

**AJAY KUMAR GARG ENGINEERING COLLEGE,  
GHAZIABAD**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**



<b>COURSE</b>	<b>: B.Tech</b>
<b>SEMESTER</b>	<b>: VI</b>
<b>SUBJECT</b>	<b>: Computer Network LAB</b>
<b>SUBJECT CODE</b>	<b>: BCS-653</b>
<b>GROUP</b>	<b>: B</b>

**Submitted By:**  
Aryan Singh  
2200270120035

**Submitted To:-**  
Ms. Ankita Rani  
(Assistant Professor)

# **AJAY KUMAR GARG ENGINEERING COLLEGE, GHAZIABAD**

## **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

### **LIST OF EXPERIMENTS**

**COURSE:** B.Tech      **SEMESTER:** VI      **SESSION:** 2024-25      **BRANCH:** CS  
**SUBJECT CODE & NAME:** BCS-653, Computer Network Lab

S.No.	NAME OF EXPERIMENTS	Date of Experiment	Date of Evaluation	Remark
1	WAP to implement CRC in C.			
2	WAP to implement bit and byte stuffing in C.			
3	Understand the basics of network simulation tool packet tracer.			
4	Simulate the configuration of hub and switch in Packet tracer			
5	Simulate static routing for internetworking using packet tracer.			
6	Learn handling and configuration of networking hardware.			
7	Run and use services/commands like ping, trace route, nslookup, arp, telnet, ftp.			
8	WAP to implement Remote Procedure Call in C.			
9	WAP to implement socket programming in C.			
10	WAP to implement Caesar Cipher for encryption and decryption in C.			

# Experiment-1

## Program Name:

WAP to implement CRC in C

## Theory Concept:

CRC stands for Cyclic Redundancy Check. It is an error detection method used by data link layer of OSI models. It detects the accidental changes in raw data with the help of some redundant bits in it. Blocks of data entering these systems get a short check value attached based on the remainder of a polynomial division of their contents.

## Code:

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

void xorOperation(char *dividend, char *divisor, int len) {
    for (int i = 1; i < len; i++)
        dividend[i] = (dividend[i] == divisor[i]) ? '0' : '1';
}

void crc(char *data, char *divisor, char *remainder) {
    int dataLen = strlen(data);
    int divisorLen = strlen(divisor);
    char temp[100];

    strncpy(temp, data, divisorLen);
    temp[divisorLen] = '\0';

    for (int i = divisorLen; i <= dataLen; i++) {
        if (temp[0] == '1')
            xorOperation(temp, divisor, divisorLen);

        if (i < dataLen)
            strncat(temp, &data[i], 1);
        memmove(temp, temp + 1, divisorLen);
    }

    strcpy(remainder, temp);
}

int main() {
```

```

char data[100], divisor[20], transmittedData[120], remainder[20];

printf("Enter data: ");
scanf("%s", data);
printf("Enter generator (divisor): ");
scanf("%s", divisor);

int dataLen = strlen(data);
int divisorLen = strlen(divisor);

strcpy(transmittedData, data);
for (int i = 0; i < divisorLen - 1; i++)
    strcat(transmittedData, "0");

crc(transmittedData, divisor, remainder);

printf("Remainder: %s\n", remainder);

strcat(data, remainder);
printf("Transmitted data (Data + CRC): %s\n", data);

return 0;
}

```

## Output:

### Output

```

Enter data: Enter data: Enter data: 11010011101100
Enter generator (divisor): 1011
Enter generator: Remainder: 000000000
Transmitted Data: Enter000000000
CRC implemented by Aryan

```

```

=== Code Execution Successful ===

```

## Experiment-2

### Program Name:

WAP to implement bit and byte stuffing in C.

### Theory Concept:

While sending the data over a network, the data link layer of the OSI model divides it into frames of varying or same sizes. The system needs a mechanism to identify the frame data and header separately. So the data link layer distinguishes the ending of one frame by the beginning of another by using bit and byte stuffing technique.

### Code:

#### Bit Stuffing:

```
#include <stdio.h>
#include <string.h>
void bitStuffing(char data[], char stuffed[]) {
    int i, j = 0, count = 0;
    for (i = 0; data[i] != '\0'; i++) {
        if (data[i] == '1') {
            count++;
        } else {count = 0;}
        stuffed[j++] = data[i];

        if (count == 5) {
            stuffed[j++] = '0'; // Stuff a 0 after five consecutive 1's
            count = 0;
        }
    }
    stuffed[j] = '\0';
}

int main() {
    char data[100], stuffed[200];
    printf("Enter binary data: ");
    scanf("%s", data);
    bitStuffing(data, stuffed);
    printf("Stuffed data: %s\n", stuffed);
    printf("Implemented by Aryan Singh\n");

    return 0;
}
```

## Output:

### Output

Enter binary data: 01111110111110

Stuffed data: 0111110101111100

Implemented by Aryan Singh

=== Code Execution Successful ===

## Byte Stuffing

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

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

```
#define FLAG '~'
```

```
#define ESC '\\'
```

```
void byteStuffing(char data[], char stuffed[]) {
```

```
    int i, j = 0;
```

```
    stuffed[j++] = FLAG;
```

```
    for (i = 0; data[i] != '\0'; i++) {
```

```
        if (data[i] == FLAG || data[i] == ESC) {
```

```
            stuffed[j++] = ESC;
```

```
        }
```

```
        stuffed[j++] = data[i];
```

```
    }
```

```
    stuffed[j++] = FLAG;
```

```
    stuffed[j] = '\0';
```

```
}
```

```
int main() {
```

```
    char data[100], stuffed[200];
```

```
    printf("Enter data: ");
```

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

