

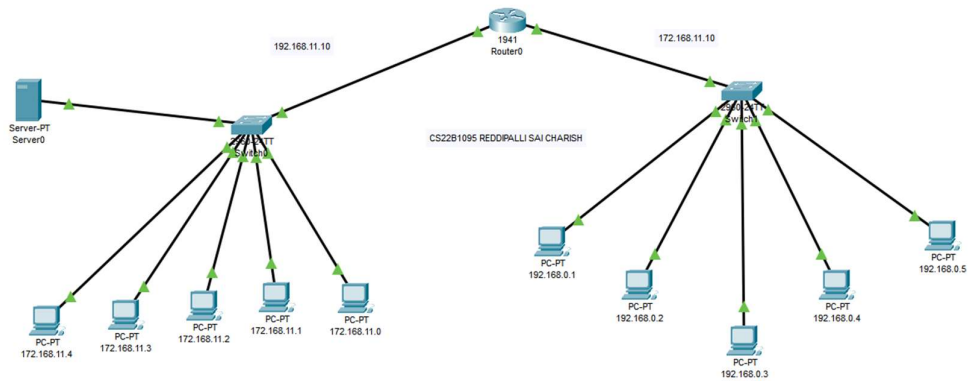
# COMPUTER NETWORKS LAB ASSIGNMENT-03

REDDIPALLI SAI CHARISH

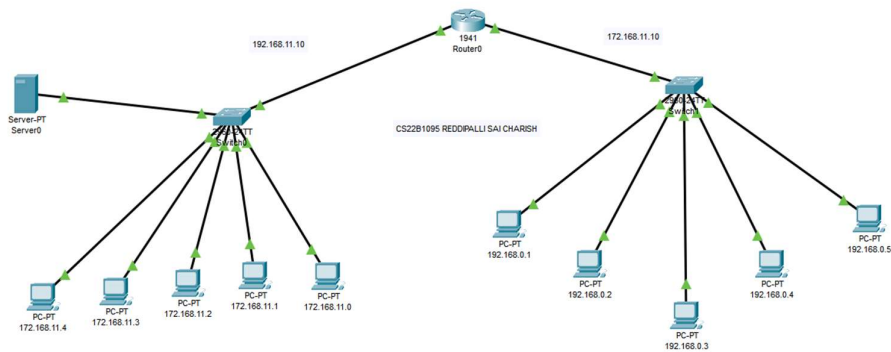
CS22B1095

QUESTION 1:

NETWORK diagram



Packet transfer successful



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	172.1...	192.168.0.1	ICMP		0.000	N	0	(edit)	(delete)
	Successful	172.1...	192.168.0.3	ICMP		0.000	N	1	(edit)	(delete)
	Successful	172.1...	192.168.0.1	ICMP		0.000	N	2	(edit)	(delete)
	Successful	172.1...	192.168.0.2	ICMP		0.000	N	3	(edit)	(delete)

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	172.1...	192.168.0.1	ICMP		0.000	N	0	(edit)	(delete)
	Successful	172.1...	192.168.0.3	ICMP		0.000	N	1	(edit)	(delete)
	Successful	172.1...	192.168.0.1	ICMP		0.000	N	2	(edit)	(delete)
	Successful	172.1...	192.168.0.2	ICMP		0.000	N	3	(edit)	(delete)

