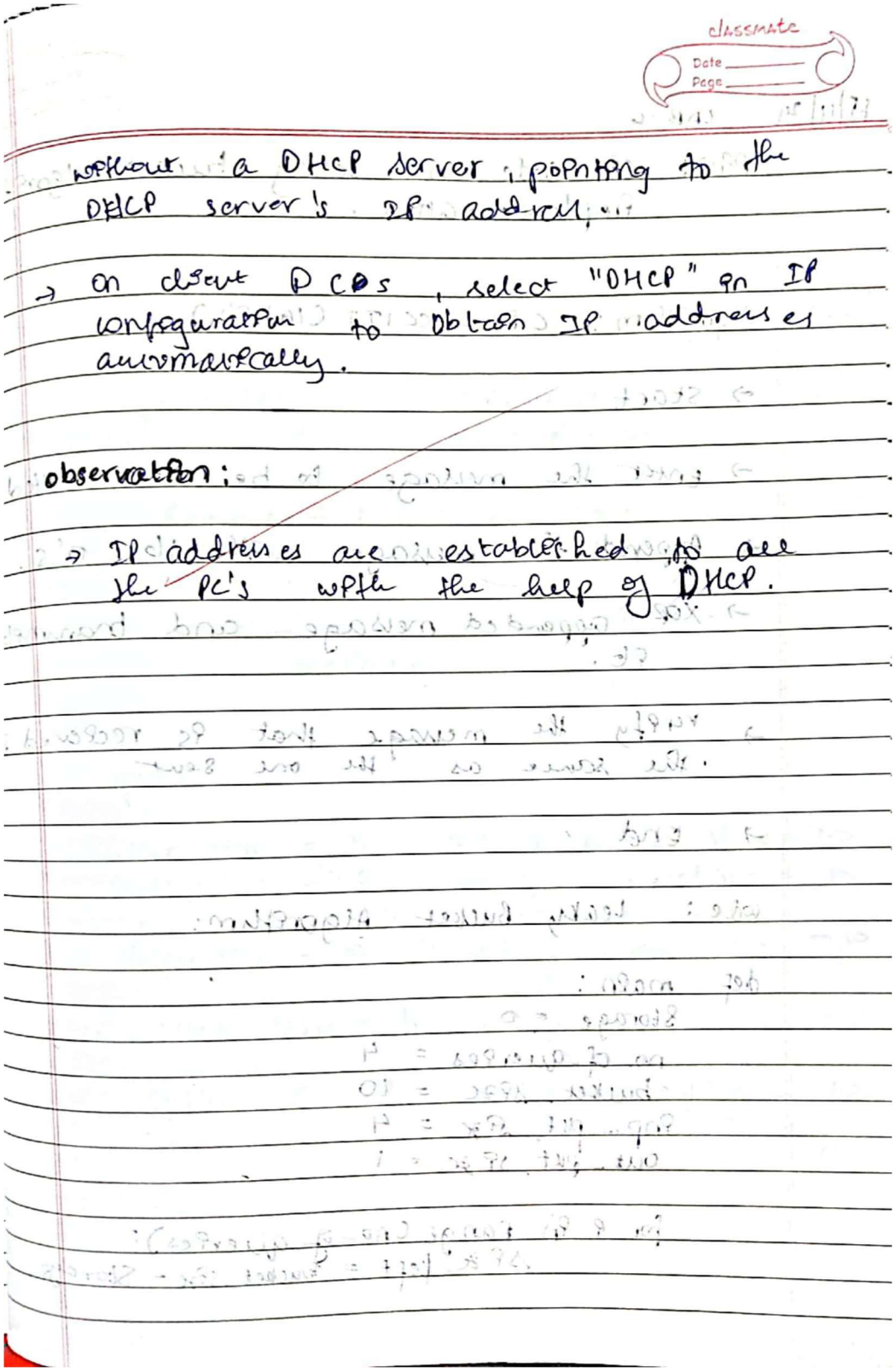
observation:

→ IP addresses are established to all the PC's with the help of DHCP.

iii. Screen shots/ output:



iv. Observation:



without a DHCP server, pointing to the DHCP server's IP address.

→ On client PCs, select "DHCP" on IP configuration to obtain IP addresses automatically.

observation:

→ IP addresses are established to all the PC's with the help of DHCP.

Program 5

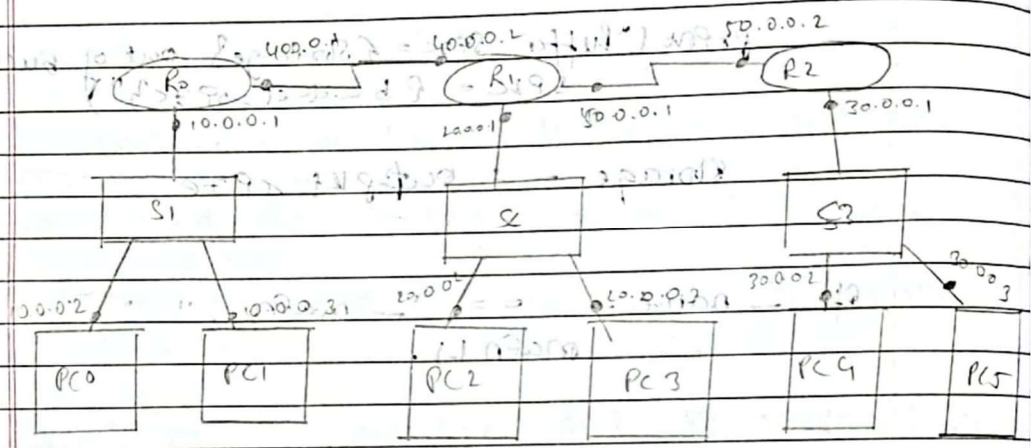
i. Aim of the program: Configure RIP routing Protocol in Routers

ii. Procedure along with the topology:

22/11/24 LAB - 7

Aim: Demonstrate RIP (Routing Information Protocol).

Topology:



Procedure:

→ Configure all end devices with respect to ip addresses.

→ Configure routers and their fast Ethernet ports and serial ports as shown in the figure.

→ Configure the gateways on end devices

→ Open the CLI of each router, open config Terminal and type 'router rip' after that type all the networks

to be connected

→ Simulate the packet transmission from one device to another device.

OBSERVATION:

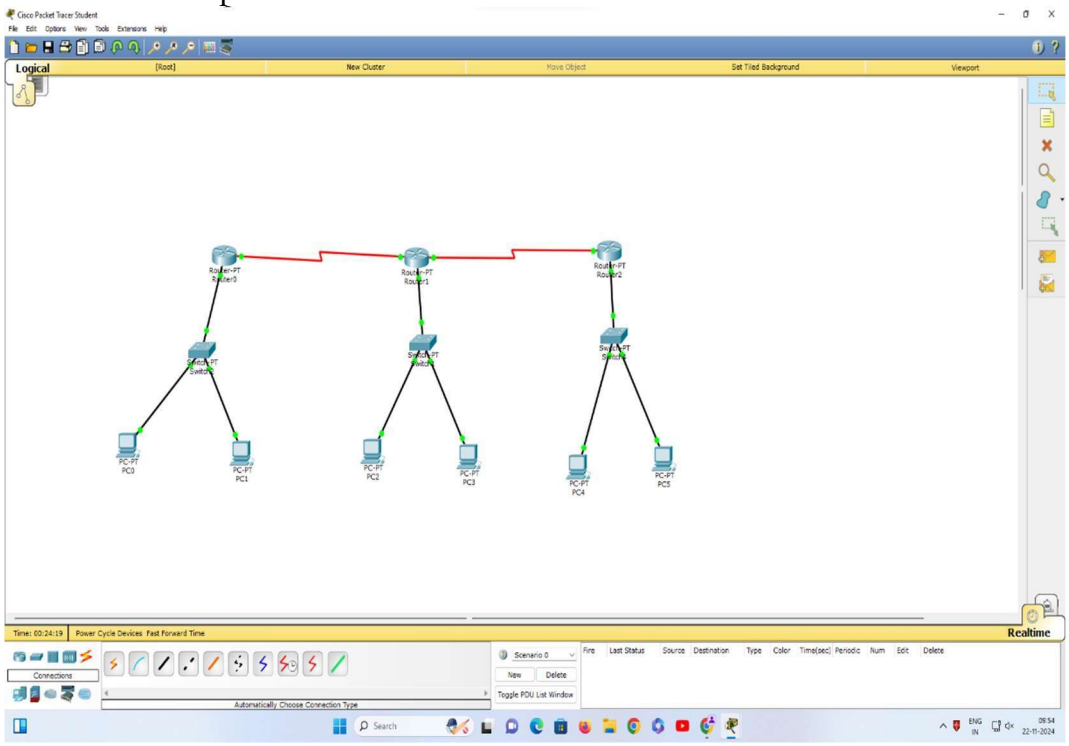
→ Packet transmission from source to destination is successful.

→ IP address are established.

→ TTL originally starts with a value 255

→ TTL decrements at each hop during Packet transmission.

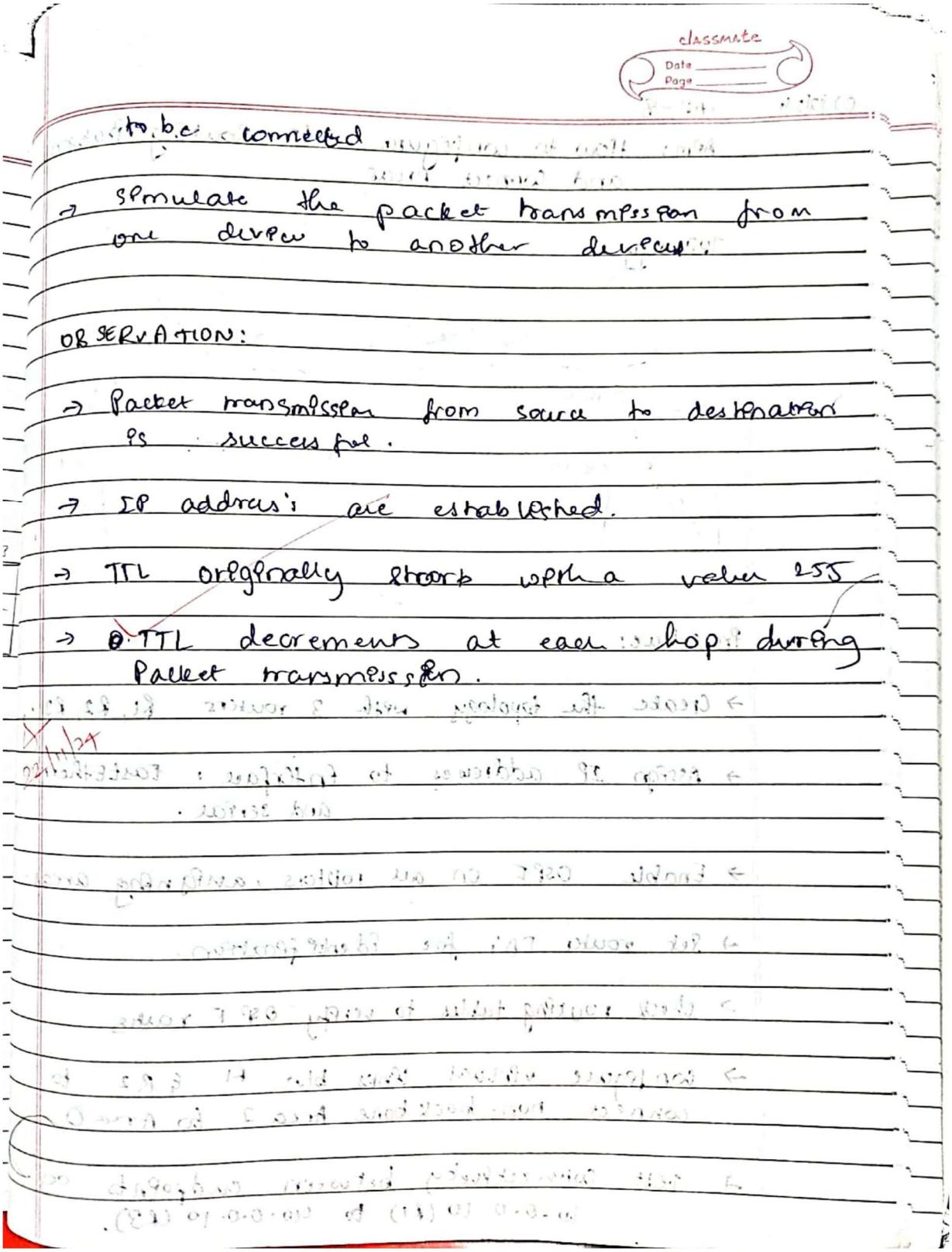
iii. Screen shots/ output:




```
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial3/0, changed state to up

Router(config-if)#exit
Router(config)#router rip
Router(config-router)#network 40.0.0.0
Router(config-router)#network 20.0.0.0
Router(config-router)#network 50.0.0.0
Router(config-router)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

R    10.0.0.0/8 [120/1] via 40.0.0.1, 00:00:17, Serial2/0
C    20.0.0.0/8 is directly connected, FastEthernet0/0
R    30.0.0.0/8 [120/1] via 50.0.0.2, 00:00:25, Serial3/0
C    40.0.0.0/8 is directly connected, Serial2/0
C    50.0.0.0/8 is directly connected, Serial3/0
Router#
```

```
% Unknown command or computer name, or unable to find computer address

Router#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#router rip
Router(config-router)#network 10.0.0.0
Router(config-router)#network 40.0.0.0
Router(config-router)#network 50.0.0.0
Router(config-router)#exit
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console
s
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

C    10.0.0.0/8 is directly connected, FastEthernet0/0
R    20.0.0.0/8 [120/1] via 40.0.0.2, 00:00:23, Serial2/0
R    30.0.0.0/8 [120/2] via 40.0.0.2, 00:00:15, Serial2/0
C    40.0.0.0/8 is directly connected, Serial2/0
R    50.0.0.0/8 [120/1] via 40.0.0.2, 00:00:23, Serial2/0
Router#
```

iv. Observation:



to be connected

→ Simulate the packet transmission from one device to another device.

OBSERVATION:

→ Packet transmission from source to destination is successful.

→ IP address are established.

→ TTL originally starts with a value 255

→ TTL decrements at each hop during Packet transmission.

Program 6

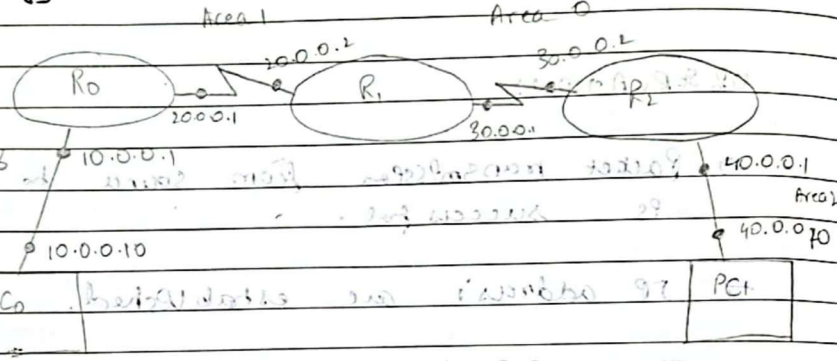
i. Aim of the program: Configure OSPF routing protocol

ii. Procedure along with the topology:

02/12/24 LAB-8

Aim: How to configure OSPF Routing Protocol and Connect Areas

Topology:



Procedure:

→ Create the topology with 3 routers R1, R2, R3.

→ Assign IP addresses to interface: FastEthernet and serial.

→ Enable OSPF on all routers, assigning areas.

