VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Computer Networks (25CS5PCCON)

Submitted by

Dharunya Balavelavan (1BM23CS090)

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
Sep-2024 to Jan-2025

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 Networks (25CS5PCCON)" carried out by **Dharunya Balavelavan (1BM23CS090),** who is a bonafide student of **B.M.S. College of Engineering.** It is in partial fulfillment 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 in respect of Computer Networks (25CS5PCCON) work prescribed for the said degree.

Sarala D V
Assistant Professor
Department of CSE, BMSCE

Dr. Joythi S Nayak
Professor & HOD
Department of CSE, BMSCE

Index

15.	12/11/25	Using UDP sockets, write a client-server program to make the client send the file name and the server to send back the contents of the requested file if present.	36
16.	29/10/25	Write a program for error detecting code using CRC-CCITT (16-bits).	37

Github Link:
https://github.com/Dharunya21/Computer-networks

1.Configure DHCP within a LAN and outside LAN.

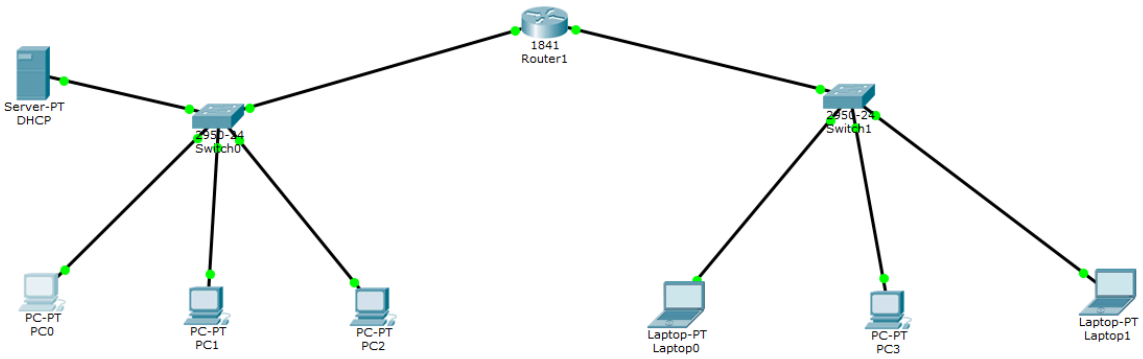
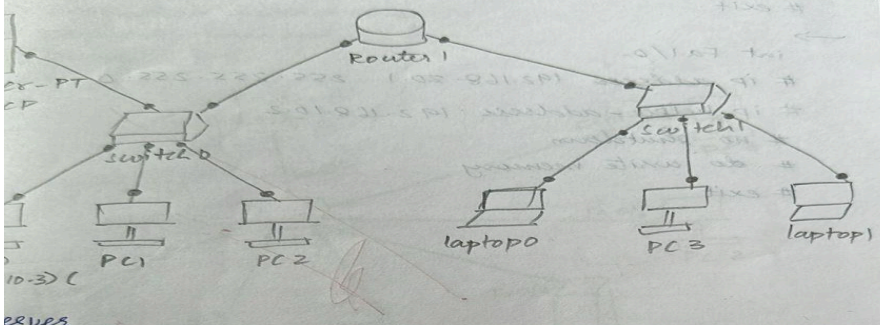


figure DHCP within a band and outside the band.

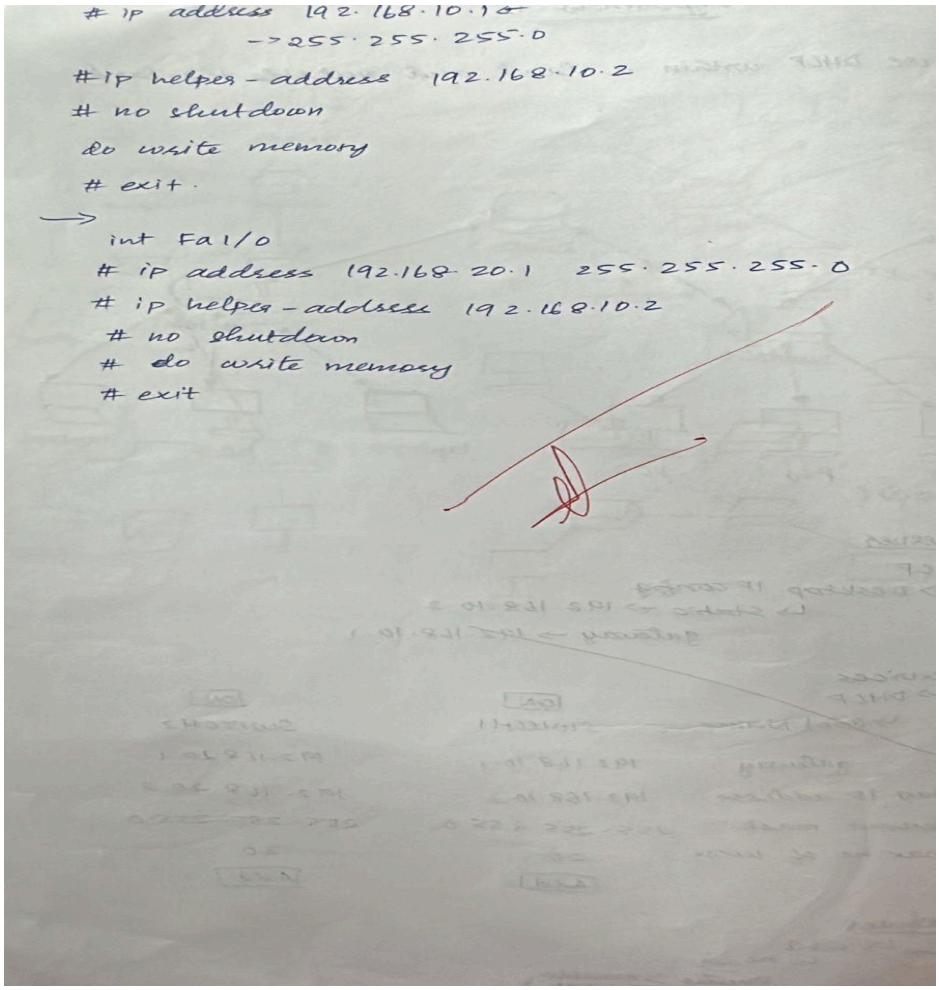


erver
CP
> Desktop IP config
↳ static → 192.168.10.2
gateway → 192.168.10.1

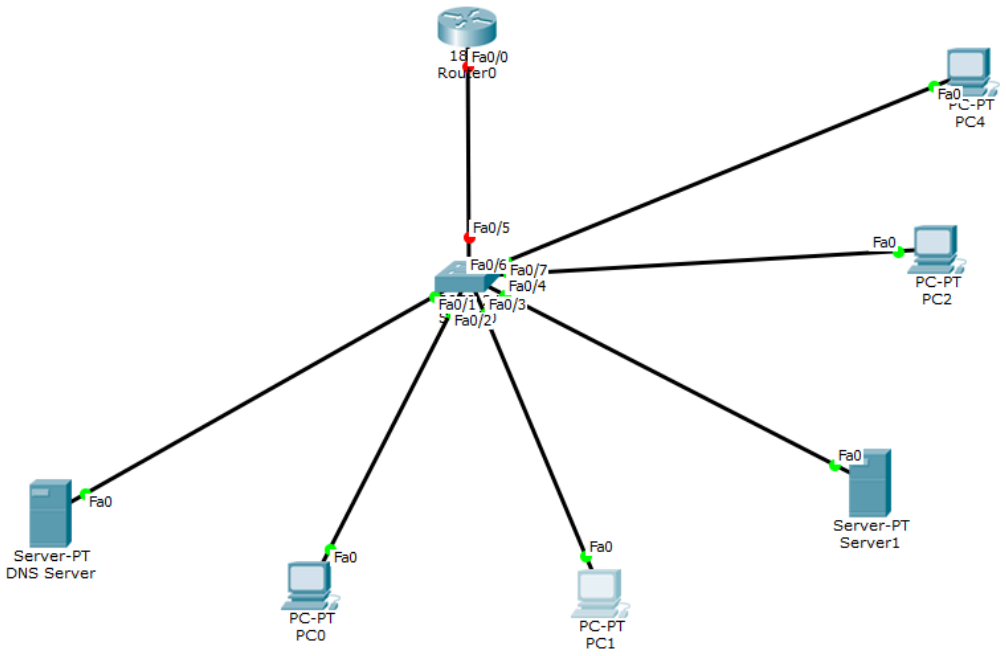
vices
> DHCP

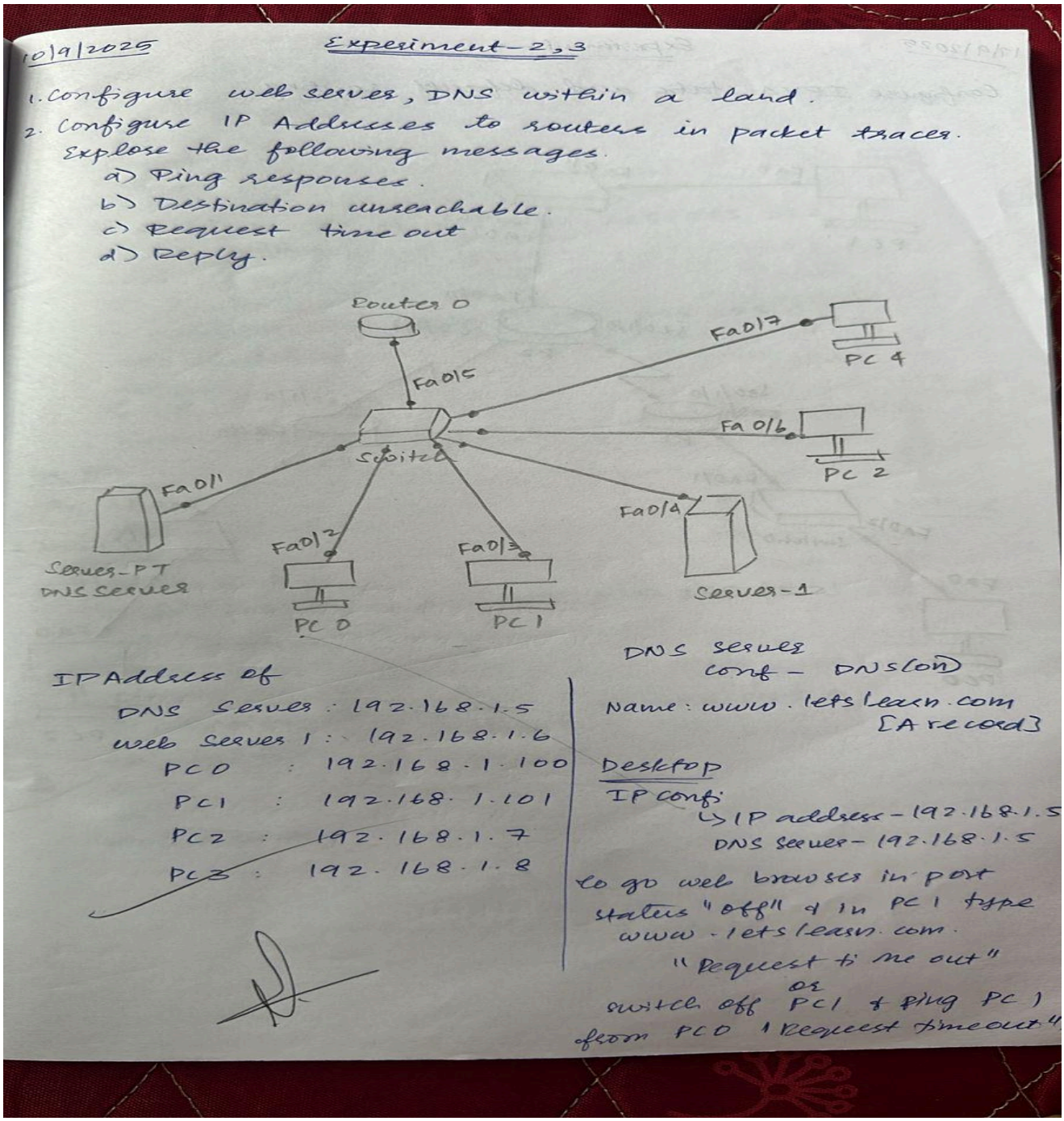
	ON	ON
↳ Pool Name	SWITCH1	SWITCH2
gateway	192.168.10.1	192.168.20.1
t IP address	192.168.10.3	192.168.20.2
bmit mask	255.255.255.0	255.255.255.0
x. no. of users	20	20
	Add	Add

uter
↳ CLI
↳ no ↵
Router > enable
conf t
int Fa 0/0

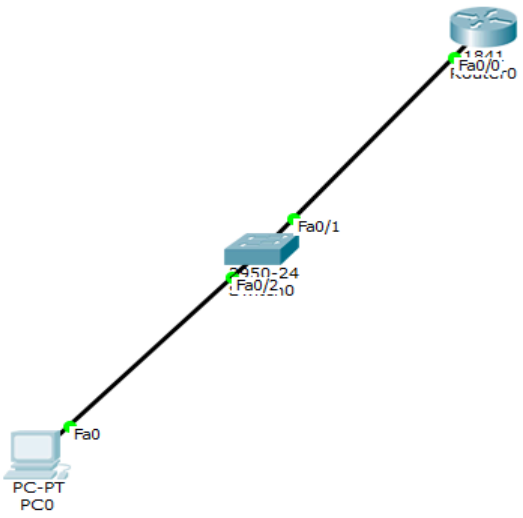


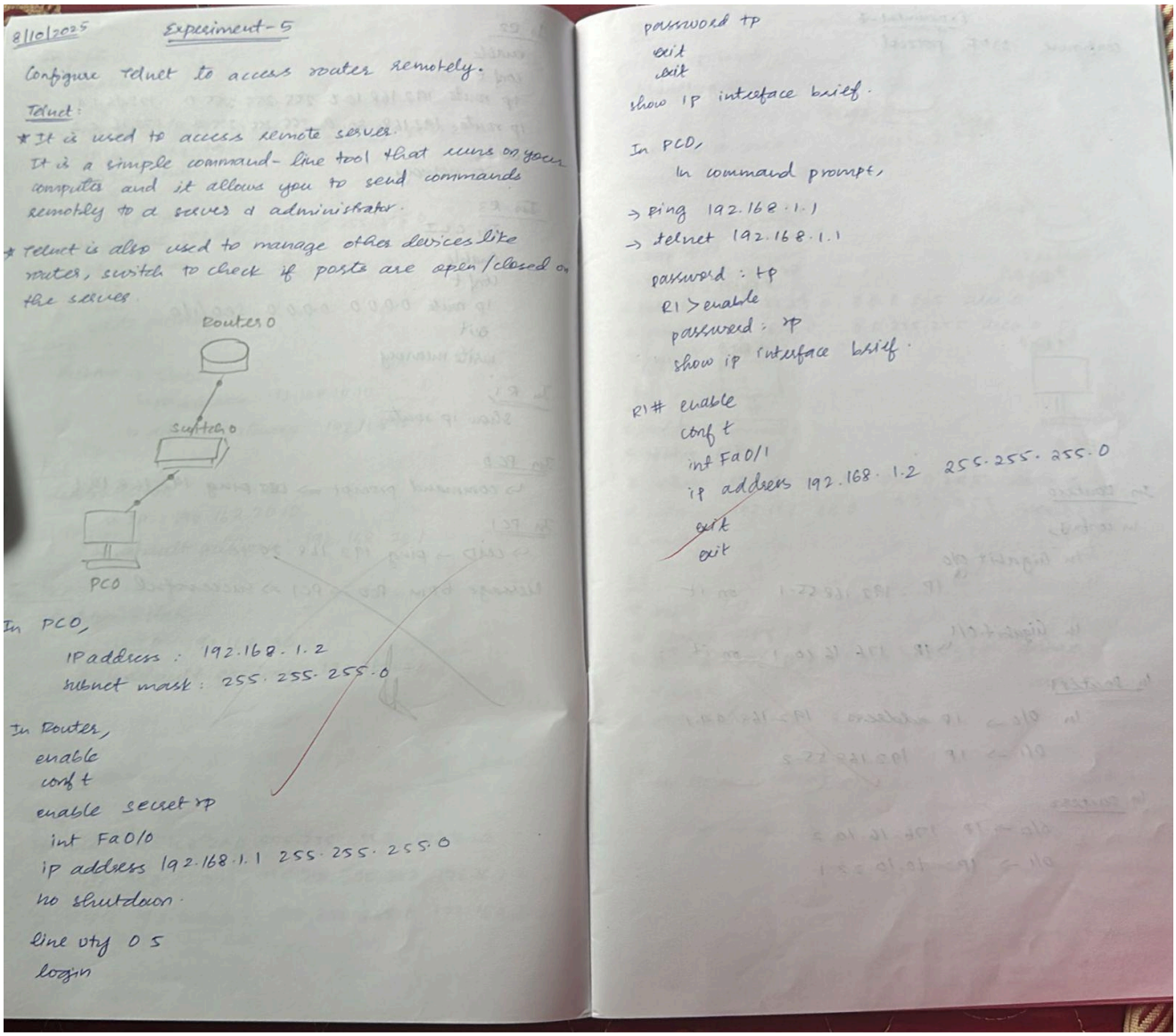
2.Configure Web Server, DNS within a LAN.