DHCP assigned IP

Server-PT  
Server0

MANUAL assigned IP

172.168.11.4

Physical Config Desktop Programming Attributes

IP Configuration

Interface FastEthernet0

IP Configuration

☒ DHCP ☐ Static

IPv4 Address 172.168.11.4

Subnet Mask 255.255.0.0

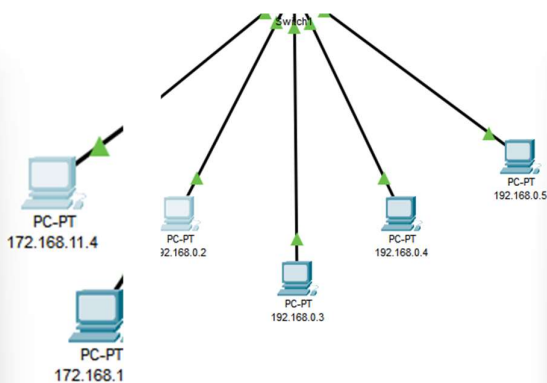
Default Gateway 192.168.11.10

DNS Server 0.0.0.0

IPv6 Configuration

☒ Automatic ☐ Static

IPv6 Address



192.168.0.2

Physical Config Desktop Programming Attributes

IP Configuration

Interface FastEthernet0

IP Configuration

☐ DHCP ☒ Static

IPv4 Address 192.168.11.2

Subnet Mask 255.255.0.0

Default Gateway 172.168.11.10

DNS Server 0.0.0.0

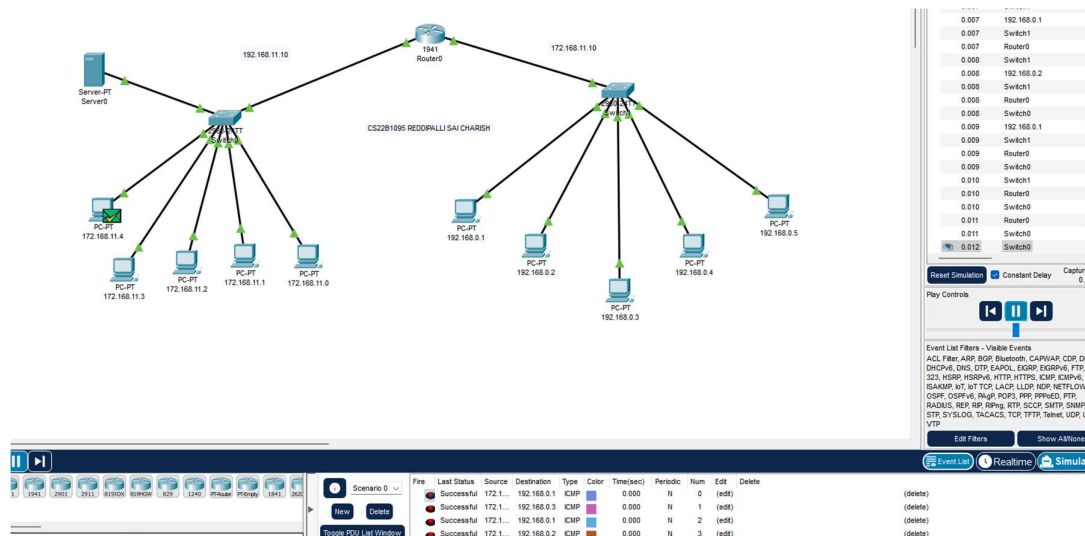
IPv6 Configuration

☐ Automatic ☒ Static

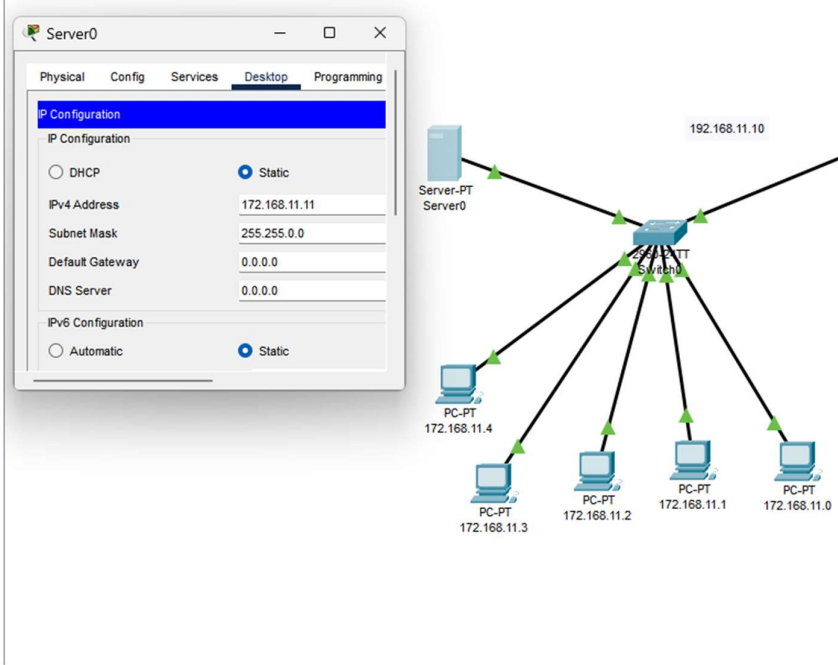
IPv6 Address

Link Local Address FE80::201:97FF:FE2E:3E66

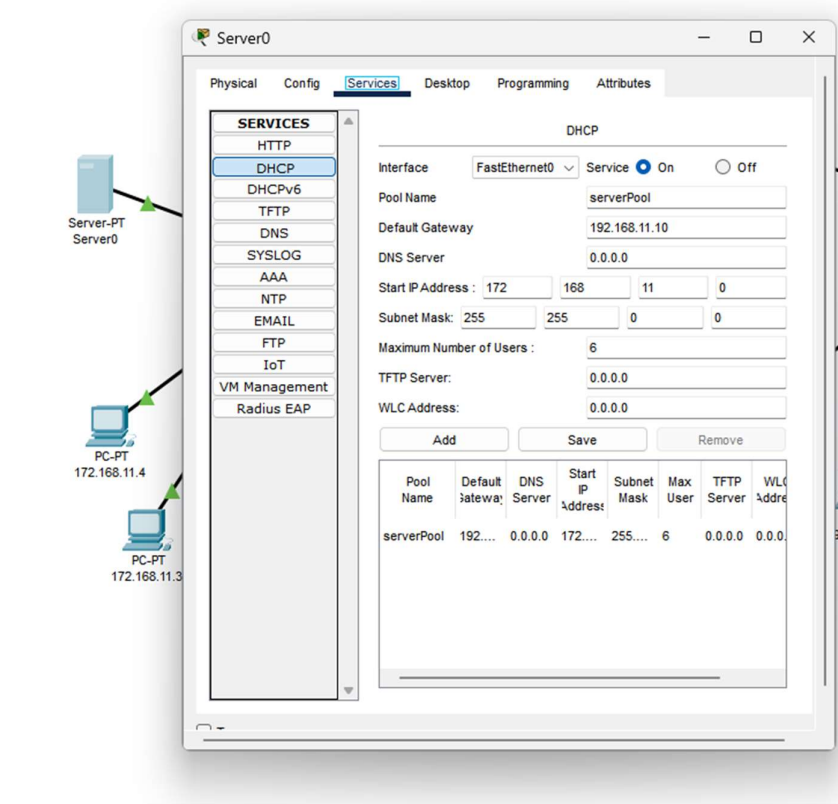
SIMULATION



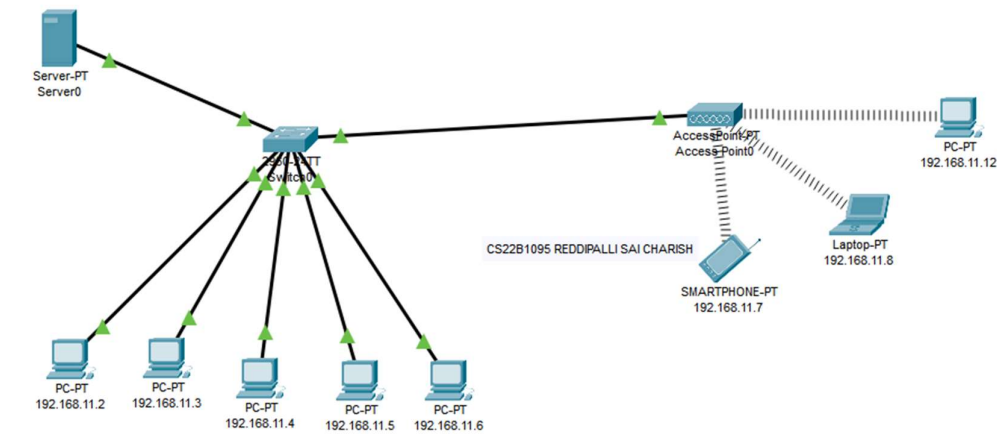
SERVER CONFIGURATION



DEFAULT GATEWAY :