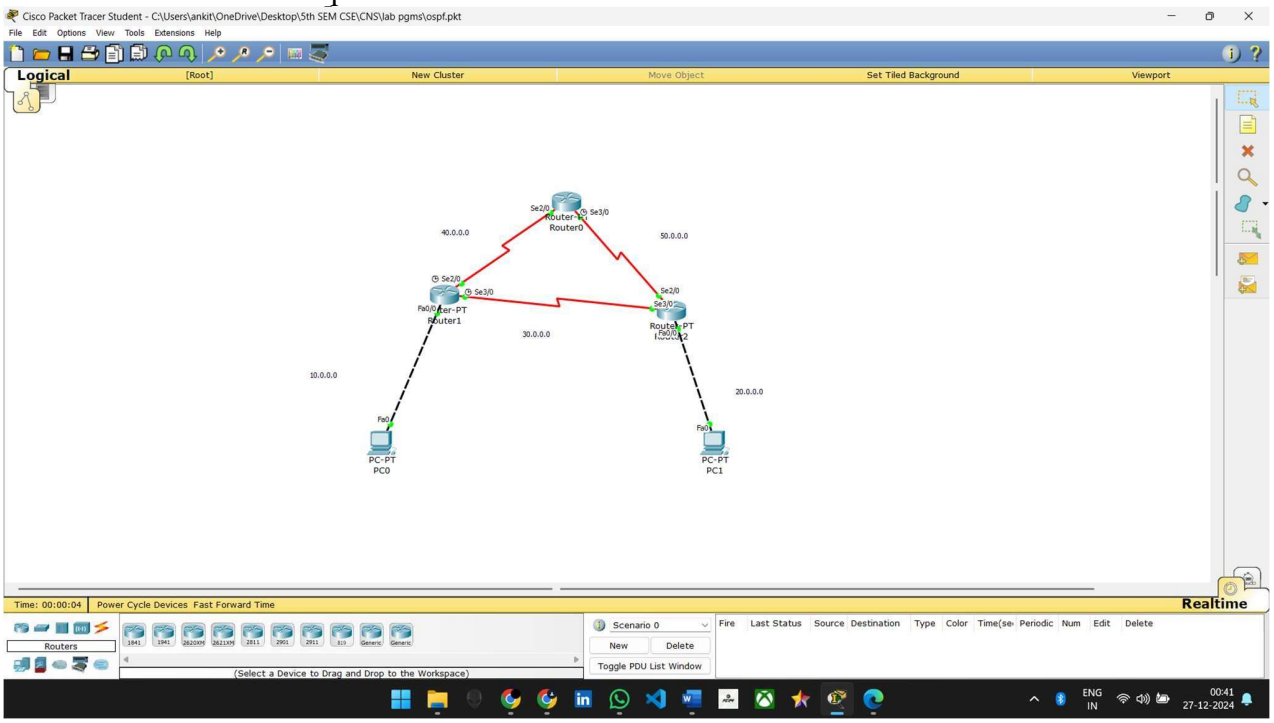
→ Set router ID's for identification.

→ check routing tables to verify OSPF routes.

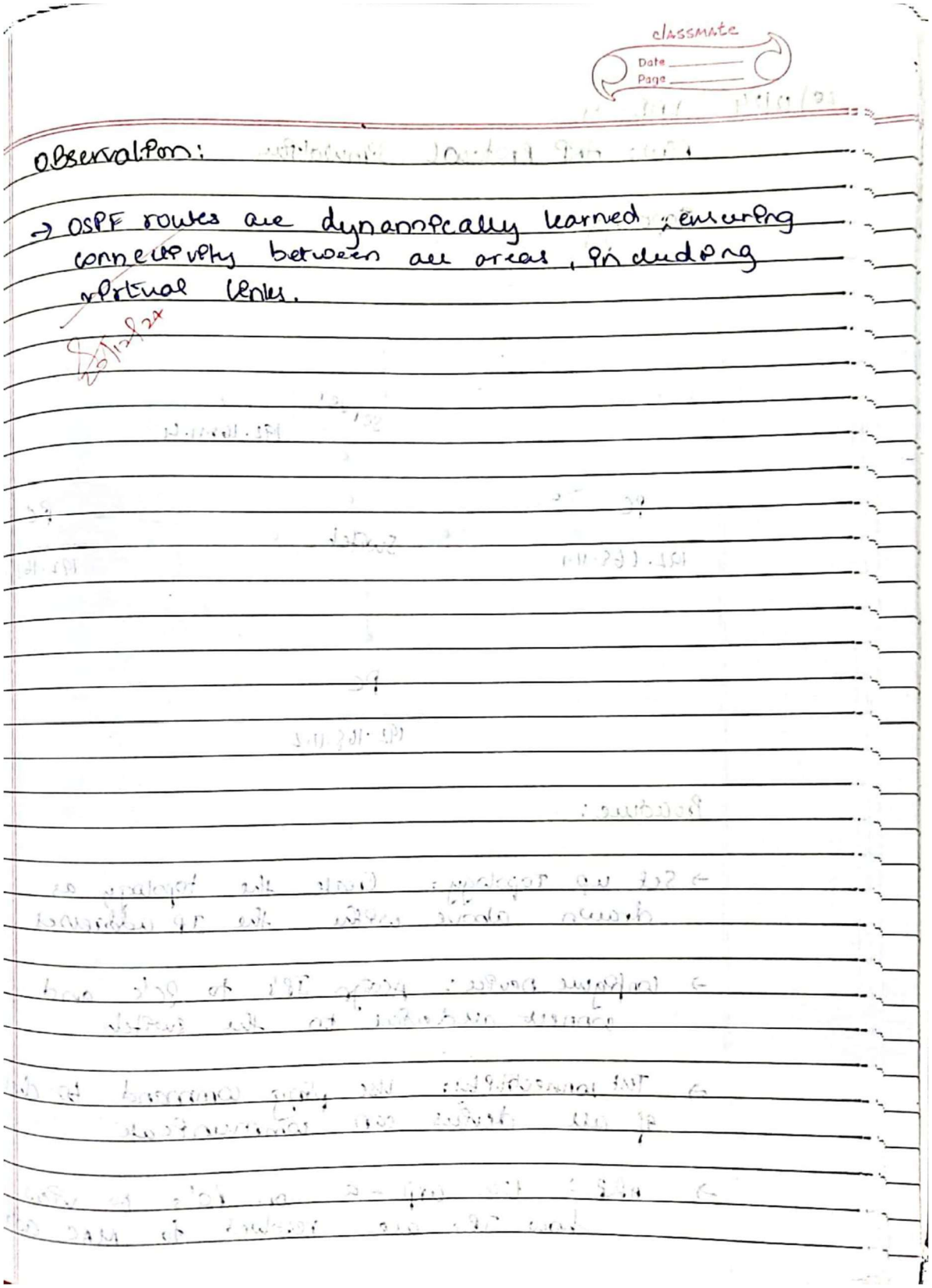
→ configure virtual links b/w R1 & R2 to connect non-backbone Area 1 to Area 0.

→ Test connectivity between endpoints eg. 10.0.0.10 (R1) to 40.0.0.10 (R3).

iii. Screen shots/ output:



iv. Observation:



Program 7

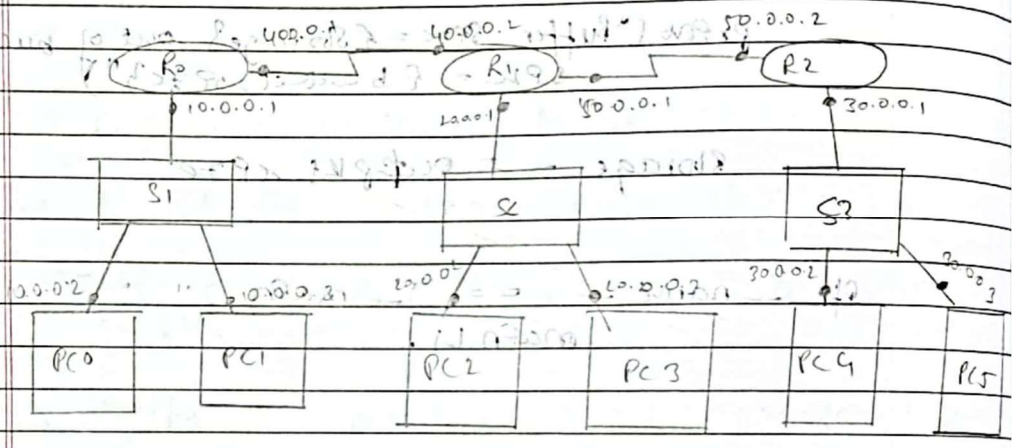
i. Aim of the program: Demonstrate the TTL/ Life of a Packet

ii. Procedure along with the topology:

22/11/24 LAB - 7

Aim: Demonstrate RIP (Routing Information Protocol).

Topology:



Procedure:

→ Configure all end devices with respective IP addresses.

→ Configure routers and their fast Ethernet ports and serial ports as shown in the figure.

→ Configure the gateways on end devices

→ Open the CLI of each router, open config Terminal and type 'router rip' after that type all the networks

to be connected

→ Simulate the packet transmission from one device to another device.

OBSERVATION:

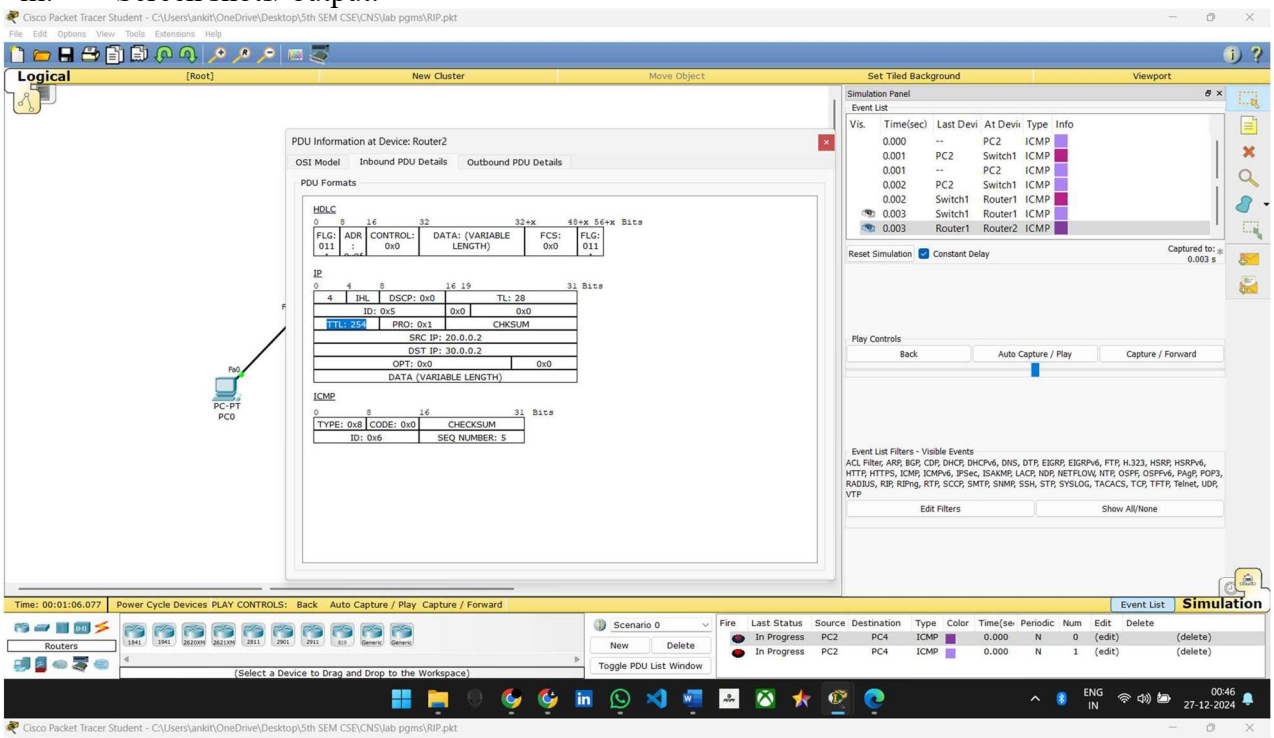
→ Packet transmission from source to destination is successful.

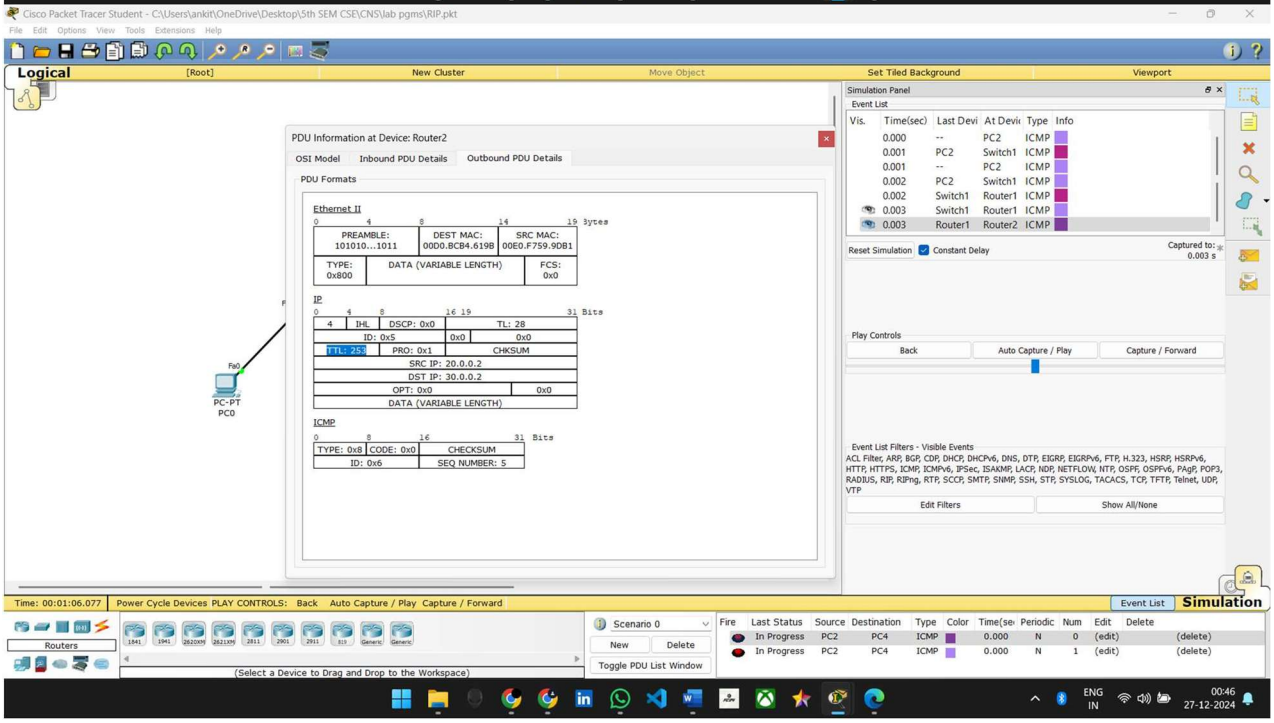
→ IP address are established.

→ TTL originally starts with a value 255.