Fa0/0
Router0
Fa0
PC-PT
PC4
Fa0/5
Fa0/6
Fa0/7
Fa0/4
Fa0/3
Fa0
PC-PT
PC2
Fa0
Server-PT
DNS Server
Fa0
PC-PT
PC0
Fa0
PC-PT
PC1
Fa0
Server-PT
Server1

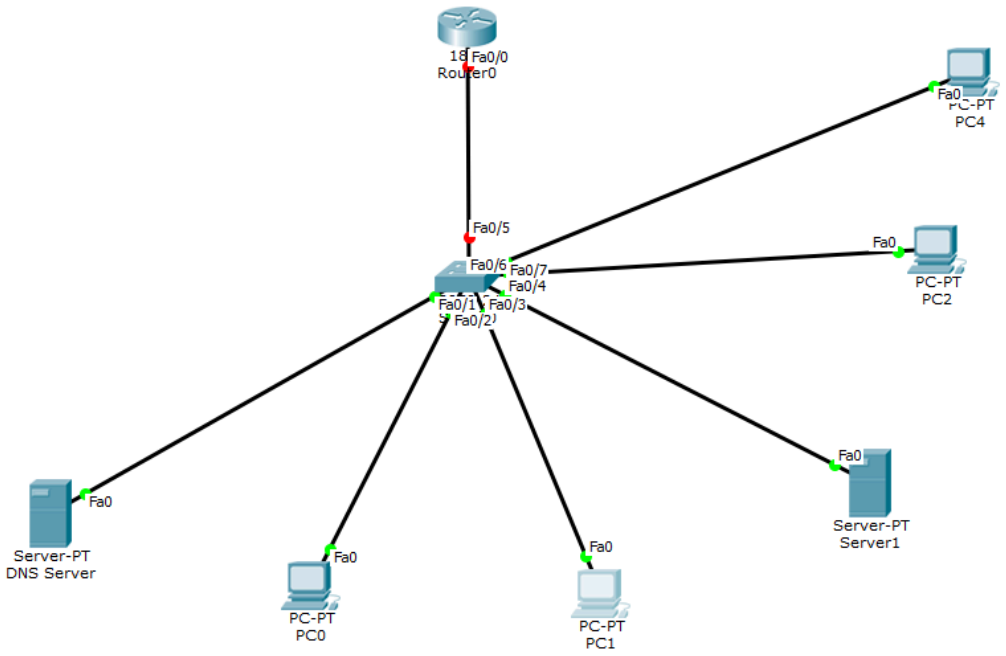


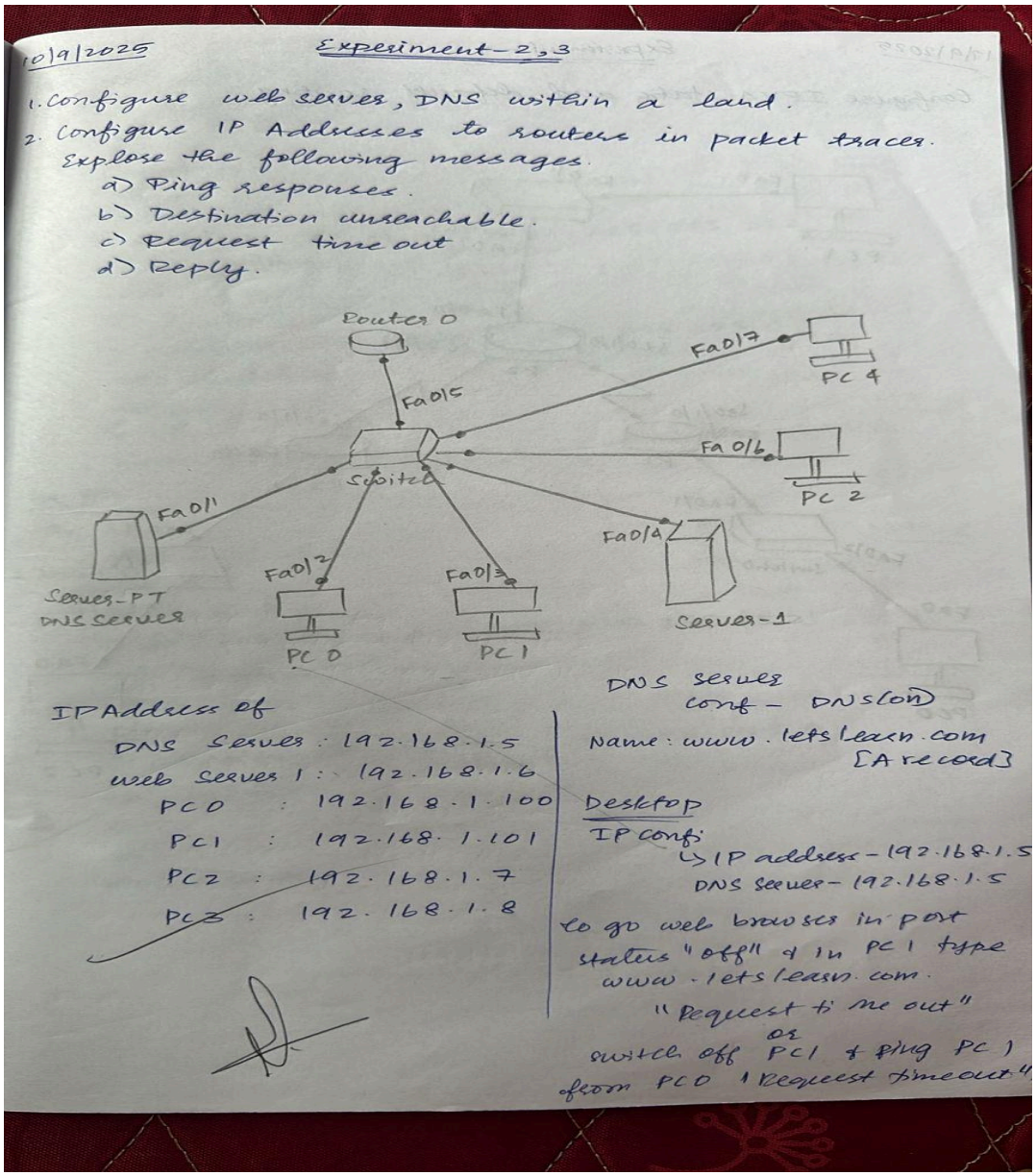
3.To understand the operation of TELNET by accessing the router in the server room from a PC in the IT office.


Fa0/1
Fa0/2
Fa0
PC-PT
PC0

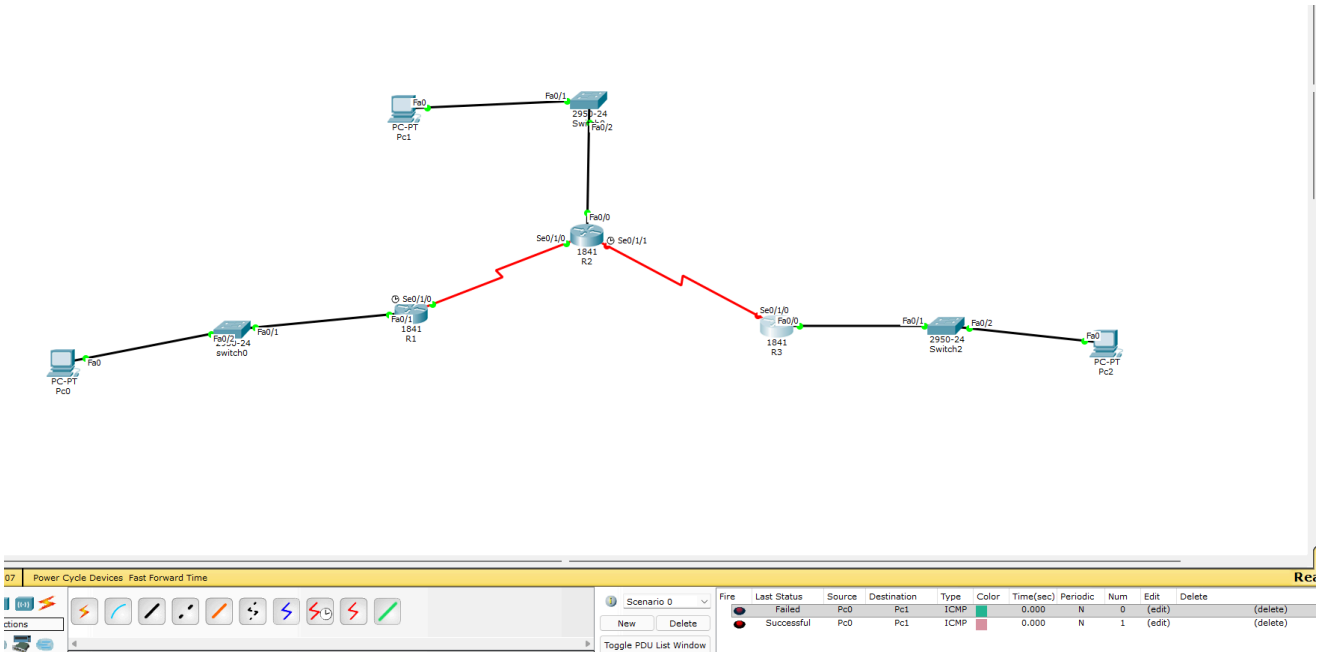


4.Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.


Fa0/0
Router0
Fa0
PC-PT
PC4
Fa0/5
Fa0/6
Fa0/7
Fa0/4
Fa0/3
Fa0
PC-PT
PC2
Fa0
Server-PT
DNS Server
Fa0
PC-PT
PC0
Fa0
PC-PT
PC1
Fa0
Server-PT
Server1



5.Configure default route, static route to the Router.

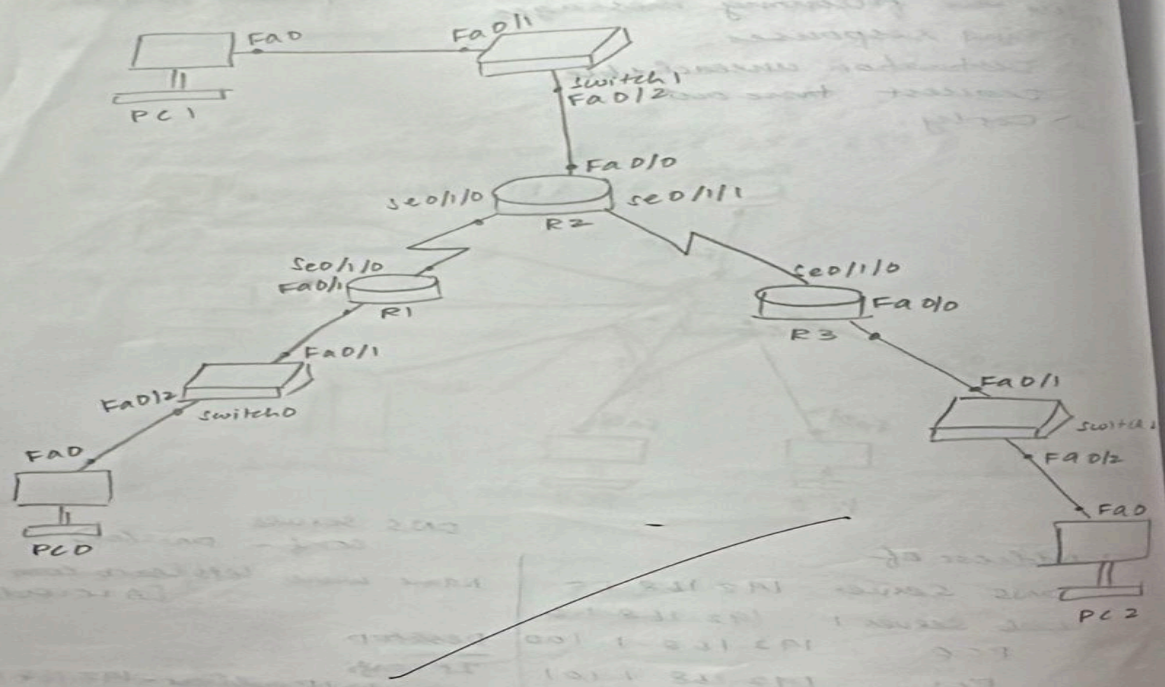

Fa0
PC-PT
Pc1
Fa0/1
2950-24
Fa0/2
Fa0/0
Se0/1/0
1841
R2
Se0/1/1
Se0/1/0
Fa0/1
1841
R1
Fa0/2
2950-24
switch0
Fa0/1
Fa0
PC-PT
Pc0
Se0/1/0
Fa0/0
1841
R3
Fa0/1
2950-24
Switch2
Fa0/2
Fa0
PC-PT
Pc2

07
Power Cycle Devices
Fast Forward Time
Rea
ctions
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time(sec)
Periodic
Num
Edit
Delete
Failed
Pc0
Pc1
ICMP
0.000
N
0
(edit)
(delete)
Successful
Pc0
Pc1
ICMP
0.000
N
1
(edit)
(delete)