Question 2:  
NETWORK DIAGRAM



PACKET TRANSFER SUCCESSFUL

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	192.1...	192.168.11.7	ICMP		0.000	N	0	(edit)	(delete)
	Successful	192.1...	192.168.11.8	ICMP		0.000	N	1	(edit)	(delete)
	Successful	192.1...	192.168.1...	ICMP		0.000	N	2	(edit)	(delete)

SIMULATION

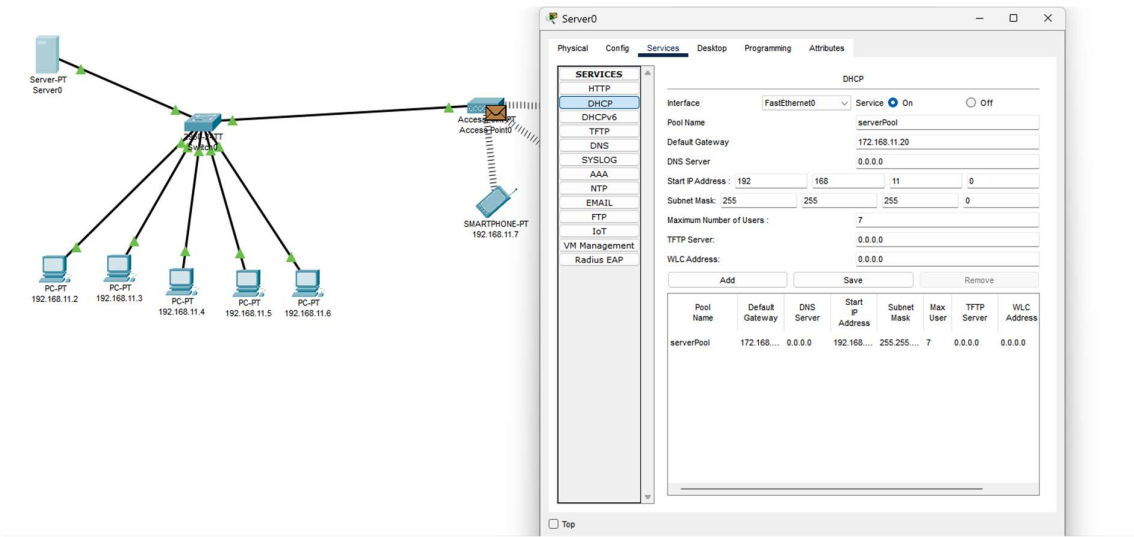
The simulation interface displays the network diagram on the left and a packet capture log on the right. The log shows the following events:

Time(sec)	Last Device
0.012	Switch0
0.012	Switch0
0.012	--
0.013	192.168.11.3
0.013	--
0.014	Access Point0
0.014	Access Point0
0.014	Access Point0
0.014	Switch0
0.015	--
0.016	Access Point0
0.016	Access Point0
0.016	Access Point0
0.017	--
0.018	192.168.11.7
0.018	Access Point0
0.020	Switch0
0.020	--
0.020	--

The bottom of the interface shows a packet capture log with the following entries:

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	In Progress	192.1...	192.168.11.7	ICMP		0.000	N	0	(edit)	(delete)
	In Progress	192.1...	192.168.11.8	ICMP		0.000	N	1	(edit)	(delete)
	In Progress	192.1...	192.168.1...	ICMP		0.000	N	2	(edit)	(delete)