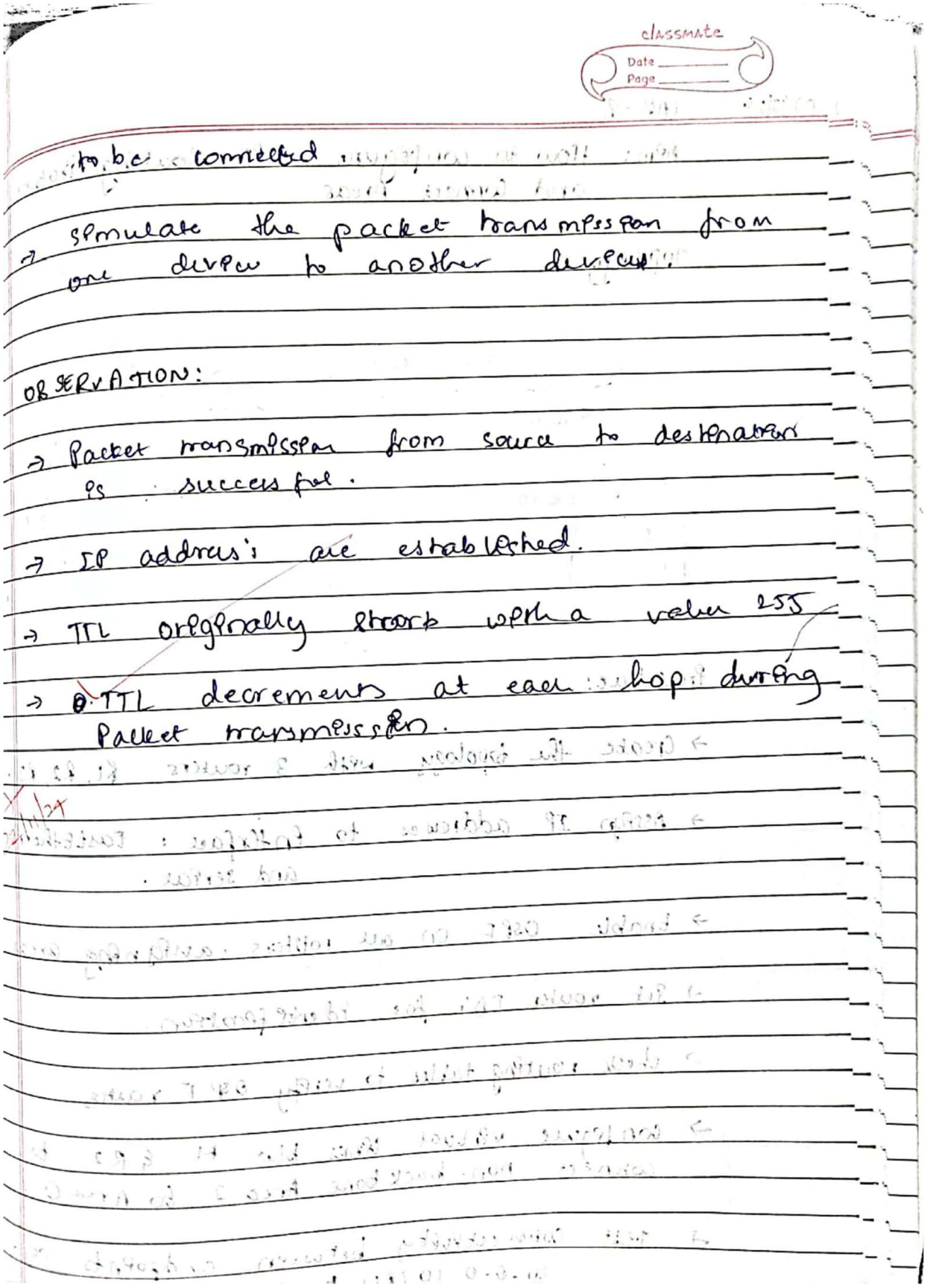
→ TTL decrements at each hop during Packet transmission.

iii. Screen shots/ output:





Observation:



to be connected

→ Simulate the packet transmission from one device to another device.

OBSERVATION:

→ Packet transmission from source to destination is successful.

→ IP address are established.

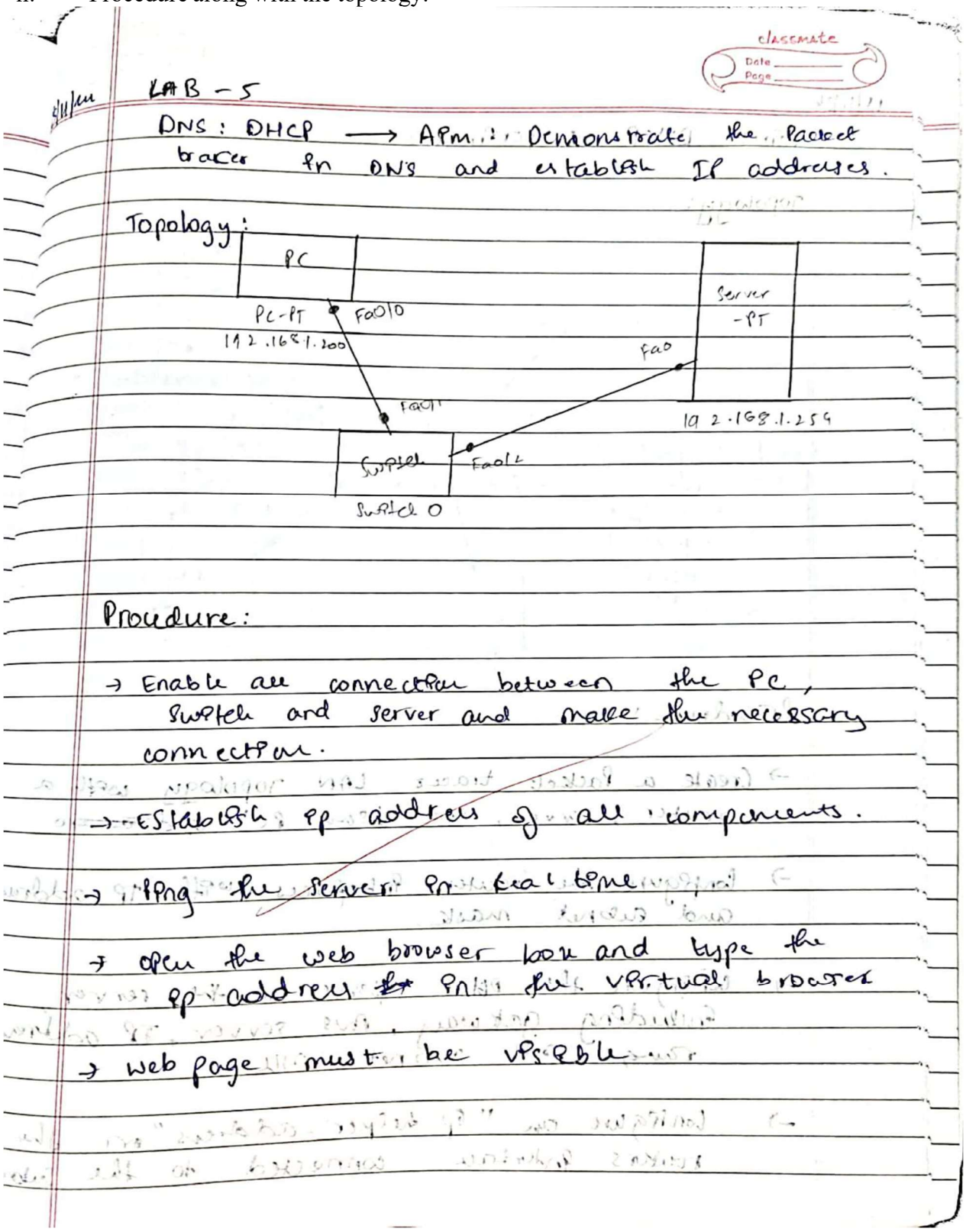
→ TTL originally starts with a value 255

→ TTL decrements at each hop during Packet transmission.

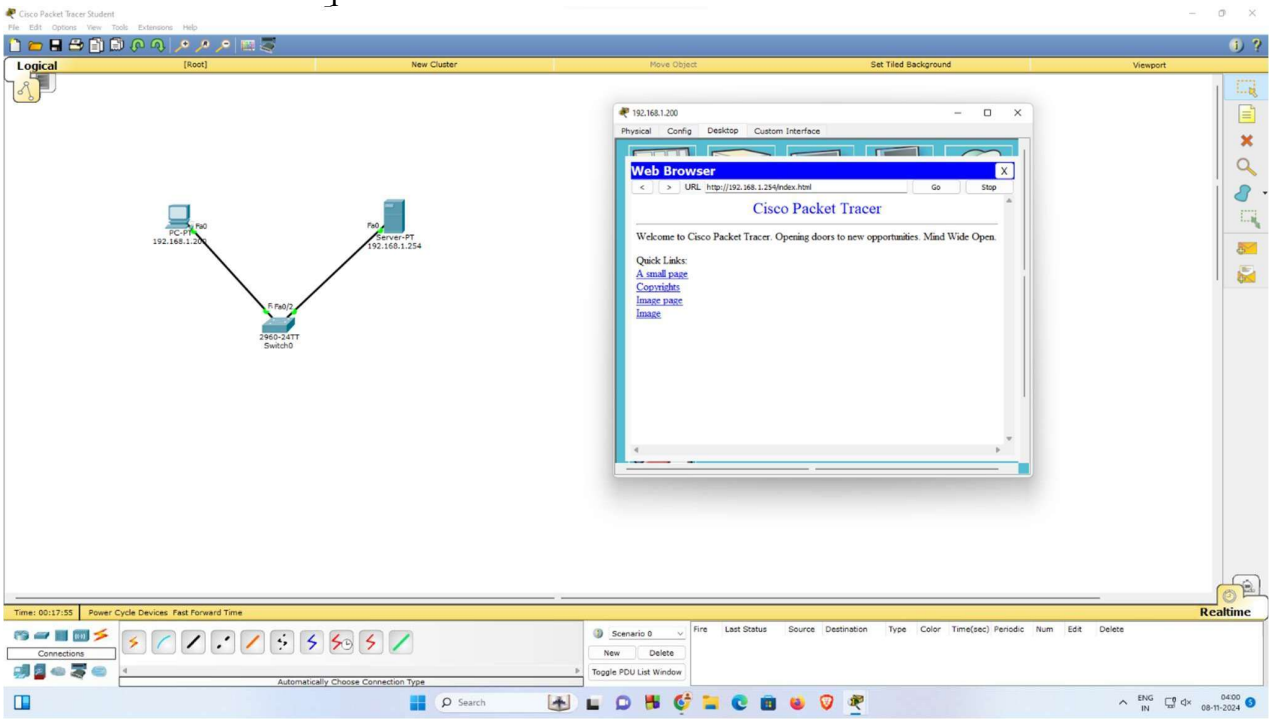
Program 8

i. Aim of the program: Configure Web Server, DNS within a LAN.

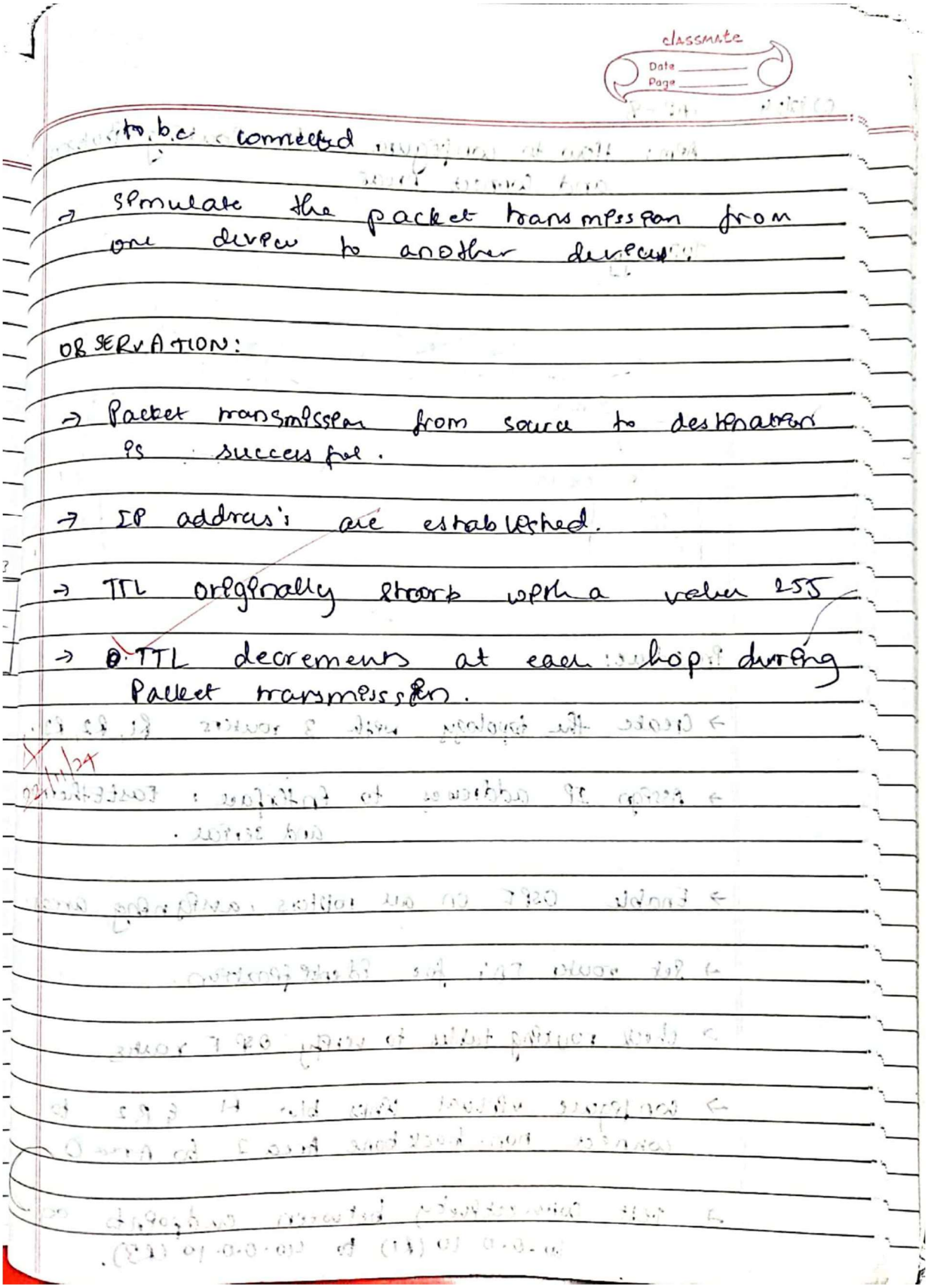
ii. Procedure along with the topology:



iii. Screen shots/ output:



iv. Observation:



to be connected

→ Simulate the packet transmission from one device to another device.

OBSERVATION:

→ Packet transmission from source to destination is successful.

→ IP address are established.