17/9/2025

Experiment – 4

Configure IPV4 static and default routing



For R1

In physical → off switch
drag & drop HWIC-2T

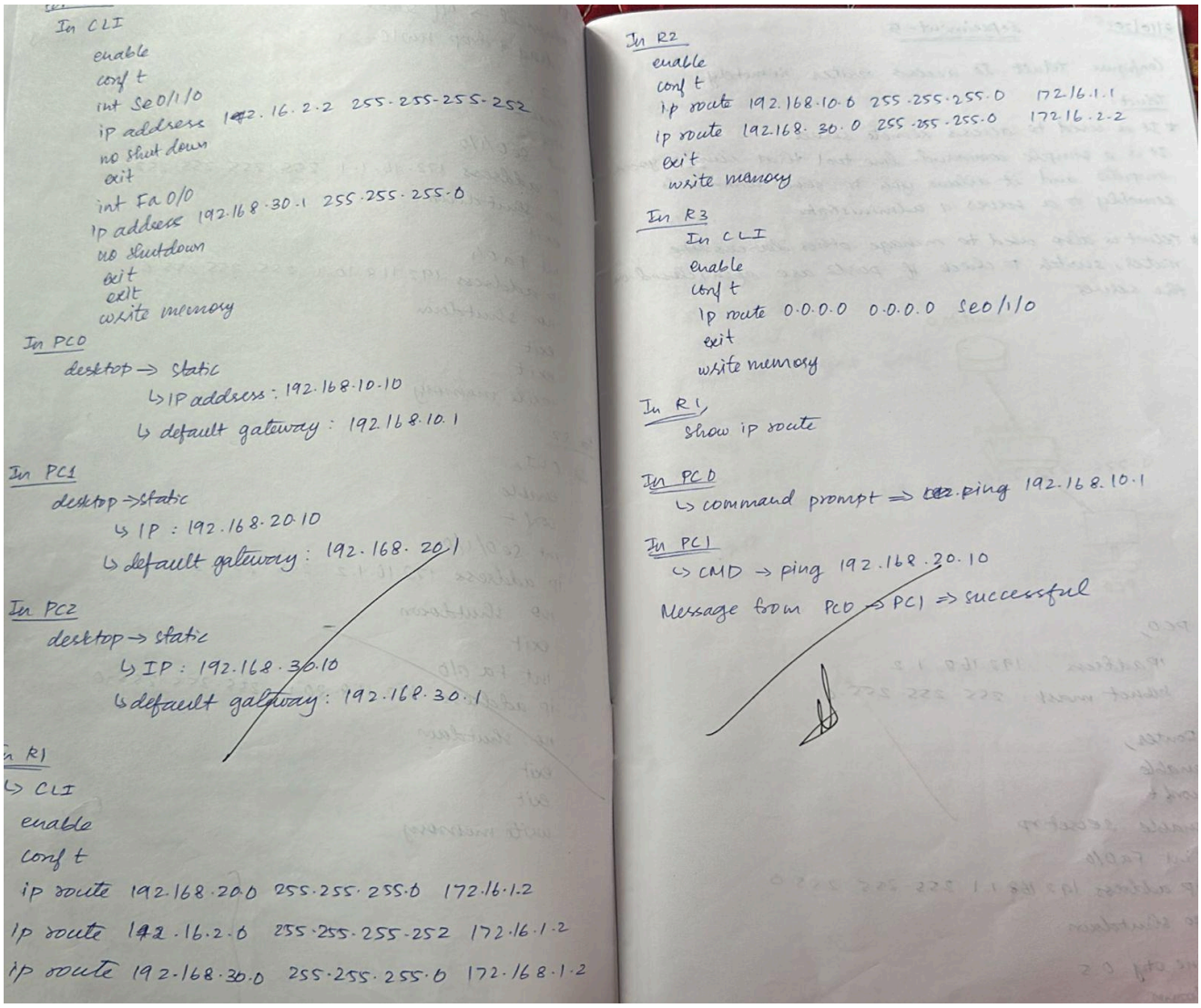
In CLI,

```
enable
conf t
int Se0/1/0
ip address 172.16.1.1 255.255.255.252
no shutdown
exit
int Fa0/1
ip address 192.168.10.1 255.255.255.0
no shutdown
exit
exit
write memory
```

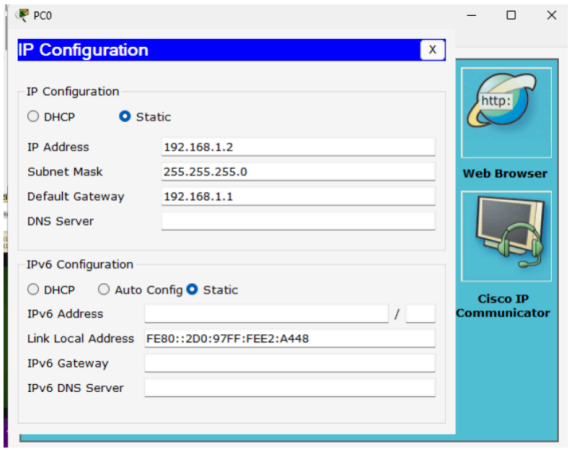
For R2

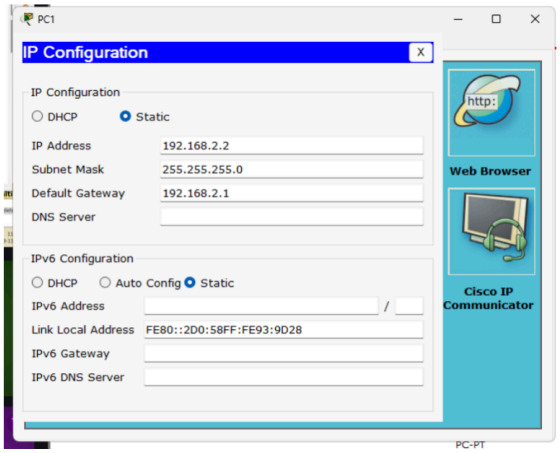
In CLI,

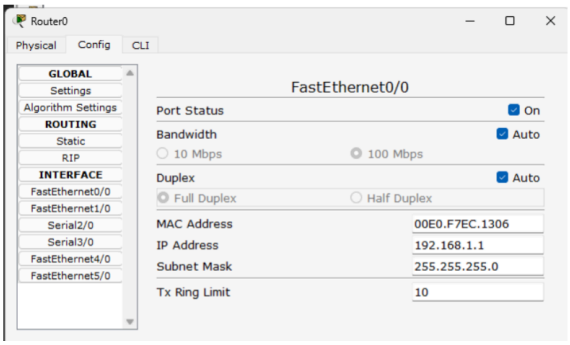
```
enable
conf t
int Se0/1/0
ip address 172.16.1.2 255.255.255.252
no shutdown
exit
int Fa 0/0
ip address 192.168.20.1 255.255.255.0
no shutdown
exit
exit
write memory
```

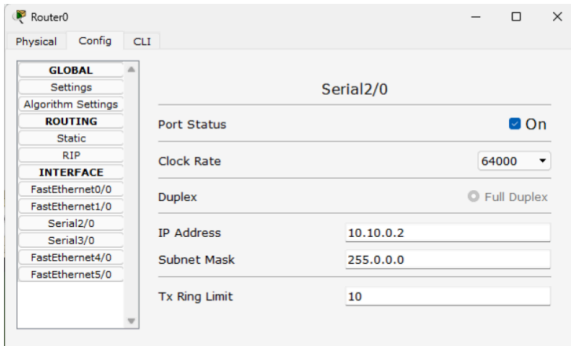


6.Configure RIP routing Protocol in Routers.

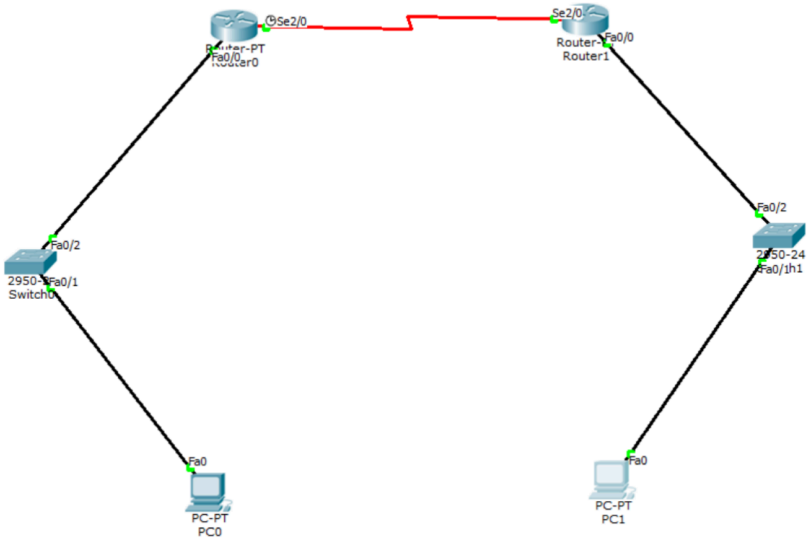

PC0
IP Configuration
IP Configuration
DHCP
Static
IP Address 192.168.1.2
Subnet Mask 255.255.255.0
Default Gateway 192.168.1.1
DNS Server
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
Link Local Address FE80::2D0:97FF:FEE2:A448
IPv6 Gateway
IPv6 DNS Server
Web Browser
Cisco IP Communicator

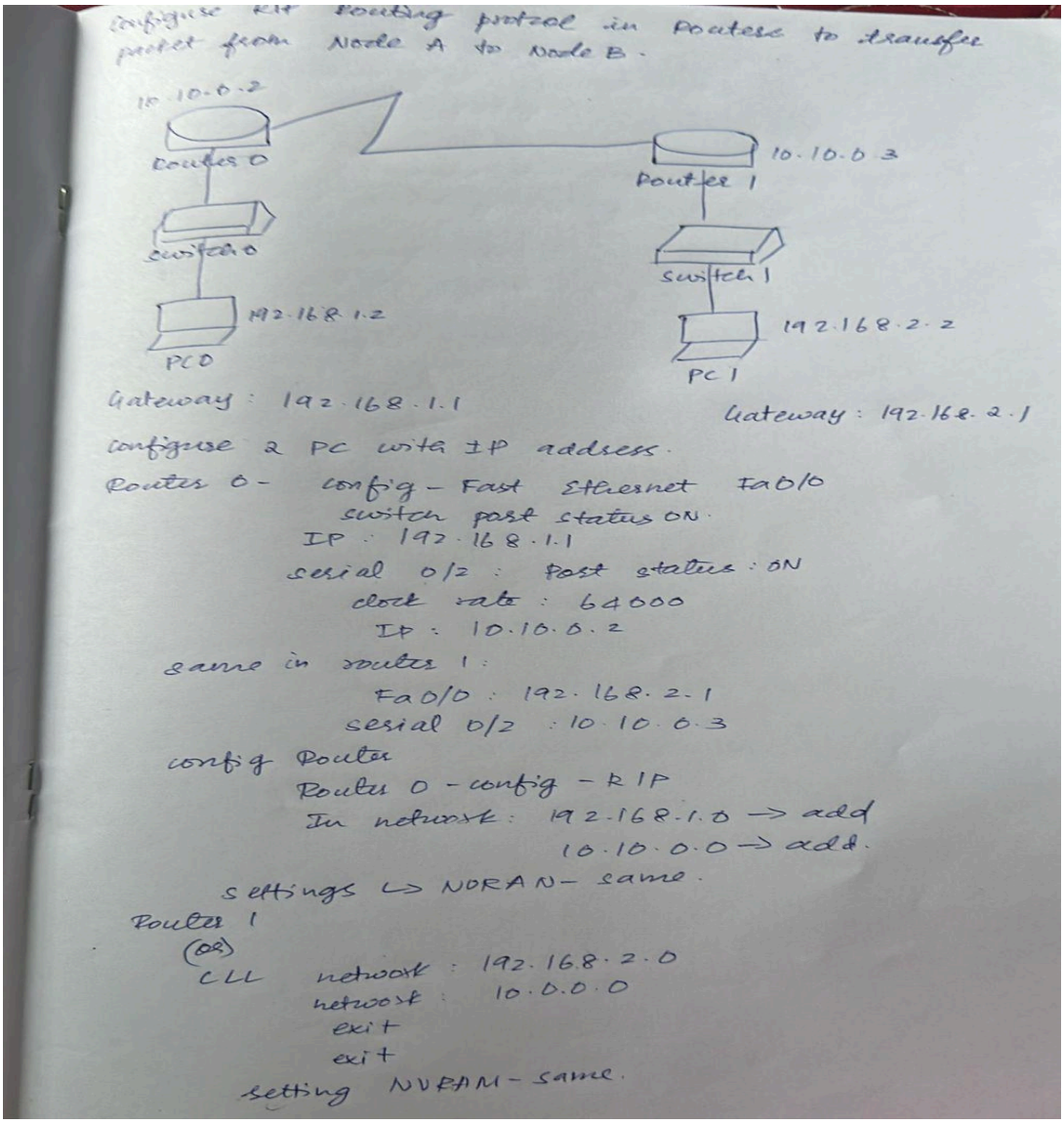

PC1
IP Configuration
IP Configuration
DHCP
Static
IP Address 192.168.2.2
Subnet Mask 255.255.255.0
Default Gateway 192.168.2.1
DNS Server
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
Link Local Address FE80::2D0:58FF:FE93:9D28
IPv6 Gateway
IPv6 DNS Server
Web Browser
Cisco IP Communicator
PC-PT


Router0
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
FastEthernet0/0
Port Status On
Bandwidth Auto
10 Mbps
100 Mbps
Duplex Auto
Full Duplex
Half Duplex
MAC Address 00E0.F7EC.1306
IP Address 192.168.1.1
Subnet Mask 255.255.255.0
Tx Ring Limit 10