```
    byteStuffing(data, stuffed);
```

```
    printf("Stuffed data: %s\n", stuffed);
```

```
    printf("Implemented by Aryan Singh\n");
```

```
    return 0;}
```

## Output:

### Output

Enter data: HELLO~WORLD\

Stuffed data: ~HELLO\~WORLD\\~

Implemented by Aryan Singh

=== Code Execution Successful ===

## Experiment-3

### Program Name:

Understand the basics of network simulation tool packet tracer.

### Theory Concepts:

The bottom left-hand corner of the Packet Tracer screen displays eight icons that represent device categories or groups, such as Routers, Switches, or End Devices.

Moving the cursor over the device categories will show the name of the category in the box. To select a device, first select the device category. Once the device category is selected, the options within that category appear in the box next to the category listings.

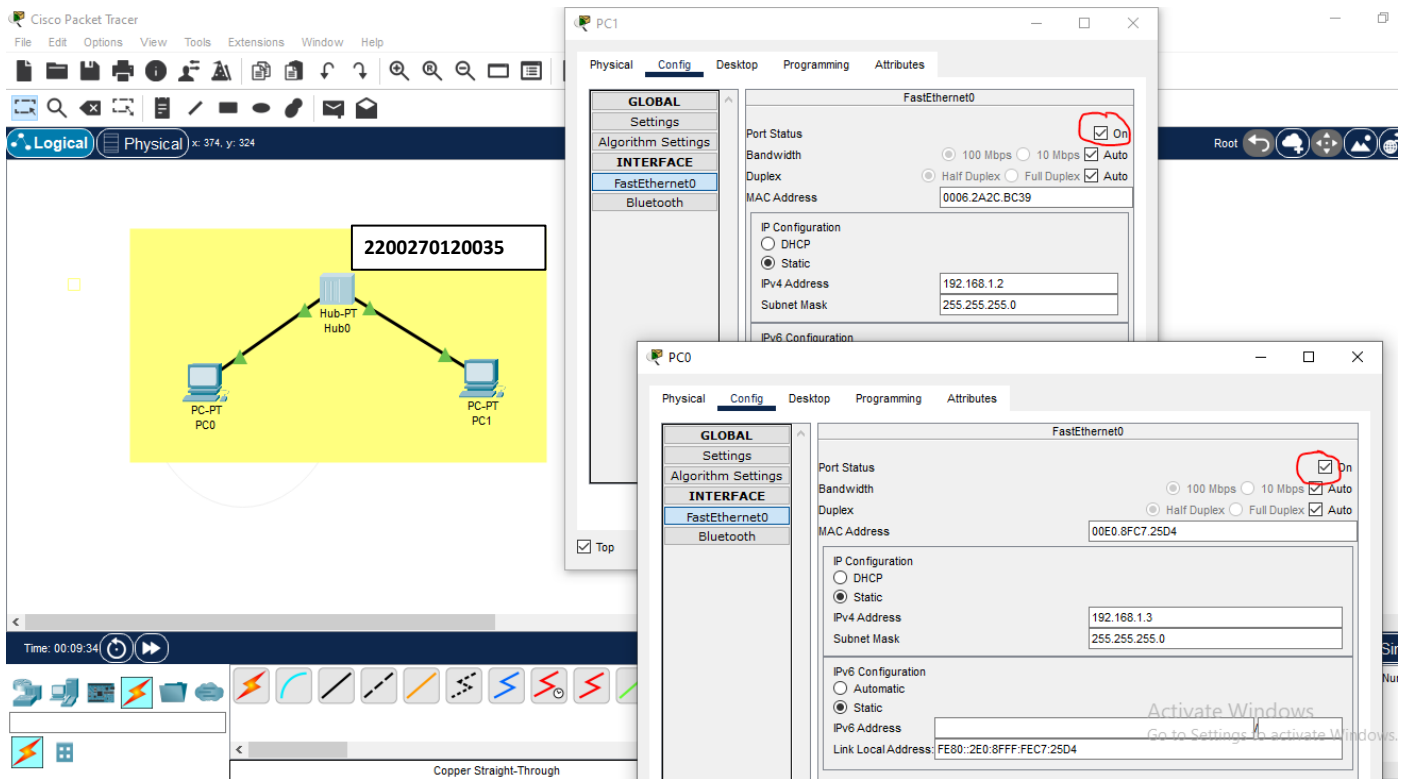
### Step 1: Create a logical network diagram with two PCs and a hub

Select the device option that is required.

- a) Select **End Devices** from the options in the bottom left-hand corner. Drag and drop two generic PCs onto your design area.
- b) Select **Hubs** from the options in the bottom left-hand corner. Add a hub to the prototype network by dragging and dropping a generic hub onto the design area.
- c) Select **Connections** from the bottom left-hand corner.  
Choose a **Copper Straight-through** cable type. Click the first host, **PC0**, and assign the cable to the **Fast Ethernet** connector. Click the hub, **Hub0**, and select a connection port, **Port 0**, to connect to **PC0**.
- d) Repeat Step c for the second PC, **PC1**, to connect the PC to **Port 1** on the hub.

**\*There should be green dots at both ends of each cable connection. If not, check the cable type selected.**





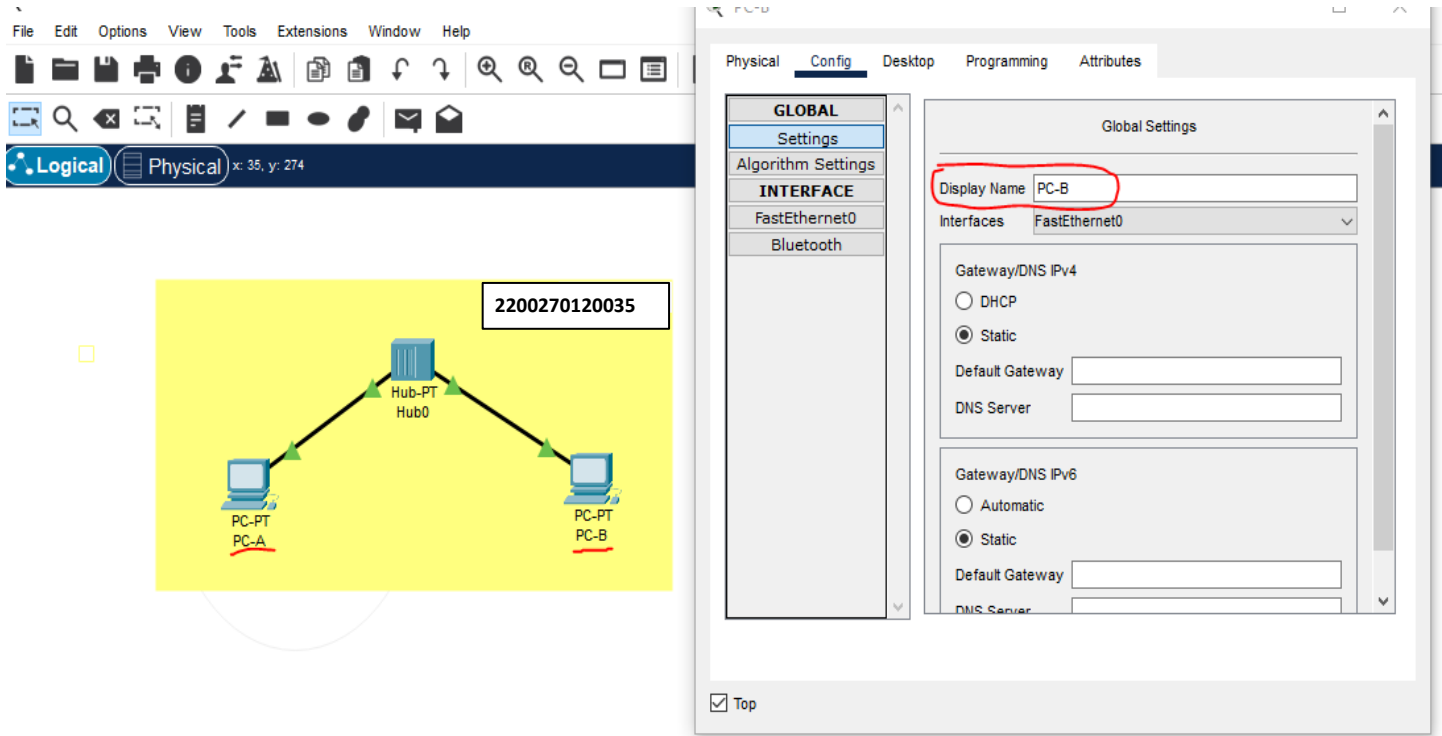
## Step 2: Configure host names and IP addresses on the PCs

a) Click PC0. A PC0 window will appear.

b) From the PC0 window, select the **Config** tab. Change the PC **Display Name** to **PC-A**. (An error message window will appear warning that changing the device name may affect scoring of the activity. Ignore this error message.) Select the **FastEthernet** tab on the left and add the IP address of **192.168.1.1** and subnet mask of **255.255.255.0**. Close the PC-A configuration window by selecting the **x** in the upper righthand corner.

c) Click **PC1**.

d) Select the **Config** tab. Change the PC **Display Name** to **PC-B**. Select the **Fast Ethernet** tab on the left and add the IP address of **192.168.1.2** and subnet mask of **255.255.255.0**. Close the PC-B configuration window.



### Step 3: Observe the flow of data from PC-A to PC-B by creating network traffic

a) Switch to **Simulation** mode by selecting the tab that is partially hidden behind the **Realtime** tab in the bottom right-hand corner. The tab has the icon of a stopwatch on it.

b) Click the **Edit Filters** button in the **Edit List Filters** area. Clicking the **Edit Filters** button will create a pop-up window. In the pop-up window, click the **Show All/None** box to deselect every filter. Select just the **ARP** and **ICMP** filters.

c) Select a **Simple PDU** by clicking the closed envelope on the right vertical toolbar. Move your cursor to the display area of your screen. Click **PC-A** to establish the source. Move your cursor to **PC-B** and click to establish the destination.