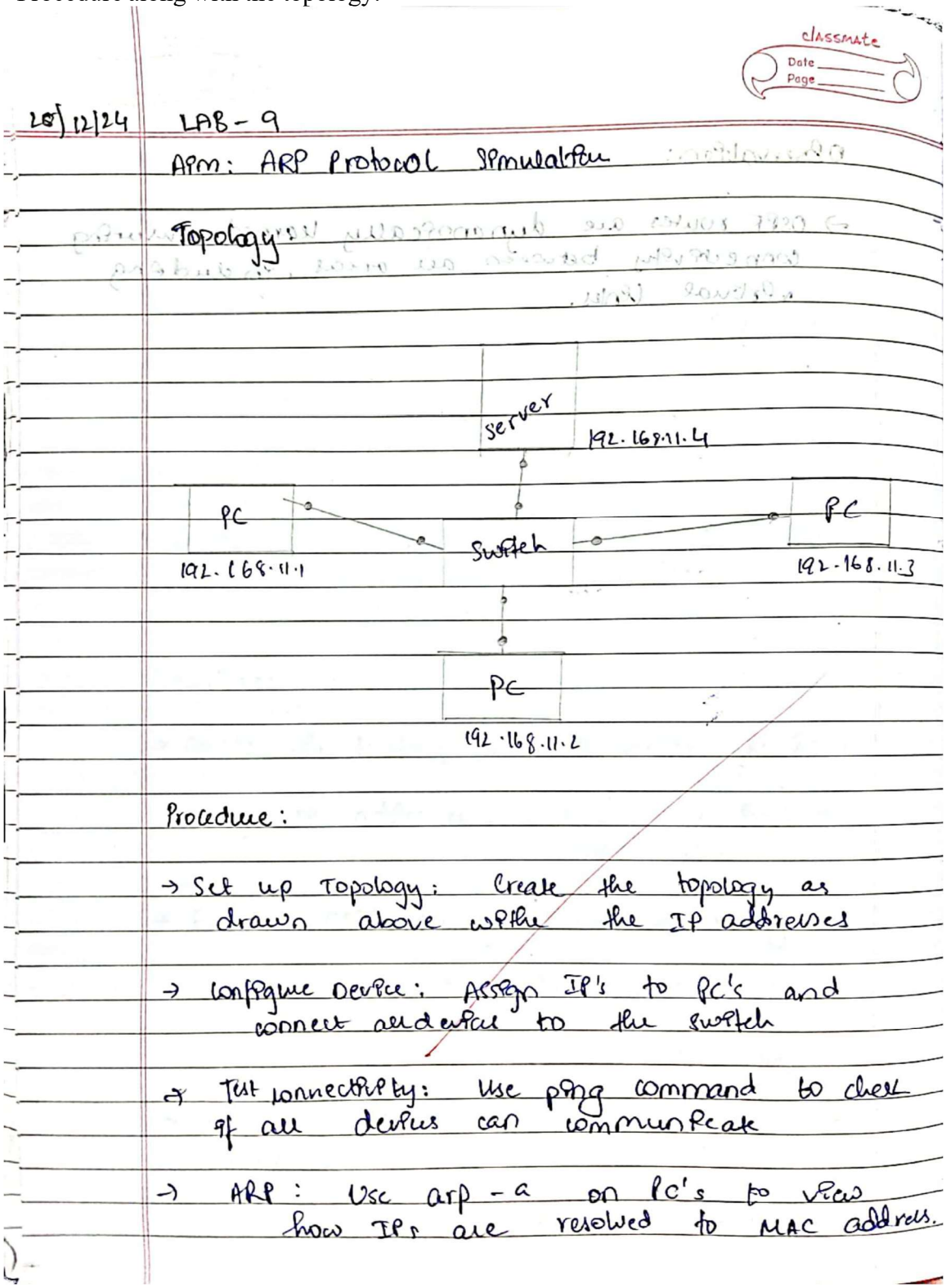
→ TTL originally starts with a value 255

→ TTL decrements at each hop during Packet transmission.

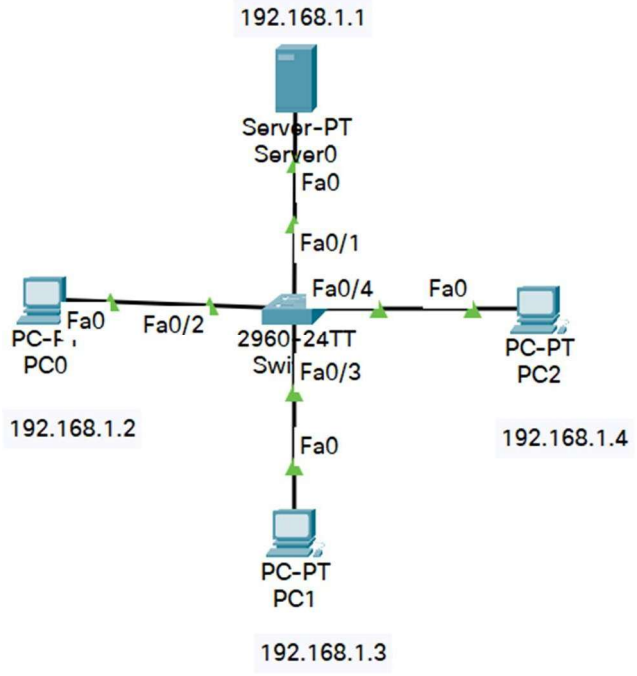
Program 9

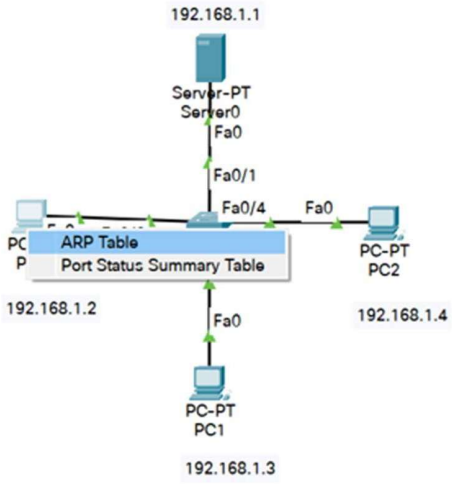
i. Aim of the program: To construct simple LAN and understand the concept and operation of Address Resolution Protocol (ARP)

ii. Procedure along with the topology:



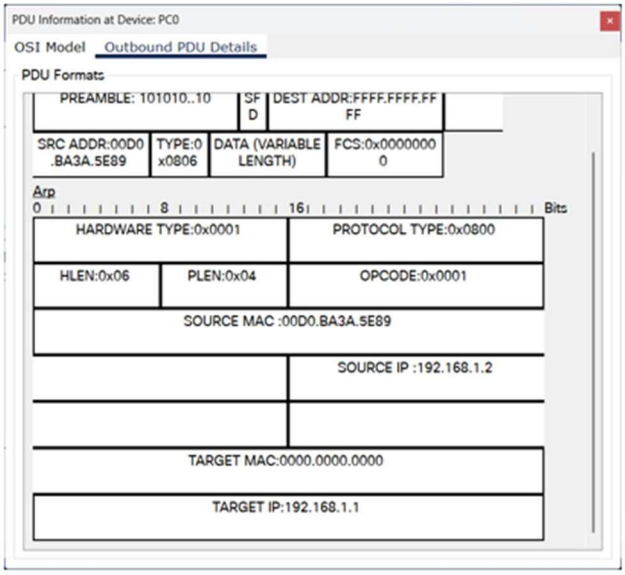
iii. Screen shots/ output:

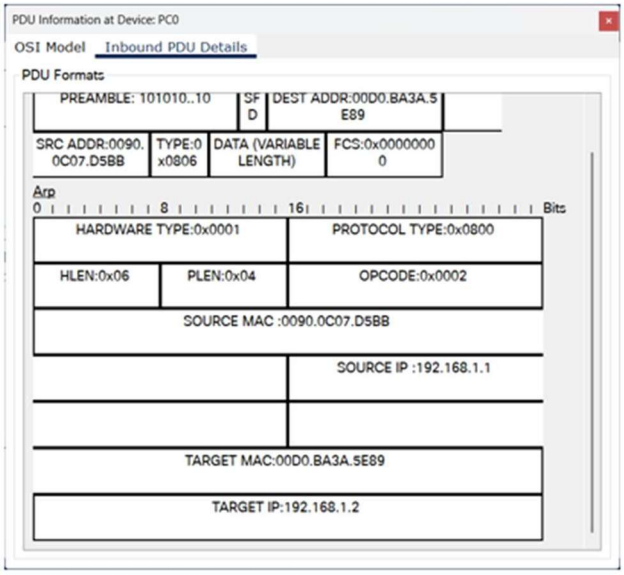




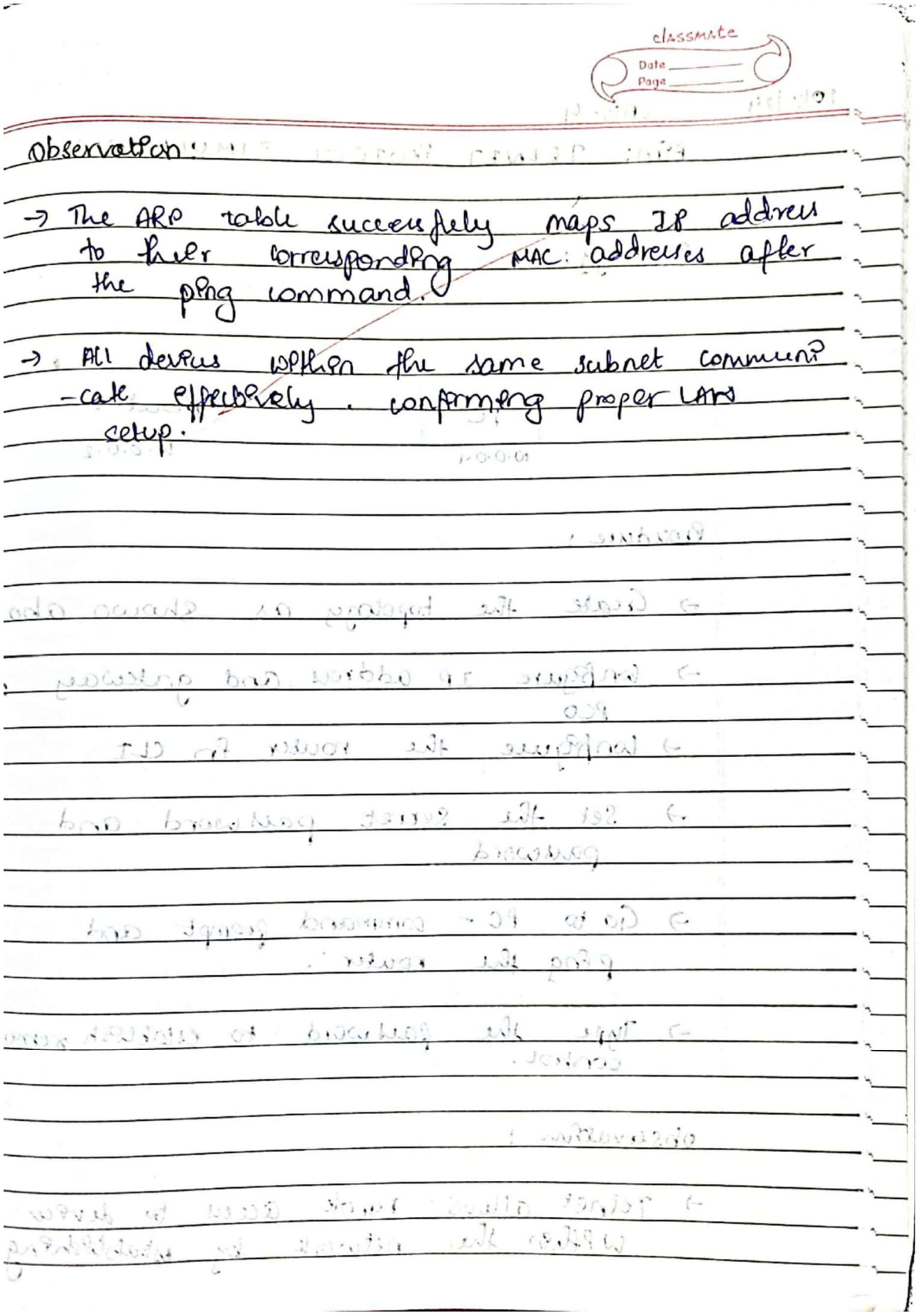
```
Packet Tracer PC Command Line 1.0
C:\>arp -a
No ARP Entries Found
C:\>
```

Pinging in Simulation Mode





iv. Observation:



Observation:

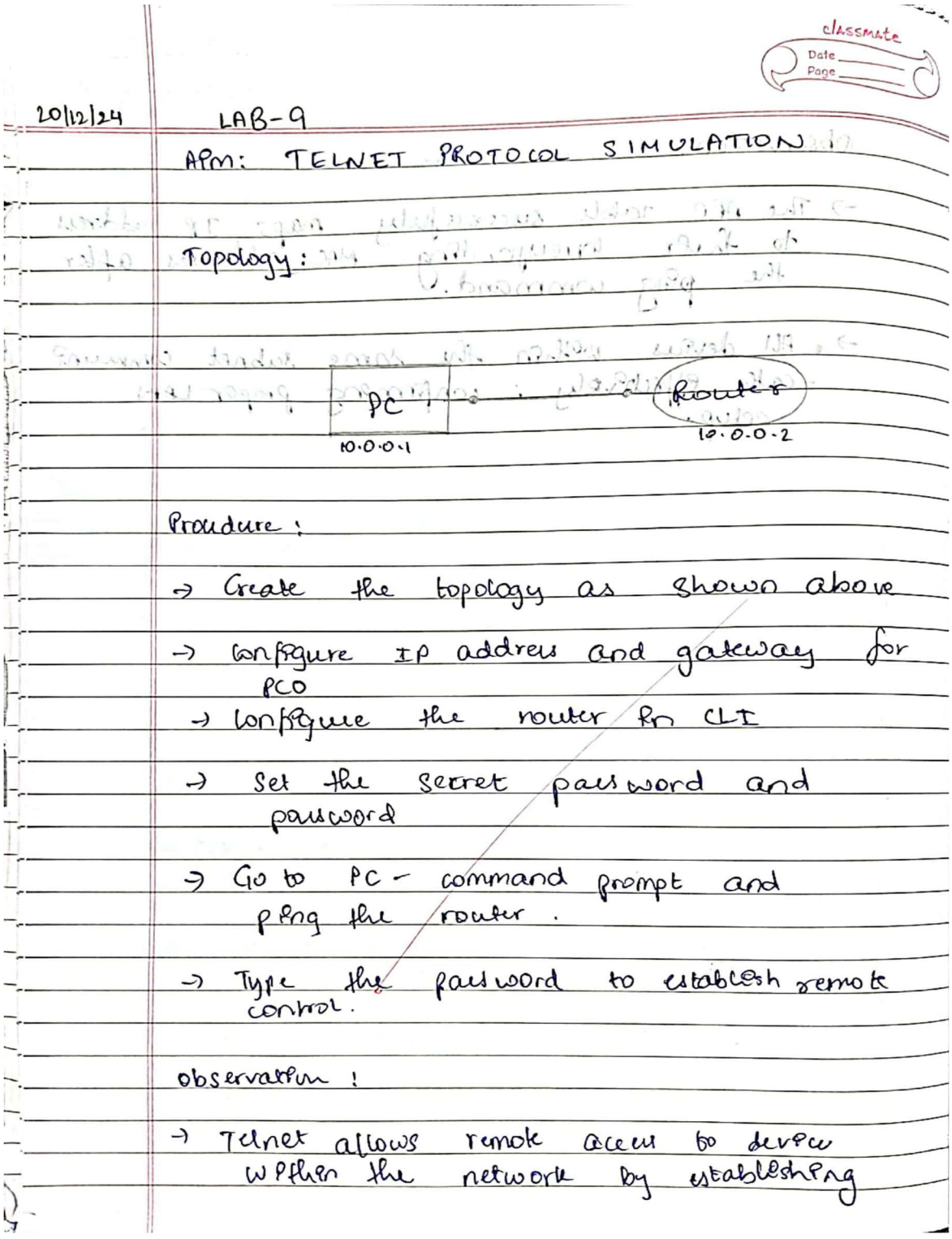
→ The ARP table successfully maps IP address to their corresponding MAC addresses after the ping command.

→ All devices within the same subnet communicate effectively, confirming proper LAN setup.