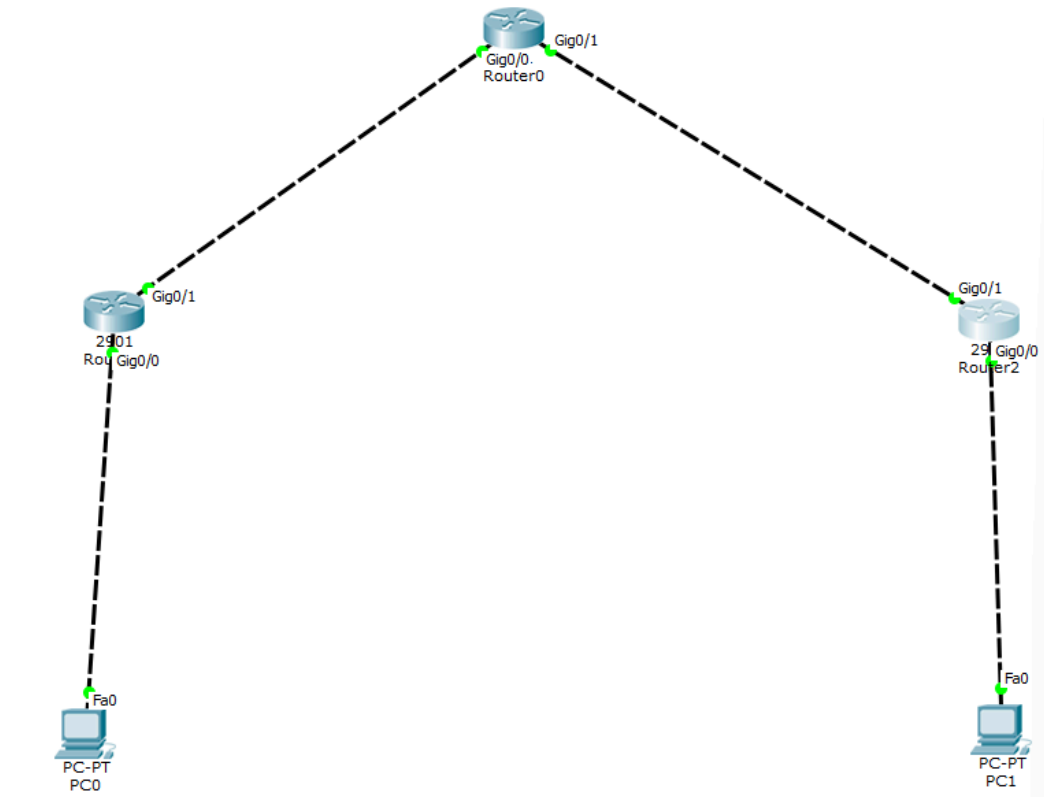

Router0
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
Serial2/0
Port Status On
Clock Rate 64000
Duplex Full Duplex
IP Address 10.10.0.2
Subnet Mask 255.0.0.0
Tx Ring Limit 10

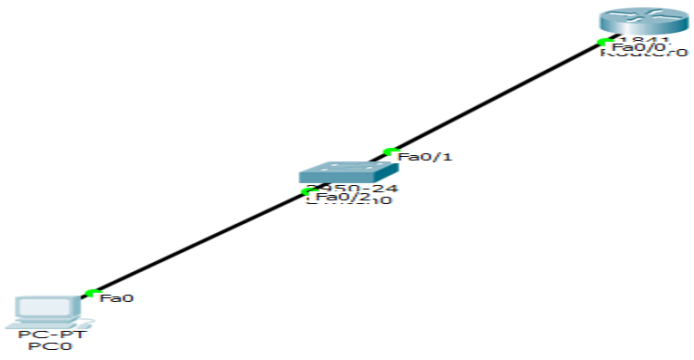

Router1
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
RIP Routing
Network
Add
Network Address
10.0.0.0
192.168.2.0
Remove
Equivalent IOS Commands
Router(config-if)#exit
Router(config)#interface Serial2/0
Router(config-if)#
Router(config-if)#exit
Router(config)#router rip
Router(config-router)#
Tx Ring Limit 10

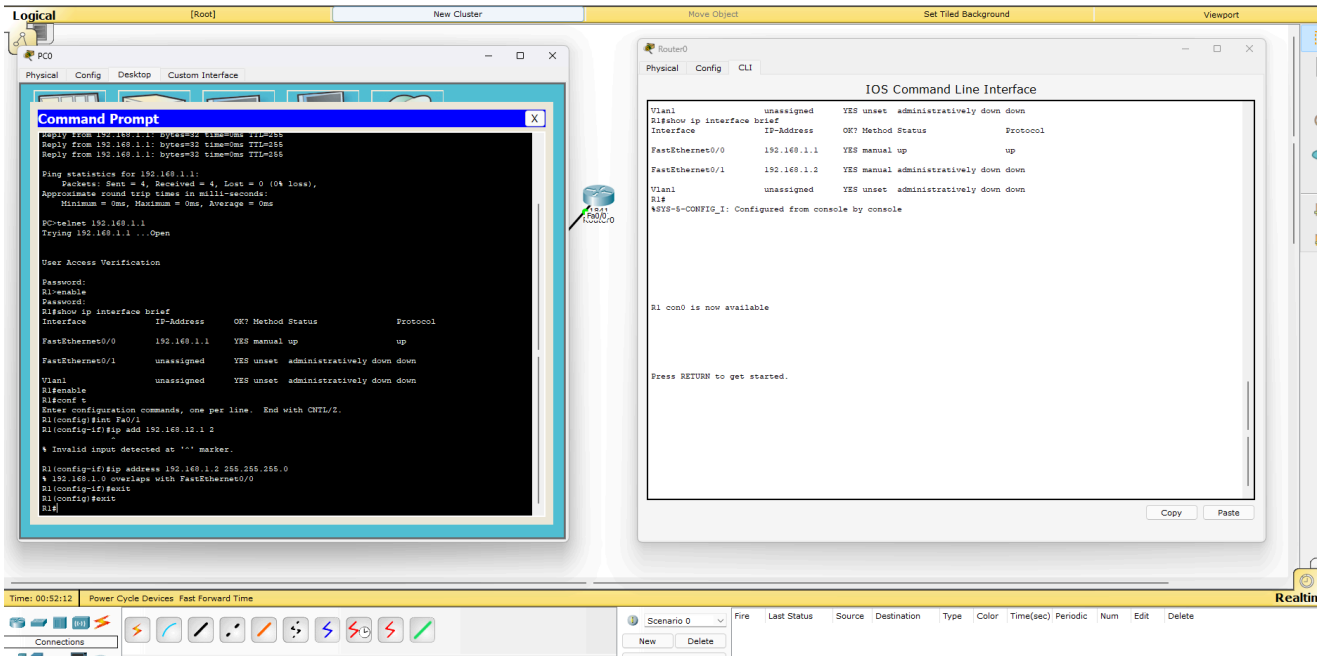

Se2/0
Fa0/0
Router1
Fa0/2
Fa0/2
Fa0/1
Switch0
Fa0
PC-PT
PC0
Fa0
PC-PT
PC1

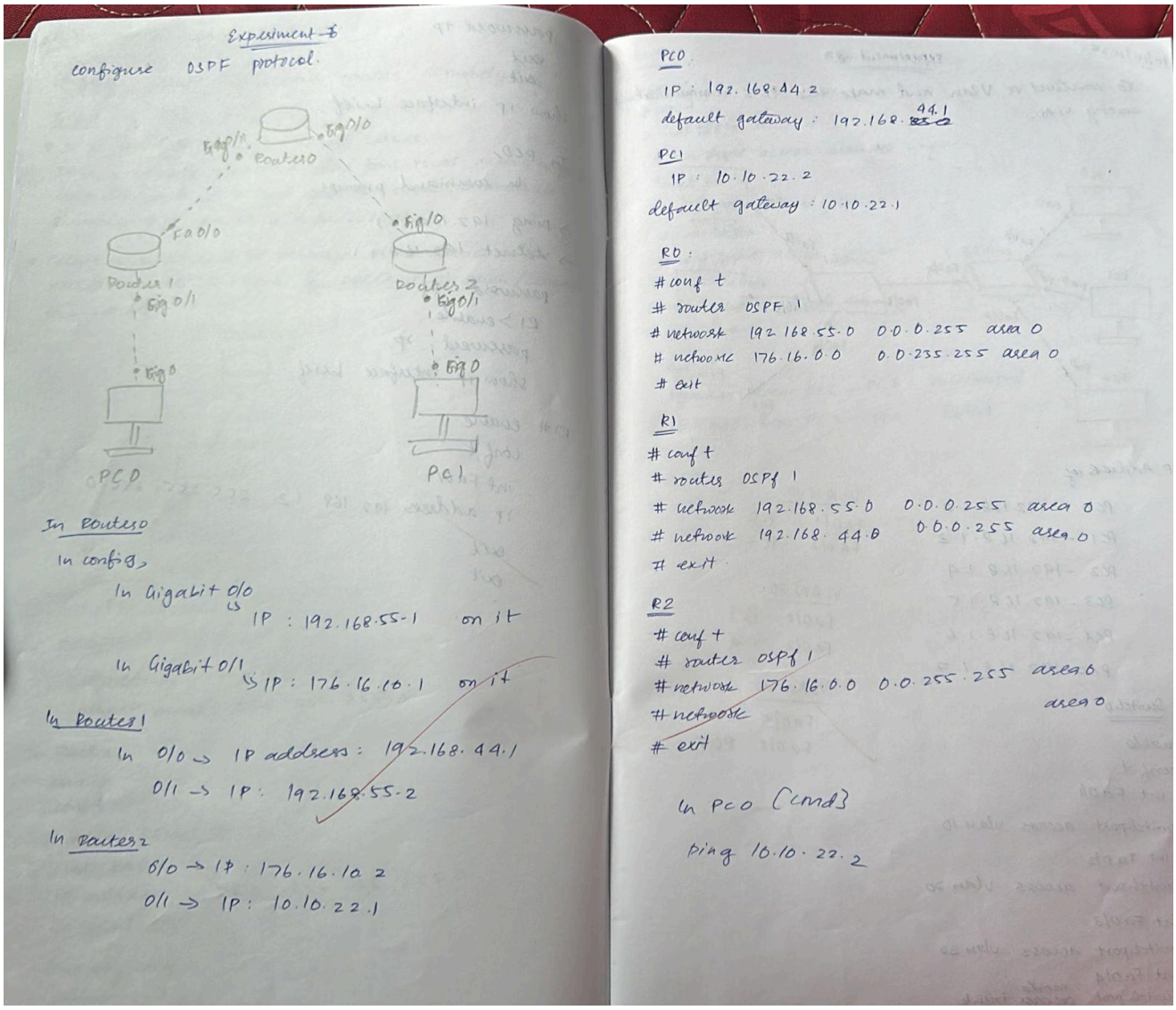


7.Configure OSPF routing protocol.

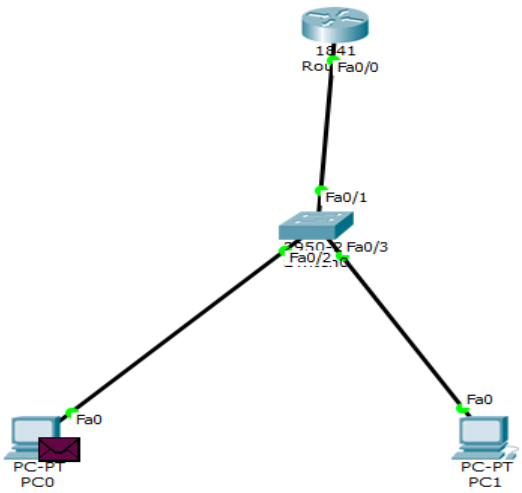

Gig0/1
Gig0/0
Router0
Gig0/1
2901
Gig0/0
Gig0/1
Gig0/0
Router2
Fa0
PC-PT
PC0
Fa0
PC-PT
PC1

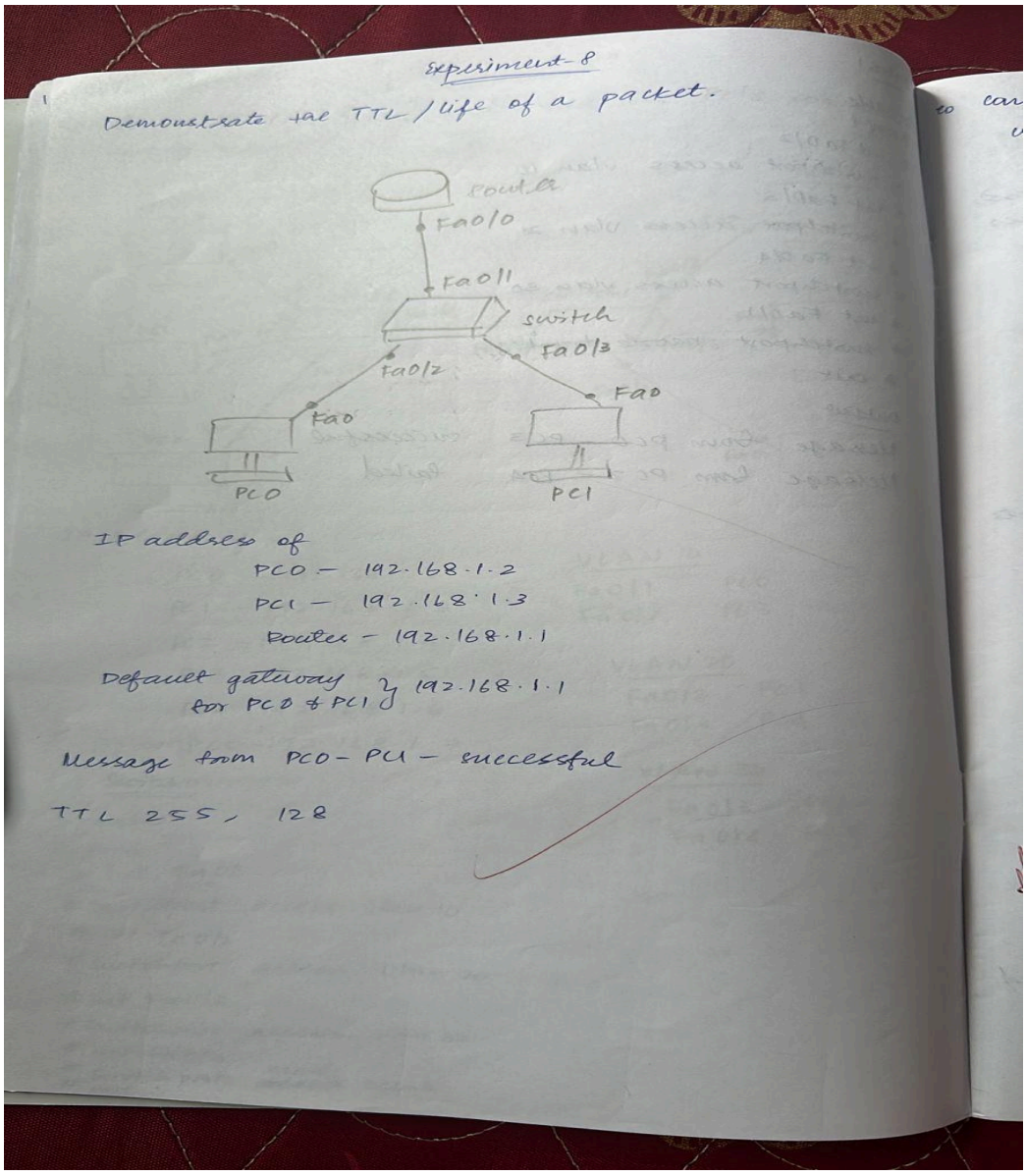

Fa0/1
Fa0
PC-PT
PC0


Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
PC0
Physical
Config
Desktop
Custom Interface
Command Prompt
Reply from 192.168.1.1: bytes=32 time=0ms TTL=255
Reply from 192.168.1.1: bytes=32 time=0ms TTL=255
Reply from 192.168.1.1: bytes=32 time=0ms 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
PC>telnet 192.168.1.1
Trying 192.168.1.1 ...Open
User Access Verification
Password:
R1>enable
Password:
R1#show ip interface brief
Interface IP-Address OK? Method Status Protocol
FastEthernet0/0 192.168.1.1 YES manual up up
FastEthernet0/1 unassigned YES unset administratively down down
Vlan1 unassigned YES unset administratively down down
R1#enable
R1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#int Fa0/1
R1(config-if)#ip add 192.168.12.1 2
% Invalid input detected at '^' marker.
R1(config-if)#ip address 192.168.1.2 255.255.255.0
% 192.168.1.0 overlaps with FastEthernet0/0
R1(config-if)#exit
R1(config)#exit
R1#
Router0
Physical
Config
CLI
IOS Command Line Interface
Vlan1 unassigned YES unset administratively down down
R1#show ip interface brief
Interface IP-Address OK? Method Status Protocol
FastEthernet0/0 192.168.1.1 YES manual up up
FastEthernet0/1 192.168.1.2 YES manual administratively down down
Vlan1 unassigned YES unset administratively down down
R1#
%SYS-5-CONFIG_I: Configured from console by console
R1 con0 is now available
Press RETURN to get started.
Copy
Paste
Time: 00:52:12
Power Cycle Devices
Fast Forward Time
Connections
Scenario 0
New
Delete
Fire
Last Status
Source
Destination
Type
Color
Time(sec)
Periodic
Num
Edit
Delete