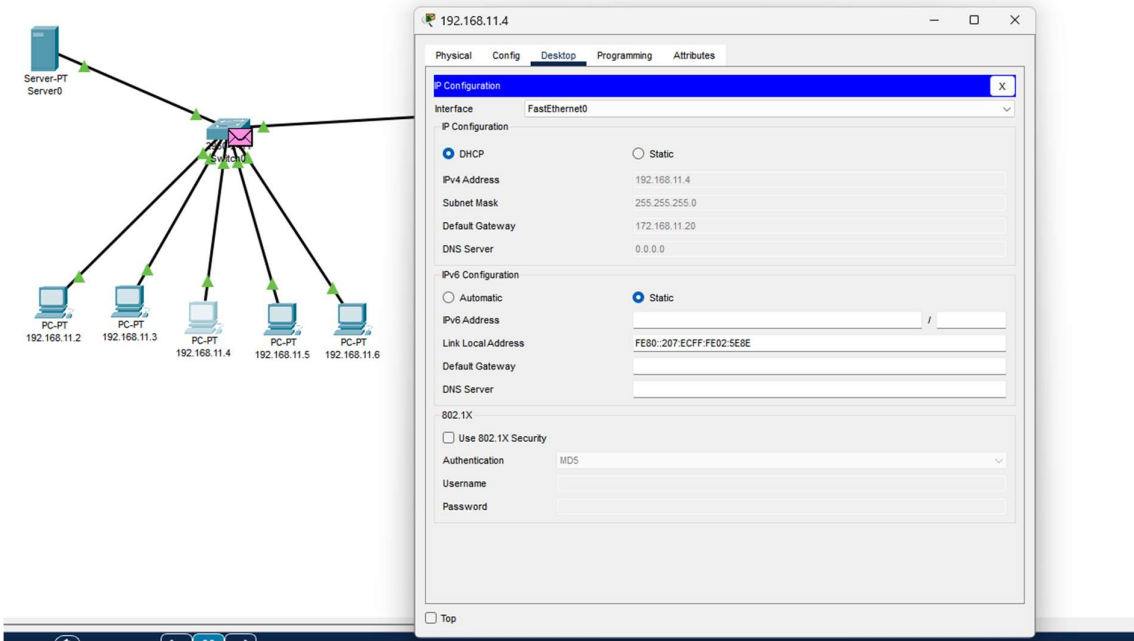
SERVER CONFIGURATION



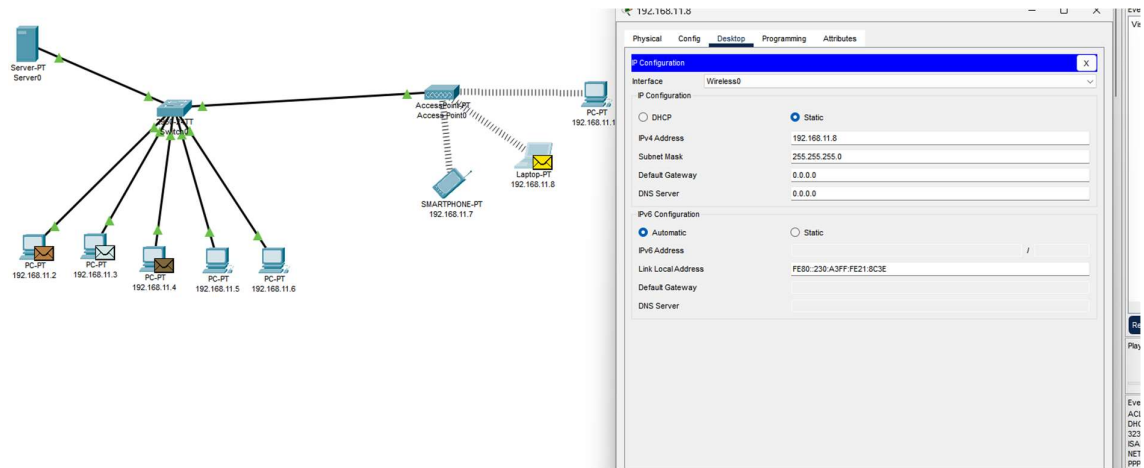
The network diagram shows a central switch connected to a Server-PT (Server0) and a Smartphone-PT (192.168.11.7). The switch is also connected to five PC-PTs with IP addresses 192.168.11.2, 192.168.11.3, 192.168.11.4, 192.168.11.5, and 192.168.11.6. The DHCP configuration window for Server0 is shown, with the DHCP service enabled on the FastEthernet0 interface. The configuration includes a pool named 'serverPool' with a default gateway of 172.168.11.20, a start IP address of 192.168.11.168, and a subnet mask of 255.255.255.0. The maximum number of users is set to 7. The TFTP server is configured with a server address of 0.0.0.0 and a WLC address of 0.0.0.0.