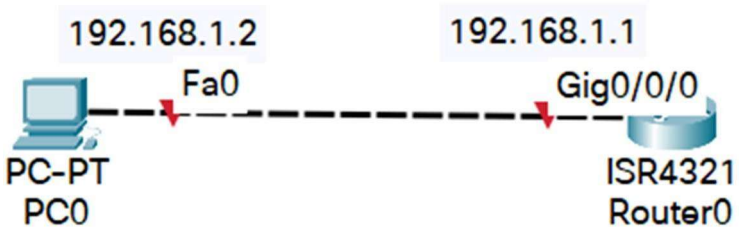
Program 10

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

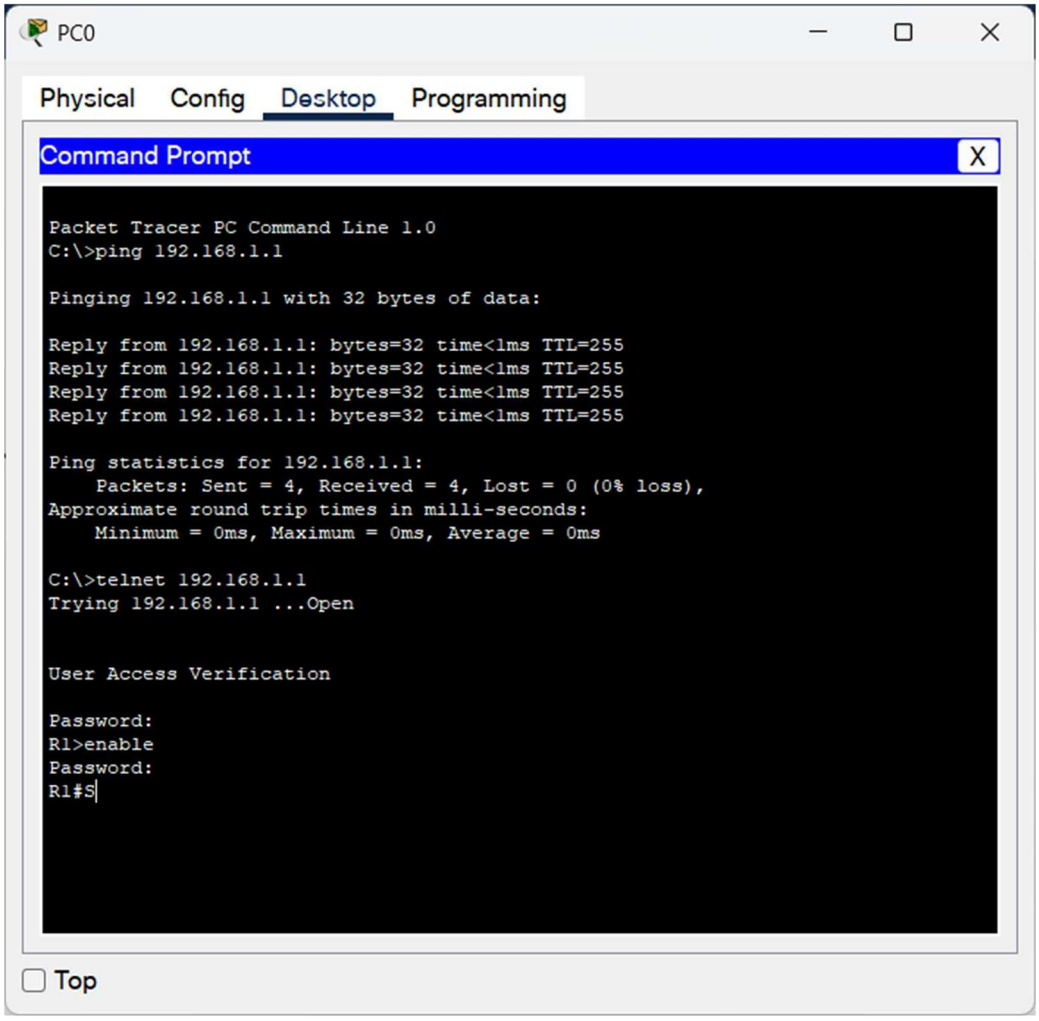
ii. Procedure along with the topology:



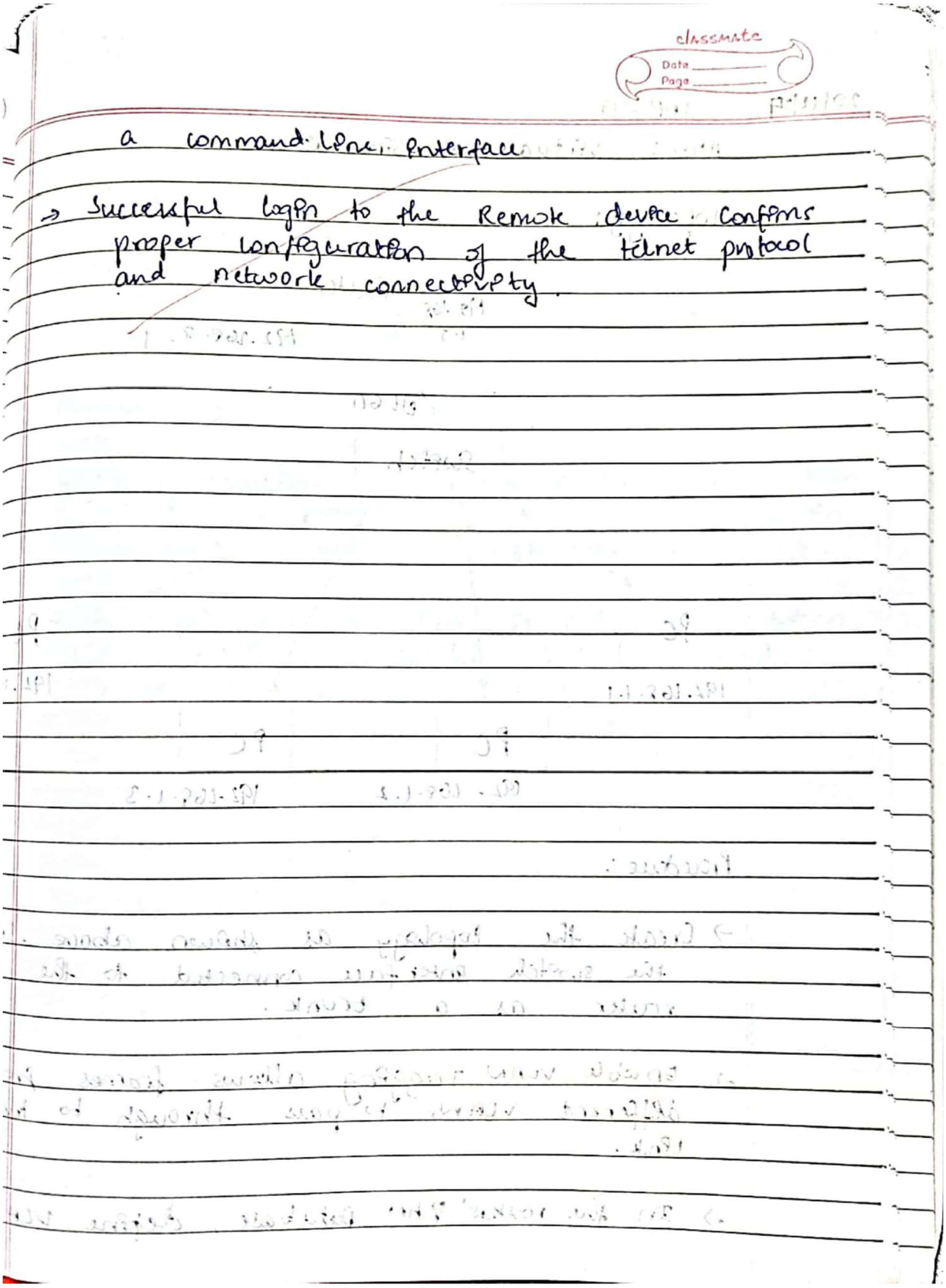
iii. Screen shots/ output:




```
Router>enable
Router#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#hostname R1
R1(config)#enable secret hello
R1(config)#interface g0/0/0
R1(config-if)#ip address 192.168.1.1 255.255.255.0
R1(config-if)#no shutdown

R1(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0/0, changed state to up

R1(config-if)#line vty 0 5
R1(config-line)#login
% Login disabled on line 2, until 'password' is set
% Login disabled on line 3, until 'password' is set
% Login disabled on line 4, until 'password' is set
% Login disabled on line 5, until 'password' is set
% Login disabled on line 6, until 'password' is set
% Login disabled on line 7, until 'password' is set
R1(config-line)#password pass
R1(config-line)#end
R1#
%SYS-5-CONFIG_I: Configured from console by console

R1#wr
Building configuration...
[OK]
R1#
```


PC0
Physical
Config
Desktop
Programming
Command Prompt
X
Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.1
Pinging 192.168.1.1 with 32 bytes of data:
Reply from 192.168.1.1: bytes=32 time<1ms TTL=255
Reply from 192.168.1.1: bytes=32 time<1ms TTL=255
Reply from 192.168.1.1: bytes=32 time<1ms TTL=255
Reply from 192.168.1.1: bytes=32 time<1ms TTL=255
Ping statistics for 192.168.1.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\>telnet 192.168.1.1
Trying 192.168.1.1 ...Open
User Access Verification
Password:
R1>enable
Password:
R1#S
Top

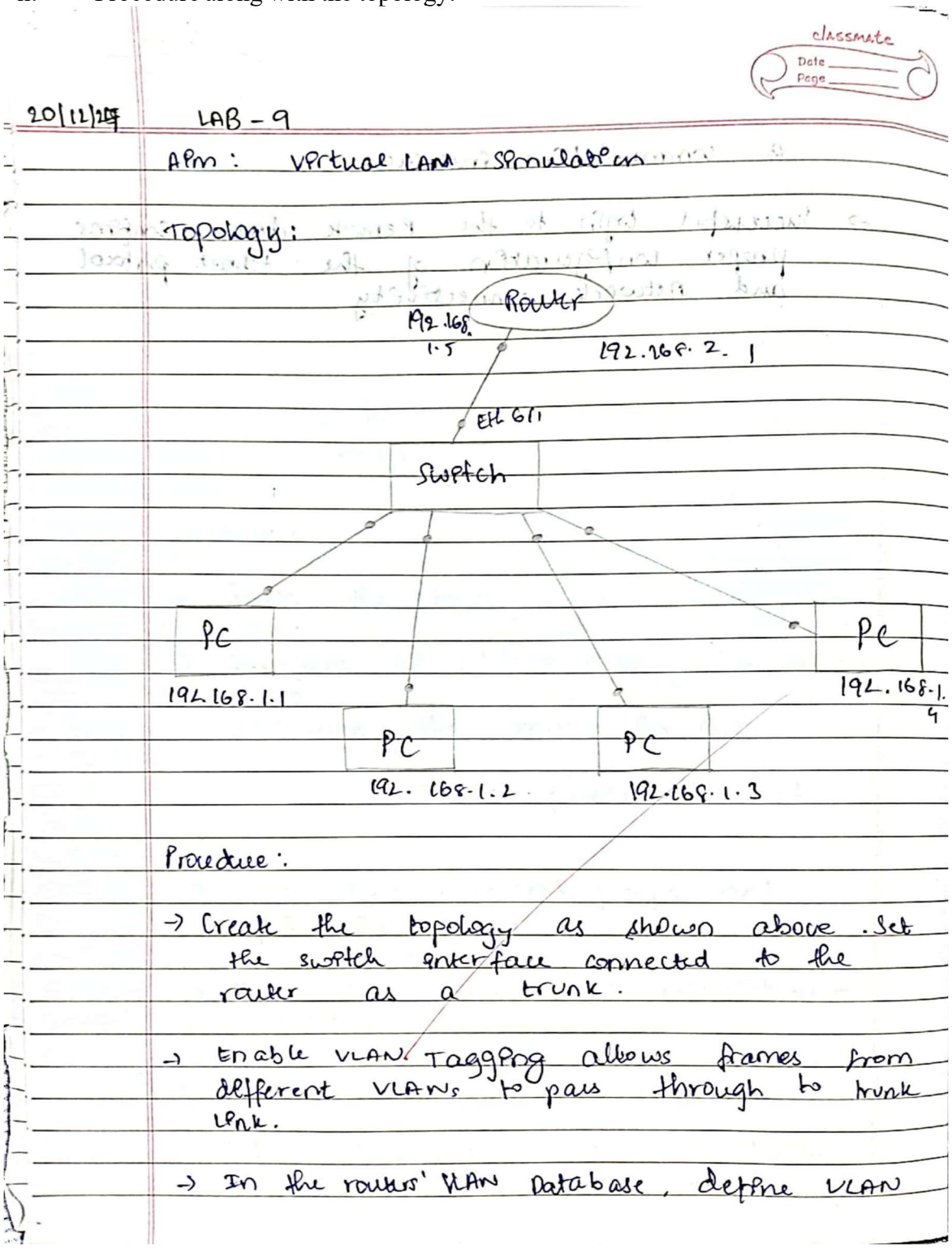
iv. Observation:



Program 11

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

ii. Procedure along with the topology:



number and name. Assign a sub-interface for the VLAN with dot1q encapsulation and an IP address.

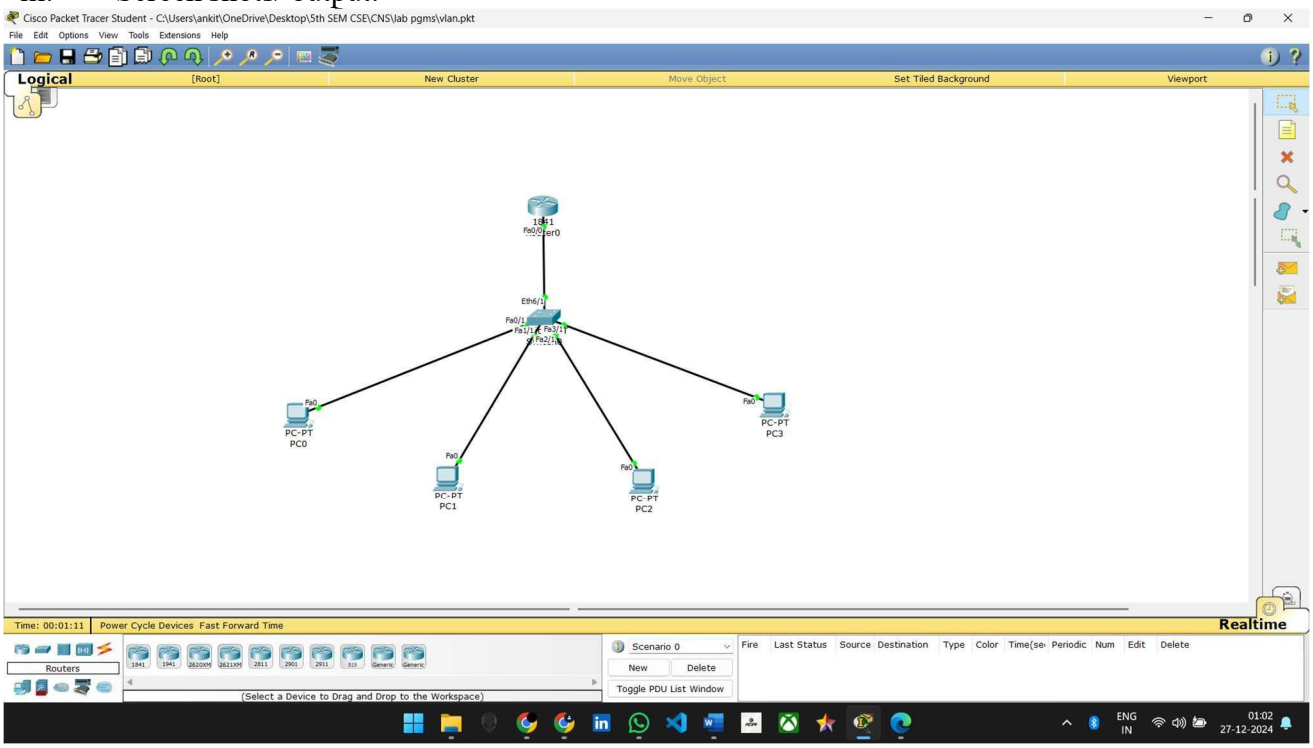
→ Activate VLAN by using no shut command. Save it and verify the configuration.