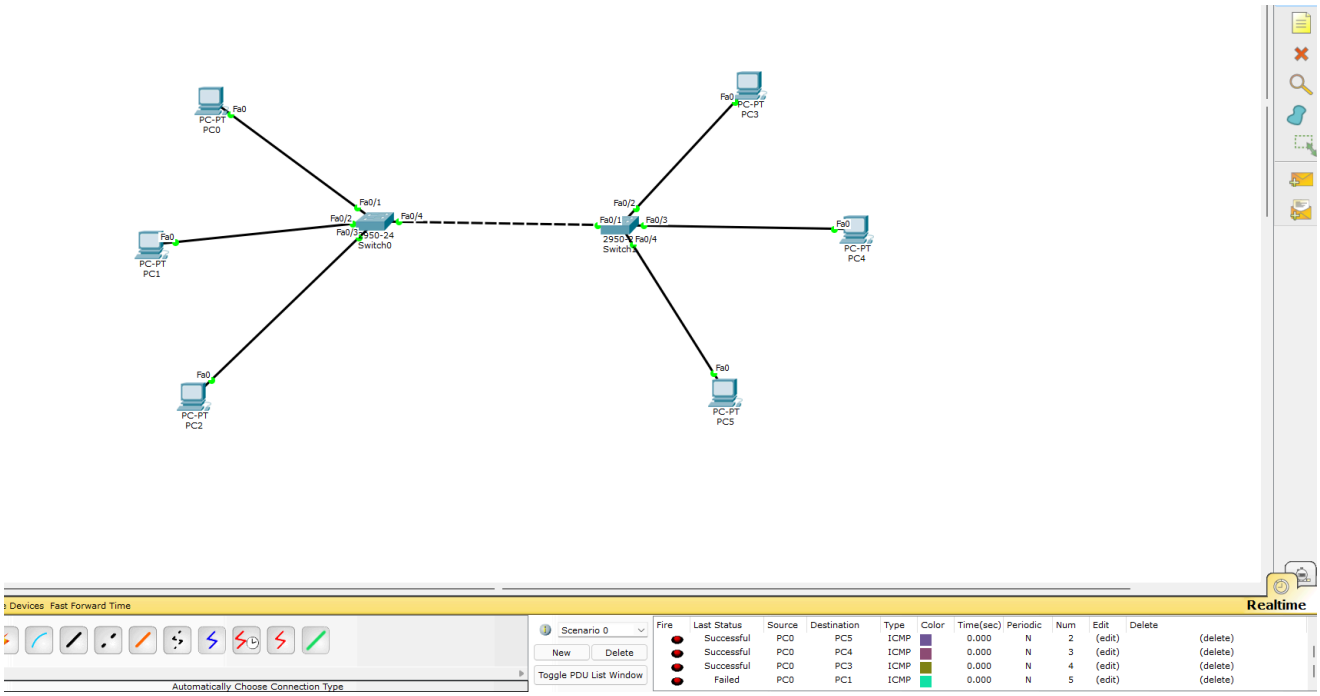


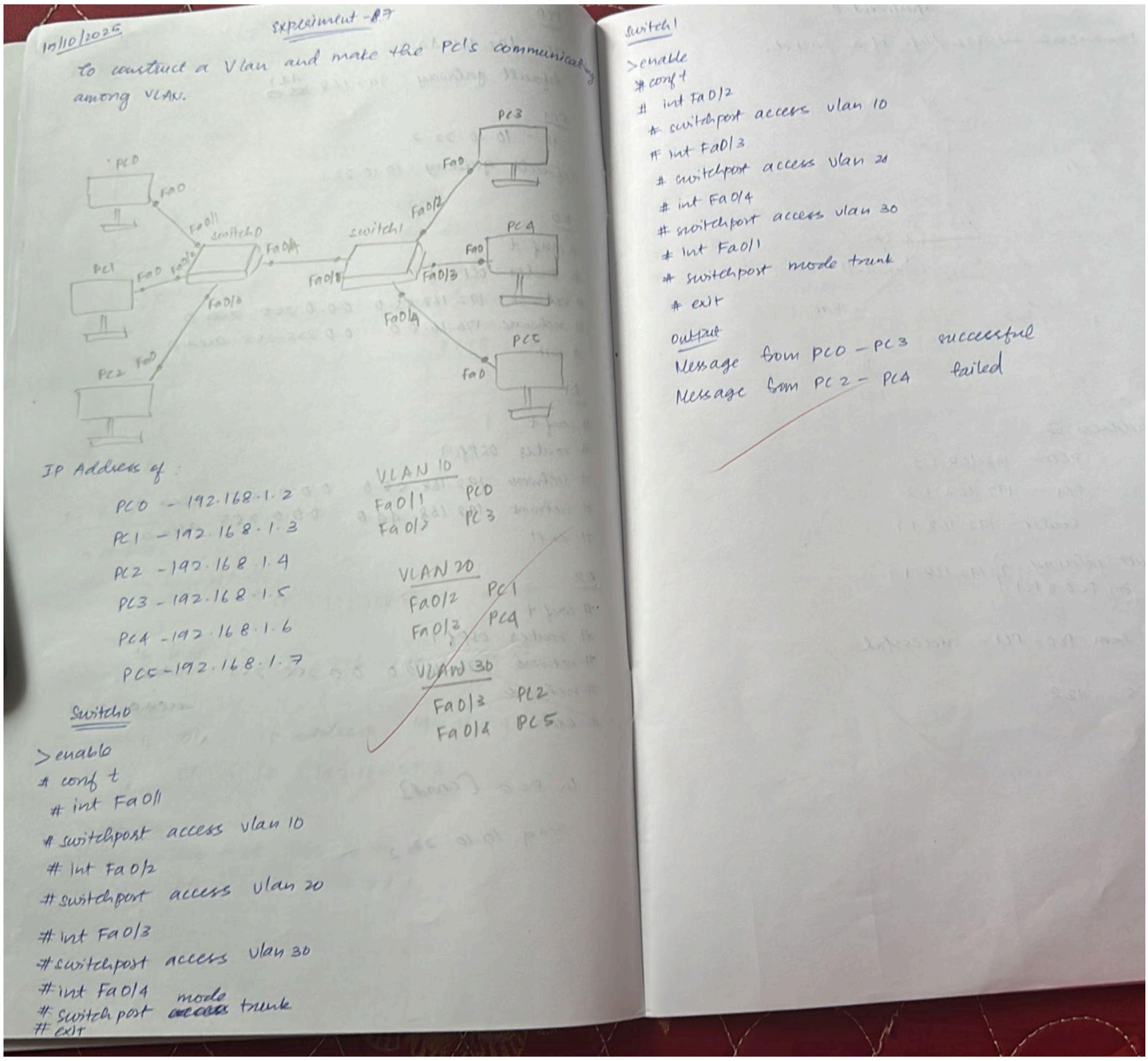
8.Demonstrate the TTL/ Life of a Packet.


Fa0/0
Fa0/1
Fa0/3
Fa0
Fa0
PC-PT
PC0
PC-PT
PC1

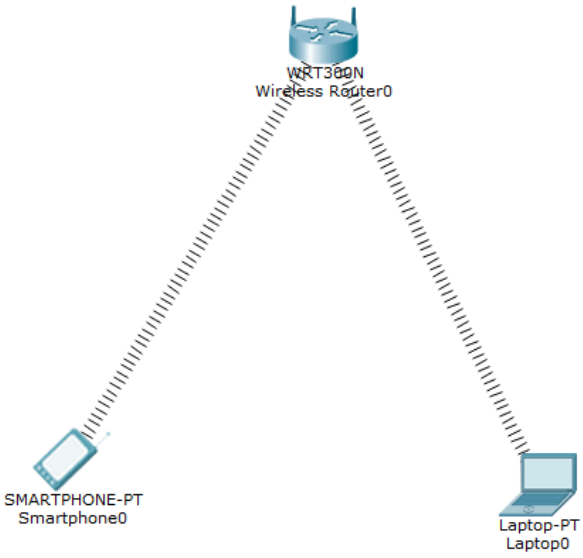


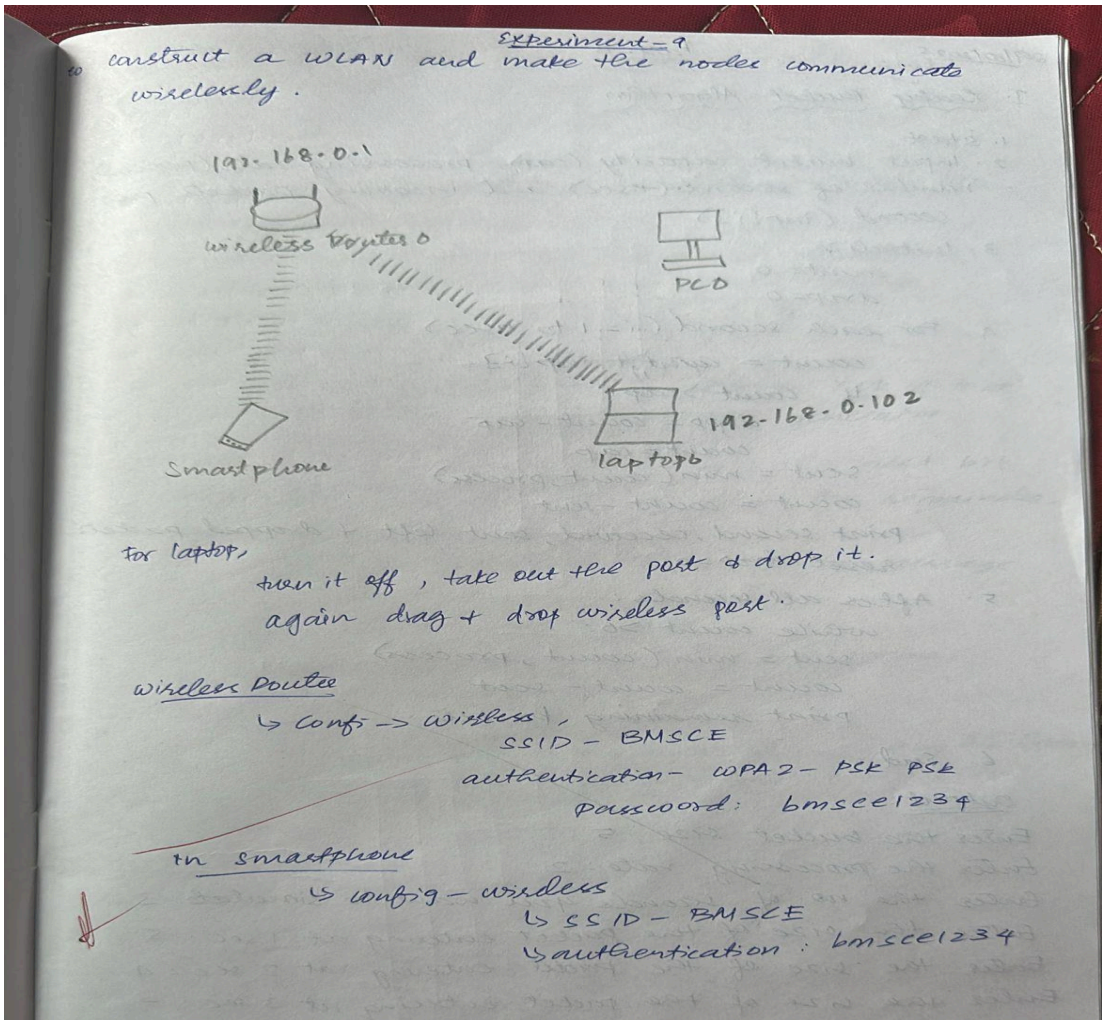
9.To construct a VLAN and make the PC's communicate among a VLAN.


PC-PT PC0
Fa0
PC-PT PC1
Fa0
PC-PT PC2
Fa0
Fa0/1
Fa0/2
Fa0/3
Fa0/4
2950-24 Switch0
Fa0/1
Fa0/2
Fa0/3
Fa0/4
2950-24 Switch1
PC-PT PC3
Fa0
PC-PT PC4
Fa0
PC-PT PC5
Fa0
Devices Fast Forward Time
Realtime
Automatically Choose Connection Type
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time(sec)
Periodic
Num
Edit
Delete
Successful PC0 PC5 ICMP 0.000 N 2 (edit) (delete)
Successful PC0 PC4 ICMP 0.000 N 3 (edit) (delete)
Successful PC0 PC3 ICMP 0.000 N 4 (edit) (delete)
Failed PC0 PC1 ICMP 0.000 N 5 (edit) (delete)



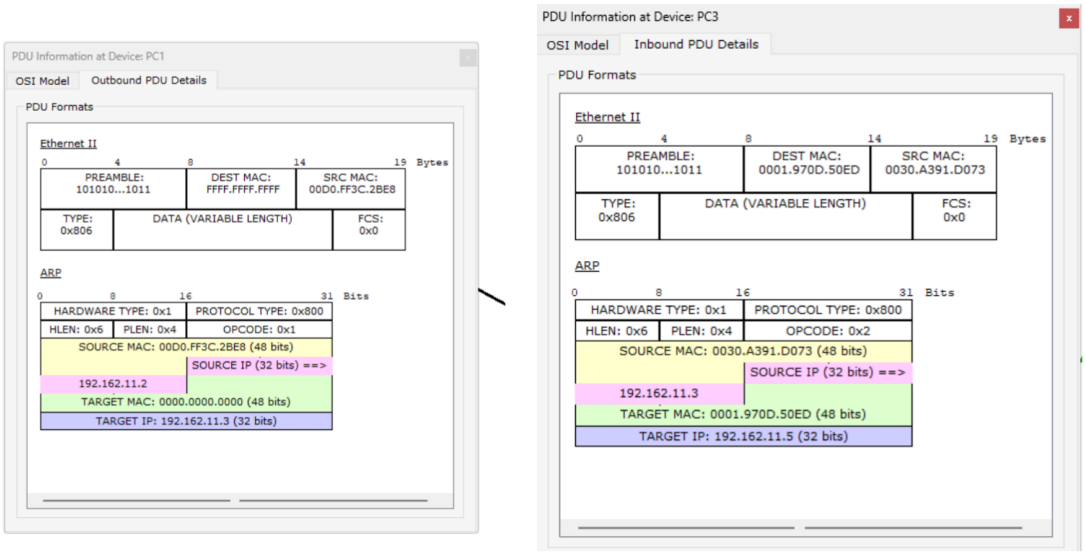
10.To construct a WLAN and make the nodes communicate wirelessly.