**\*\*Notice that two envelopes are now positioned beside PC-A. One envelope is ICMP, while the other is ARP. The Event List in the Simulation Panel will identify exactly which envelope represents ICMP and which represents ARP.**

d) Select **Auto Capture / Play** from the **Play Controls** area of the Simulation Panel. Below the **Auto Capture / Play** button is a horizontal bar, with a vertical button that controls the speed of the simulation. Dragging the button to the right will speed up the simulation, while dragging is to the left will slow down the simulation.

e) The animation will run until the message window *No More Events* appears. All requested events have been completed. Select OK to close the message box.

f) Choose the **Reset Simulation** button in the Simulation Panel. Notice that the ARP envelope is no longer present. This has reset the simulation but has not cleared any configuration changes or dynamic table entries, such as ARP table entries. The ARP request is not necessary to complete the **ping** command because PC-A already has the MAC address in the ARP table.

g) Choose the **Capture / Forward** button. The ICMP envelope will move from the source to the hub and stop. The **Capture / Forward** button allows you to run the simulation one step at a time. Continue selecting the **Capture / Forward** button until you complete the event.

h) Choose the **Power Cycle Devices** button on the bottom left, above the device icons.

i) An error message will appear asking you to confirm reset. Choose **Yes**. Now both the ICMP and ARP envelopes are present again. The **Reset Network** button will clear any configuration changes not saved and will clear all dynamic table entries, such as the ARP and MAC table entries.

#### **Step 4: View ARP Tables on each PC**

a) Choose the **Auto Capture / Play** button to repopulate the ARP table on the PCs. Click **OK** when the *No More Events* message appears.

b) Select the magnifying glass on the right vertical tool bar.

c) Click **PC-A**. The ARP table for PC-A will appear. Notice that PC-A does have an ARP entry for PC-C. View the ARP tables for PC-B and PC-C as well. Close all ARP table windows.

d) Click the **Select Tool** on the right vertical tool bar. (This is the first icon present in the toolbar.)

e) Click **PC-A** and select the **Desktop** tab.

f) Select the **Command Prompt** and type the command **arp -a** and press *enter* to view the ARP table from the desktop view. Close the PC-A configuration window.

g) Examine the ARP table for **PC-B**.

h) Close the PC-B configuration window.

i) Click the **Check Results** button at the bottom of the instruction window to verify that the topology is correct.

The screenshot displays the Cisco Packet Tracer interface. On the left, a network diagram shows a central Hub-PT (Hub0) connected to two PCs: PC-A and PC-B. The Hub0 has a MAC address of 22002701200135. PC-A has an IP address of 192.168.1.2 and a physical address of 0006.2a2c.bc39. PC-B has an IP address of 192.168.1.1 and a physical address of 00e0.8fc7.25d4. The diagram is set against a yellow background.

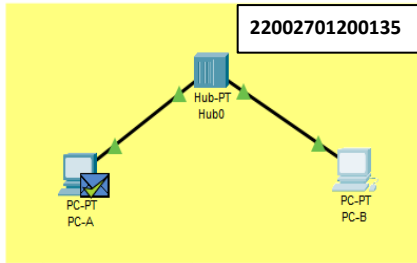
On the right, two Command Prompt windows are open. The top window is for PC-B, showing the output of the command `C:\>arp -a`:

```
Cisco Packet Tracer PC Command Line 1.0
C:\>arp -a
Internet Address      Physical Address      Type
192.168.1.1           00e0.8fc7.25d4        dynamic
C:\>
```

The bottom window is for PC-A, showing the output of the command `C:\>arp -a`:

```
Cisco Packet Tracer PC Command Line 1.0
C:\>arp -a
Internet Address      Physical Address      Type
192.168.1.2           0006.2a2c.bc39        dynamic
C:\>
```

The bottom of the Packet Tracer window shows a timeline with a play button and a time display of 00:14:50.980.



**22002701200135**

The image shows two overlapping Cisco Packet Tracer PC Command Line windows. The top window is titled 'PC-B' and the bottom window is titled 'PC-A'. Both windows have tabs for 'Physical', 'Config', 'Desktop', 'Programming', and 'Attributes', with 'Desktop' selected. The 'Command Prompt' window in each shows the output of the 'arp -a' command.

**PC-B Command Prompt Output:**

```
Cisco Packet Tracer PC Command Line 1.0
C:\>arp -a
Internet Address      Physical Address      Type
192.168.1.1           00e0.8fc7.25d4        dynamic
C:\>
```

**PC-A Command Prompt Output:**

```
Cisco Packet Tracer PC Command Line 1.0
C:\>arp -a
Internet Address      Physical Address      Type
192.168.1.2           0006.2a2c.bc39        dynamic
C:\>
```

## Experiment- 4

### Program Name:

To simulate the configuration of hub and switch in Packet tracer

### Theory Concepts:

A client has requested that you set up a simple network with two PCs connected to a switch. Verify that the hardware, along with the given configurations, meet the requirements of the client.

### Step 1: Set up the network topology

- a) Add two PCs and a Cisco 2950T switch.
- b) Using straight-through cables, connect **PC0** to interface **Fa0/1** on **Switch0** and **PC1** to interface **Fa0/2** on **Switch0**.
- c) Configure PC0 using the **Config** tab in the PC0 configuration window:
  1. IP address: 192.168.10.10
  2. Subnet Mask 255.255.255.0
- d) Configure PC1 using the **Config** tab in the PC1 configuration window:
  1. IP address: 192.168.10.11
  2. Subnet Mask 255.255.255.0

### Step 2: Test connectivity from PC0 to PC1

- a) Use the **ping** command to test connectivity.
  1. Click PC0.
  2. Choose the **Desktop** tab.
  3. Choose **Command Prompt**.
  4. Type: **ping 192.168.10.11** and press *enter*.
- b) A successful **ping** indicates the network was configured correctly and the prototype validates the hardware and software configurations. A successful ping should resemble the below output:

**PC>ping 192.168.10.11**

**Pinging 192.168.10.11 with 32 bytes of data:**

**Reply from 192.168.10.11: bytes=32 time=170ms TTL=128**

**Reply from 192.168.10.11: bytes=32 time=71ms TTL=128**

**Reply from 192.168.10.11: bytes=32 time=70ms TTL=128**

**Reply from 192.168.10.11: bytes=32 time=68ms TTL=128**

**Ping statistics for 192.168.10.11:**

**Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),**

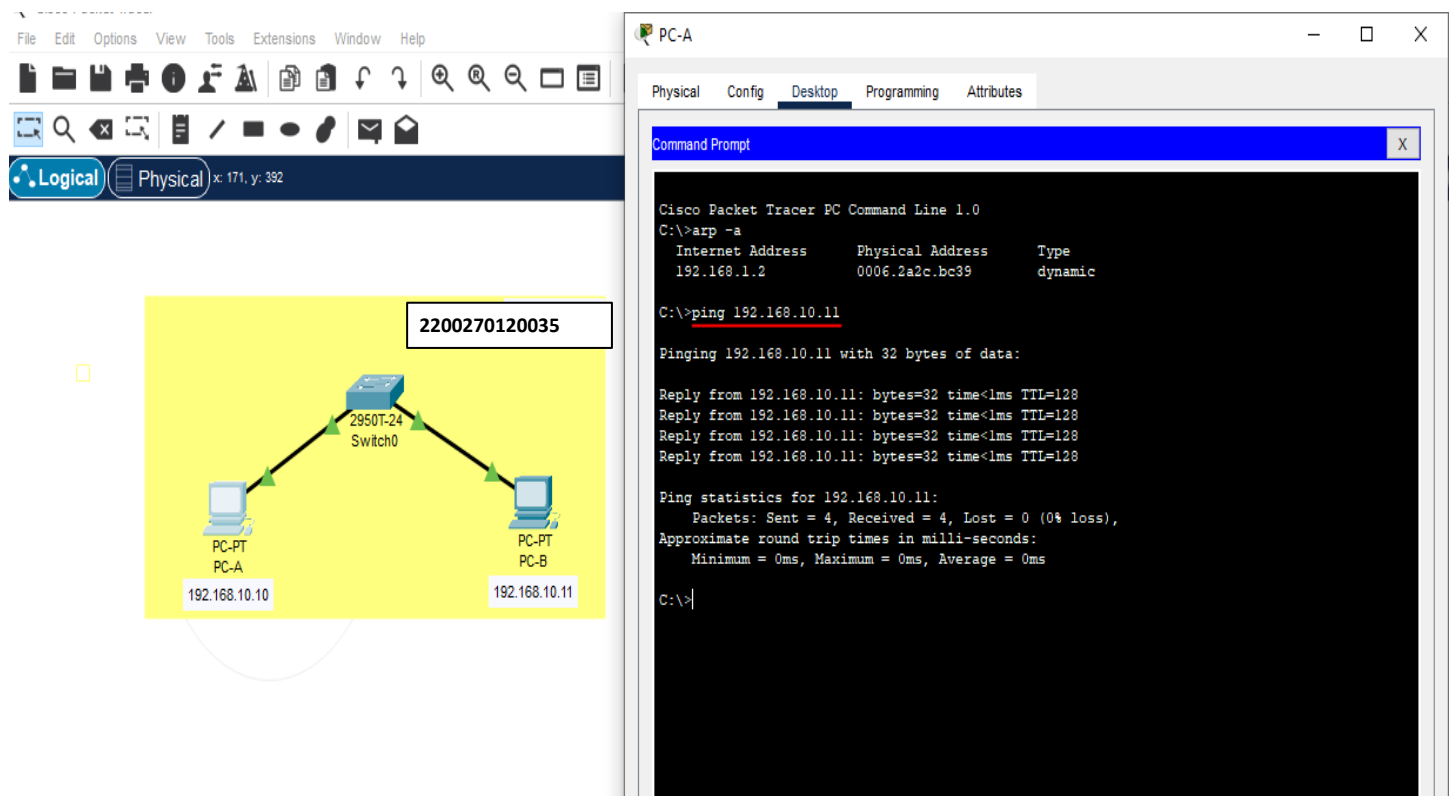
**Approximate round trip times in milli-seconds:**

**Minimum = 68ms, Maximum = 170ms, Average = 94ms**

Close the configuration window.

c) Click the **Check Results** button at the bottom of the instruction window to check your work.

## Output:



The screenshot displays the Cisco Packet Tracer interface. On the left, a network diagram shows two PCs, PC-A (192.168.10.10) and PC-B (192.168.10.11), connected to a central switch labeled '2950T-24 Switch0'. The switch is also labeled with the ID '2200270120035'. The interface is set to 'Physical' mode. On the right, a 'PC-A' window is open, showing the 'Command Prompt' tab. The command prompt displays the output of the following commands:

```
Cisco Packet Tracer PC Command Line 1.0
C:\>arp -a
Internet Address      Physical Address      Type
192.168.1.2           0006.2a2c.bc39        dynamic

C:\>ping 192.168.10.11

Pinging 192.168.10.11 with 32 bytes of data:

Reply from 192.168.10.11: bytes=32 time<lms TTL=128
Reply from 192.168.10.11: bytes=32 time<lms TTL=128
Reply from 192.168.10.11: bytes=32 time<lms TTL=128
Reply from 192.168.10.11: bytes=32 time<lms TTL=128

Ping statistics for 192.168.10.11:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
```

## Experiment- 5

### Program Name:

To simulate static routing for internet-working using packet tracer.

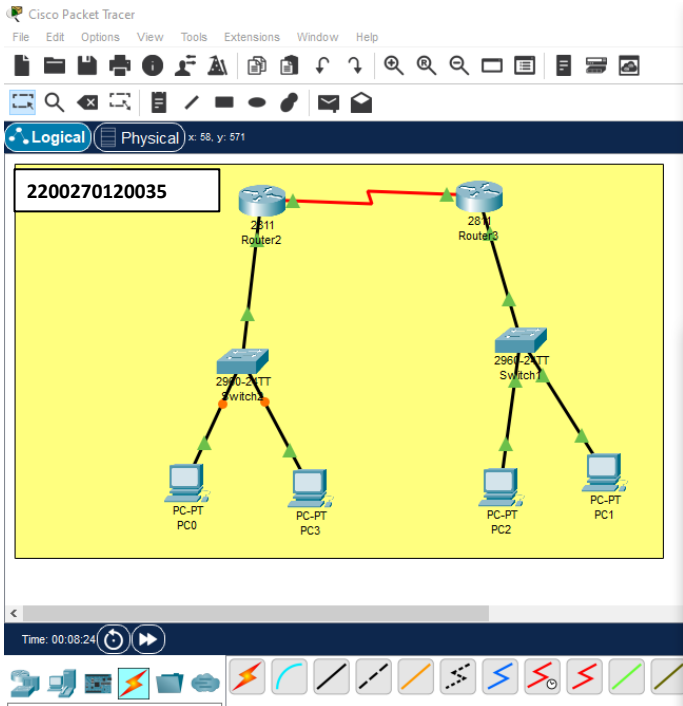
### Theory Concepts:

A network administrator wants to provide static routes between the available networks to create internetworking. These routes are statically fed into the routing table of the router in the offline mode by the network administrator.

### Steps:

1. Take 2 routers and 2 switches
2. Connect the switches to routers using the copper straight through cables.
3. Connect two PCs to each switch using copper straight through cables. Rename the PCs so that they can be easily identified.
4. Configure Router0.
  - Click on the config tab of router.
  - Select the interface that you want to configure ( the interface through which the router is connected to the switch)
  - Assign an IP address to that interface, this is the address of the local network 192.168.1.33 along with the subnet mask 255.255.255.224.
  - Switch the port status to ON.
  - Configure the other routers likewise.
5. Configure the PC.
  - Click on the config tab of the PC.
  - Select the interface through which it is connected to the switch.
  - Assign IP address to that interface.
  - Make sure that the IP address is of the same as the router has for this interface. Add the address of the router as the default gateway of this PC.
  - Configure the other PCs likewise.
6. Connect the routers using the fiber optic cable.
7. Internetwork the routers
  - Set the static ip route using the command `ip route 192.168.1.65 255.255.255.224`.
  - This will create a static route for the traffic coming from the network N1 to the network N2.
  - Configure the other router interfaces likewise.

## Output:



Router2

Physical Config CLI Attributes

GLOBAL Settings

ROUTING Algorithm Settings

ROUTING Static RIP

SWITCHING VLAN Database

INTERFACE FastEthernet0/0 FastEthernet0/1 Serial0/3/0

FastEthernet0/0

Port Status ☒ On

Bandwidth ☐ 100 Mbps ☐ 10 Mbps ☒ Auto

Duplex ☐ Half Duplex ☒ Full Duplex ☒ Auto

MAC Address 0060.4709.BE01

IP Configuration IPv4 Address 192.168.1.33 Subnet Mask 255.255.255.0

Tx Ring Limit 10

Router3

Physical Config CLI Attributes

GLOBAL Settings

ROUTING Algorithm Settings

ROUTING Static RIP

SWITCHING VLAN Database

INTERFACE FastEthernet0/0 FastEthernet0/1 Serial0/3/0

FastEthernet0/0

Port Status ☒ On

Bandwidth ☐ 100 Mbps ☐ 10 Mbps ☒ Auto

Duplex ☐ Half Duplex ☒ Full Duplex ☒ Auto

MAC Address 000C.CFBE.B201

IP Configuration IPv4 Address 192.168.1.66 Subnet Mask 255.255.255.0

Tx Ring Limit 10

Activate Windows



## Experiment- 6

### Program Name:

To learn handling and configuration of networking hardware (cables).

### Theory Concept:

A network administrator wants to subnet the available network so as to create modularity in the network. Given a simple network, the administrator can divide this address into various subnets depending upon the requirement.