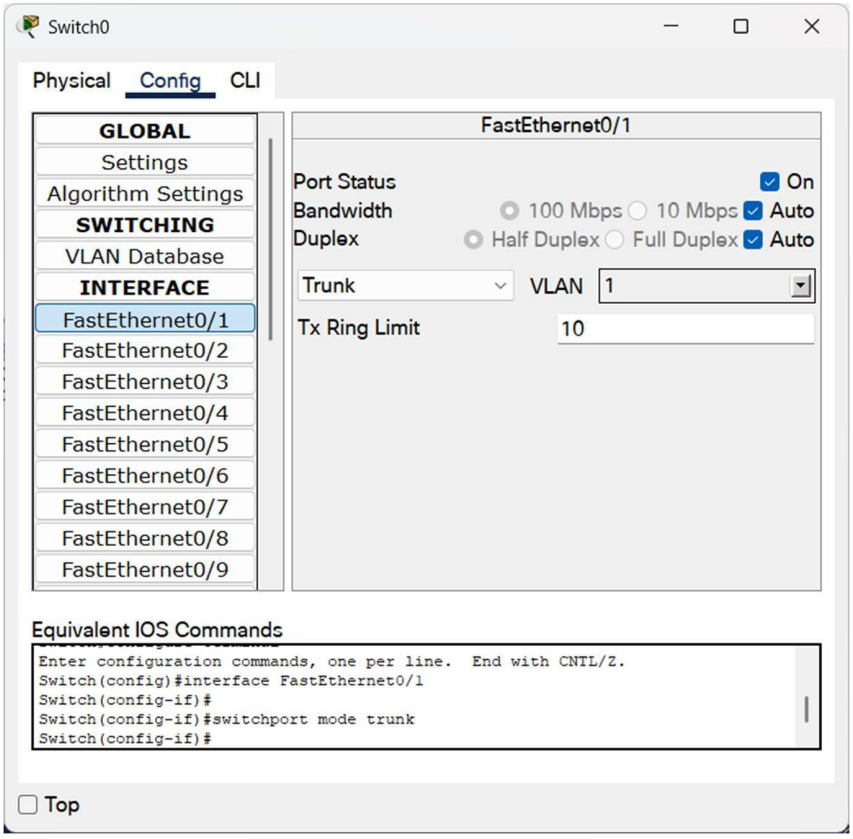
Observation:

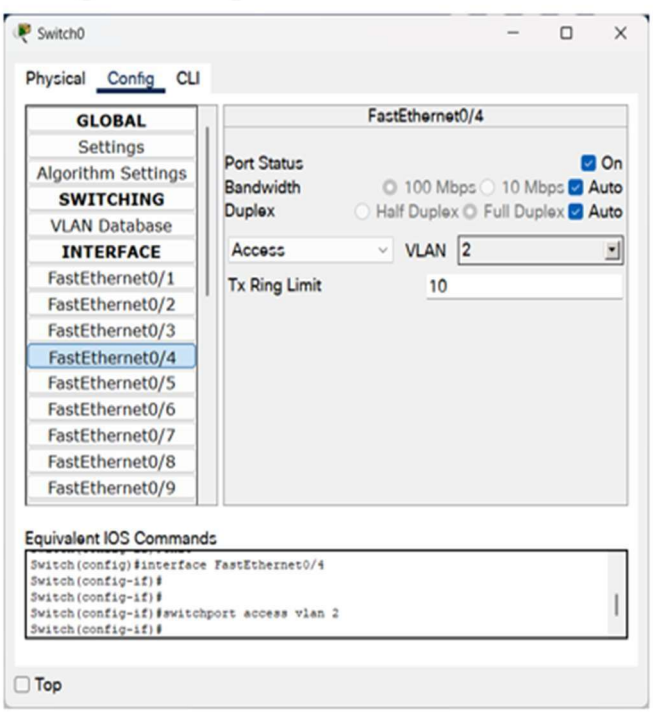
→ VLAN Tagging ensures proper data segregation and communication between devices in different VLANs through the trunk link.

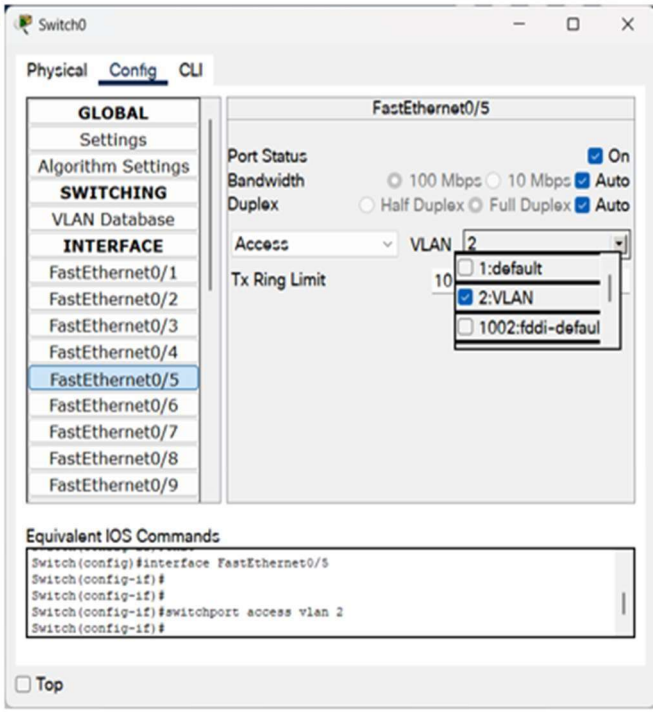
→ Successful communication is established when the router and switch are configured correctly with VLAN Tagging.

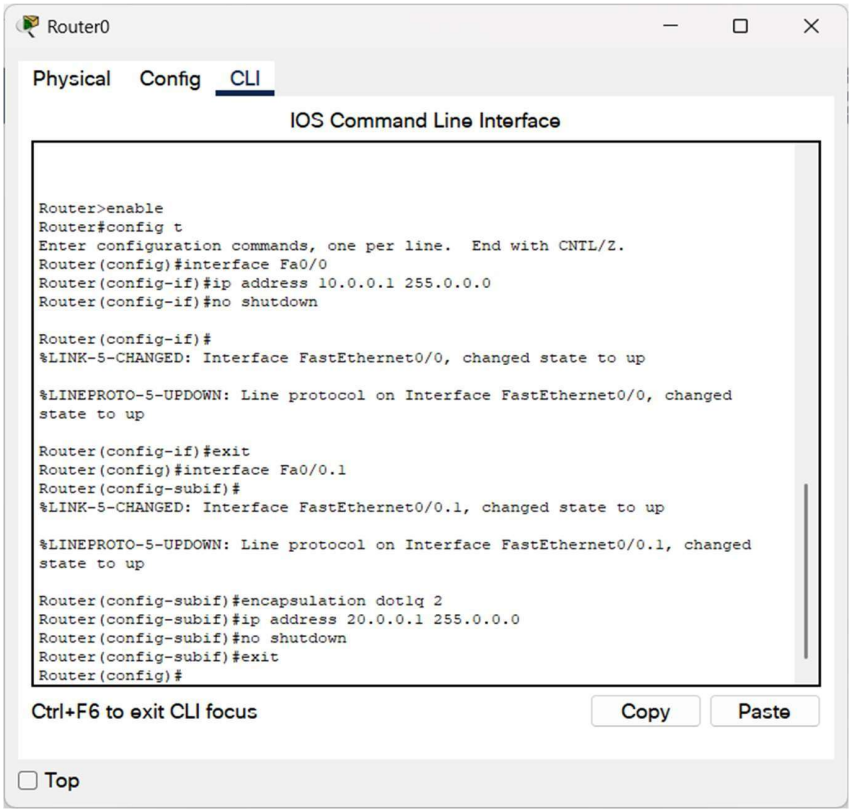
iii. Screen shots/ output:






Switch0
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
SWITCHING
VLAN Database
INTERFACE
FastEthernet0/1
FastEthernet0/2
FastEthernet0/3
FastEthernet0/4
FastEthernet0/5
FastEthernet0/6
FastEthernet0/7
FastEthernet0/8
FastEthernet0/9
FastEthernet0/4
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
Access
VLAN
2
Tx Ring Limit
10
Equivalent IOS Commands
Switch(config)#interface FastEthernet0/4
Switch(config-if)#
Switch(config-if)#
Switch(config-if)#switchport access vlan 2
Switch(config-if)#
Top


Switch0
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
SWITCHING
VLAN Database
INTERFACE
FastEthernet0/1
FastEthernet0/2
FastEthernet0/3
FastEthernet0/4
FastEthernet0/5
FastEthernet0/6
FastEthernet0/7
FastEthernet0/8
FastEthernet0/9
FastEthernet0/5
Port Status
On
Bandwidth
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
Access
VLAN
2
1:default
2:VLAN
1002:fddi-defaul
Tx Ring Limit
10
Equivalent IOS Commands
Switch(config)#interface FastEthernet0/5
Switch(config-if)#
Switch(config-if)#
Switch(config-if)#switchport access vlan 2
Switch(config-if)#
Top


Router0
Physical
Config
CLI
IOS Command Line Interface
Router>enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface Fa0/0
Router(config-if)#ip address 10.0.0.1 255.0.0.0
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
Router(config-if)#exit
Router(config)#interface Fa0/0.1
Router(config-subif)#
%LINK-5-CHANGED: Interface FastEthernet0/0.1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0.1, changed state to up
Router(config-subif)#encapsulation dot1q 2
Router(config-subif)#ip address 20.0.0.1 255.0.0.0
Router(config-subif)#no shutdown
Router(config-subif)#exit
Router(config)#
Ctrl+F6 to exit CLI focus
Copy
Paste
Top

iv. Observation:

number and name i.e. Assign a sub-interface for the VLAN with dot1q encapsulation and an IP address.

→ Activate VLAN by using no shut command. Save it and verify the configuration.

Observation:

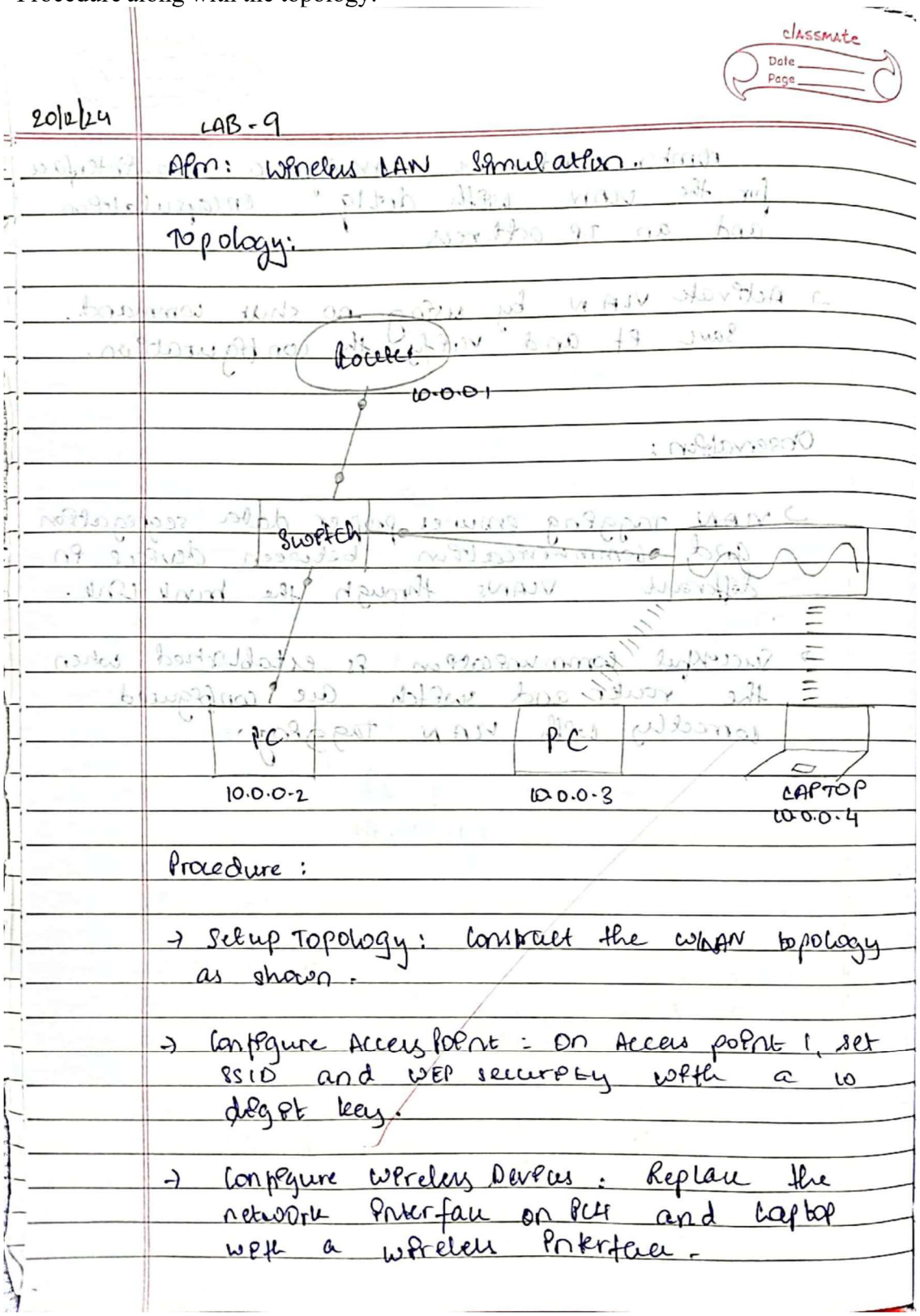
→ VLAN Tagging ensures proper data segregation and communication between devices in different VLANs through the trunk link.

→ Successful communication is established when the router and switch are configured correctly with VLAN Tagging.

Program 12

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

ii. Procedure along with the topology:



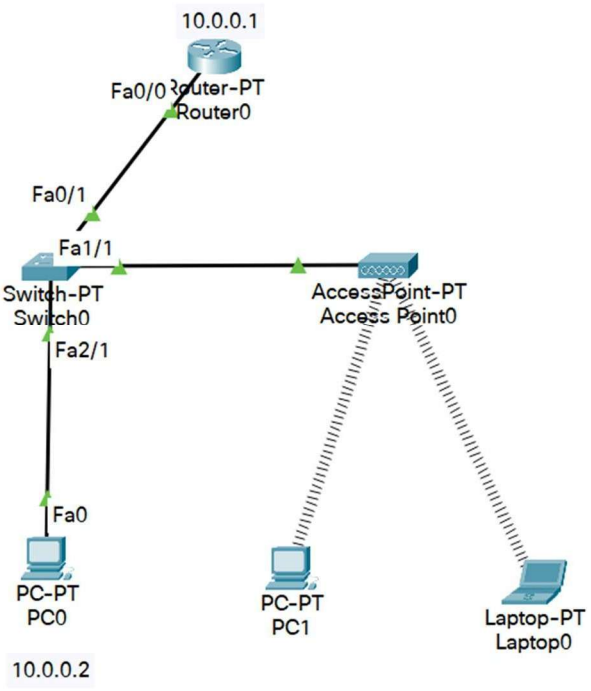
→ Ping all devices within the network to verify communication.