WRT300N
Wireless Router0
SMARTPHONE-PT
Smartphone0
Laptop-PT
Laptop0

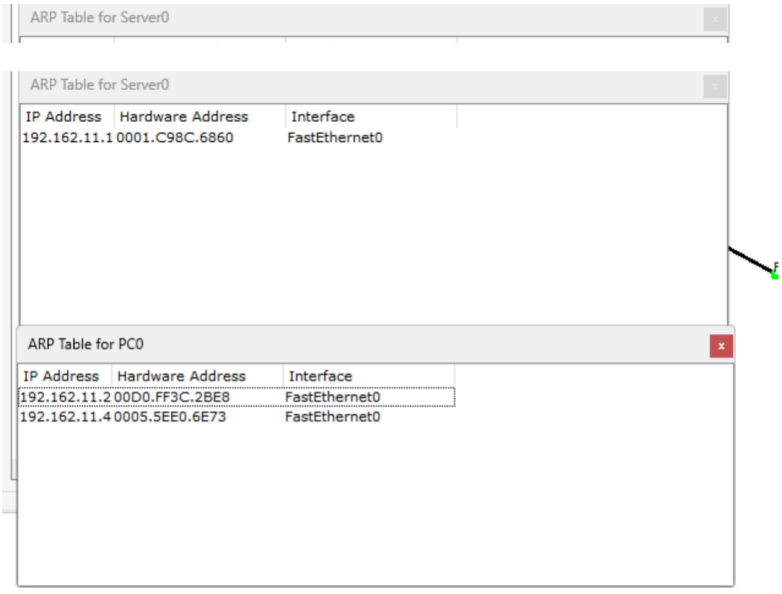


11.To construct a simple LAN and understand the concept and operation of Address Resolution Protocol (ARP).

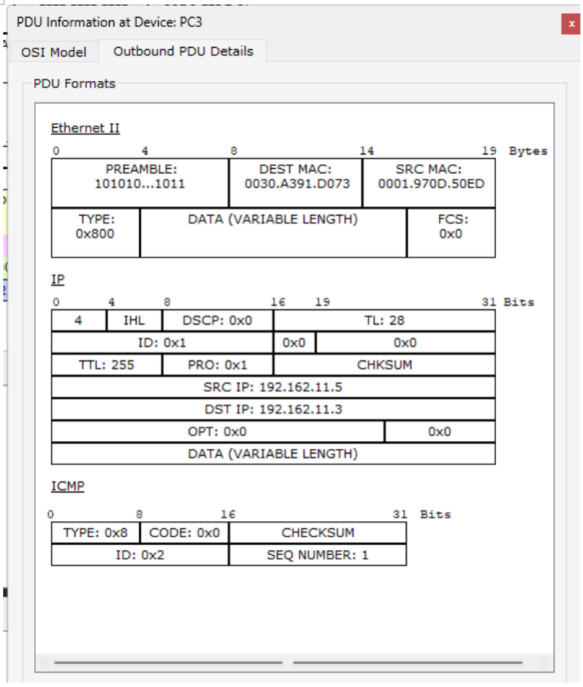
ARP



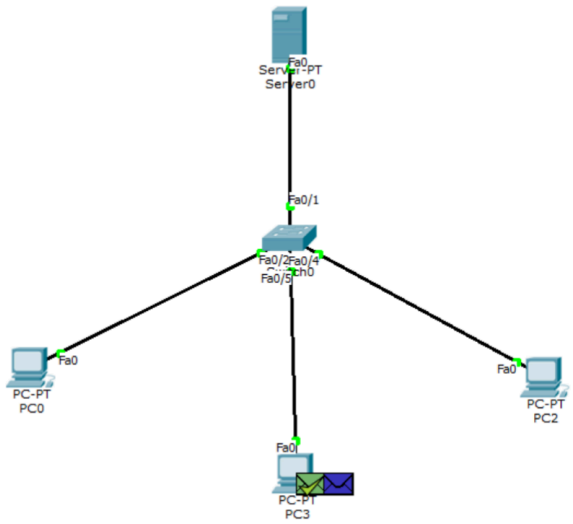
THE MAC ADDRESS before and after encrypting and sending



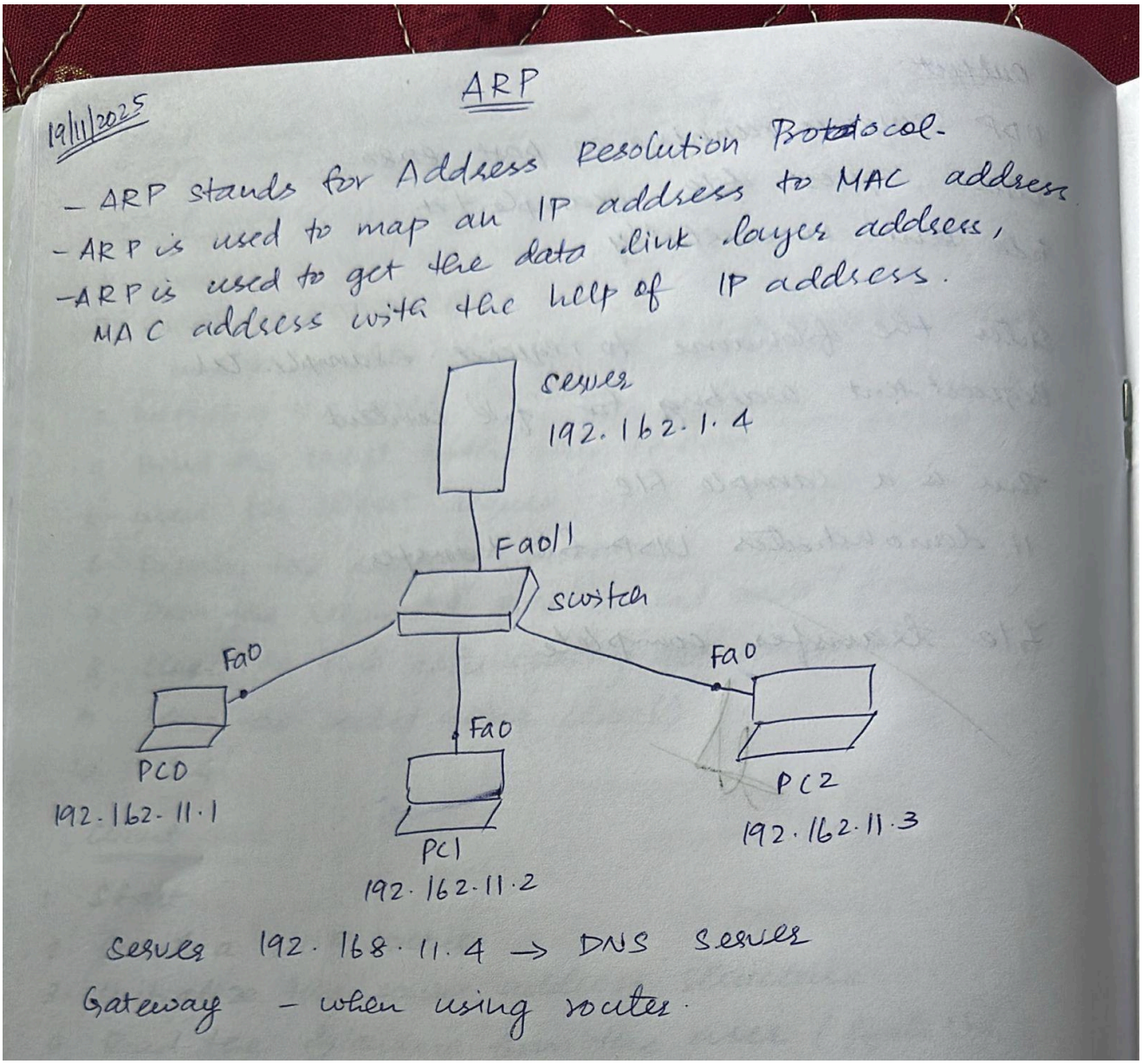
ARP table of pc0 and server after sending a simple PDU



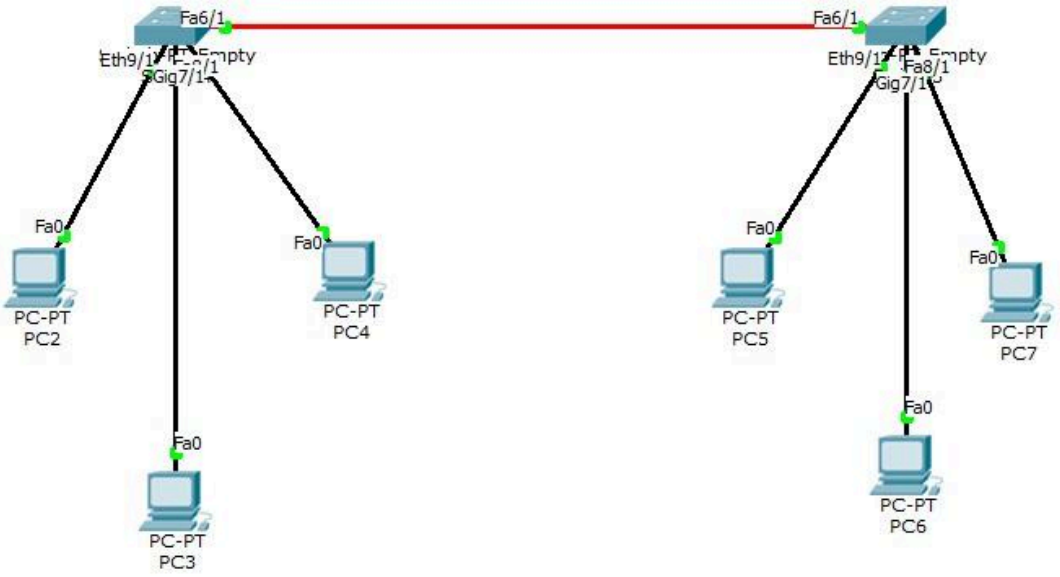
The address of sent one

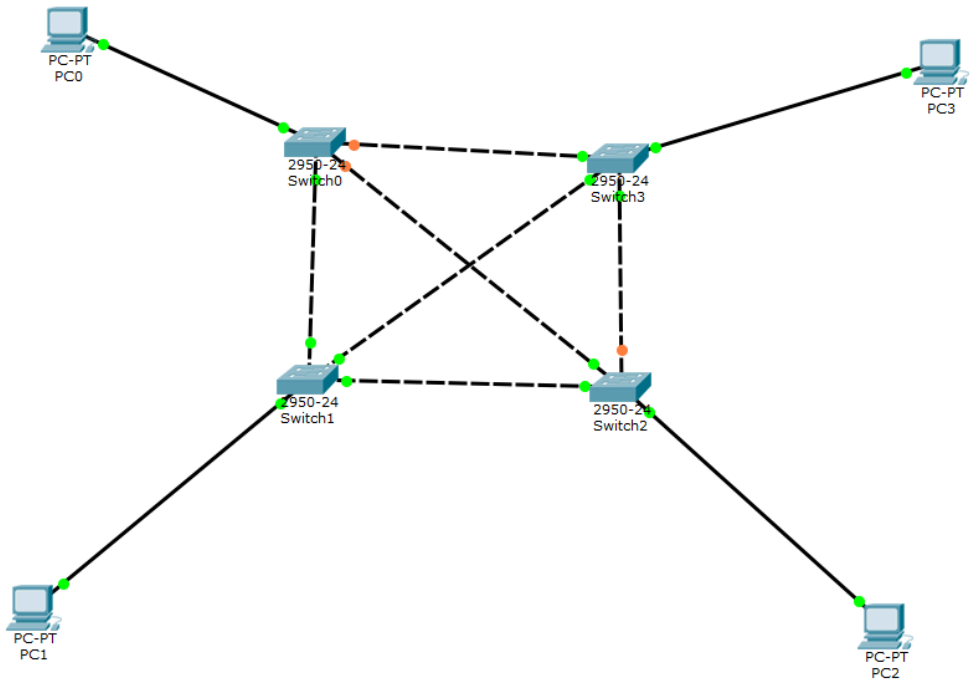


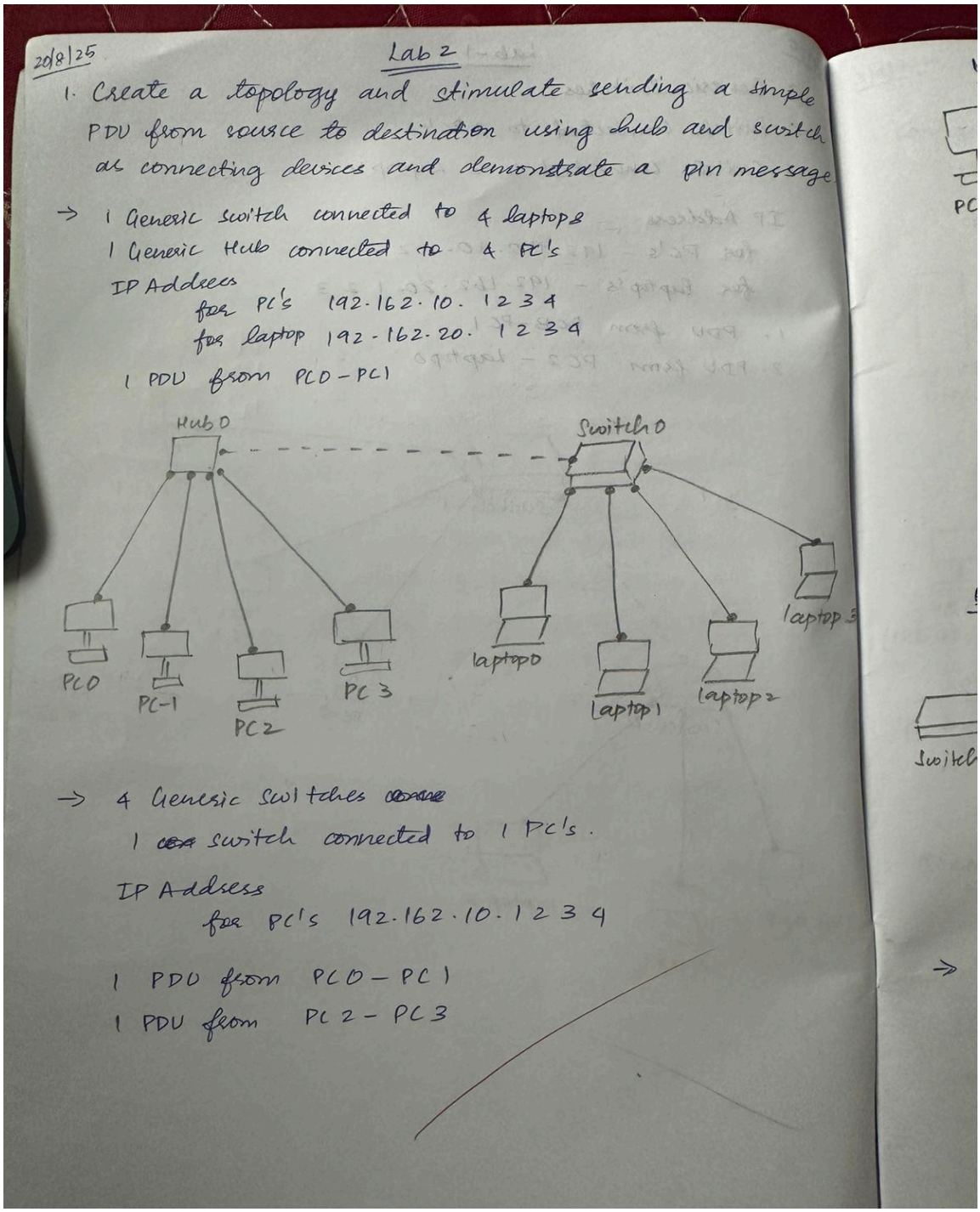
topology



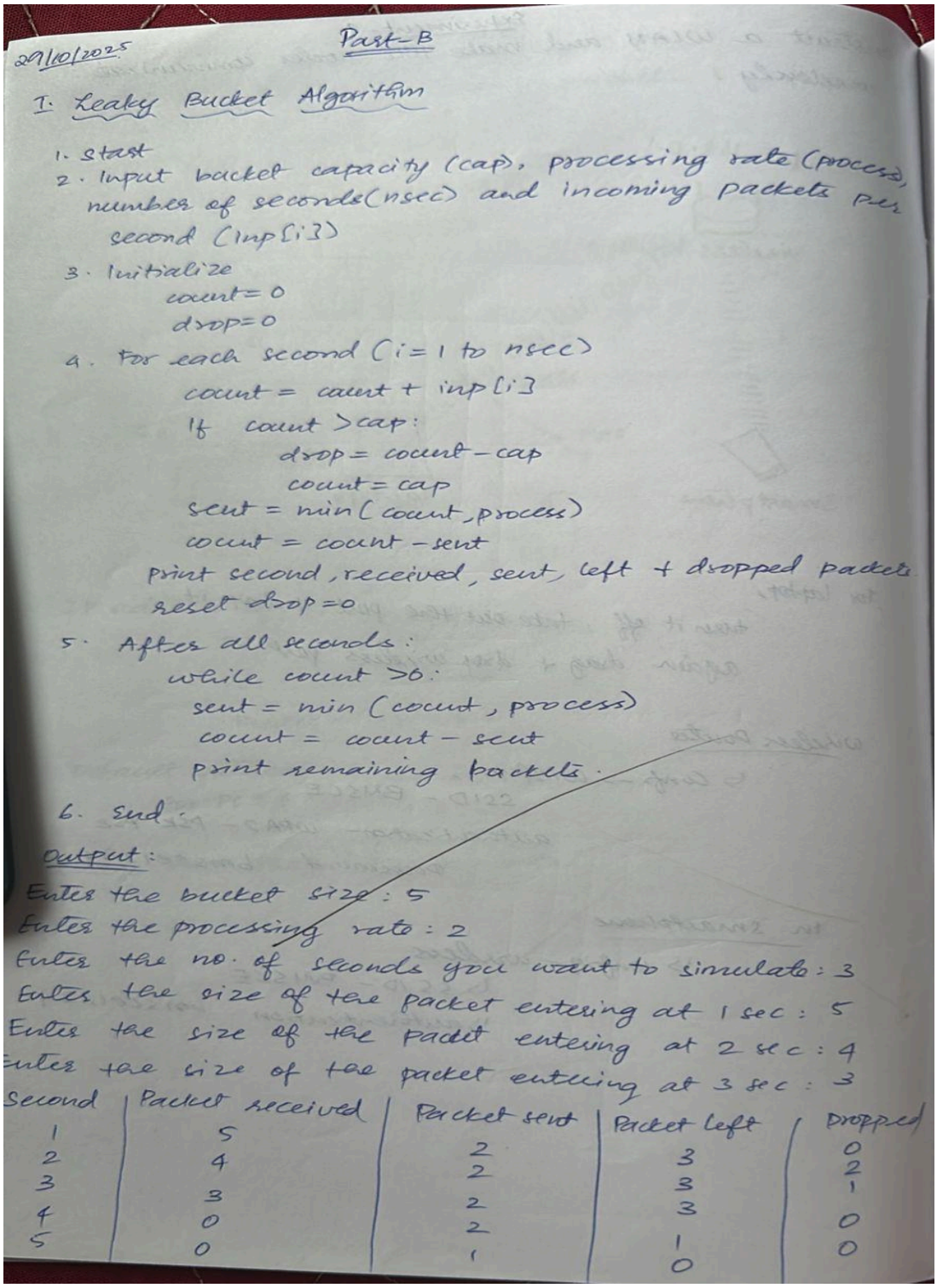
12.Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping messages.


Fa6/1
Fa6/1
Fa0
Fa0
Fa0
Fa0
Fa0
PC-PT
PC2
PC-PT
PC4
PC-PT
PC5
PC-PT
PC7
PC-PT
PC3
PC-PT
PC6


PC-PT
PC0
PC-PT
PC3
2950-24
Switch0
2950-24
Switch3
2950-24
Switch1
2950-24
Switch2
PC-PT
PC1
PC-PT
PC2



13.Write a program for congestion control using Leaky bucket algorithm.



29/10/2025

Part-B

1. Leaky Bucket Algorithm

1. start
2. Input bucket capacity (cap), processing rate (process), number of seconds(nsec) and incoming packets per second (inp[i])
3. Initialize
   count = 0
   drop = 0
4. For each second (i = 1 to nsec)
   count = count + inp[i]
   If count > cap:
   drop = count - cap
   count = cap
   sent = min(count, process)
   count = count - sent
   print second, received, sent, left + dropped packets
   reset drop = 0
5. After all seconds:
   while count > 0:
   sent = min(count, process)
   count = count - sent
   print remaining packets.
6. End.

Output:

Enter the bucket size : 5
Enter the processing rate : 2
Enter the no. of seconds you want to simulate : 3
Enter the size of the packet entering at 1 sec : 5
Enter the size of the packet entering at 2 sec : 4
Enter the size of the packet entering at 3 sec : 3

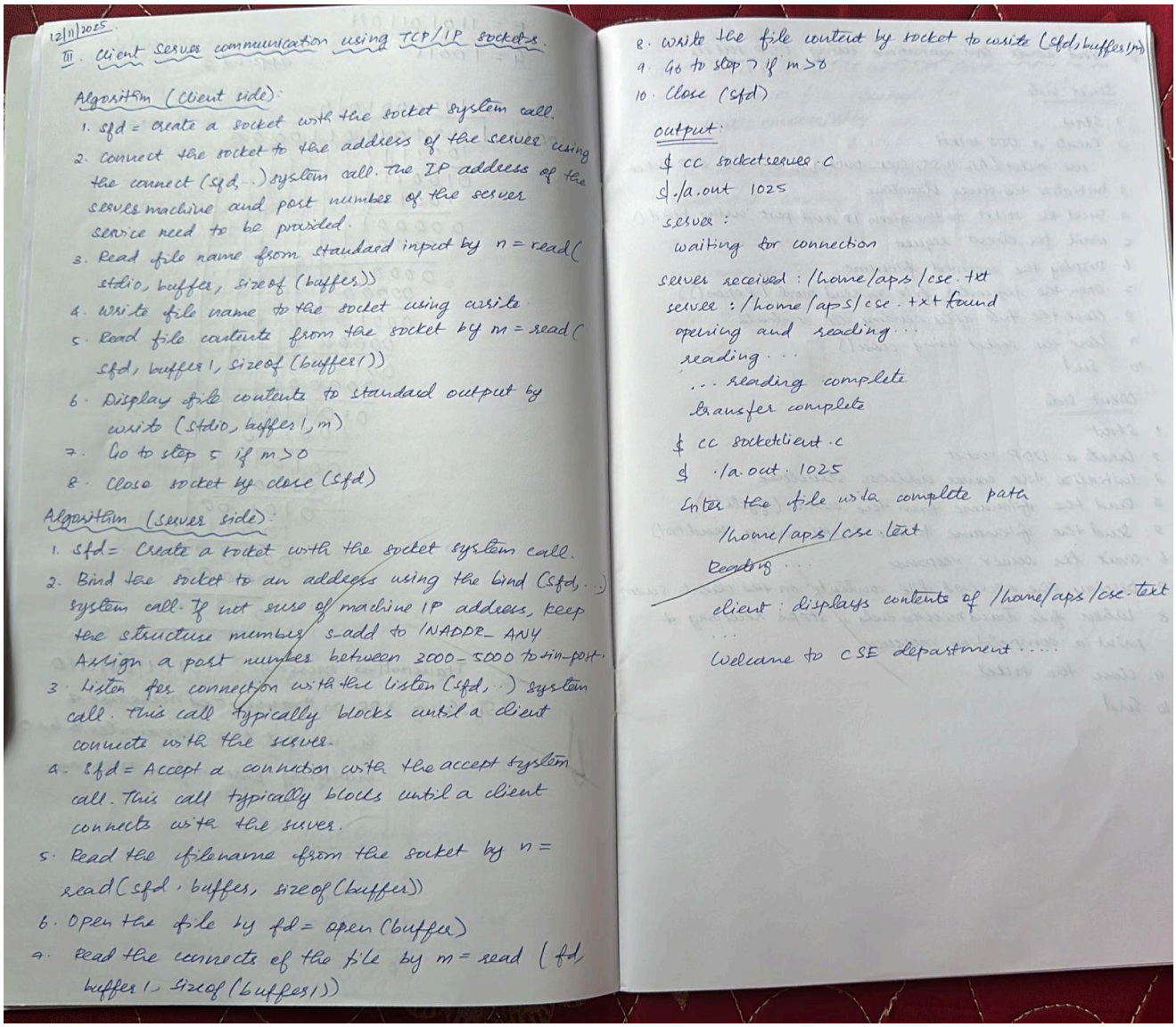
Second	Packet received	Packet sent	Packet left	Dropped
1	5	2	3	0
2	4	2	3	2
3	3	2	3	1
4	0	2	1	0
5	0	1	0	0

```
#include <stdio.h>

int min(int x, int y)
{
   return (x < y) ? x : 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);
      scanf("%d", &inp[i]);
   }

   printf("\n Second | 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("%6d | %15d |", i + 1, inp[i]);

      mini = min(count, process);
      printf(" %11d |", mini);

      count -= mini;
      printf(" %12d | %7d\n", count, drop);

      drop = 0;
   }

   // Process remaining packets after all seconds
   for (; count != 0; i++)
   {
      if (count > cap)
```

```
        {
            drop = count - cap;
            count = cap;
        }

        printf("%6d | %15d |", i + 1, 0);

        mini = min(count, process);
        printf(" %11d |", mini);

        count -= mini;
        printf(" %12d | %7d\n", count, drop);
    }

    return 0;
}
```

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



12/11/2025

III. Client Server communication using TCP/IP sockets.

Algorithm (client side):

1. sfd = create a socket with the socket system call.
2. Connect the socket to the address of the server using the connect (sfd, ..) system call. The IP address of the servers machine and port number of the server service need to be provided.
3. Read file name from standard input by n = read(stdio, buffer, sizeof (buffer))
4. Write file name to the socket using write
5. Read file contents from the socket by m = read (sfd, buffer1, sizeof (buffer1))
6. Display file contents to standard output by write (stdio, buffer1, m)
7. Go to step 5 if m > 0
8. Close socket by close (sfd)

Algorithm (server side):

1. sfd = Create a socket with the socket system call.
2. Bind the socket to an address using the bind (sfd, ..) system call. If not sure of machine IP address, keep the structure member s_add to INADDR_ANY Assign a port number between 3000 - 5000 to sin_port.
3. Listen for connection with the listen (sfd, ..) system call. This call typically blocks until a client connects with the server.
4. sfd = Accept a connection with the accept system call. This call typically blocks until a client connects with the server.
5. Read the filename from the socket by n = read(sfd, buffer, sizeof(buffer))
6. Open the file by fd = open (buffer)
7. Read the contents of the file by m = read (fd, buffer1, sizeof(buffer1))
8. Write the file content by socket to write (sfd, buffer1, m)
9. Go to step 7 if m > 0
10. Close (sfd)

output:

$ cc socketserver.c
$ ./a.out 1025
server :
waiting for connection
server received : /home/aps/cse.txt
server : /home/aps/cse.txt found
opening and reading ...
reading ...
... reading complete
transfer complete
$ cc socketclient.c
$ ./a.out 1025
Enter the file with complete path
/home/aps/cse.text
Reading ...
client : displays contents of /home/aps/cse.text
...
Welcome to CSE department ....

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

int main(int argc, char *argv[])
{
   int sockfd, portno, n;
   struct sockaddr_in serv;
   struct hostent *server;
   char buffer[256];
   char c[20000];
```

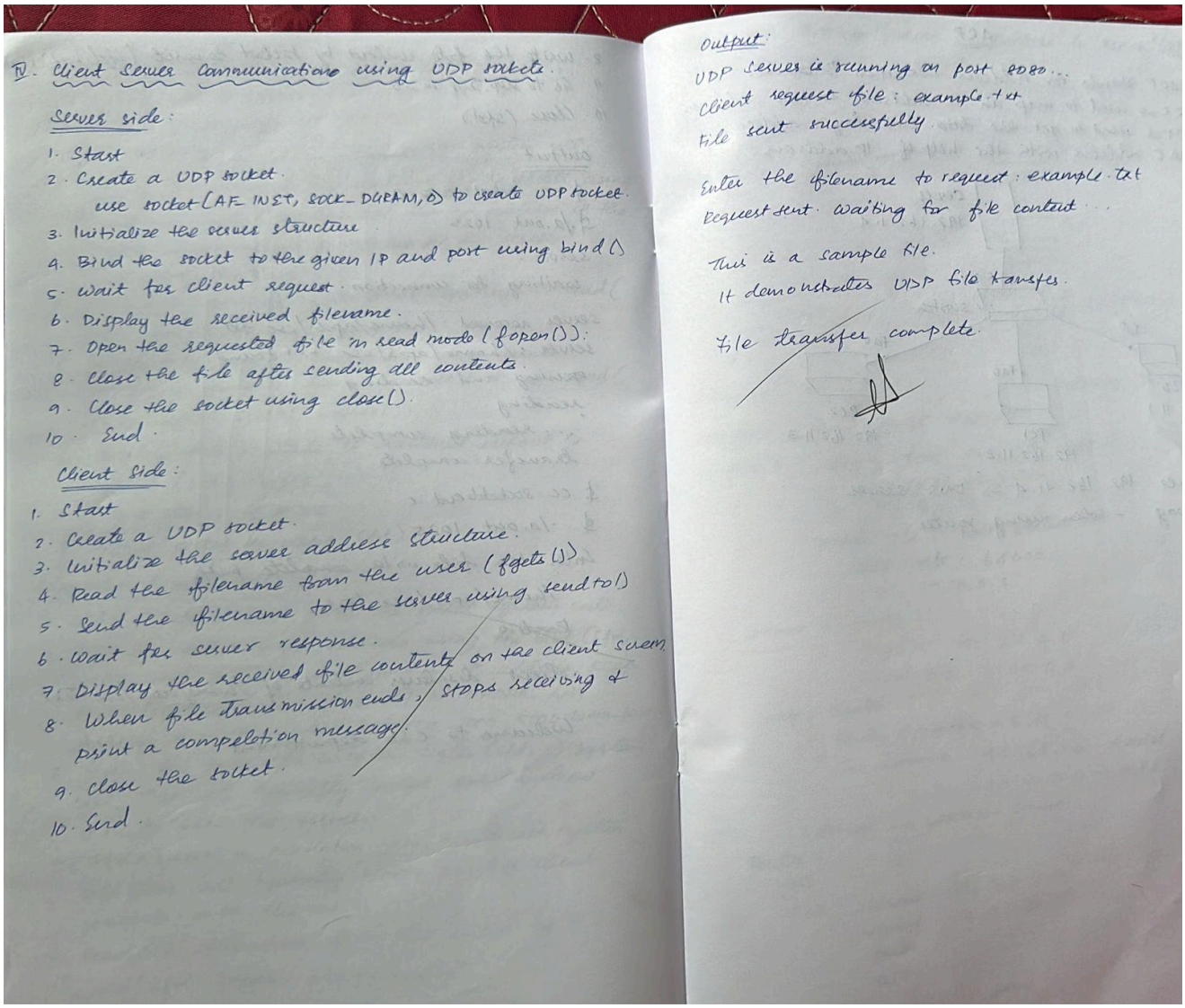
```
if (argc < 3) {
   printf("Error: insufficient arguments.\n");
   printf("Usage: %s <hostname> <port>\nExample: %s 127.0.0.1 7777\n", argv[0], argv[0]);
   exit(1);
}

portno = atoi(argv[2]);