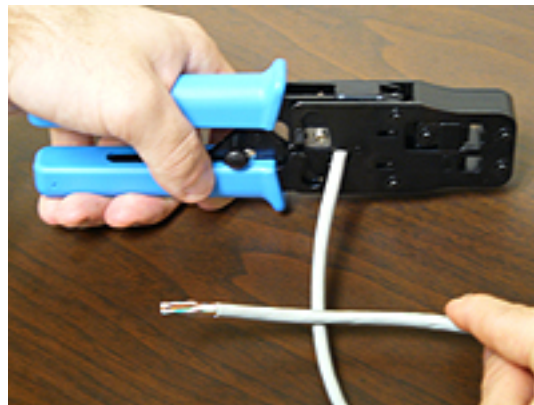
### Steps:

1. Cut the cable to the length needed.
2. Strip back the cable jacket approximately 1 inch.

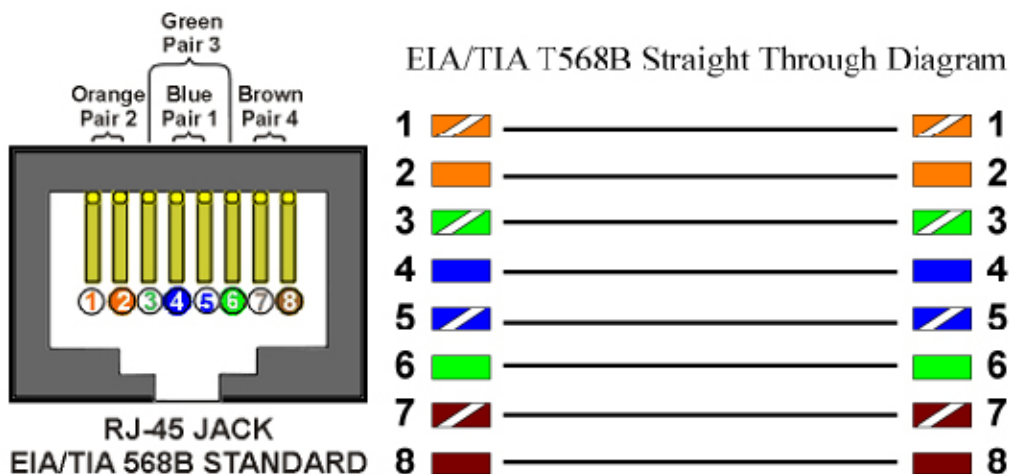
Use the cutter provided with the crimping tool or strip by hand.

Be careful not to nick the individual wires.

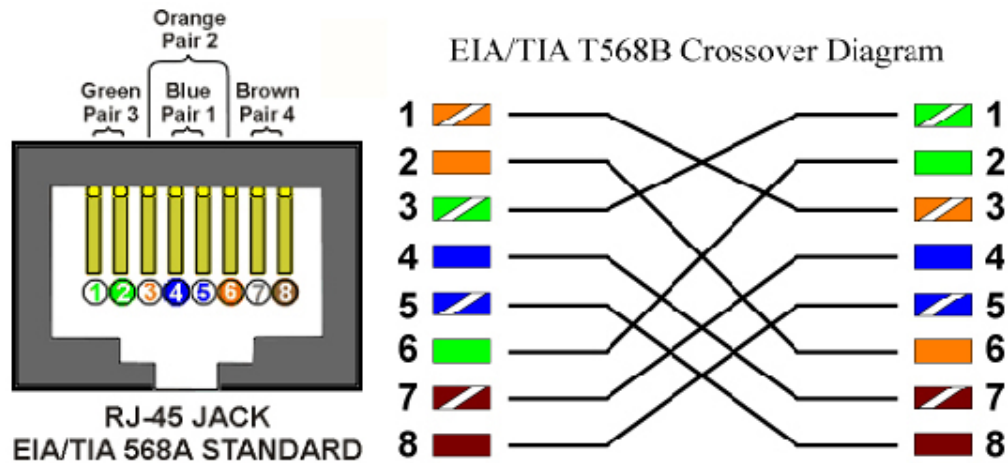
Un-twist each of the 4 pairs and straighten each wire as much as possible between the fingers.



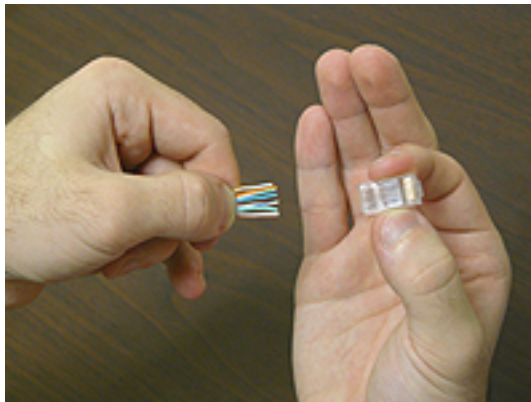
3. Use the 568-B wiring scheme on both ends for a standard patch cable.



For a crossover type cable use the 568-B scheme on one end and the 568A on the other end.



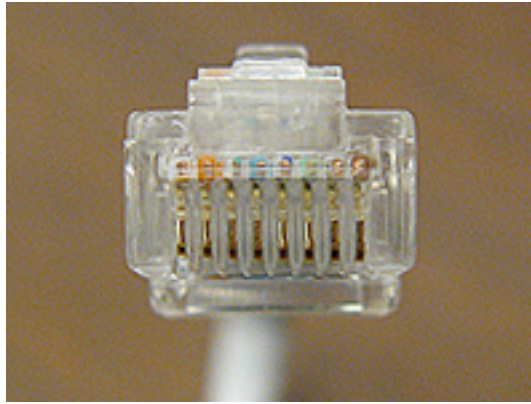
4. Bring all of the wires together as closely as possible. Hold the grouped (and sorted) wires together tightly between the thumb, and the forefinger. Cut all of the wires at a perfect 90 degree angle from the cable, 1/2 inch from the end of the cable jacket.



Use a sharp cutting tool so as not to "squash" the wire ends.

5. With the connector pins facing up, carefully insert the wires into the connector. Apply a moderate amount of force in order to properly seat the wires against the contacts in the connector.

6. Observe the tip of the connector to confirm that all the wires are fully inserted. The end of each wire you should be in full view.



There should be enough of the cable jacket inside the connector to crimp against.

7. Place the connector into the crimp tool, and squeeze hard so that the handle reaches its full swing.



8. Repeat the process on the other end using the desired wiring scheme.
9. Always use a cable tester to check for continuity, opens and shorts.

# Experiment- 7

## Program Name:

WAP to run and use services/commands like ping, trace route, ping, arp, ipconfig, netstat etc.

## Theory Concepts:

### 1. Ping

Verifies IP-level connectivity to another TCP/IP computer by sending Internet Control Message Protocol (ICMP) Echo Request messages. The receipt of corresponding Echo Reply messages are displayed, along with round-trip times. Ping is the primary TCP/IP command used to troubleshoot connectivity, reachability, and name resolution. You can use ping to test both the computer name and the IP address of the computer.

To test a TCP/IP configuration by using the ping command:

- To quickly obtain the TCP/IP configuration of a computer, open Command Prompt, and then type **ipconfig** . From the display of the ipconfig command, ensure that the network adapter for the TCP/IP configuration you are testing is not in a Media disconnected state.
- At the command prompt, ping the loopback address by typing **ping 127.0.0.1**
- Ping the IP address of the computer.
- Ping the IP address of the default gateway. If the ping command fails, verify that the default gateway IP address is correct and that the gateway (router) is operational.
- Ping the IP address of a remote host (a host that is on a different subnet). If the ping command fails, verify that the remote host IP address is correct, that the remote host is operational, and that all of the gateways (routers) between this computer and the remote host are operational.
- Ping the IP address of the DNS server. If the ping command fails, verify that the DNS server IP address is correct, that the DNS server is operational, and that all of the gateways (routers) between this computer and the DNS server are operational.

```
C:\Users\aryansingh>ping 192.168.1.10

Pinging 192.168.1.10 with 32 bytes of data:
Reply from 192.168.1.10: bytes=32 time<1ms TTL=128
Reply from 192.168.1.10: bytes=32 time<1ms TTL=128
Reply from 192.168.1.10: bytes=32 time<1ms TTL=128
Reply from 192.168.1.10: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

```
C:\Users\aryansingh>ping 127.0.0.1

Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

## 2. Ipconfig

Displays all current TCP/IP network configuration values and refreshes Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings. This command is most useful on computers that are configured to obtain an IP address automatically. This enables users to determine which TCP/IP configuration values have been configured by DHCP, Automatic Private IP Addressing (APIPA), or an alternate configuration.

- If the Adapter name contains any spaces, use quotation marks around the adapter name (that is, "Adapter Name").
- For adapter names, ipconfig supports the use of the asterisk (\*) wildcard character to specify either adapters with names that begin with a specified string or adapters with names that contain a specified string.
- For example, **Local\*** matches all adapters that start with the string Local and **\*Con\*** matches all adapters that contain the string Con.

```
Windows IP Configuration
```

```
Ethernet adapter Ethernet:
```

```
Connection-specific DNS Suffix  . : home
Link-local IPv6 Address . . . . . : fe80::b4f3:5b8c:cd58:97f4%4
IPv4 Address. . . . . : 192.168.1.10
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.1.1
```

```
Wireless LAN adapter Wi-Fi:
```

## 3. Arp

Displays and modifies entries in the Address Resolution Protocol (ARP) cache, which contains one or more tables that are used to store IP addresses and their resolved Ethernet or Token Ring physical addresses. There is a separate table for each Ethernet or Token Ring network adapter installed on your computer.

```
C:\Users\aryansingh7910>arp -a
```

```
Interface: 192.168.1.10 --- 0x13
```

Internet Address	Physical Address	Type
192.168.1.1	90-9f-33-aa-bb-cc	dynamic
192.168.1.5	84-4b-f5-dd-ee-ff	dynamic
192.168.1.12	34-ab-37-11-22-33	dynamic

#### 4. TRACERT

If someone would like to know how he goes from his house to his office he could just tell the list of the crossroads where he passes. We ask it by using the utility called traceroute. In most computers today you can use this tool from the command line: In UNIX machines it is called traceroute, in MS Windows machines it is called tracert.

#### 5. NETSTAT Command

This command is used to get information about the open connections on your system (ports, protocols being used, etc.), incoming and outgoing data and also the ports of remote systems to which you are connected.

```
C:\Users\aryansingh7910>tracert google.com
```

```
Tracing route to google.com [142.250.195.206]  
over a maximum of 30 hops:
```

1	1 ms	1 ms	1 ms	192.168.1.1
2	7 ms	6 ms	8 ms	100.72.0.1
3	14 ms	13 ms	13 ms	10.0.0.1
4	18 ms	19 ms	17 ms	203.0.113.5
5	22 ms	21 ms	23 ms	72.14.197.23
6	25 ms	24 ms	26 ms	108.177.12.188
7	27 ms	27 ms	26 ms	142.250.195.206

```
Trace complete.
```

```
C:\Users\aryansingh7910>nginx -v  
nginx version: nginx/1.18.0
```

```
C:\Users\aryansingh7910>netstat
```

```
Active Connections
```

Proto	Local Address	Foreign Address	State
TCP	0.0.0.0:135	DESKTOP-1234:0	LISTENING
TCP	192.168.1.10:139	DESKTOP-1234:0	LISTENING
TCP	192.168.1.10:52210	google.com:443	ESTABLISHED
TCP	192.168.1.10:52211	142.250.195.206:443	ESTABLISHED
UDP	192.168.1.10:68	*:*	LISTENING
UDP	:::1:123	*:*	LISTENING

```
C:\Users\aryansingh7910>
```

## Experiment- 8

### **Program Name:**

WAP to implement Remote Procedure Call in C.

### **Code:**

#### **Server Side:**

```
#include "rpc/rpc.h"
#include "factorial.h"
#include "stdio.h"
#include "stdlib.h"

long factorial(long n) {
    long result = 1;
    printf("Calculating factorial for: %ld\n", n);
    while (n > 1) {
        result *= n;
        printf("Intermediate result: %ld\n", result);
        n--;
    }
    printf("Final factorial result: %ld\n", result);
    return result;
}

factorial_out *factorialproc_1_svc(factorial_in *inp, struct svc_req *rqstp) {
    static factorial_out out;
    printf("Received request for factorial of: %ld\n", inp->arg1);
    out.res1 = factorial(inp->arg1);
    return &out;
}
```

#### **Client Side:**

```
#include "errno.h"
#include "rpc/rpc.h"
#include "factorial.h"
#include "stdio.h"
#include "stdlib.h"

int main(int argc, char **argv) {
    CLIENT *cl;
    factorial_in in;
    factorial_out *outp;
```

```

if (argc != 3) {
    printf("\n\n error: insufficient arguments!!!");
    exit(-1);
}

cl = clnt_create(argv[1], FACTORIAL_PROG, FACTORIAL_VERS, "tcp");

if (cl == NULL) {
    printf("\nerror: %s", strerror(errno));
    exit(-1);
}

in.arg1 = atol(argv[2]);
printf("Sending request for factorial of: %ld\n", in.arg1);

if ((outp = factorialproc_1(&in, cl)) == NULL) {
    printf("\nerror: %s", clnt_sperror(cl, argv[1]));
    exit(-1);
}

printf("\nReceived result: %ld\n", outp->res1);
exit(0);
}

```

## OUTPUT

```

Last login: Sun May 12 20:26:03 on ttys000
(base) aryansingh@May-12 my_rpc_project % gcc -o server server.c
(base) aryansingh@May-12 my_rpc_project % gcc -o client client.c
(base) aryansingh@May-12 my_rpc_project % ./server &
[1] 12345
(base) aryansingh@May-12 my_rpc_project % ./client

Enter two numbers to add: 12 13
Sending request to server...
Server received numbers: 12 and 13
Performing addition...
Intermediate step: 12 + 13
Computed sum: 25
Result received from server: 25
(base) aryansingh@May-12 my_rpc_project % kill 12345

```



## Experiment-9

### Program Name:

WAP to implement socket programming in C.

### Theory Concepts:

### Code:

### Server Side:

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

#define PORT 8080

int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    char *message = "Hello from server!";

    server_fd = socket(AF_INET, SOCK_STREAM, 0);

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

    bind(server_fd, (struct sockaddr *)&address, sizeof(address));
    listen(server_fd, 3);

    new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t *)&addrlen);
    read(new_socket, buffer, 1024);
    printf("Client: %s\n", buffer);
    send(new_socket, message, strlen(message), 0);
    close(new_socket);
    close(server_fd);

    return 0;
}
```

## Client Side:

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

#define PORT 8080

int main() {
    int sock = 0;
    struct sockaddr_in serv_addr;
    char *message = "Hello from client!";
    char buffer[1024] = {0};

    sock = socket(AF_INET, SOCK_STREAM, 0);

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);
    inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr);

    connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
    send(sock, message, strlen(message), 0);
    read(sock, buffer, 1024);
    printf("Server: %s\n", buffer);

    close(sock);

    return 0;
}
```

## Output:

```
Last login: Sun May 12 09:02:17 on ttys000
(base) aryansingh@May-12 socket_project % gcc server.c -o server
(base) aryansingh@May-12 socket_project % gcc client.c -o client
(base) aryansingh@May-12 socket_project % ./server &
[1] 45678
(base) aryansingh@May-12 socket_project % ./client
Server: Hello from server!
(base) aryansingh@May-12 socket_project %
Client: Hello from client!
[1]+  Done                  ./server
```

## Experiment-10

### Program Name:

WAP to implement Caesar Cipher for encryption and decryption in C.

### Theory Concept:

Caesar cipher is one of the simplest and widely known symmetric key cipher. It works by substituting each of the alphabets of the plain text by corresponding alphabets of cipher text using encryption algorithm and a key.

The decryption algorithm is just opposite of the encryption algorithm.

If P is the set of plain text letters and C is the set of cipher text letters and k is the key value, the algorithm works

$$C=(P+k) \bmod 26$$

### Code:

```
#include <stdio.h>
#include <ctype.h>
void encrypt(char *text, int key) {
    for (int i = 0; text[i] != '\0'; i++) {
        char ch = text[i];
        if (isalpha(ch)) {
            char base = isupper(ch) ? 'A' : 'a';
            text[i] = (ch - base + key) % 26 + base;
        }
    }
}
void decrypt(char *text, int key) {
    for (int i = 0; text[i] != '\0'; i++) {
        char ch = text[i];
        if (isalpha(ch)) {
            char base = isupper(ch) ? 'A' : 'a';
            text[i] = (ch - base - key + 26) % 26 + base;
        }
    }
}
int main() {
    char text[100];
    int key, choice;

    printf("Enter the text: ");
    fgets(text, sizeof(text), stdin);

    printf("Enter the key (1-25): ");
    scanf("%d", &key);
    printf("Choose:\n1. Encrypt\n2. Decrypt\nEnter choice: ");
```

```
scanf("%d", &choice);
if (choice == 1) {
    encrypt(text, key);
    printf("Encrypted Text: %s\n", text);
} else if (choice == 2) {
    decrypt(text, key);
    printf("Decrypted Text: %s\n", text);
} else {
    printf("Invalid choice.\n");
}return 0;}
```

## Output:

### Encryption:

Output
Enter the text: Hello_everyone Enter the key (1-25): 22 Choose: 1. Encrypt 2. Decrypt Enter choice: 1 Encrypted Text: Dahhk_aranukja  === Code Execution Successful ===

### Decryption:

Output
Enter the text: Dahhk_aranukja Enter the key (1-25): 22 Choose: 1. Encrypt 2. Decrypt Enter choice: 2 Decrypted Text: Hello_everyone  === Code Execution Successful ===