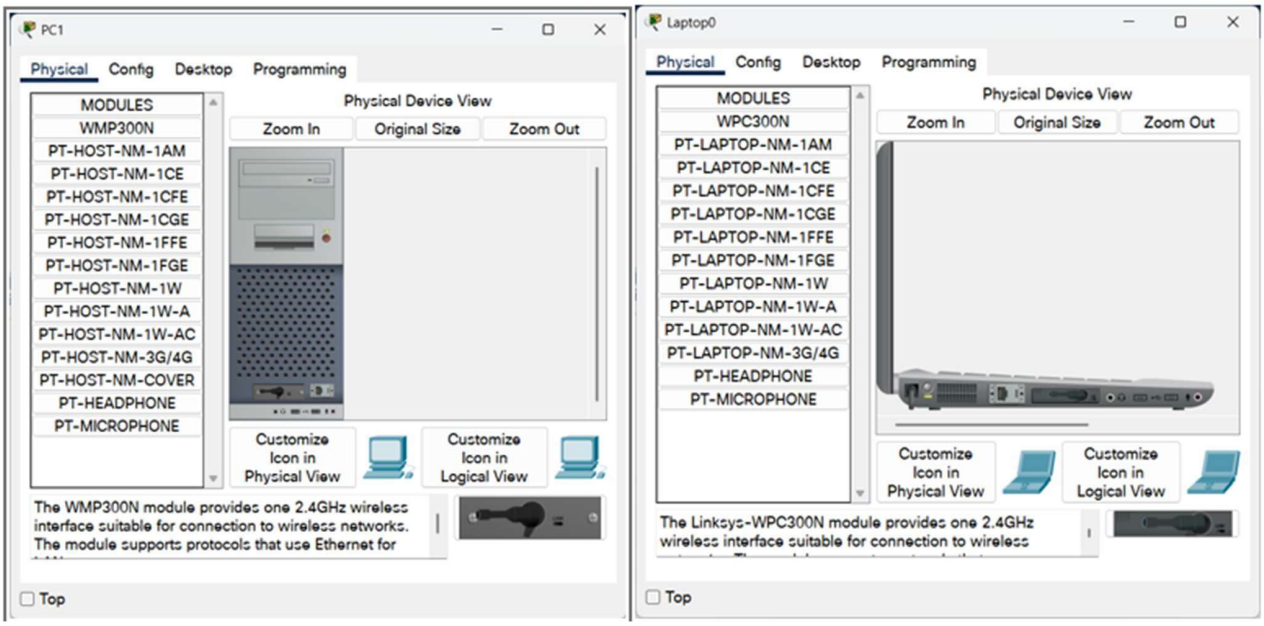
observation:

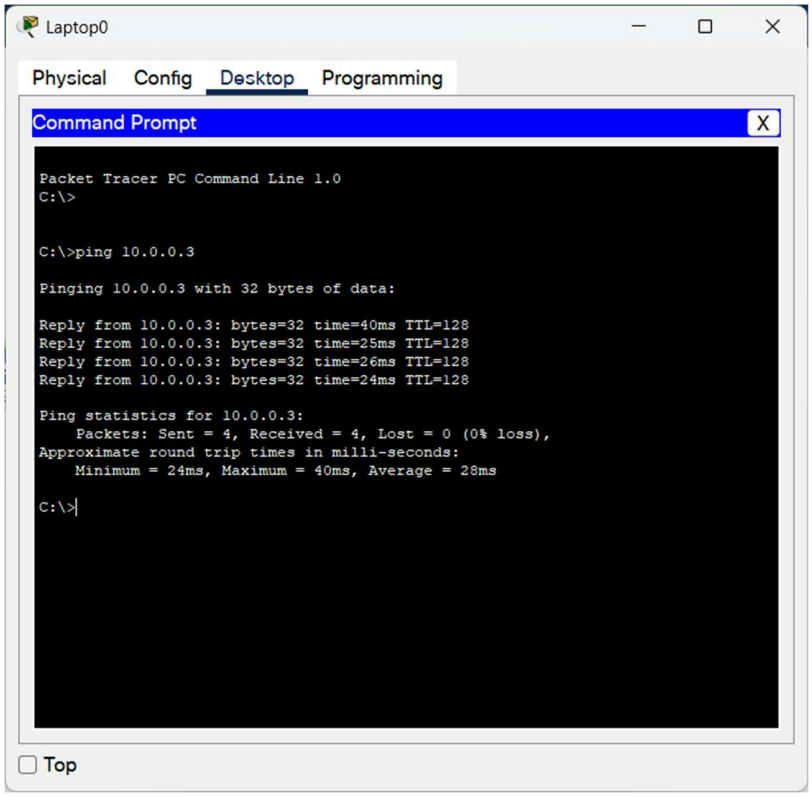
→ Successful wireless communication occurs between all devices when properly configured with matching SSID.

→ Devices respond to pings, confirming a fully functional WLAN setup.

iii. Screen shots/ output:






Laptop0
Physical
Config
Desktop
Programming
Command Prompt
X
Packet Tracer PC Command Line 1.0
C:\>
C:\>ping 10.0.0.3
Pinging 10.0.0.3 with 32 bytes of data:
Reply from 10.0.0.3: bytes=32 time=40ms TTL=128
Reply from 10.0.0.3: bytes=32 time=25ms TTL=128
Reply from 10.0.0.3: bytes=32 time=26ms TTL=128
Reply from 10.0.0.3: bytes=32 time=24ms TTL=128
Ping statistics for 10.0.0.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 24ms, Maximum = 40ms, Average = 28ms
C:\>
Top

iv. Observation:

→ Ping all devices within the network to verify communication.

observation:

→ Successful wireless communication occurs between all devices when properly configured with matching SSID.

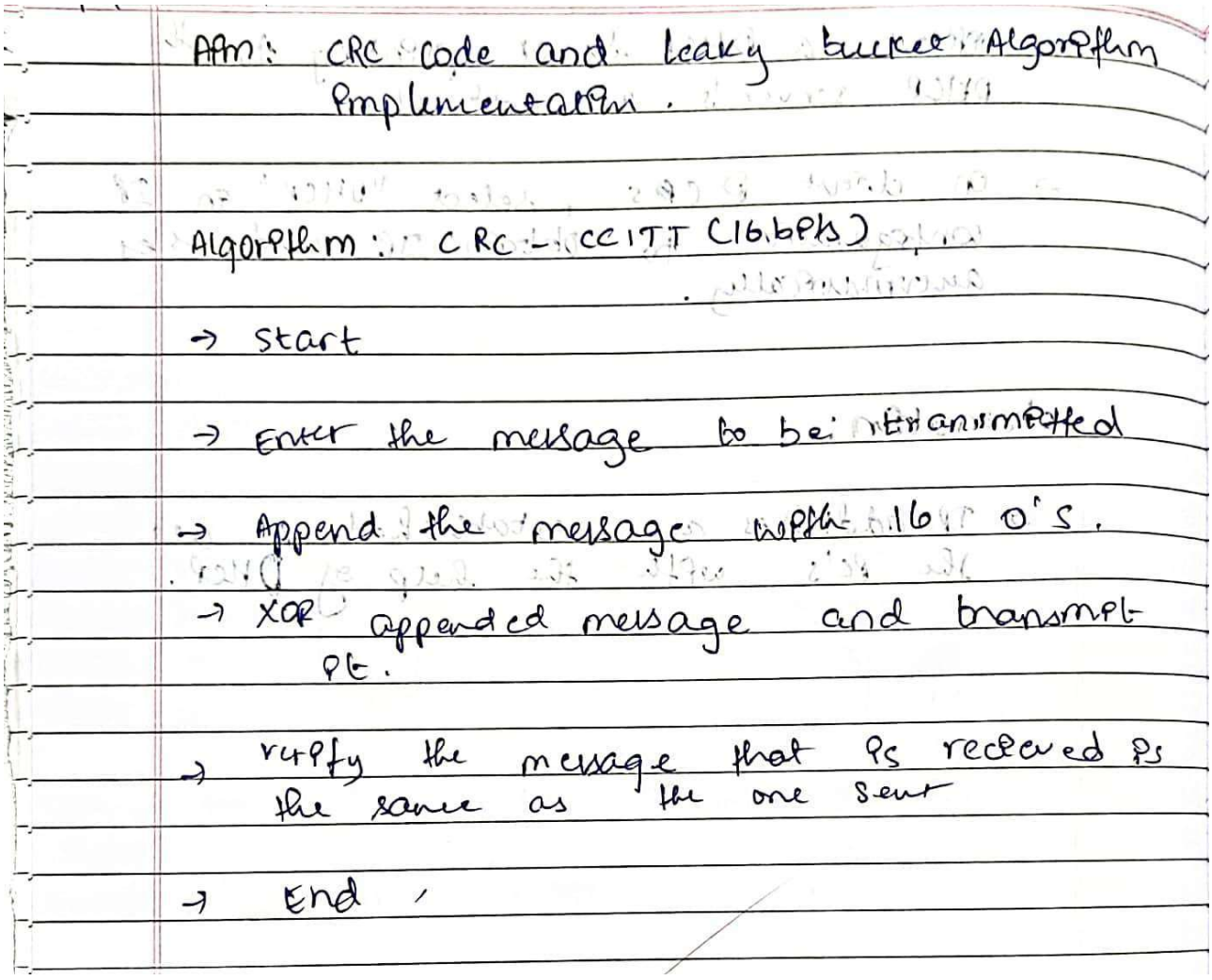
→ Devices respond to pings, confirming a fully functional WLAN setup.

CYCLE – 2

Program 1

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

ii. Procedure:



iii. Screen shots code:

```python
def crc(input_msg, poly, mode):
    # Prepare the output message
    output_msg = input_msg
    if mode:
        output_msg += '0' * (len(poly) - 1)

    # Perform XOR on the message with the selected polynomial
    output_msg = list(output_msg)
    poly_len = len(poly)
    for i in range(len(input_msg)):
        if output_msg[i] == '1':
            for j in range(poly_len):
                if i + j < len(output_msg):
                    output_msg[i + j] = '0' if output_msg[i + j] == poly[j] else '1'

    # Check for errors and return 0 if error detected
    return '1' not in output_msg[-(poly_len-1):]

def main():
    poly = "10001000000100001"

    # Input the original message
    input_msg = input("Enter the input message in binary: ")

    # Calculate transmitted message
    crc(input_msg, poly, 1)
    transmitted_msg = input_msg + '0' * (len(poly) - 1)

    print("The transmitted message is:", transmitted_msg)

    # Input the received message
    received_msg = input("Enter the received message in binary: ")

    # Check for errors in received message
    if crc(received_msg, poly, 0):
        print("No error in data")
    else:
        print("Error in data transmission has occurred")

if __name__ == "__main__":
    main()
```

iv. Observation:

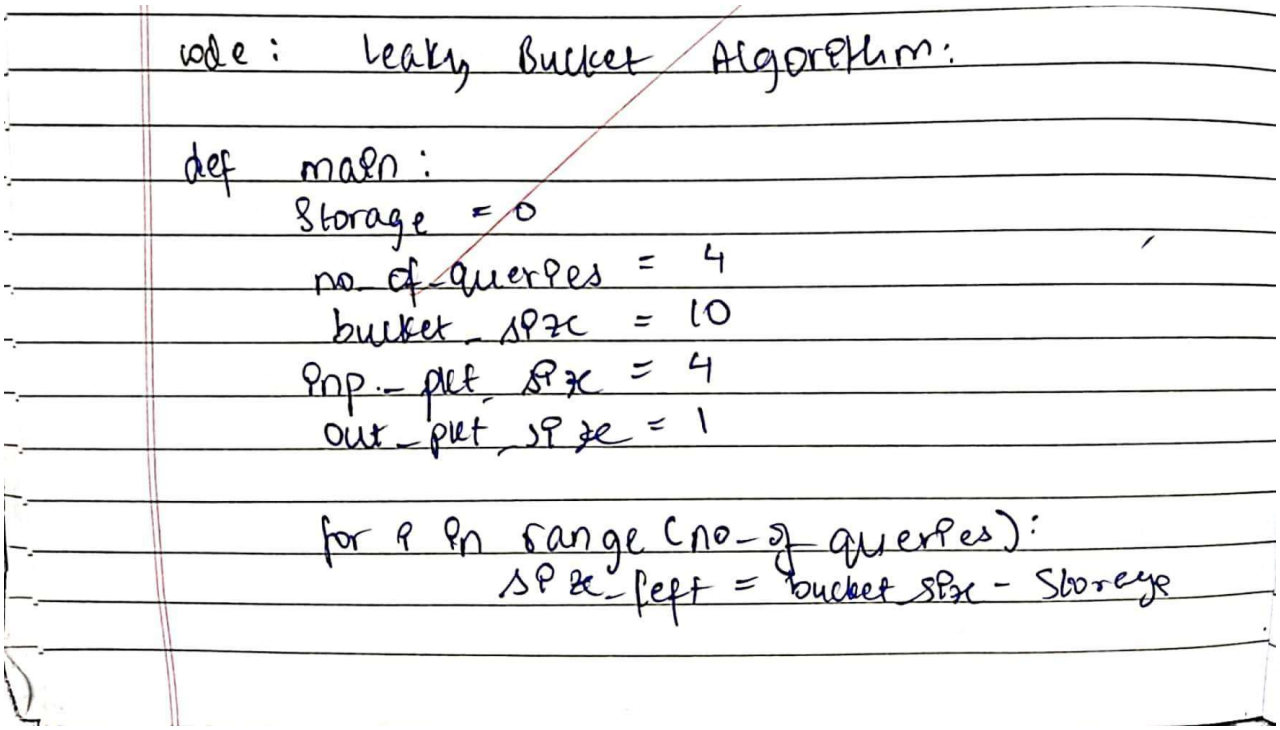
```
Enter the input message in binary: 11111
The transmitted message is: 11111000000000000000
Enter the received message in binary: 11101
No error in data
```

```
Enter the input message in binary:
11111
The transmitted message is: 111111110001111011110
Enter the received message in binary:
11101
Error in data transmission has occurred.
```

Program 2

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

ii. Procedure:



```
ode : Leaky Bucket Algorithm:

def main :
    Storage = 0
    no_of_querpes = 4
    bucket_sPz = 10
    inp._put_sPz = 4
    out_put_sPze = 1

    for P in range (no_of_querpes):
            sPze_left = bucket_sPz - Storeye
```

```
if pnp_pkt_sze <= sze_left :
        storage += pnp_pkt_sze

else :
    print("Packet loss = {pnp_pkt_sze}")

print("Buffer sze = {storage} out of bucket
        sze = {bucket_sze}")

storage -= out_pkt_sze

if __name__ == '__main__' :
        main()
```

Output :

Buffer size = 4 out of Bucket size = 10
Buffer size = 7 out of Bucket size = 10
Buffer size = 10 out of Bucket size = 10

Packet loss = 4

Buffer size = 9 out of Bucket size = 10

iii. Screen shots code:

```
storage = 0
no_of_queries = 4
bucket_size = 10
input_pkt_size = 4
output_pkt_size = 1

for i in range(no_of_queries):
    # Calculate space left in the bucket
    size_left = bucket_size - storage

    if input_pkt_size <= size_left:
        # Update storage if space is available
        storage += input_pkt_size
    else:
        # If not enough space, print packet loss
        print(f"Packet loss = {input_pkt_size}")

    # Print current buffer status
    print(f"Buffer size = {storage} out of bucket size = {bucket_size}")

    # Update storage by removing output packets
    storage -= output_pkt_size
```