// Create socket
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0) {
   perror("Error opening socket");
   exit(1);
}

// Get server by name/IP
server = gethostbyname(argv[1]);
if (server == NULL) {
   fprintf(stderr, "Error: no such host.\n");
   exit(1);
}

// Zero out the structure
bzero((char *)&serv, sizeof(serv));
serv.sin_family = AF_INET;
bcopy((char *)server->h_addr, (char *)&serv.sin_addr.s_addr, server->h_length);
serv.sin_port = htons(portno);

// Connect to server
if (connect(sockfd, (struct sockaddr *)&serv, sizeof(serv)) < 0) {
   perror("Error connecting");
   exit(1);
}

printf("Enter the file path (complete path): ");
scanf("%s", buffer);

// Send filename to server
n = write(sockfd, buffer, strlen(buffer));
if (n < 0) {
   perror("Error writing to socket");
   exit(1);
}

bzero(c, sizeof(c));
printf("Reading file contents from server...\n");
```

```
    // Read file contents
    n = read(sockfd, c, sizeof(c) - 1);
    if (n < 0) {
        perror("Error reading from socket");
        exit(1);
    }

    printf("\nClient: Display content of %s\n----------------------------------\n", buffer);
    fputs(c, stdout);
    printf("\n----------------------------------\n");

    close(sockfd);
    return 0;
}
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

int main(int argc, char *argv[])
{
    int sockfd, newsockfd, portno, len, n;
    char buffer[256], c[2000], cc[20000];
    struct sockaddr_in serv, cli;
    FILE *fd;

    if (argc < 2) {
        printf("Error: no port number provided.\n");
        printf("Usage: %s <port>\nExample: %s 7777\n", argv[0], argv[0]);
        exit(1);
    }

    // Create socket
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        perror("Error opening socket");
        exit(1);
    }

    // Initialize server address structure
    bzero((char *)&serv, sizeof(serv));
    portno = atoi(argv[1]);
    serv.sin_family = AF_INET;
    serv.sin_addr.s_addr = INADDR_ANY;
```

```
serv.sin_port = htons(portno);

// Bind socket
if (bind(sockfd, (struct sockaddr *)&serv, sizeof(serv)) < 0) {
    perror("Error on binding");
    close(sockfd);
    exit(1);
}

// Listen for incoming connections
listen(sockfd, 5);
len = sizeof(cli);
printf("Server: waiting for connection on port %d...\n", portno);

// Accept a client
 newsockfd = accept(sockfd, (struct sockaddr *)&cli, (socklen_t *)&len);
if (newsockfd < 0) {
    perror("Error on accept");
    close(sockfd);
    exit(1);
}

// Read filename from client
bzero(buffer, 255);
n = read(newsockfd, buffer, 255);
if (n < 0) {
    perror("Error reading from socket");
    close(newsockfd);
    close(sockfd);
    exit(1);
}

printf("Server received filename: %s\n", buffer);

// Try to open the file
fd = fopen(buffer, "r");
if (fd != NULL) {
    printf("Server: file '%s' found, reading and sending...\n", buffer);
    bzero(cc, sizeof(cc));
    while (fgets(c, sizeof(c), fd) != NULL) {
        strcat(cc, c);
     }
    fclose(fd);

    // Send file content to client
    n = write(newsockfd, cc, strlen(cc));
```

```
        if (n < 0)
            perror("Error writing to socket");
        else
            printf("File transfer complete.\n");
    } else {
        printf("Server: file not found.\n");
        n = write(newsockfd, "Error: file not found.\n", 24);
        if (n < 0)
            perror("Error writing to socket");
    }

    close(newsockfd);
    close(sockfd);
    return 0;
}
```

15.Using UDP sockets, write a client-server program to make the client send the file name and the server to send back the contents of the requested file if present.



IV. Client Server Communication using UDP sockets.

Server side:

1. Start
2. Create a UDP socket.
   use socket(AF_INET, SOCK_DGRAM, 0) to create UDP socket.
3. Initialize the server structure.
4. Bind the socket to the given IP and port using bind()
5. Wait for client request.
6. Display the received filename.
7. Open the requested file in read mode (fopen()).
8. Close the file after sending all contents.
9. Close the socket using close().
10. End.

Client side:

1. Start
2. Create a UDP socket.
3. Initialize the server address structure.
4. Read the filename from the user (fgets())
5. Send the filename to the server using sendto()
6. Wait for server response.
7. Display the received file contents on the client screen
8. When file transmission ends, stops receiving & print a completion message.
9. Close the socket.
10. End.

Output:

UDP server is running on port 8080...
Client request file: example.txt
File sent successfully

Enter the filename to request: example.txt
Request sent. Waiting for file content...

This is a sample file.
It demonstrates UDP file transfer.

File transfer complete.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define MAXLINE 1024

int main() {
    int sockfd;
    char buffer[MAXLINE];
    struct sockaddr_in servaddr, cliaddr;
    socklen_t len;
    ssize_t n;

    // Create UDP socket
```

```
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));
    memset(&cliaddr, 0, sizeof(cliaddr));

    // Server info
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(PORT);

    // Bind socket to the port
    if (bind(sockfd, (const struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
        perror("bind failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

    printf("UDP Server is running on port %d...\n", PORT);

    len = sizeof(cliaddr);

    // Receive filename from client
    n = recvfrom(sockfd, (char *)buffer, MAXLINE, 0, (struct sockaddr *)&cliaddr, &len);
    buffer[n] = '\0';
    printf("Client requested file: %s\n", buffer);

    FILE *fp = fopen(buffer, "r");
    if (fp == NULL) {
        char *msg = "File not found!";
        sendto(sockfd, msg, strlen(msg), 0, (struct sockaddr *)&cliaddr, len);
        printf("File not found, message sent to client.\n");
    } else {
        // Read and send file content
        while (fgets(buffer, MAXLINE, fp) != NULL) {
            sendto(sockfd, buffer, strlen(buffer), 0, (struct sockaddr *)&cliaddr, len);
        }
        fclose(fp);
        printf("File sent successfully.\n");
    }

    close(sockfd);
    return 0;
}
#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define MAXLINE 1024

int main() {
    int sockfd;
    char buffer[MAXLINE];
    struct sockaddr_in servaddr;
    socklen_t len;

    // Create UDP socket
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));

    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = INADDR_ANY;

    printf("Enter the filename to request: ");
    fgets(buffer, MAXLINE, stdin);
    buffer[strcspn(buffer, "\n")] = '\0';  // remove newline

    // Send filename to server
    sendto(sockfd, buffer, strlen(buffer), 0, (const struct sockaddr *)&servaddr, sizeof(servaddr));

    printf("Request sent. Waiting for file content...\n\n");

    len = sizeof(servaddr);

    // Receive file contents
    ssize_t n;
    while ((n = recvfrom(sockfd, buffer, MAXLINE, 0, (struct sockaddr *)&servaddr, &len)) > 0) {
        buffer[n] = '\0';
        printf("%s", buffer);
        if (n < MAXLINE - 1) break;  // assume end of file
    }

    printf("\n\nFile transfer complete.\n");
```

```
    close(sockfd);
    return 0;
}
```

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

Output

```
Enter the generator polynomial:
101
Generator polynomial (CRC-CCITT): 101
Enter the message:
110101

Message polynomial appended with zeros:
11010100

Checksum (Remainder):
11

Transmitted message (with checksum):
11010111

Enter the received message:
11010111

Received message is error-free ✅


=== Code Execution Successful ===
```

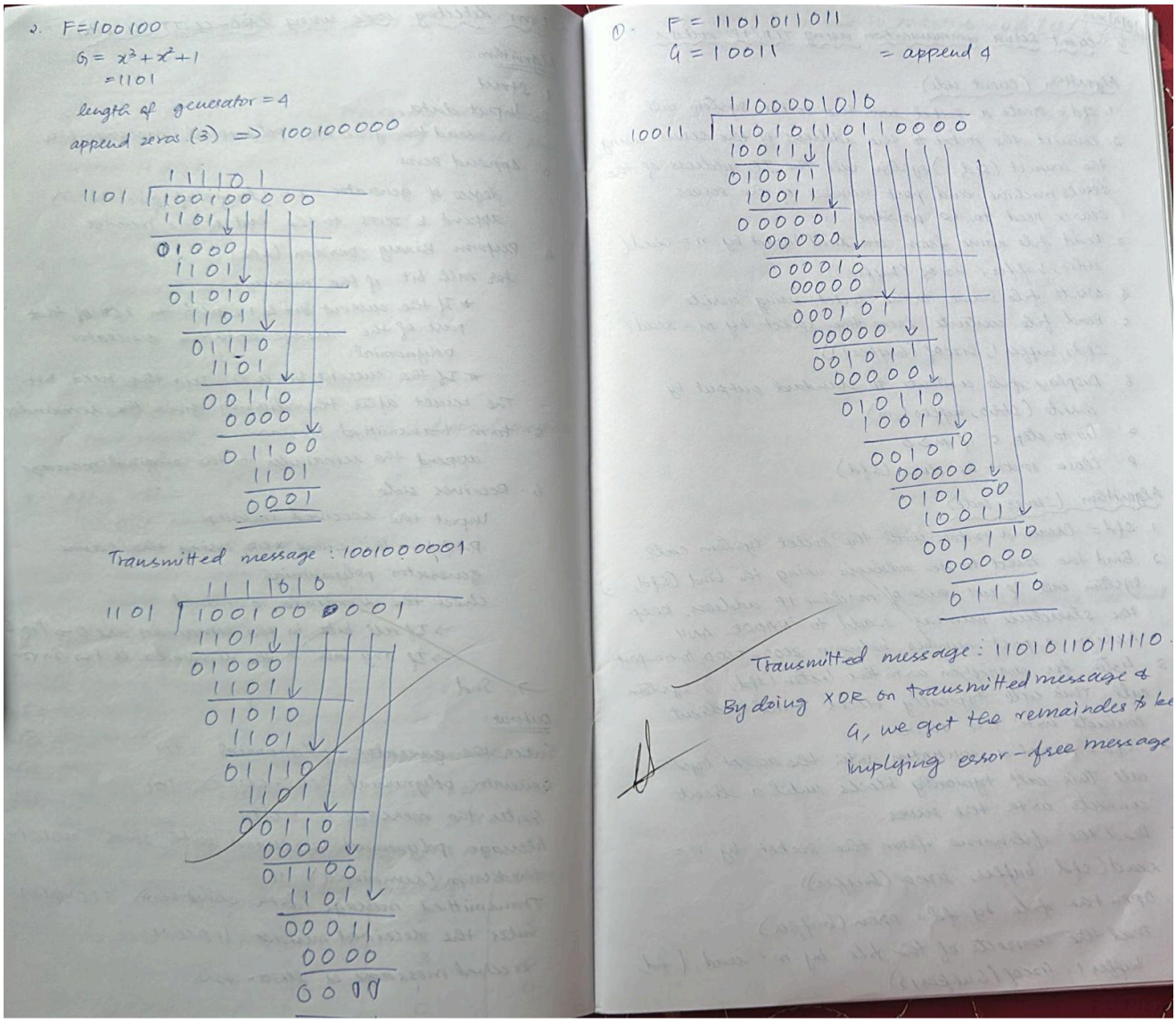
1) Error detecting code using CRC-CCITT (16-bits)

Algorithm:

1. Start
2. Input data
   read the generator polynomial, message bits.
3. Append zeros.
   degree of generator = k = (length of generator - 1)
   append k zeros to the end of the message.
4. Perform Binary Division (XOR)
   for each bit of the message
   * If the current bit is 1, perform XOR of that part of the message with the generator polynomial.
   * If the current bit is 0, skip the next bit.
   The result after the division gives the remainder.
5. form transmitted Message
   append the remainder to the original message.
6. Receives side.
   Input the received message.
   Perform the same XOR using the same generator polynomial
   check the remainder obtained.
   -> If all bits in the remainder are 0 -> No Error.
   -> If any bits in the remainder is 1 -> Error.
7. End.

Output:

Enter the generator polynomial : 101
Generator polynomial (CRC-CCITT) : 101
Enter the message : 110101
Message polynomial appended with zeros : 11010100
Checksum (remainder) : 11
Transmitted message (with checksum) : 11010111
Enter the received message : 11010111
Received message is error-free




```
#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 generator polynomial:\n");
   fgets(gen, sizeof(gen), stdin);
   gen[strcspn(gen, "\n")] = '\0';  // remove newline if present
   printf("Generator polynomial (CRC-CCITT): %s\n", gen);

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

```
printf("Enter the message:\n");
fgets(msj, sizeof(msj), stdin);
msj[strcspn(msj, "\n")] = '\0';  // remove newline

n = strlen(msj);

// Append k zeros to the message
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);

// Division (XOR)
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++;
      }
   }
}

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

printf("\nChecksum (Remainder):\n");
puts(rem);

// Append checksum to message
printf("\nTransmitted message (with checksum):\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);

  // Receiver side
  printf("\nEnter the received message:\n");
  fgets(s, sizeof(s), stdin);
  s[strcspn(s, "\n")] = '\0';  // remove newline
  n = strlen(s);

  // Division on received message
  for (i = 0; i < n - k; 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 - k + i];
  rem[k] = '\0';

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

  if (flag == 0)
    printf("\nReceived message is error-free ✅\n");
  else
    printf("\nReceived message contains errors ❌\n");

  return 0;
}
```