Pool Name	Default Gateway	DNS Server	Start IP Address	Subnet Mask	Max User	TFTP Server	WLC Address
serverPool	172.168.11.20	0.0.0.0	192.168.11.168	255.255.255.0	7	0.0.0.0	0.0.0.0

DHCP assigned IP

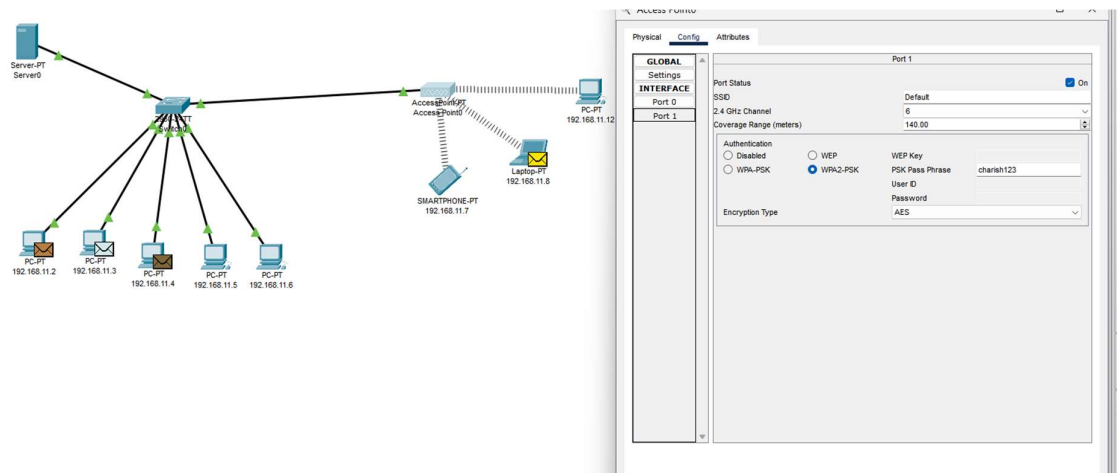


The network diagram shows the same setup as the previous image. The PC configuration window for the PC with IP 192.168.11.4 is shown, with the DHCP service selected. The configuration shows the IP address 192.168.11.4, subnet mask 255.255.255.0, default gateway 172.168.11.20, and DNS server 0.0.0.0. The IPv6 configuration is set to static, with a link local address of FE80::207:ECFF:FE02:5E8E. The 802.1X security is disabled, and the authentication is set to MD5.



MANUAL assignment of IP to wireless devices

SETTING PASSWRODS:



PT-LAPTOP-NM-1FGE

PT-LAPTOP-NM-1W

PT-LAPTOP-NM-1W-A

PT-LAPTOP-NM-1W-AC

PT-LAPTOP-NM-3G/4G

PT-HEADPHONE

PT-MICROPHONE

Customize  
Icon in  
Physical View

Customize  
Icon in  
Logical View

The Linksys-WPC300N module provides one 2.4GHz wireless interface suitable for connection to wireless networks. The module supports protocols that use Ethernet for LAN access.

## WPA2-Personal Needed for Connection

This wireless network has WPA2-Personal enabled. To connect to this network, enter the required passphrase in the appropriate field below. Then click the **Connect** button.

Security

WPA2-Personal

Please select the wireless security method used by your existing wireless network.

Pre-shared Key

charish123

Please enter a Pre-shared Key that is 8 to 63 characters in length.

Cancel

Connect

### Question 3 :

```
//CS22B1095
```

```
//REDDIPALLI SAI CHARISH
```

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

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

```
void bitStuffing(char *inputMsg, char *stuffedMsg) {
```

```
    int index = 0, oneCount = 0, msgLength;
```

```
    msgLength = strlen(inputMsg);
```

```
    for (index = 0; index < msgLength; index++) {
```

```
        if (oneCount == 5) {
```

```
            strcat(stuffedMsg, "0");
```

```
            oneCount = 0;
```

```
        }
```

```
        if (inputMsg[index] == '1') {
```

```
            oneCount++;
```

```
        } else {
```

```
            oneCount = 0;
```

```
        }
```

```
        strncat(stuffedMsg, &inputMsg[index], 1);
```

```
    }
```

```
    if (oneCount == 5) {
```

```
        strcat(stuffedMsg, "0");
```

```
    }
```

```
}
```

```
void reverseBitStuffing(char *stuffedMsg, char *retrievedMsg) {
```

```
    int index = 0, oneCount = 0, msgLength;
```

```
    msgLength = strlen(stuffedMsg);
```

```
    for (index = 0; index < msgLength; index++) {
```

```
        if (oneCount == 5 && stuffedMsg[index] == '0') {
```

```
            oneCount = 0;
```

```
            continue;
```

```
        }
```

```
        if (stuffedMsg[index] == '1') {
```

```
            oneCount++;
```

```
        } else {
```

```
            oneCount = 0;
```



```

    }

    strncat(retrievedMsg, &stuffedMsg[index], 1);
}
}

int main() {
    char inputMsg[1000], stuffedMsg[2000], retrievedMsg[1000], finalMsg[2000];

    while (1) {
        strcpy(stuffedMsg, "01111110");
        retrievedMsg[0] = '\0';
        finalMsg[0] = '\0';

        printf("Enter the message (or type 'exit' to stop): ");
        scanf("%s", inputMsg);

        if (strcmp(inputMsg, "exit") == 0) {
            break;
        }

        bitStuffing(inputMsg, stuffedMsg);
        strcat(stuffedMsg, "01111110");
        printf("Stuffed message: %s\n", stuffedMsg);

        reverseBitStuffing(stuffedMsg + 8, retrievedMsg);
        strcpy(finalMsg, "01111110");
        strcat(finalMsg, retrievedMsg);
        printf("Retrieved message: %s\n", finalMsg);
    }

    return 0;
}

```

```
Enter the message (or type 'exit' to stop): 1010100101100
Stuffed message: 01111110101010010110001111110
Retrieved message: 01111110101010010110001111110
Enter the message (or type 'exit' to stop): 1111101001
Stuffed message: 011111101111100100101111110
Retrieved message: 01111110111110100101111110
Enter the message (or type 'exit' to stop): 111110111110
Stuffed message: 011111101111100111110001111110
Retrieved message: 0111111011111011111001111110
Enter the message (or type 'exit' to stop): 1010111111
Stuffed message: 011111101010111110101111110
Retrieved message: 01111110101011111011111110
Enter the message (or type 'exit' to stop): 1111111111111111
Stuffed message: 011111101111101111101111101101111110
Retrieved message: 011111101111111111111111101111110
Enter the message (or type 'exit' to stop): exit
PS C:\Users\CHARISH\OneDrive\Desktop\5th sem\COMPUTER NETWORKS (CN )\output> |
```

#### Question 4:

```
#include <stdio.h>

#include <string.h>

void addZeros(char *data, int divisorLength) {

    int length = strlen(data);

    for (int i = 0; i < divisorLength - 1; i++) {

        data[length + i] = '0';

    }

    data[length + divisorLength - 1] = '\0';

}

void performDivision(char *data, char *divisor, char *remainder) {

    int dataLen = strlen(data);

    int divisorLen = strlen(divisor);

    strcpy(remainder, data);

    for (int i = 0; i <= dataLen - divisorLen; i++) {

        if (remainder[i] == '1') {

            for (int j = 0; j < divisorLen; j++) {

                remainder[i + j] = remainder[i + j] == divisor[j] ? '0' : '1';

            }

        }

    }

}

void generateFinalMessage(char *data, char *remainder, int divisorLength) {

    int dataLen = strlen(data);

    int remainderLen = strlen(remainder);

    data[dataLen - (divisorLength - 1)] = '\0';

    strcat(data, remainder + remainderLen - (divisorLength - 1));

}

int checkError(char *remainder) {

    int remainderLen = strlen(remainder);

    for (int i = 0; i < remainderLen; i++) {

        if (remainder[i] == '1')

            return 0;

    }

}
```

```

    }

    return 1;
}

int main() {

    char data[100];

    char divisor[] = "10000111";

    char crcRemainder[200];

    char receivedRemainder[200];

    while (1) {

        printf("Enter the binary data (or type 'exit' to quit): ");

        scanf("%s", data);

        if (strcmp(data, "exit") == 0) {

            break;

        }

        int divisorLength = strlen(divisor);

        addZeros(data, divisorLength);

        performDivision(data, divisor, crcRemainder);

        generateFinalMessage(data, crcRemainder, divisorLength);

        printf("Final message to be sent with CRC: %s\n", data);

        performDivision(data, divisor, receivedRemainder);

        if (checkError(receivedRemainder))

            printf("received correctly\n");

        else

            printf("error\n");

    }

    return 0;
}

```

```
Enter the binary data (or type 'exit' to quit): 11010011101100
Final message to be sent with CRC: 110100111011001111101
received correctly
Enter the binary data (or type 'exit' to quit): 10101010
Final message to be sent with CRC: 101010101000100
received correctly
Enter the binary data (or type 'exit' to quit): 1111
Final message to be sent with CRC: 11110101101
received correctly
Enter the binary data (or type 'exit' to quit): 00000000
Final message to be sent with CRC: 000000000000000
received correctly
Enter the binary data (or type 'exit' to quit): 11111111
Final message to be sent with CRC: 111111111100110
received correctly
Enter the binary data (or type 'exit' to quit): █
```

#### Question 5:

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

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

```
void addZeros(char *data, int divisorLength) {
    int length = strlen(data);

    for (int i = 0; i < divisorLength - 1; i++) {
        data[length + i] = '0';
    }

    data[length + divisorLength - 1] = '\0';
}
```

```

void performDivision(char *data, char *divisor, char *remainder) {

    int dataLen = strlen(data);

    int divisorLen = strlen(divisor);

    strcpy(remainder, data);

    for (int i = 0; i <= dataLen - divisorLen; i++) {

        if (remainder[i] == '1') {

            for (int j = 0; j < divisorLen; j++) {

                remainder[i + j] = remainder[i + j] == divisor[j] ? '0' : '1';

            }

        }

    }

}

void generateFinalMessage(char *data, char *remainder, int divisorLength) {

    int dataLen = strlen(data);

    int remainderLen = strlen(remainder);

    data[dataLen - (divisorLength - 1)] = '\0';

    strcat(data, remainder + remainderLen - (divisorLength - 1));

}

int checkError(char *remainder) {

    int remainderLen = strlen(remainder);

    for (int i = 0; i < remainderLen; i++) {

        if (remainder[i] == '1')

            return 0;

    }

    return 1;

}

int main() {

    char data[100];

    char divisor[100];

    char crcRemainder[200];

    char receivedRemainder[200];

    char command[10];

    while (1) {

        printf("Enter the binary data (or type 'exit' to quit): ");

```

```
scanf("%s", data);

if (strcmp(data, "exit") == 0) {
    break;
}

printf("Enter the binary divisor: ");
scanf("%s", divisor);

int divisorLength = strlen(divisor);

addZeros(data, divisorLength);
performDivision(data, divisor, crcRemainder);
generateFinalMessage(data, crcRemainder, divisorLength);

printf("Final message to be sent with CRC: %s\n", data);

performDivision(data, divisor, receivedRemainder);
if (checkError(receivedRemainder))
    printf("received correctly\n");
else
    printf("error\n");
}

return 0;
}
```

```
PS C:\Users\CHARISH\OneDrive\Desktop\5th sem\COMPUTER NETWORKS (CN )\output> & .\'5.exe'
Enter the binary data (or type 'exit' to quit): 11010011101100
Enter the binary divisor: 10011011
Final message to be sent with CRC: 110100111011000100110
received correctly
Enter the binary data (or type 'exit' to quit): 10101010
Enter the binary divisor: 10011011
Final message to be sent with CRC: 101010100111100
received correctly
Enter the binary data (or type 'exit' to quit): 1111
Enter the binary divisor: 10011011
Final message to be sent with CRC: 11110000010
received correctly
Enter the binary data (or type 'exit' to quit): 00000000
Enter the binary divisor: 10011011
Final message to be sent with CRC: 000000000000000
received correctly
Enter the binary data (or type 'exit' to quit): 11111111
Enter the binary divisor: 10011011
Final message to be sent with CRC: 111111110100010
received correctly
Enter the binary data (or type 'exit' to quit): exit
PS C:\Users\CHARISH\OneDrive\Desktop\5th sem\COMPUTER NETWORKS (CN )\output> 
```