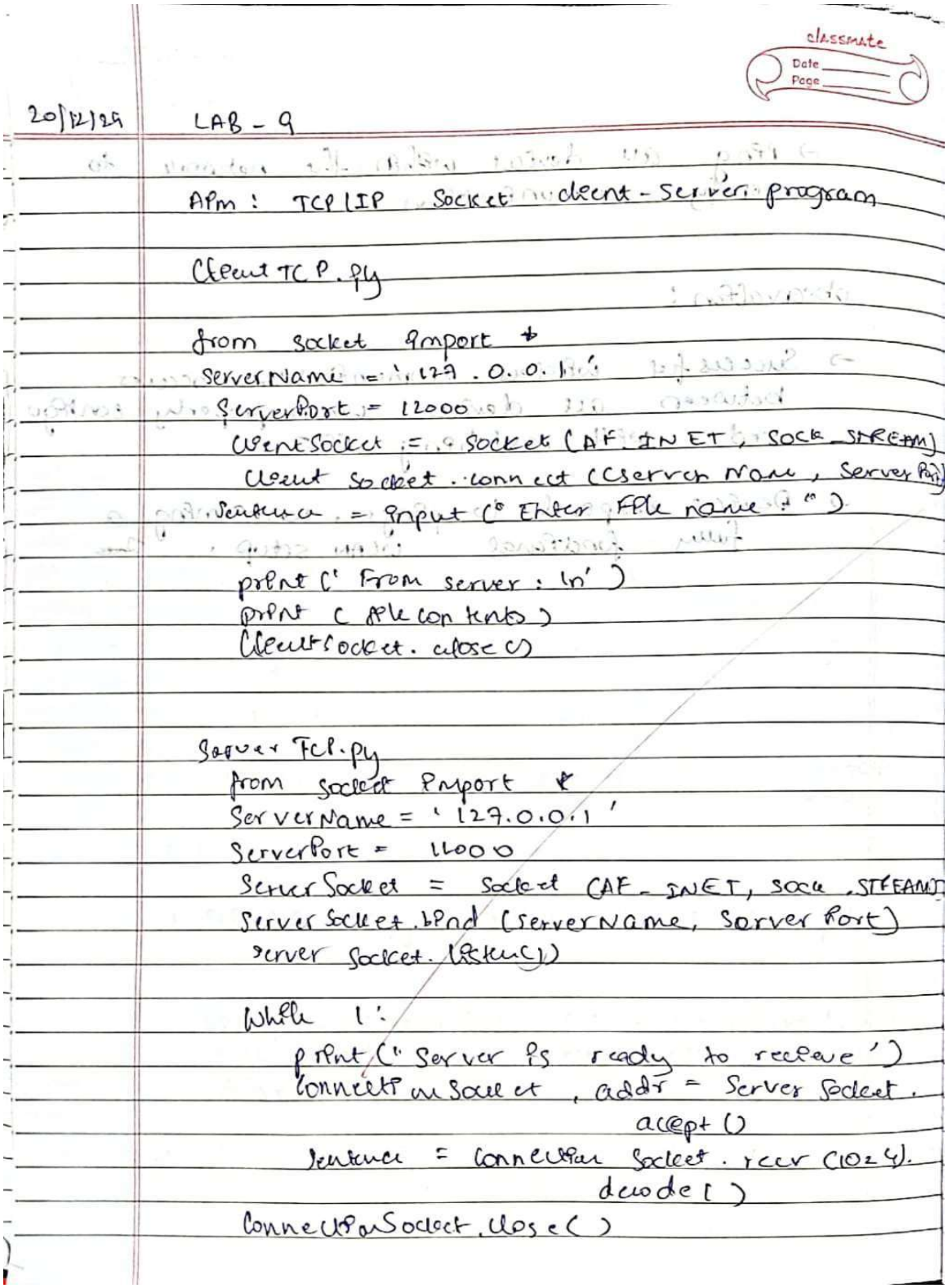
iv. Observation:

```
Buffer size = 4 out of bucket size = 10
Buffer size = 7 out of bucket size = 10
Buffer size = 10 out of bucket size = 10
Packet loss = 4
Buffer size = 9 out of bucket size = 10
```

Program 3

i. Aim of the program: 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.

ii. Procedure:



20/12/24 LAB - 9

Aim : TCP IP Socket client - server program

Client TCP.py

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

Server TCP.py

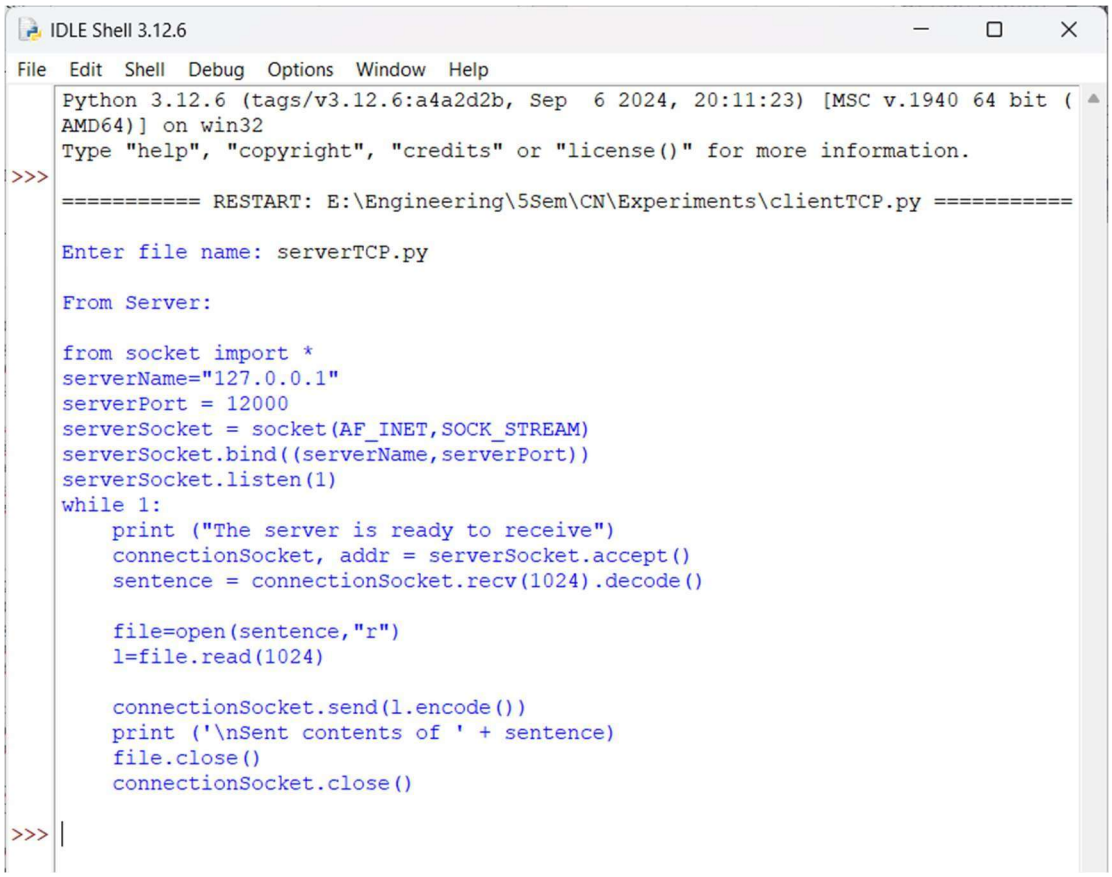
```
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)

while 1:
    print('Server is ready to receive')
    connectionSocket, addr = serverSocket.
                             accept()
    sentence = connectionSocket.recv(1024).
                             decode()
connectionSocket.close()
```

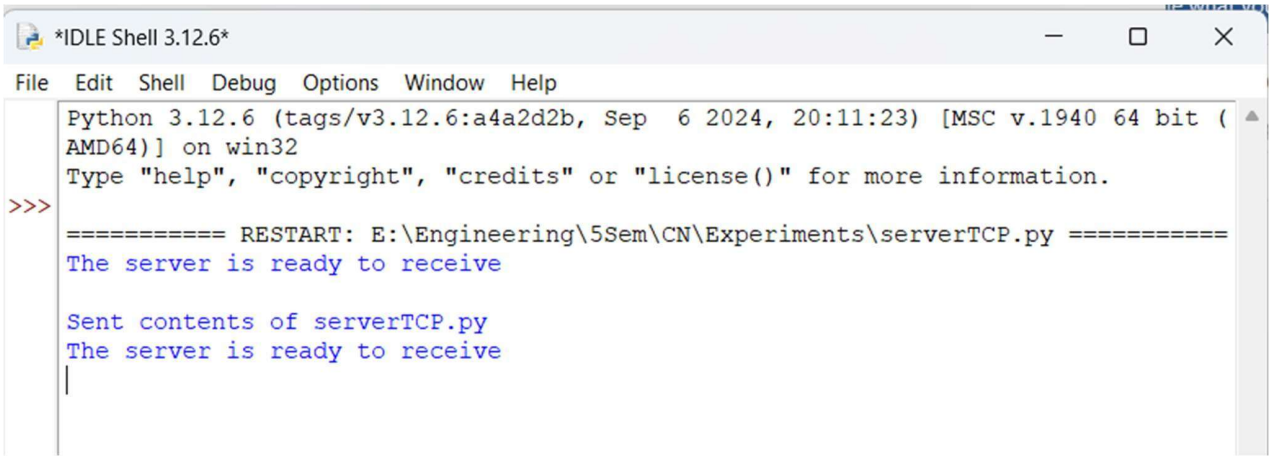
iii. Screen shot code:

```
from socket import *

serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)

while 1:
    print("The server is ready to receive")

    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()

    file = open(sentence, "r")
    l = file.read(1024)
    connectionSocket.send(l.encode())

    print(f'\nSent contents of {sentence}')

    file.close()
    connectionSocket.close()
```

```
from socket import *
serverName = '127.0.0.1'  # IP address of the server (localhost)
serverPort = 12000        # Port number on which the server is listening
clientSocket = socket(AF_INET, SOCK_STREAM)  # Create a TCP socket (IPv4, TCP)

clientSocket.connect((serverName, serverPort))  # Connect to the server

# Input the file name from the user
sentence = input("\nEnter file name: ")

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

# Receive the file contents from the server
filecontents = clientSocket.recv(1024).decode()

# Print the contents received from the server
print('\nFrom Server:\n')
print(filecontents)

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

iv. Observation:



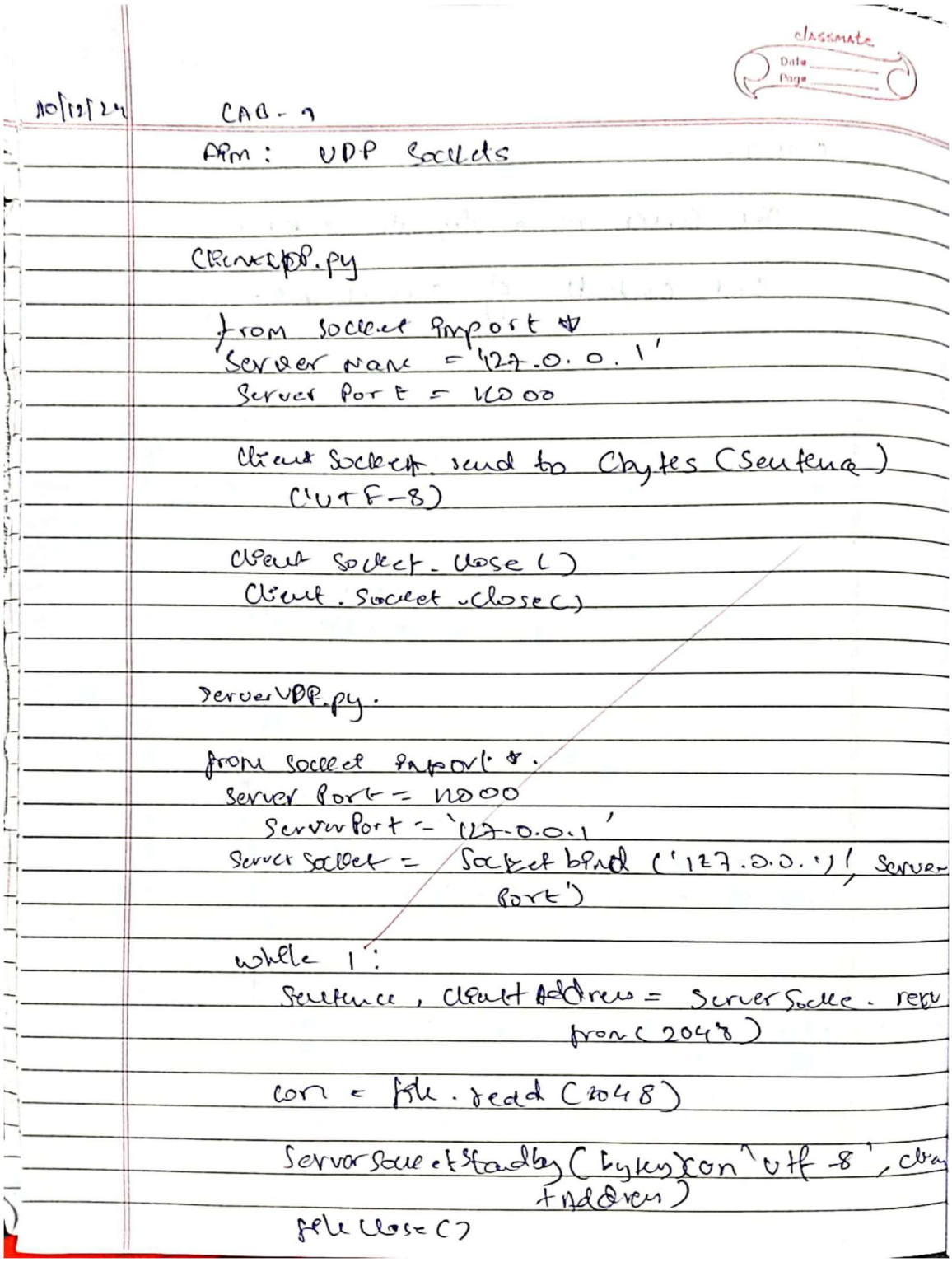


Program 4

i. Aim of the program:

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.

ii. Procedure:



10/12/24 CAO - 7

Aim: UDP Sockets

ClientUDP.py

```
from socket import *
Server Name = '127.0.0.1'
Server Port = 12000

Client Socket. send to (bytes (Sentence)
(UTF-8)

Client Socket. close ()
Client. Socket. close()
```

ServerUDP.py.

```
from socket import *.
Server Port = 12000
ServerPort = '127.0.0.1'
Server Socket = Socket bind ('127.0.0.1', Server
Port')

while 1:
Sentence, Client Address = ServerSocke. rec
from (2048)

con = file. read (2048)

ServerSocketStadby (bytes(con 'utf-8', cli
+nddress)

file close ()
```

iii. Screen shots code:

clientUDP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)

sentence = input("\nEnter file name:  ")

clientSocket.sendto(bytes(sentence,"utf-8"),(serverName,  serverPort))

filecontents,serverAddress = clientSocket.recvfrom(2048)
print ('\nReply from Server:\n')
print (filecontents.decode("utf-8"))
# for i in filecontents:
   # print(str(i), end = '')
clientSocket.close()
clientSocket.close()
```

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 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file=open(sentence,"r")
    con=file.read(2048)

    serverSocket.sendto(bytes(con,"utf-8"),clientAddress)

    print ('\nSent contents of ', end = ' ')
    print (sentence)
   #  for i in sentence:
      #  print (str(i), end = '')
    file.close()
```

iv. Observation: