Solutions

Part A

1. **Perform a detailed study of the following:**
    a. **Types of Network cables.**
    b. **Network Devices.**
    c. **Network IP.**
    d. **Basic network command and Network configuration commands.**

a. **Types of network cables**

Fiber optic cable, twisted pair cable, and coaxial cable are the three main types of network cables used in communication systems. Each of them is different and suitable for various applications.

Fiber Optic Cable

Fiber optic cable consists of a bundle of glass threads, each of which is capable of transmitting messages modulated onto light waves.

Fiber Optic cable has a complicated design and structure. This type of cable has an outer optical casing that surrounds the light and traps it within a central core. The inside of the cable (the core) must be configured in two different ways – Single-mode and multi-mode; although the difference may seem small, it makes a tremendous difference to the performance and the usage of fiber optic cables.

Twisted Pair Cable

Twisted pair cable is a type of ordinary wiring which connects home and many business computers to the telephone company. It is made by putting two separate insulated wires together in a twisted pattern and running them parallel to each other, which helps to reduce crosstalk or electromagnetic induction between pairs of wires. Twisted pair cable is suitable for transferring balanced differential signals. The method of transmitting signals dates to the early days of the telegraph and radio. The advantages of improved signal-to-noise ratio, crosstalk, and ground bounce that balanced signal transmission brings are particularly valuable in wide bandwidth and high-fidelity systems.

Coaxial Cable

Coaxial cable, or coax cable, is another type of copper cable which has an inner conductor surrounded by foam insulation, symmetrically wrapped by a woven braided metal shield, then covered by in a plastic jacket (as shown in the following image). This unique design allows coaxial cable runs to installed next to metal objects such as gutters without the power losses that occur in other types of transmission lines. The coaxial cable acts as a high-frequency transmission cable made up of a single solid copper core and compared to twisted pair cable. It has 80 times or more transmission capability. This kind of cable is mainly adopted in feedlines connecting radio transmitters and receivers with their antennas, computer network connections, and distributing cable television signals.

Cross-wired and straight through cable

An Ethernet cable is a network cable used for high-speed wired network connections between two devices. This network cable is made of four-pair cable, which is consists of twisted pair conductors. It is used for data transmission at both ends of the cable, which is called RJ45 connector.

The Ethernet cables are categorized as Cat 5, Cat 5e, Cat 6, and UTP cable. Cat 5 cable can support a 10/100 Mbps Ethernet network while Cat 5e and Cat 6 cable to support Ethernet network running at 10/100/1000 Mbps.

T568A and T568B Wiring Standard Basis

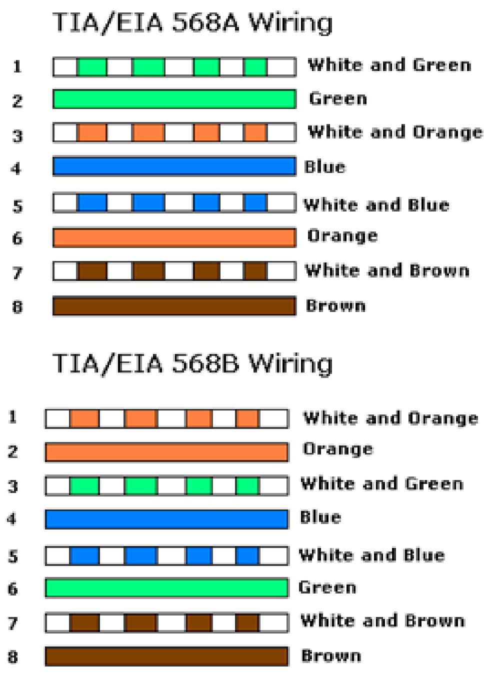
A RJ45 connector is a modular 8 position, 8 pin connector used for terminating Cat5e or Cat6 twisted pair cable. A pinout is a specific arrangement of wires that dictate how the connector is terminated. There are two standards recognized by ANSI, TIA and EIA for wiring Ethernet cables. The first is the T568A wiring standard and the second is T568B. T568B has surpassed 568A and is seen as the default wiring scheme for twisted pair structured cabling. If you are unsure of which to use, choose 568B.

Straight Through Cable

A straight through cable is a type of twisted pair cable that is used in local area networks to connect a computer to a network hub such as a router. This type of cable is also sometimes called a patch cable and is an alternative to wireless connections where one or more computers access a router through a wireless signal. On a straight through cable, the wired pins match. Straight through cable use one wiring standard: both ends use T568A wiring standard or both ends use T568B wiring standard.

Crossover Cable

An Ethernet crossover cable is a type of Ethernet cable used to connect computing devices together directly. Unlike straight through cable, crossover cables use two different wiring standards: one end uses the T568A wiring standard, and the other end uses the T568B wiring standard. The internal wiring of Ethernet crossover cables reverses the transmit and receive signals. It is most often used to connect two devices of the same type: e.g., two computers (via network interface controller) or two switches to each other.



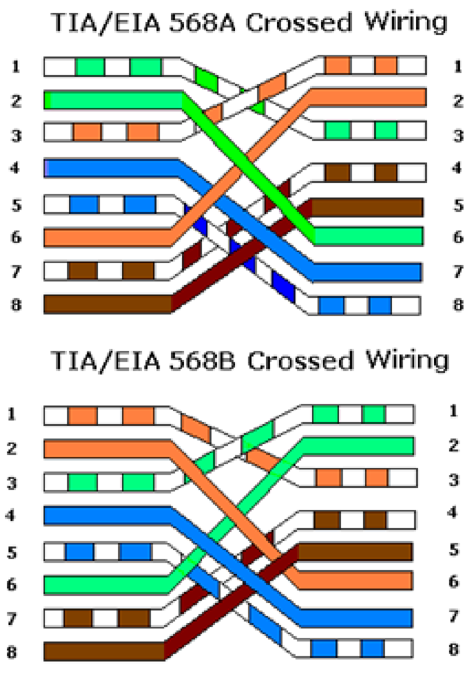


Figure A

Shows the Pin Out of Straight through Cables

Figure B

Shows the Pin Out of Crossover Cables

b. Network Devices.

1. Hub
Hubs connect multiple computer networking devices together. A hub also acts as a repeater in that it amplifies signals that deteriorate after traveling long distances over connecting cables. A hub is the simplest in the family of network connecting devices because it connects LAN components with identical protocols.

A hub can be used with both digital and analog data, provided its settings have been configured to prepare for the formatting of the incoming data. For example, if the incoming data is in digital format, the hub must pass it on as packets; however, if the incoming data is analog, then the hub passes it on in signal form.

Hubs do not perform packet filtering or addressing functions; they just send data packets to all connected devices. Hubs operate at the Physical layer of the Open Systems Interconnection (OSI) model. There are two types of hubs: simple and multiple port.

2. Switch
Switches generally have a more intelligent role than hubs. A switch is a multiport device that improves network efficiency. The switch maintains limited routing information about nodes in the internal network, and it allows connections to systems like hubs or routers. Strands of LANs are usually connected using switches. Generally,

switches can read the hardware addresses of incoming packets to transmit them to the appropriate destination.

Using switches improves network efficiency over hubs or routers because of the virtual circuit capability. Switches also improve network security because the virtual circuits are more difficult to examine with network monitors. You can think of a switch as a device that has some of the best capabilities of routers and hubs combined. A switch can work at either the Data Link layer or the Network layer of the OSI model. A multilayer switch is one that can operate at both layers, which means that it can operate as both a switch and a router. A multilayer switch is a high-performance device that supports the same routing protocols as routers.

Switches can be subject to distributed denial of service (DDoS) attacks; flood guards are used to prevent malicious traffic from bringing the switch to a halt. Switch port security is important so be sure to secure switches: Disable all unused ports and use DHCP snooping, ARP inspection and MAC address filtering.

3. Router
Routers help transmit packets to their destinations by charting a path through the sea of interconnected networking devices using different network topologies. Routers are intelligent devices, and they store information about the networks they're connected to. Most routers can be configured to operate as packet-filtering firewalls and use access control lists (ACLs). Routers, in conjunction with a channel service unit/data service unit (CSU/DSU), are also used to translate from LAN framing to WAN framing. This is needed because LANs and WANs use different network protocols. Such routers are known as border routers. They serve as the outside connection of a LAN to a WAN, and they operate at the border of your network.

Routers are also used to divide internal networks into two or more subnetworks. Routers can also be connected internally to other routers, creating zones that operate independently. Routers establish communication by maintaining tables about destinations and local connections. A router contains information about the systems connected to it and where to send requests if the destination isn't known. Routers usually communicate routing and other information using one of three standard protocols: Routing Information Protocol (RIP), Border Gateway Protocol (BGP) or Open Shortest Path First (OSPF).

Routers are your first line of defence, and they must be configured to pass only traffic that is authorized by network administrators. The routes themselves can be configured as static or dynamic. If they are static, they can only be configured manually and stay that way until changed. If they are dynamic, they learn of other routers around them and use information about those routers to build their routing tables.

Routers are general-purpose devices that interconnect two or more heterogeneous networks. They are usually dedicated to special-purpose computers, with separate input and output network interfaces for each connected network. Because routers and gateways are the backbone of large computer networks like the internet, they have special features that give them the flexibility and the ability to cope with varying network addressing schemes and frame sizes through segmentation of big packets into smaller sizes that fit the new network components. Each router interface has its own Address Resolution Protocol (ARP) module, its own LAN address (network card address) and its own Internet Protocol (IP) address. The router, with the help of a routing table, has knowledge of routes a packet could take from its source to its destination. The routing table, like in the bridge and switch, grows dynamically. Upon receipt of a packet, the router removes the packet headers and trailers and analyzes the IP header by determining the source and destination addresses and data type and noting the arrival time. It also updates the router table with new addresses not already in the table. The IP header and arrival time information is entered in the routing table. Routers normally work at the Network layer of the OSI model.

4. Bridge

Bridges are used to connect two or more hosts or network segments together. The basic role of bridges in network architecture is storing and forwarding frames between the different segments that the bridge connects. They use hardware Media Access Control (MAC) addresses for transferring frames. By looking at the MAC address of the devices connected to each segment, bridges can forward the data or block it from crossing. Bridges can also be used to connect two physical LANs into a larger logical LAN.

Bridges work only at the Physical and Data Link layers of the OSI model. Bridges are used to divide larger networks into smaller sections by sitting between two physical network segments and managing the flow of data between the two.

Bridges are like hubs in many respects, including the fact that they connect LAN components with identical protocols. However, bridges filter incoming data packets, known as frames, for addresses before they are forwarded. As it filters the data packets, the bridge makes no modifications to the format or content of the incoming data. The bridge filters and forwards frames on the network with the help of a dynamic bridge table. The bridge table, which is initially empty, maintains the LAN addresses for each computer in the LAN and the addresses of each bridge interface that connects the LAN to other LANs. Bridges, like hubs, can be either simple or multiple port.

Bridges have mostly fallen out of favour in recent years and have been replaced by switches, which offer more functionality. In fact, switches are sometimes referred to as “multiport bridges” because of how they operate.

5. Gateway
Gateways normally work at the Transport and Session layers of the OSI model. At the Transport layer and above, there are numerous protocols and standards from different vendors; gateways are used to deal with them. Gateways provide translation between networking technologies such as Open System Interconnection (OSI) and Transmission Control Protocol/Internet Protocol (TCP/IP). Because of this, gateways connect two or more autonomous networks, each with its own routing algorithms, protocols, topology, domain name service, and network administration procedures and policies.

Gateways perform all the functions of routers and more. In fact, a router with added translation functionality is a gateway. The function that does the translation between different network technologies is called a protocol converter.

6. Modem
Modems (modulators-demodulators) are used to transmit digital signals over analog telephone lines. Thus, digital signals are converted by the modem into analog signals of different frequencies and transmitted to a modem at the receiving location. The receiving modem performs the reverse transformation and provides a digital output to a device connected to a modem, usually a computer. The digital data is usually transferred to or from the modem over a serial line through an industry standard interface, RS-232. Many telephone companies offer DSL services, and many cable operators use modems as end terminals for identification and recognition of home and personal users. Modems work on both the Physical and Data Link layers.

7. Repeater
A repeater is an electronic device that amplifies the signal it receives. You can think of repeater as a device which receives a signal and retransmits it at a higher level or higher power so that the signal can cover longer distances, more than 100 meters for standard LAN cables. Repeaters work on the Physical layer.

8. Access Point
While an access point (AP) can technically involve either a wired or wireless connection, it commonly means a wireless device. An AP works at the second OSI layer, the Data Link layer, and it can operate

either as a bridge connecting a standard wired network to wireless devices or as a router passing data transmissions from one access point to another.

Wireless access points (WAPs) consist of a transmitter and receiver (transceiver) device used to create a wireless LAN (WLAN). Access points typically are separate network devices with a built-in antenna, transmitter, and adapter. APs use the wireless infrastructure network mode to provide a connection point between WLANs and a wired Ethernet LAN. They also have several ports, giving you a way to expand the network to support additional clients. Depending on the size of the network, one or more APs might be required to provide full coverage. Additional APs are used to allow access to more wireless clients and to expand the range of the wireless network. Each AP is limited by its transmission range – the distance a client can be from an AP and still obtain a usable signal and data process speed. The actual distance depends on the wireless standard, the obstructions and environmental conditions between the client and the AP. Higher end APs have high-powered antennas, enabling them to extend how far the wireless signal can travel.

APs might also provide many ports that can be used to increase the network's size, firewall capabilities and Dynamic Host Configuration Protocol (DHCP) service. Therefore, we get APs that are a switch, DHCP server, router, and firewall.

c. Network IP.

IP address is the unique numerical address of a device in a computer network that uses Internet Protocol for communication. The IP address allow you to pinpoint a particular device from the billions of devices on the Internet. One computer needs the IP address of another computer to communicate with it.

An IP address consists of four numbers; each can contain one to three digits. These numbers are separated with a single dot (.). These four numbers can range from 0 to 255.

Types of IP addresses

The IP addresses can be classified into two. They are listed below.
1) Static IP addresses
2) Dynamic IP addresses

Static IP Addresses
Static IP addresses usually never change but they may be changed because of network administration. They serve as a permanent Internet address and provide a simple and reliable way for the communication.

From the static IP address of a system, we can get many details such as the continent, country, region, and city in which a computer is located, The Internet Service Provider (ISP) that serves that particular computer and non-technical information such as precise latitude and longitude of the country, and the locale of the computer. There are many websites providing IP address lookups. You can find out your IP addresses at http://whatismyip.org/.

Dynamic IP Addresses

Dynamic IP address are the second category. These are temporary IP addresses. These IP addresses are assigned to a computer when they get connected to the Internet each time. They are borrowed from a pool of IP addresses, shared over various computers. Since limited number of static IP addresses are available, ISPs usually reserve the portion of their assigned addresses for sharing among their subscribers in this way.

Static IP addresses are considered as less secure than dynamic IP addresses because they are easier to track.

IP Version 4

IP Version 4 (IPv4) was defined in 1981. It has not undergone much changes from that time. Unfortunately, there is a need of IP addresses more than IPv4 could supply.

IPv4 uses 32-bit IP address. So, the maximum number of IP address is 2^{32} or 4,294,967,296.

This is a little more than four billion IP addresses. An IPv4 address is typically formatted as four 8-bit fields. Each 8-bit field represents a byte of the IPv4 address. As we have seen earlier, each field will be separated with dots. This method of representing the byte of an IPv4 address is referred to as the dotted-decimal format. The bytes of the IPv4 are further classified into two parts. The network part and the host part.

Network Part

This part specifies the unique number assigned to your network. It also identifies the class of network assigned. The network part takes two bytes of the IPv4 address.

Host Part

This is the part of the IPv4 address that you can assign to each host. It uniquely identifies this machine on your network. For all hosts on your network, the network part of the IP address will be the same and host part will be changing.

IP address and classes

The IP hierarchy contains many classes of the IP addresses. Broadly, the IPv4 addressing system is divided into five classes of IP address. All the five classes are identified by the first octet of the IP address.

The classes of IPv4 addresses

The different classes of the IPv4 address are the following:

- Class A address
- Class B address
- Class C address
- Class D address
- Class E address

Class A Address

The first bit of the first octet is always set to zero. So that the first octet ranges from 1 – 127. The class A address only include IP starting from 1.x.x.x to 126.x.x.x. The IP range 127.x.x.x is reserved for loop back IP addresses. The default subnet mask for class A IP address is 255.0.0.0. This means it can have 126 networks (27-2) and 16777214 hosts (224-2).

Class B Address

Here the first two bits in the first two bits is set to zero. Class B IP Addresses range from 128.0.x.x to 191.255.x.x. The default subnet mask for Class B is 255.255.x.x. Class B has 16384 (214) Network addresses and 65534 (216-2) Host addresses.

Class C Address

The first octet of this class has its first 3 bits set to 110. Class C IP addresses range from 192.0.0.x to 223.255.255.x. The default subnet mask for Class C is 255.255.255.x. Class C gives 2097152 (221) Network addresses and 254 (28-2) Host addresses.

Class D Address

The first four bits of the first octet in class D IP address are set to 1110. Class D has IP address rage from 224.0.0.0 to 239.255.255.255. Class D is reserved for Multicasting. In multicasting data is not intended for a particular host, but multiple ones. That is why there is no need to extract host address from the class D IP addresses. The Class D does not have any subnet mask.

Class E Address

The class E IP addresses are reserved for experimental purpose only for R&D or study. IP addresses in the class E ranges from 240.0.0.0 to 255.255.255.254. This class too is not equipped with any subnet mask.

d. Basic network command and Network configuration commands.

1. Ping (Packet InterNet Groper)
   Ping is used to test the ability of one network host to communicate with another. Simply enter the Ping command, followed by the name or the IP address of the destination host. Assuming that there are no network problems or firewalls preventing the ping from completing, the remote host will respond to the ping with four packets. Receiving these packets confirms that a valid and functional network path exists between the two hosts.
   ping domain_name
   ping ip_address

2. NetStat
   If there are problems with network communications, then network statistics can sometimes help point toward the root cause of the problem. This command has several different functions, but the most useful of these is to display network summary information for the device. To see this type of summary information, just type *NetStat -e*

3. ARP
   The ARP command corresponds to the Address Resolution Protocol. Although it is easy to think of network communications in terms of IP addressing, packet delivery is ultimately dependent on the Media Access Control (MAC) address of the device's network adapter. This is where the Address Resolution Protocol comes into play. Its job is to map IP addresses to MAC addresses.
   Windows devices maintain an ARP cache, which contains the results of recent ARP queries. To see the contents of this cache, use the ARP -A command. If there is a problem communicating with a specific host, then append the remote host's IP address to the ARP -A command.
   ARP -A

4. Hostname
   Typing Hostname at the command prompt returns the local computer name.

5. Tracert
   Tracert works similarly to Ping. The major difference is that Tracert sends a series of ICMP echo requests. This allows the utility to display the routers through which packets are passing to be identified. When possible, Windows displays the duration and IP address or fully qualified domain name of each hop.

6. Ipconfig

The Ipconfig command will display basic IP address configuration information for the device. Type Ipconfig at the Windows command prompt, and it will display the IP address, subnet mask, and default gateway that the device is currently using.

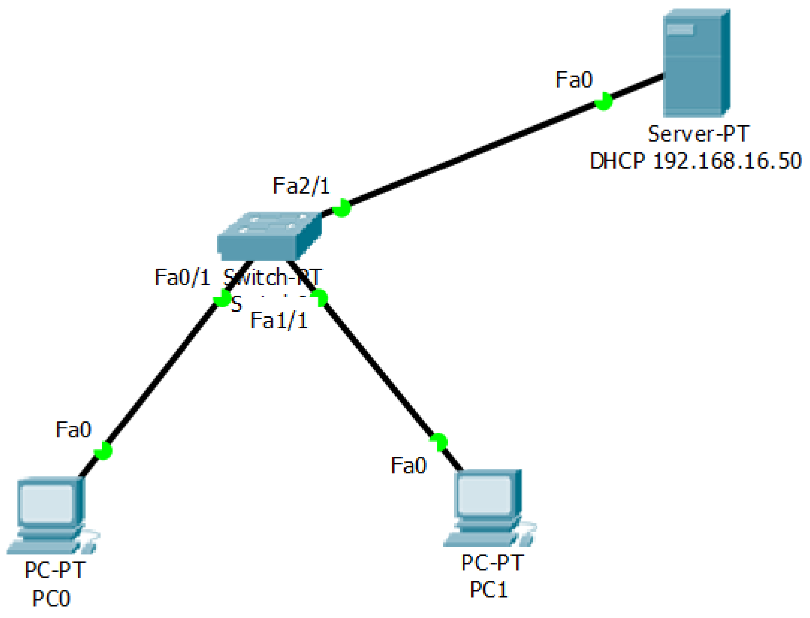
7. NSLookup
   NSLookup is a utility for diagnosing DNS name resolution problems. Just type the NSLookup command, and Windows will display the name and IP address of the device's default DNS server.

8. Route
   IP networks use routing tables to direct packets from one subnet to another. The Windows Route utility allows users to view the device's routing tables. To do so, simply type *Route Print*.
   The Route command allows users to make changes. Commands such as Route Add, Route Delete, and Route Change allow users to make routing table modifications on an as needed basis. The changes can be persistent or nonpersistent, depending on whether users use the -P switch.

9. PathPing
   PathPing tool is a utility that combines the best aspects of Tracert and Ping.
   Entering the PathPing command followed by a host name initiates what looks like a somewhat standard Tracert process. Once this process completes however, the tool takes 300 seconds (five minutes) to gather statistics, and then reports latency and packet loss statistics that are more detailed than those provided by Ping or Tracert.

10. NetDiag (Till Windows 8)
   The NetDiag command is designed to run a series of tests on the computer to help the technician figure out why the computer is experiencing networking problems.
   Entering the NetDiag command by itself will cause all the available tests to be run. In some cases, NetDiag can not only identify problems, but can also fix those problems. Obviously, NetDiag cannot automatically correct every problem that it finds, but appending the /Fix parameter to the command will tell NetDiag to attempt to fix the problem automatically.

2. Configure and implement DHCP service in a Local Area Network.



Topology

Aim: To Configure and use DHCP service to assign IP addresses in a Local Area Network.

Procedure:

1. Assign static IP address to DHCP Server.

   Click on the DHCP server and assign the static IP address and respective default gateway. Ex. 192.168.16.50 and 255.255.255.0

2. Enable DHCP service.

   Under the "services" tab of the DHCP server, select "DHCP", enable DHCP and provide the following and click on "Save":

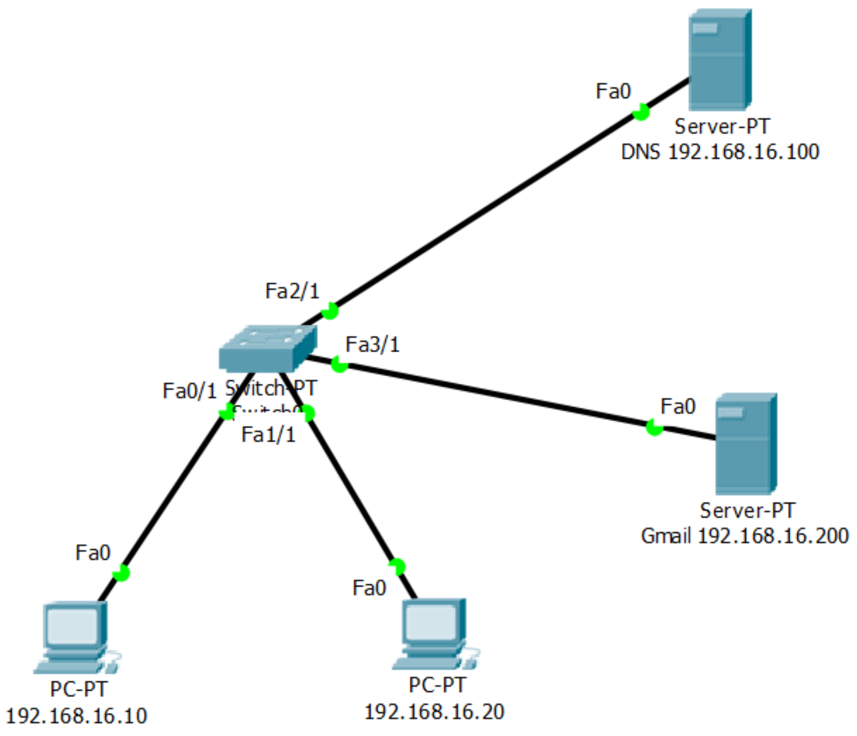
   a. Start IP and subnet mask (Ex. 192.168.16.100 and 255.255.255.0)
   b. Maximum number of hosts. (Ex. 100)

3. Assign IP addresses to hosts using DHCP.

   Click on the PC, select "IP address" under "Desktop" and select "DHCP" instead of DHCP. If DHCP is configured properly, first available IP address in the pool will be assigned.

   Repeat this for the other PC.

3. Configure and implement DNS service.



Topology

Aim: To configure and use DNS service.

Procedure:

Planned IP addresses for the experiment are:

PC 0:

IP - 192.168.16.10

Subnet – 255.255.255.0

DNS – 192.168.16.100

PC 1:

IP - 192.168.16.20

Subnet – 255.255.255.0

DNS – 192.168.16.100

Server 0 (DNS):

IP - 192.168.16.100

Subnet – 255.255.255.0

Server 1 (Gmail):

IP - 192.168.16.200

Subnet – 255.255.255.0

Assign IP addresses to all devices as planned (Sample plan mentioned above).

Configure DNS Service.

a. Click on the DNS server and select “DNS” under services tab.
b. Enable DNS service, provide name and address of the Gmail server. Ex. Name can be gmail.com and IP address is 192.168.16.200

Configure the index page of Gmail.

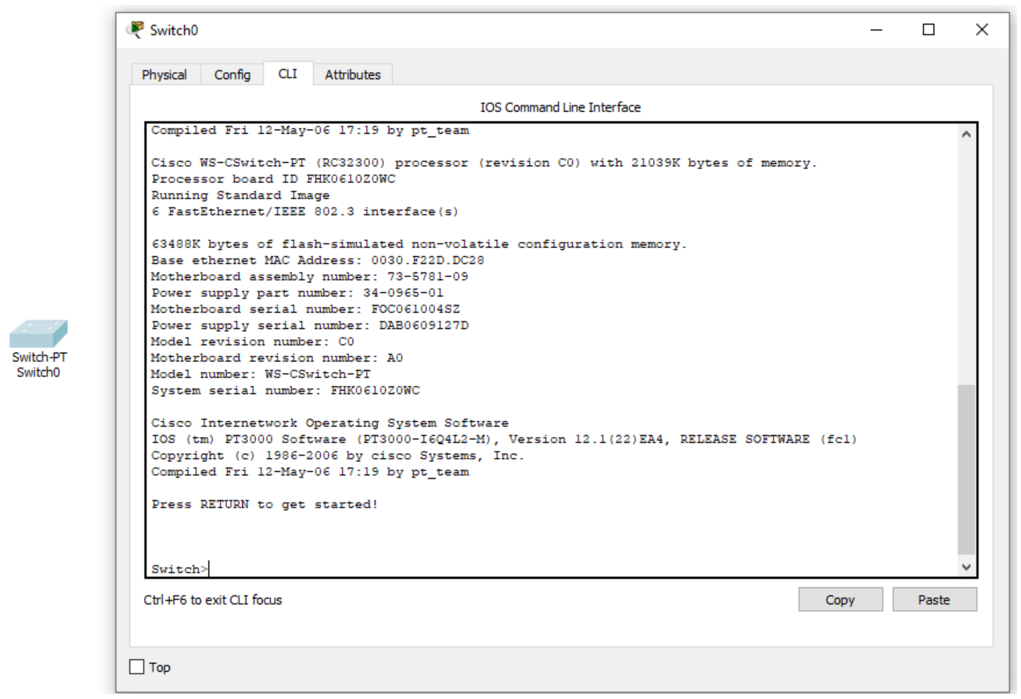
a. Click on the Gmail server and select “HTTP” under services tab.
b. Turn on the service and make minor modifications in the index.html page.

Check if DNS is working.

c. Click on the PC and select “Web Browser” under “Desktop” tab.
d. In the address bar enter the domain name (Ex. Gmail.com) and check if Gmail homepage is visible.

4. Performing an Initial Switch Configuration.

(Host name, Console password, vty password, Privileged EXEC mode password, Privileged EXEC mode secret, IP address on VLAN1 interface, Default gateway)



```
Compiled Fri 12-May-06 17:19 by pt_team

Cisco WS-CSwitch-PT (RC32300) processor (revision C0) with 21039K bytes of memory.
Processor board ID FHK0610Z0WC
Running Standard Image
6 FastEthernet/IEEE 802.3 interface(s)

63488K bytes of flash-simulated non-volatile configuration memory.
Base ethernet MAC Address: 0030.F22D.DC28
Motherboard assembly number: 73-5781-09
Power supply part number: 34-0965-01
Motherboard serial number: FOC061004SZ
Power supply serial number: DAB0609127D
Model revision number: C0
Motherboard revision number: A0
Model number: WS-CSwitch-PT
System serial number: FHK0610Z0WC

Cisco Internetwork Operating System Software
IOS (tm) PT3000 Software (PT3000-I6Q4L2-M), Version 12.1(22)EA4, RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2006 by cisco Systems, Inc.
Compiled Fri 12-May-06 17:19 by pt_team

Press RETURN to get started!


Switch>
```

Access Switch in CLI mode

Aim: To perform initial switch configuration of a switch using CLI mode.

Procedure: The operations to be performed are:

1. Configure the switch host name.
2. Set Privileged EXEC mode password and secret.
3. Set console password.
4. Set vty password.
5. Create a VLAN interface and assign an IP address.
6. Assign the default gateway.
7. Check the setting in the running configuration.

Click on the Router and select “CLI” tab.

1. Configure the switch host name.

   To configure hostname, user needs to be in the Global configuration mode.

```
Switch> enable
Switch# configure terminal
Switch(config)# hostname NIESwitch
```

2. Set Privileged EXEC mode password and secret.

   a. Setting Password

   User needs to be in the global configuration mode.

```
NIESwitch(config)# enable password cisco
```

   b. Setting Secret

   User needs to be in the global configuration mode.

```
NIESwitch(config)# enable secret cisco123
```

3. Set Console Password

   User needs to be in the global configuration mode.

```
NIESwitch(config)# line console 0
NIESwitch (config-line)# password cisco123
NIESwitch (config-line)# login
NIESwitch (config-line)# exit
```

   Connect a PC to the Switch using Console cable and access the switch using PC’s terminal.

4. Set vty password

   User needs to be in the global configuration mode.

```
NIESwitch(config)# line vty 0 15
NIESwitch (config-line)# password cisco123
NIESwitch (config-line)# login
NIESwitch (config-line)# exit
NIESwitch (config)#
```

5. Create a VLAN interface and assign an IP address.

   User needs to be in the global configuration mode.

```
NIESwitch (config)# interface vlan 1
NIESwitch (config-if)# ip address 192.168.16.10 255.255.255.0
NIESwitch (config-if)# no shutdown
NIESwitch (config-if)# exit
```

6. Assign the default gateway.

From global configuration mode, assign the default gateway to 192.168.16.1.

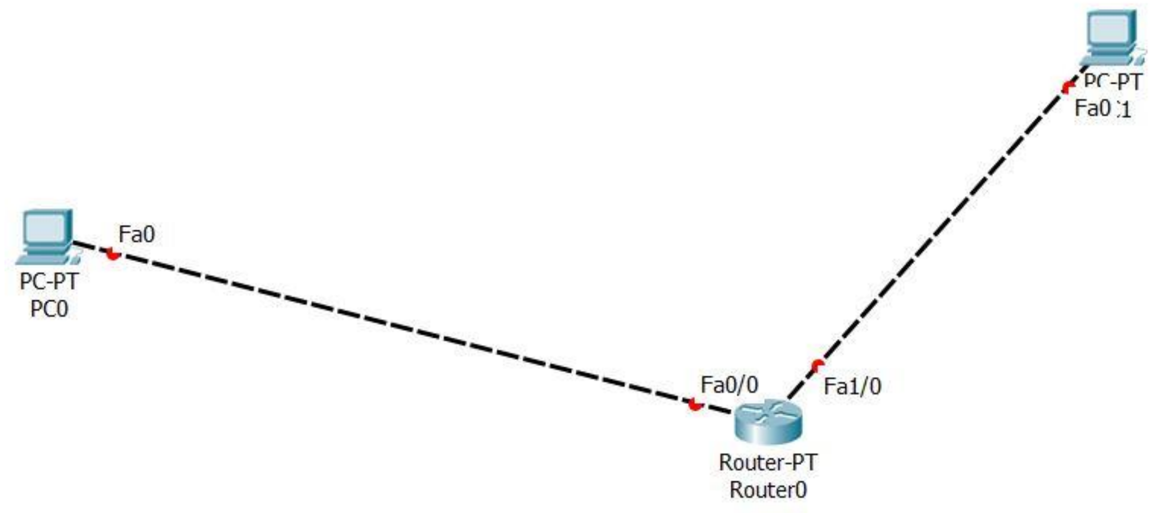
```
NIESwitch(config)#ip default-gateway 192.168.16.1
```

7. Check the setting in the running configuration.

To display the current running configuration, enter the show running-config command.

5. Performing an Initial Router Configuration

(Configure the router host name, Configure passwords, Configure banner messages, Verify the router configuration)



Topology

Aim: To perform initial router configuration of a router using CLI mode.

Procedure: The operations to be performed are:

1. Configure the router host name.
2. Configure passwords.
3. Configure password encryption.
4. Configure banner messages.
5. Turn off domain server lookup.
6. Save the running configuration to the startup configuration.

Click on the Router and select "CLI" tab.

1. Configure the router host name.

   To configure hostname, user needs to be in the Global configuration mode.

```
Router> enable
Router# configure terminal
Router(config)# hostname NIERouter
```

2. Configure passwords.

   a. Setting Password

User needs to be in the global configuration mode.

```
NIERouter(config)# enable password cisco
```

b. Setting Secret

User needs to be in the global configuration mode.

```
NIERouter(config)# enable secret cisco123
```

c. Setting Console Password

User needs to be in the global configuration mode.

```
NIERouter(config)# line console 0
NIERouter(config-line)# password cisco123
NIERouter(config-line)# login
NIERouter(config-line)# exit
NIERouter(config)#
```

d. Setting vty password

User needs to be in the global configuration mode.

```
NIERouter(config)# line vty 0 4
NIERouter(config-line)# password cisco123
NIERouter(config-line)# login
NIERouter(config-line)# exit
NIERouter(config)#
```

3. Configure password encryption.

   User needs to be in the global configuration mode.

   ```
   NIERouter(config)# service password-encryption
   ```

   Use the show running-config command again to verify that the passwords are encrypted.

4. Configure banner messages.

   User needs to be in the global configuration mode.

   ```
   NIERouter(config)#banner motd $Authorized Access Only!$
   ```

   Log out of the router by typing the exit command twice. The banner displays before the prompt for a password.

5. Turn off domain server lookup.
   Make a spelling mistake in any of the commands. The router pauses while trying to locate an IP address for the mistyped word entered.

   ```
   NIERouter>emable
   Translating "emable"...domain server (255.255.255.255)
   ```

   To stop this, use the following command in the Global configuration mode:

   ```
   NIERouter(config)#no ip domain-lookup
   ```

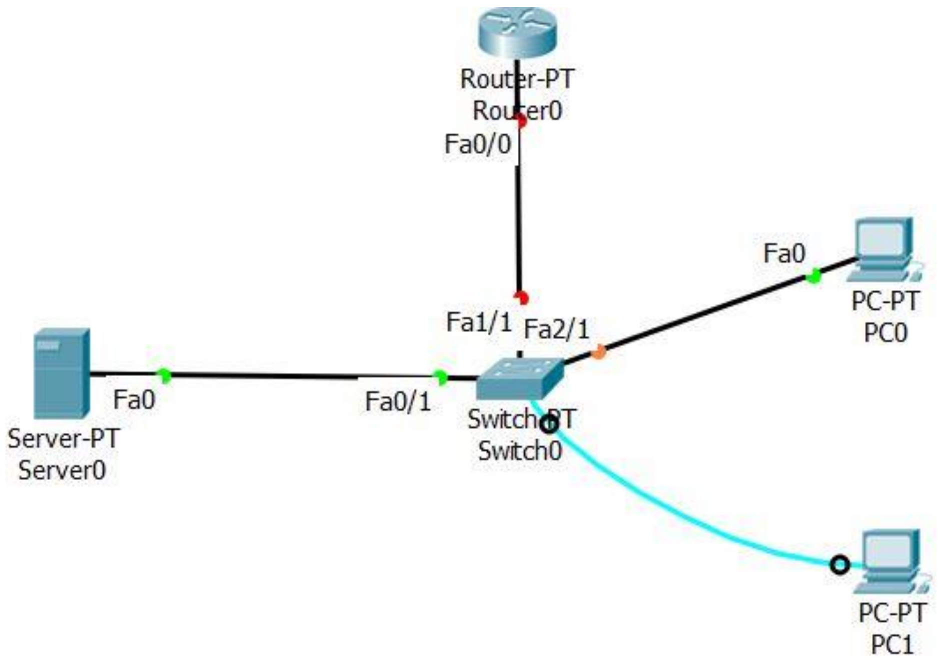
6. Save the running configuration to the startup configuration.

   ```
   NIERouter(config)#end
   NIERouter# copy run start
   ```

   - The running configuration is stored in RAM; the startup configuration is stored in NVRAM.
   - To display the current running configuration, enter the show running-config command.
   - Enter the copy running-config startup-config command to save the current running configuration to the startup configuration file in NVRAM.

6. Configuring and Troubleshooting a Switched Network

(Establish console connection to the switch, Configure the host name and VLAN1, Use the help feature to configure the clock, Configure passwords and console/Telnet access, Configure login banners, Configure the router, Configure port security, Secure unused ports)



Topology

Aim: To perform initial router configuration of a router using CLI mode.

Procedure: The operations to be performed are:

1. Establish console connection to the switch.
2. Configure the host name and VLAN1.
3. Use the help feature to configure the clock.
4. Configure passwords and console/Telnet access.
5. Configure login banner.
6. Configure port security.
7. Secure unused ports.

1. Establish console connection to the switch.
   Instead of direct access to switch configuration via CLI mode, we will configure the switch using a console session through PC 1.
   a. Connect a console cable from PC1 to Switch.

b. From PC1, open a terminal window and use the default terminal configuration. You should now have access to the CLI for the Switch.

2. Configure the host name and VLAN1.

Configure the switch host name as S1

```
Switch> enable
Switch# configure terminal
Switch(config)# hostname S1
S1(config)#
```

Configure VLAN

```
S1(config)# interface vlan 1
S1(config-if)# ip address 172.17.99.11 255.255.255.0
S1(config-if)# no shutdown
```

3. Use the help feature to configure the clock.
   a. Configure the clock to the current time. At the privileged EXEC prompt, enter clock ?
      ```
      S1# clock ?
      ```
   b. Use Help to discover the steps required to set the current time.
      ```
      S1# clock set [hh:mm:ss] [month] [day] [year]
      S1# clock set 14:30:00 May 28 2021
      ```
   c. Use the show clock command to verify that the clock is now set to the current time. Packet Tracer may not correctly simulate the time you entered.
      ```
      S1# show clock
      ```

4. Configure passwords.

a. Setting Password

User needs to be in the global configuration mode.

```
S1(config)# enable password cisco
```

b. Setting Secret

User needs to be in the global configuration mode.

```
S1(config)# enable secret cisco123
```

c. Setting Console Password

User needs to be in the global configuration mode.

```
S1(config)# line console 0
S1(config-line)# password cisco123
S1(config-line)# login
S1(config-line)# exit
S1(config)#
```

d. Setting vty password

User needs to be in the global configuration mode.

```
S1(config)# line vty 0 4
S1(config-line)# password cisco123
S1(config-line)# login
S1(config-line)# exit
S1(config)#
```

5. Configure banner messages.

   User needs to be in the global configuration mode.

   ```
   S1(config)#banner motd $Authorized Access Only!$
   ```

   Log out of the switch by typing the exit command twice. The banner is displayed before the prompt for a password.

6. Configure port security.

   Select the interface and configure the port as an access port.

   ```
   S1(config)# int fa0/1
   S1(config-if)# switchport mode access
   S1(config-if)# switchport port-security
   ```

   Define which MAC addresses can send frames through this interface.

   ```
   S1(config-if)# switchport port-security mac-address sticky
   ```

   To view the security status

   ```
   show port-security interface fa0/1
   ```

7. Secure unused ports.

In the privilege mode execute the following command

```
S1# show ip interface brief
```

In the global configuration mode

```
S1(config)# interface range fa0/3 – 24 , g0/1 – 2
S1 (config-if-range)#shutdown
```

In the privilege mode execute the following command once again

```
S1# show ip interface brief
```

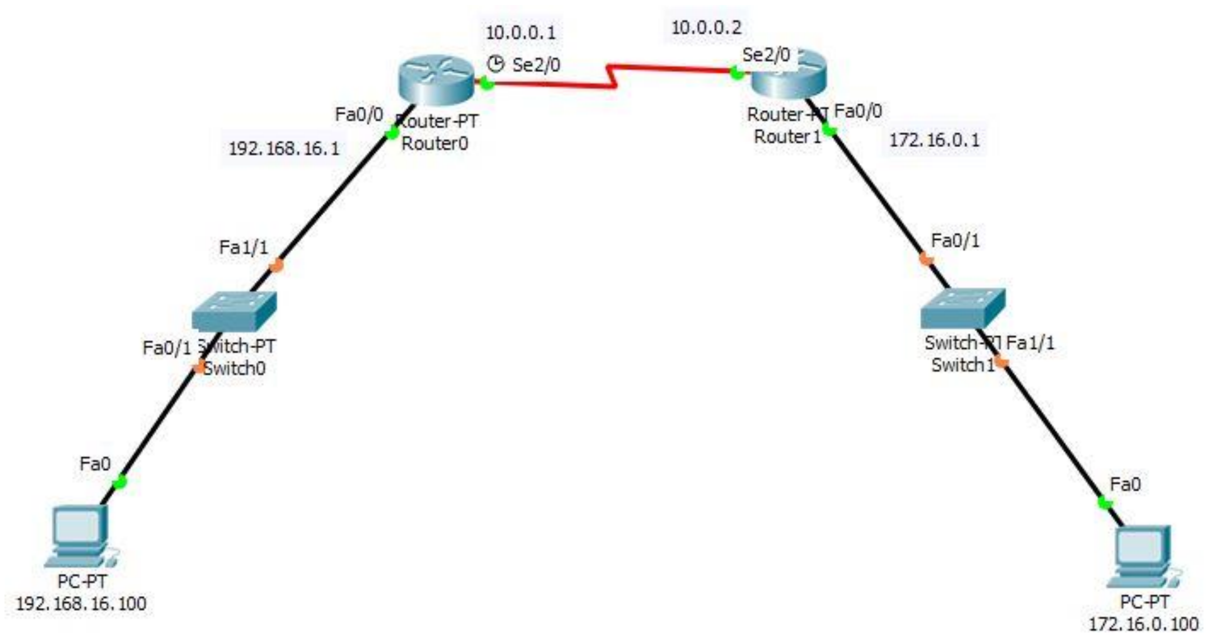
7. Implement the following types of routing:

   a. Static Routing.

   b. Dynamic Routing.

   c. Default Routes.

Static Routing.



Topology

Aim: To implement static routing.

Procedure: Build the topology as shown above and perform the following steps:

1. Assign IP addresses to both the PCs.

   PC0

IP address:	192.168.16.100
Subnet mask:	255.255.255.0
Default gateway:	192.168.16.1

   PC1

IP address:	172.16.0.100
Subnet mask:	255.255.0.0
Default gateway:	172.16.0.1

2. Assign IP addresses to interfaces fa0/0 and se2/0 in both the routers.

   Router 0 – Configurations

```
Router>enable
Router# configure terminal
Router(config)# int fa0/0
Router(config-if)# ip address 192.168.16.1 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)#exit

Router(config)# int se2/0
Router(config-if)# ip address 10.0.0.1 255.0.0.0
Router(config-if)# no shutdown
```

Router 1 – Configurations

```
Router>enable
Router# configure terminal
Router(config)# int fa0/0
Router(config-if)# ip address 172.16.0.1 255.255.0.0
Router(config-if)# no shutdown
Router(config-if)#exit
Router(config)# int se2/0
Router(config-if)# ip address 10.0.0.2 255.0.0.0
Router(config-if)# no shutdown
```

3. Add static route entry in the routing table.

   Command:

   ```
   ip route dest_network_addr subnet_mask next_hop_addr
   ```

   In Router 0

   ```
   Router(config)# ip route 172.16.0.0 255.255.0.0 10.0.0.2
   ```

   In Router 1

   ```
   Router(config)# ip route 192.168.16.0 255.255.255.0 10.0.0.1
   ```

4. Check node reachability using Ping.

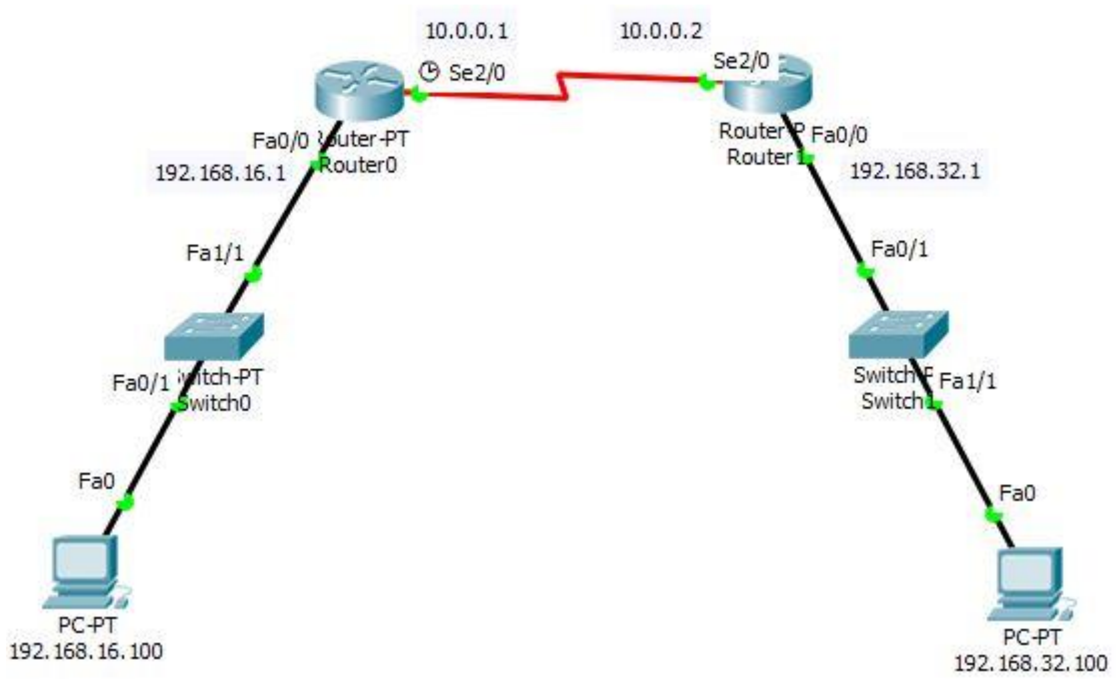
   From command prompt of PC0

   ```
   ping 172.16.0.100
   ```

   From command prompt of PC1

   ```
   ping 192.168.16.100
   ```

Dynamic Routing



Topology

Aim: To implement dynamic routing.

Procedure: Build the topology as shown above and perform the following steps:

1. Assign IP addresses to both the PCs.

   PC0

IP address:	192.168.16.100
Subnet mask:	255.255.255.0
Default gateway:	192.168.16.1

   PC1

IP address:	192.168.32.100
Subnet mask:	255.255.255.0
Default gateway:	192.168.32.1

2. Assign IP addresses to interfaces fa0/0 and se2/0 in both the routers.

   Router 0 – Configurations

   ```
   Router>enable
   Router# configure terminal
   Router(config)# int fa0/0
   ```

```
Router(config-if)# ip address 192.168.16.1 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)#exit

Router(config)# int se2/0
Router(config-if)# ip address 10.0.0.1 255.0.0.0
Router(config-if)# no shutdown
```

Router 1 – Configurations

```
Router>enable
Router# configure terminal
Router(config)# int fa0/0
Router(config-if)# ip address 192.168.32.1 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)#exit

Router(config)# int se2/0
Router(config-if)# ip address 10.0.0.2 255.0.0.0
Router(config-if)# no shutdown
```

3. Enable RIP and provide network addresses.

   In Router 0

   ```
   Router(config)# router rip
   Router(config-router)# network 192.168.16.0
   Router(config-router)# network 10.0.0.0
   ```

   In Router 1

   ```
   Router(config)# router rip
   Router(config-router)# network 192.168.32.0
   Router(config-router)# network 10.0.0.0
   ```

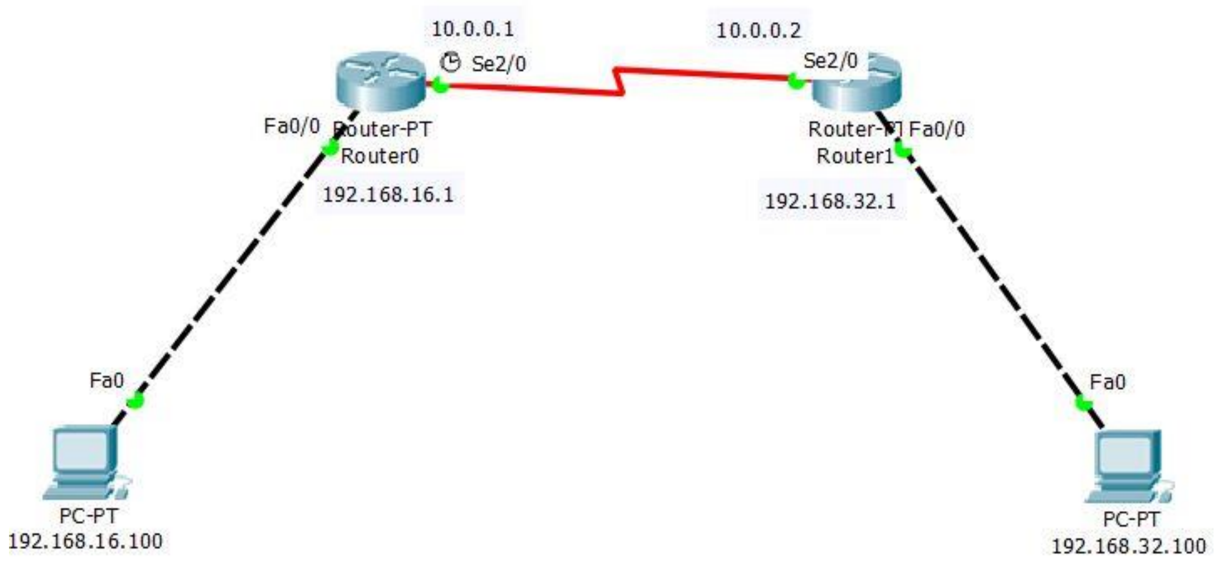
4. Check node reachability using Ping.

   From command prompt of PC0

   ```
   ping 192.168.32.100
   ```

```
From command prompt of PC1
ping 192.168.16.100
```

Default Routing



Topology

Aim: To use default routing.

Procedure: Build the topology as shown above and perform the following steps:

1. Assign IP addresses to both the PCs.

   PC0

IP address:	192.168.16.100
Subnet mask:	255.255.255.0
Default gateway:	192.168.16.1

   PC1

IP address:	192.168.32.100
Subnet mask:	255.255.255.0
Default gateway:	192.168.32.1

2. Assign IP addresses to interfaces fa0/0 and se2/0 in both the routers.

   Router 0 – Configurations

   ```
   Router>enable
   Router# configure terminal
   Router(config)# int fa0/0
   ```

```
Router(config-if)# ip address 192.168.16.1 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)#exit

Router(config)# int se2/0
Router(config-if)# ip address 10.0.0.1 255.0.0.0
Router(config-if)# no shutdown

Router 1 – Configurations
Router>enable
Router# configure terminal
Router(config)# int fa0/0
Router(config-if)# ip address 192.168.32.1 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)#exit

Router(config)# int se2/0
Router(config-if)# ip address 10.0.0.2 255.0.0.0
Router(config-if)# no shutdown
```

3. Add default route entry in the routing table.

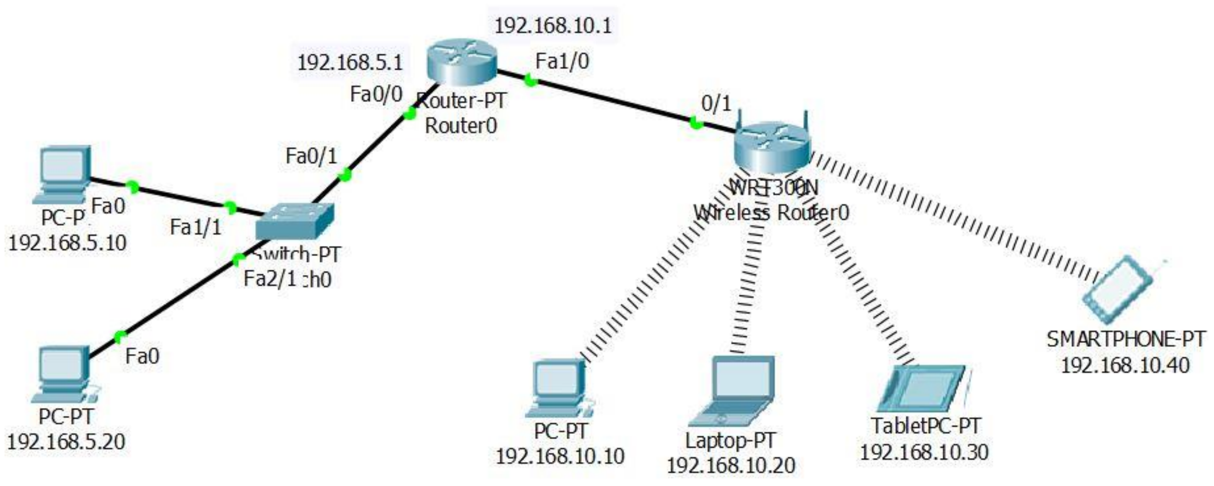
   Command: ip route dest_net_add sub_mask next_hop_add

   ```
   In Router 0
   Router(config)# ip route 0.0.0.0 0.0.0.0 10.0.0.2
   In Router 1
   Router(config)# ip route 0.0.0.0 0.0.0.0 10.0.0.1
   ```

   Run “show ip route” in privilege exec mode to view routing table.

4. Check node reachability using Ping.

8. Configuring WEP on a Wireless Router.



Topology

Aim: To enable WEP on Cisco WRT300N Wireless router and check reachability of the devices in the network.

Procedure: The operations to be performed are:

1. Build the topology as shown above.
2. Assign IP addresses to the Router 0 interfaces (Fa0/0 and Fa1/0).
3. Assign IP addresses, Subnet mask and default gateway to the PCs on the 192.168.5.X network.
4. Configure the wireless router Internet address, LAN address and enable WEP.
5. Add wireless modules in the devices connected to the wireless devices (Laptop and PC).
6. Enable wireless connection by providing SSID and WEP.
7. Assign static IPs to devices connected to wireless router.
8. Check node reachability using PING command.

Step 1: Select all the required devices and connect them using Copper Straight Through cable. The devices required are:

a. Router – 1
b. Switch – 1
c. Wireless router (WRT300N) – 1
d. PCs – At least 2
e. Laptop – At least 1
f. Tablet PC – At least 1

g. Smartphone – At least 1

Step 2: Click on the "Router 0" and enable Fa0/0 and Fa1/0 interfaces and assign IP addresses of 2 different networks to these interfaces, respectively.

Fa0/0 – 192.168.5.1

Fa1/0 – 192.168.10.1

Step 3: Assign IP addresses, Subnet mask and default gateway to the PCs on the 192.168.5.X network.

PC0

IP address	- 192.168.5.10
Subnet mask	- 255.255.255.0
Default gateway	- 192.168.5.1

PC1

IP address	- 192.168.5.20
Subnet mask	- 255.255.255.0
Default gateway	- 192.168.5.1

Step 4: Configure the wireless router Internet address, LAN address and enable WEP.

- Click on the Wireless router, select "config" tab and select "Internet" option. Enter the default gateway as 192.168.10.1. Leave rest of the fields empty.
- Under "config" tab, select "LAN" and enter the IP and subnet as 192.168.10.2 and 255.255.255.0, respectively.
- Under "config" tab, select "Wireless" and change the SSID, select "Authentication" as WEP, and enter a WEP key.

Step 5: Add wireless modules in the devices connected to the wireless devices (Laptop and PC).

- Click on the PC and click on the “Physical” tab. Turn off the PC, remove existing module, add new wireless module under “WMP300N” and turn on the PC.
- Repeat the above step for the laptop device.

Step 6: Enable wireless connection by providing SSID and WEP.

- Click on the wireless device and “wireless” option under the “config” tab.
- Provide the SSID (the one provided in the wireless router), select WEP and enter WEP key.
- Repeat this on all the wireless devices. This will enable wireless connection between the wireless router and the wireless devices.

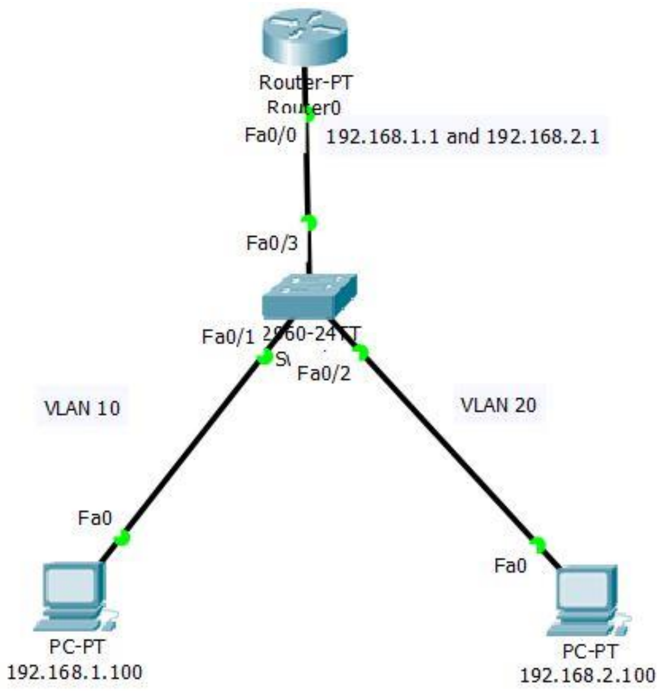
Step 7: Assign static Ips to devices connected to wireless router.

- The wireless router assigns dynamic IP addresses using DHCP, you can change it and assign static IP addresses which belong to the network 192.168.10.X
- Sample IP address assignment:
  - IP addresses: 192.168.10.10
  - Subnet mask: 255.255.255.0
  - Default-gateway: 192.168.10.1
- Repeat this on all the devices.

Step 8: Check node reachability using PING command.

- Click on any device, select “Command Prompt” under “Desktop” tab. Enter the ping command. Ex. ping 192.168.5.10

9. Demonstrating Inter-VLAN Routing in Switch.



Topology

Aim: To implement inter-VLAN routing.

Procedure: Build the topology as shown above and perform the following steps:

Step 1: Configure the PCs with IP address and Default Gateway.

PC0

IP addresses: 192.168.1.100

Subnet mask: 255.255.255.0

Default-gateway: 192.168.1.1

PC1

IP addresses: 192.168.2.100

Subnet mask: 255.255.255.0

Default-gateway: 192.168.2.1

Step 2: Configure switches

a. Create VLAN and provide mode of switchport access.

```
Switch>enable
Switch# configure terminal
Switch(config)# VLAN 10
Switch(config-vlan)# exit
Switch(config)# VLAN 20
Switch(config-vlan)# exit
Switch(config)# exit
Switch# show VLAN
```

b. Provide mode of switchport access and assign it to VLAN.

```
Switch(config)# int fa0/1
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config)# exit

Switch(config)# int fa0/2
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 20
Switch(config)# exit

Switch(config)# int fa0/3
Switch(config-if)# switchport mode trunk
Switch(config)# exit
```

Now ping from

1. PC0 to PC1, Observe the results.
2. PC1 to PC0, Observe the results.

Ping should be unsuccessful.

Step 3: Configure Router

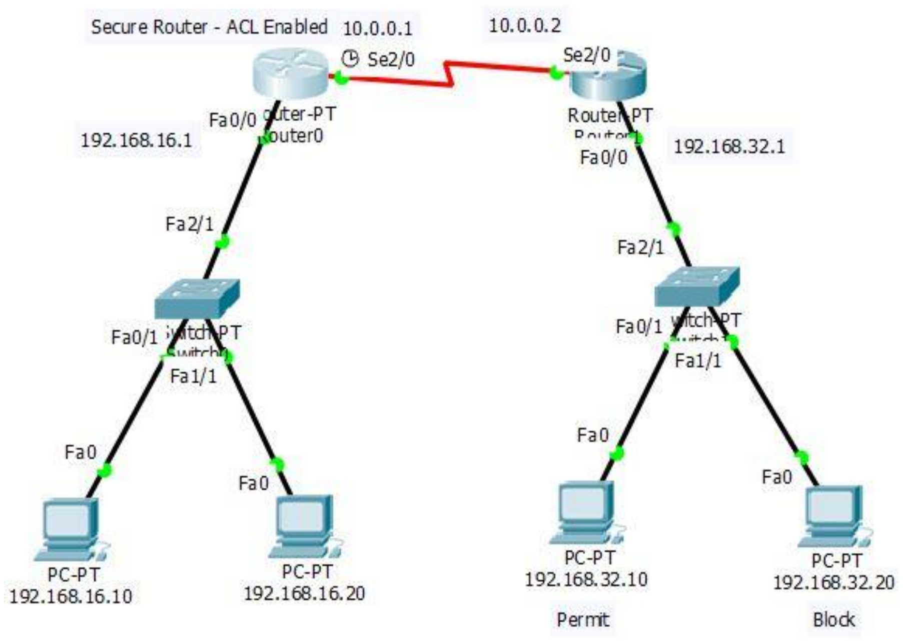
```
Router> enable
Router# configure terminal
Router(config)# int fa0/0.1
Router(config-if)# encapsulation dot1Q 10
Router(config-if)# ip address 192.168.1.1 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# int fa0/0.2
Router(config-if)# encapsulation dot1Q 20
Router(config-if)# ip address 192.168.2.1 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)# exit
```

Now ping from

1. PC0 to PC1, Observe the results.
2. PC1 to PC0, Observe the results.

Ping should be successful.

10. Defining and using Access Control Lists.



Topology

Note: Before defining ACL, hosts from one network can communicate with hosts from another network. Use RIP or Static Routes to enable communication.

In PC-192.168.16.10, try ping 192.168.32.10, this should be successful.

In PC-192.168.32.20, try ping 192.168.16.20, this should be successful.

Step 1: Defining ACL in the Router belonging to 192.168.16.0 network. Define Access control list (Deny 192.168.32.20 and permit all other hosts)

```
Router> enable

Router# configure terminal

Router(config) # access-list 1 deny host 192.168.32.20

Router(config) # access-list 1 permit any
```

Step 2: Assign this to the particular interface of the router and indicate if its inbound or outbound traffic with respect to that interface.

```
Router(config) # int fa0/0

Router(config-if) # ip access-group 1 out

Router(config-if) # exit
```

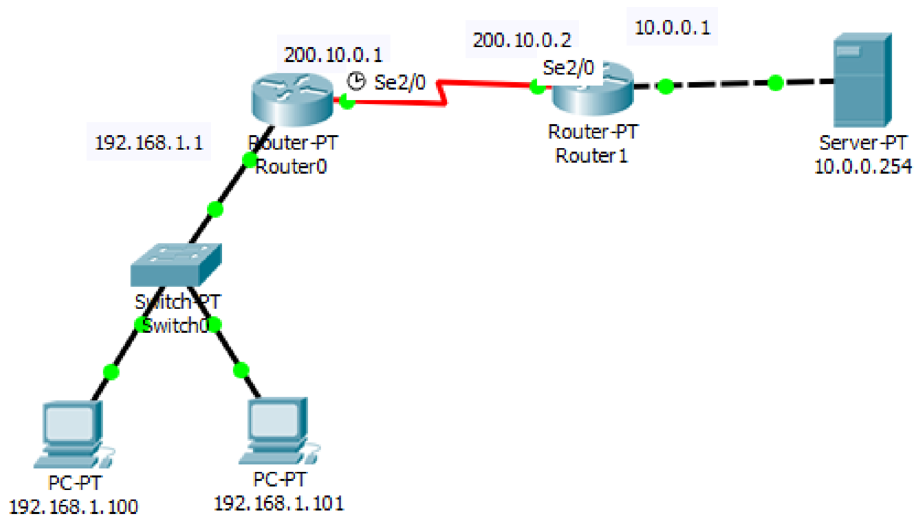
Step 3: Verify all these rules and configurations.

```
Router(config) # do sh runn
```

Step 4: Testing ACL

a) From PC-192.168.32.10, try to communicate with any PC of 192.168.16.0 network – Ex. ping 192.168.16.10
   This should be successful.
b) From PC-192.168.32.20, try to communicate with any PC of 192.168.16.0 network – Ex. ping 192.168.16.10
   This should be unsuccessful.

11. Using Network Address Translation (NAT).



Topology

Note: Before you start, HTTP service should be enabled on server and should be accessible from PC Web Browser.

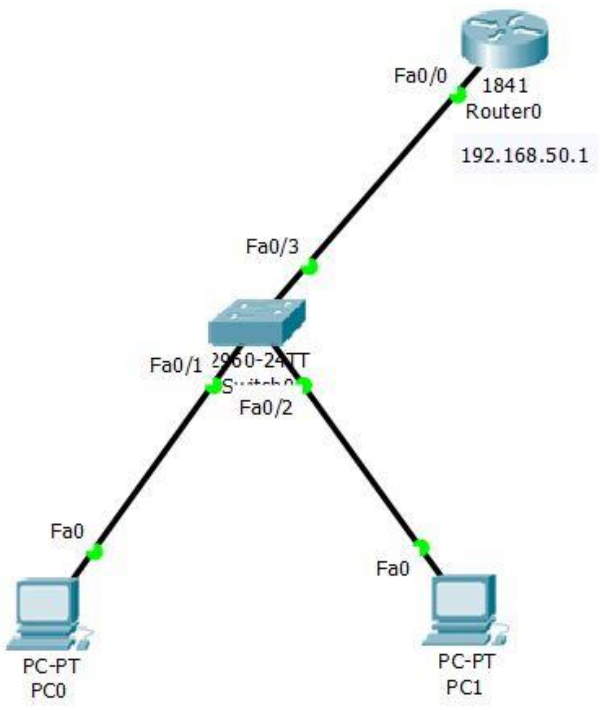
http://10.0.0.254

```
Router > enable

Router # configure terminal

Router (config) # ip nat inside source static 10.0.0.254 200.10.0.2

Router (config) # int s2/0

Router (config-if) # ip nat outside

Router (config-if) # exit

Router (config) # int fa0/0

Router (config-if) # ip nat inside

Router (config-if) # exit

Router # show run
```

(Observe the contents of running configuration file. You can optionally copy this to startup configuration file)

Note: Now test HTTP using global IP address, http://200.10.0.2

12. Configuring a Cisco Router as a DHCP Server.



Topology

Aim: To configure a Cisco 1841 router as a DHCP server.

Procedure: Build the topology as shown above and perform the following steps:

Step 1: Assigning IP address to the Router Interface.

```
Router> enable
Router# configure terminal
Router(config)# int fa 0/0
Router(config-if)# ip address 192.168.50.1 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)# exit
```

Step 2: Defining DHCP service pool and configuration details.

```
Router(config)# service dhcp
Router(config)# ip dhcp pool pool1
Router(dhcp-config)# network 192.168.50.0 255.255.255.0
Router(dhcp-config)# default-router 192.168.50.1
Router(dhcp-config)# dns-server 192.168.50.1
Router(dhcp-config)# exit
```

Step 3: To exclude range of IP addresses run the following command.

```
Router(config)# ip dhcp excluded-address 192.168.50.1 192.168.50.10
```

Step 4: Open IP configuration page of PCs and select DHCP.

Part B

1. **Write a program to implement the following Error Detection Techniques**
    a. **Single Parity Check**
    b. **Cyclic Redundancy Check (CRC)**
    c. **Checksum**

Single Parity Check

Sender side:

```
#include<stdio.h>
#include<string.h>
int main()
{
	char s[10];
	int c = 0,key,i;
	printf("Enter the message to be sent\n");
	gets(s);
	for(int i = 0 ; i < strlen(s);i++)
	{
		if(s[i] == '1')
		{
			c++;
		}
	}
	printf("Enter 1 for odd parity and 2 for even parity: ");
	scanf("%d",&key);
	if(key == 1)
	{
		if(c % 2 == 0)
			strcat(s, "1");
		else
			strcat(s, "0");
	}
	else if(key == 2)
	{
		if(c%2 != 0)
			strcat(s, "1");
		else
			strcat(s, "0");
	}
	printf("%s\n",s);
	return 0;
}
```

Receiver side :

```
#include<stdio.h>
#include <string.h>
int main(){
	char str[10];
	int i, key,c=0;
	printf("enter the message received :");
	gets(str);
	for(int i = 0 ; i < strlen(str);i++)
	{
		if(str[i] == '1')
			c++;
	}
	printf("Enter 1 for odd parity checking and 2 for even parity
:");
	scanf("%d",&key);
	if(key == 1)
	{
		if(c % 2!=0)
			printf("Correct");
		else
			printf("Incorrect");
	}
	else if(key == 2)
	{
		if(c % 2 == 0)
			printf("Correct");
		else
			printf("Incorrect");
	}
	return 0;
}
```

Cyclic Redundancy Check (CRC)

```
  #include<stdio.h>
  main()
  {
   int da[20],di[20],te[20],tem[20],l;
   int i,j,m,n,data,div,t,k,e;
   clrscr();
   printf("\nEnter the total bit of data and divisor");
   scanf("%d %d",&data,&div);
   m=data+div-1;
   printf("\nEnter the data:");
   for(i=0;i<data;i++)
```

```
    {
          scanf("%d",&da[i]);
          te[i]=da[i];
    }
for(i=data;i<m;i++)
    {
          te[i]=0;
    }
printf("\nEnter the divisor");
for(i=0;i<div;i++)
    {
          scanf("%d",&di[i]);
    }
l=div;t=0;
k=0;
for(i=0;i<data;i++)
    {
      e=0;t=0;
      for(j=1;j<div;j++)
          {
            if(((da[j]==1)&&(di[j]==1))||((da[j]==0)&&(di[j]==0)))
                {
                  tem[j-1]=0;
                  if(e!=1)
                      {
                        k=k+1;
                        t=t+1;
                        i=i+1;
                      }
                }
            else
                {
                  tem[j-1]=1;
                  e=1;
                }
          }
      j=0;
      for(e=t;e<div-1;e++)
          {
            da[j]=tem[e];
            j++;
          }
      for(j=j;j<div;j++)
          {
            if(l>=data+1)
```

```
                {
                  da[j]=0;
                }
             else
                {
                  da[j]=te[l];
                  l=l+1;
                }
            }
        }
    printf("\n The CRC BITS are\t ");
    for(i=0;i<div-1;i++)
       {
         printf(" %d",tem[i]);
       }
  }
```

Checksum

```
#include<stdio.h>
#include<math.h>
int sender(int arr[10],int n)
{
    int checksum,sum=0,i;
    printf("\n****SENDER SIDE****\n");
    for(i=0;i<n;i++)
    sum+=arr[i];
    printf("SUM IS: %d",sum);
    checksum=~sum;    //1's complement of sum
    printf("\nCHECKSUM IS:%d",checksum);
    return checksum;
}
void receiver(int arr[10],int n,int sch)
{
    int checksum,sum=0,i;
    printf("\n\n****RECEIVER SIDE****\n");
    for(i=0;i<n;i++)
        sum+=arr[i];
    printf("SUM IS:%d",sum);
    sum=sum+sch;
    checksum=~sum;    //1's complement of sum
    printf("\nCHECKSUM IS:%d",checksum);
}
void main()
{
    int n,sch,rch;
    printf("\nENTER SIZE OF THE STRING:");
    scanf("%d",&n);
    int arr[n];
    printf("ENTER THE ELEMENTS OF THE ARRAY TO CALCULATE
CHECKSUM:\n");
    for(int i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    sch=sender(arr,n);
    receiver(arr,n,sch);
}
```

2. Write a program to implement Caesar substitution cipher.

Caesar Cipher Encryption

```c
#include<stdio.h>
#include<ctype.h>
int main()
{
  char text[500], ch;
  int key;
  // taking user input
  printf("Enter a message to encrypt: ");
  scanf("%s", text);
  printf("Enter the key: ");
  scanf("%d", & key);
  // visiting character by character
  for (int i = 0; text[i] != '\0'; ++i) {
    ch = text[i];
    // check for valid character
    if (isalnum(ch)) {
      // lower case characters
      if (islower(ch)) {
        ch = (ch - 'a' + key) % 26 + 'a';
      }
      // uppercase characters
      if (isupper(ch)) {
        ch = (ch - 'A' + key) % 26 + 'A';
      }
      // numbers
      if (isdigit(ch)) {
        ch = (ch - '0' + key) % 10 + '0';
      }
    }
    // invalid character
    else {
      printf("Invalid Message");
    }
    // adding encoded answer
    text[i] = ch;
  }
  printf("Encrypted message: %s", text);
  return 0;
}
```

Caesar Cipher Decryption

```c
#include<stdio.h>
#include<ctype.h>
int main()
```

```c
{
  char text[500], ch;
  int key;
  // taking user input
  printf("Enter a message to decrypt: ");
  scanf("%s", text);
  printf("Enter the key: ");
  scanf("%d", & key);
  //visiting each character
  for (int i = 0; text[i] != '\0'; ++i) {
    ch = text[i];
    // check for valid characters
    if (isalnum(ch)) {
      // lower case characters
      if (islower(ch)) {
        ch = (ch - 'a' - key + 26) % 26 + 'a';
      }
      // uppercase characters
      if (isupper(ch)) {
        ch = (ch - 'A' - key + 26) % 26 + 'A';
      }
      // numbers
      if (isdigit(ch)) {
        ch = (ch - '0' - key + 10) % 10 + '0';
      }
    }
    // invalid characters
    else {
      printf("Invalid Message");
    }
    // asding decoded character back
    text[i] = ch;

  }
  printf("Decrypted message: %s", text);
  return 0;
}
```

3. **Write a program to implement RSA algorithm for encryption and decryption of data.**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
long int p,q,n,t,flag,e[100],d[100],temp[100],j,m[100],en[100],i;
char msg[100];
int prime(long int);
void ce();
long int cd(long int);
void encrypt();
void decrypt();
void main()
{
printf("\nENTER FIRST PRIME NUMBER\n");
scanf("%d",&p);
flag=prime(p);
if(flag==0)
{
    printf("\nWRONG INPUT\n");
    getch();
    exit(1);
}
printf("\nENTER ANOTHER PRIME NUMBER\n");
scanf("%d",&q);
flag=prime(q);
if(flag==0||p==q)
{
    printf("\nWRONG INPUT\n");
    getch();
    exit(1);
}
printf("\nENTER MESSAGE\n");
fflush(stdin);
scanf("%s",msg);
for(i=0;msg[i]!=NULL;i++)
m[i]=msg[i];
n=p*q;
t=(p-1)*(q-1);
ce();
printf("\nPOSSIBLE VALUES OF e AND d ARE\n");
for(i=0;i<j-1;i++)
printf("\n%ld\t%ld",e[i],d[i]);
```

```
encrypt();
decrypt();
getch();
}
int prime(long int pr)
{
int i;
j=sqrt(pr);
for(i=2;i<=j;i++)
{
    if(pr%i==0)
    return 0;
}
return 1;
}
void ce()
{
int k;
k=0;
for(i=2;i<t;i++)
{
    if(t%i==0)
    continue;
    flag=prime(i);
    if(flag==1&&i!=p&&i!=q)
    {
        e[k]=i;
        flag=cd(e[k]);
        if(flag>0)
        {
            d[k]=flag;
            k++;
        }
        if(k==99)
        break;
    }
}
}
long int cd(long int x)
{
long int k=1;
while(1)
{
    k=k+t;
    if(k%x==0)
    return(k/x);
}
}
void encrypt()
{
long int pt,ct,key=e[0],k,len;
i=0;
len=strlen(msg);
while(i!=len)
{
    pt=m[i];
    pt=pt-96;
    k=1;
    for(j=0;j<key;j++)
    {
        k=k*pt;
        k=k%n;
    }
    temp[i]=k;
    ct=k+96;
    en[i]=ct;
    i++;
}
en[i]=-1;
printf("\nTHE ENCRYPTED MESSAGE IS\n");
for(i=0;en[i]!=-1;i++)
printf("%c",en[i]);
}
void decrypt()
{
long int pt,ct,key=d[0],k;
i=0;
while(en[i]!=-1)
{
    ct=temp[i];
    k=1;
    for(j=0;j<key;j++)
    {
        k=k*ct;
        k=k%n;
    }
    pt=k+96;
    m[i]=pt;
    i++;
}
```

```
m[i]=-1;
printf("\nTHE DECRYPTED MESSAGE IS\n");
for(i=0;m[i]!=-1;i++)
printf("%c",m[i]);
}
```

4. **Write a program for congestion control using leaky bucket algorithm.**

```
#include<stdio.h>
#include<stdlib.h>
#define bucketSize 512
void bktInput(int a,int b)
{
   if(a>bucketSize)
   printf("\n\t\tBucket overflow");
   else
   {
         sleep(500);
         while(a>b)
         {
               printf("\n %d bytes transmitted.",b);
               a=a-b;
               delay(500);
         }
         if (a>0)
         printf("\n%d Last bytes transmitted \t",a);
         printf("\n Bucket output successful");
   }
}
void main()
{
   int op, pktSize;
   randomize();
   printf("Enter output rate : ");
   scanf("%d",&op);
   for(int i=1;i<=5;i++)
   {
         sleep(random(1000));
         pktSize=random(1000);
         printf("\n Packet size = %d",pktSize);
         bktInput(pktSize,op);
   }
}
```

5. **Write a program which uses sockets to establish connections between client and server and exchange messages.**

Client program

```
#include <netinet/in.h> //structure for storing address
information
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h> //for socket APIs
#include <sys/types.h>
int main(int argc, char const* argv[])
{
   int sockD = socket(AF_INET, SOCK_STREAM, 0);
   struct sockaddr_in servAddr;
   servAddr.sin_family = AF_INET;
   servAddr.sin_port
         = htons(9001); // use some unused port number
   servAddr.sin_addr.s_addr = INADDR_ANY;
   int connectStatus
         = connect(sockD, (struct sockaddr*)&servAddr,
                   sizeof(servAddr));
   if (connectStatus == -1) {
         printf("Error...\n");
   }
   else {
         char strData[255];
         recv(sockD, strData, sizeof(strData), 0);
         printf("Message: %s\n", strData);
   }
   return 0;
}
```

Server Program

```
#include <netinet/in.h> //structure for storing address
information
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h> //for socket APIs
#include <sys/types.h>

int main(int argc, char const* argv[])
{

   // create server socket similar to what was done in
   // client program
   int servSockD = socket(AF_INET, SOCK_STREAM, 0);
```

```
    // string store data to send to client
    char serMsg[255] = "Message from the server to the "
                       "client \'Hello Client\' ";

    // define server address
    struct sockaddr_in servAddr;
    servAddr.sin_family = AF_INET;
    servAddr.sin_port = htons(9001);
    servAddr.sin_addr.s_addr = INADDR_ANY;
    // bind socket to the specified IP and port
    bind(servSockD, (struct sockaddr*)&servAddr,
         sizeof(servAddr));
    // listen for connections
    listen(servSockD, 1);
    // integer to hold client socket.
    int clientSocket = accept(servSockD, NULL, NULL);
    // send's messages to client socket
    send(clientSocket, serMsg, sizeof(serMsg), 0);
    return 0;
}
```

6. **Write a client-server program which uses sockets to make client sending the file name and the server to send back the contents of the requested file if present.**

Client Program

```
#include <stdio.h>
#include <arpa/inet.h>
#include <fcntl.h>
#include <unistd.h>
int main()
{
    int soc, n;
    char buffer[1024], fname[50];
    struct sockaddr_in addr;
    /*  socket creates an endpoint for communication and returns a
file descriptor */
    soc = socket(PF_INET, SOCK_STREAM, 0);
    /*
     * sockaddr_in is used for ip manipulation
     * we define the port and IP for the connection.
     */
    addr.sin_family = AF_INET;
    addr.sin_port = htons(7891);
    addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    /*  keep trying to esatablish connection with server */
    while(connect(soc, (struct sockaddr *) &addr, sizeof(addr))) ;
    printf("\nClient is connected to Server");
    printf("\nEnter file name: ");
    scanf("%s", fname);
    /*  send the filename to the server */
    send(soc, fname, sizeof(fname), 0);
    printf("\nRecieved response\n");
    /*  keep printing any data received from the server */
    while ((n = recv(soc, buffer, sizeof(buffer), 0)) > 0)
        printf("%s", buffer);

    return 0;
}
```

Server Program

```
#include <stdio.h>
#include <arpa/inet.h>
#include <fcntl.h>
#include <unistd.h>
int main()
{
    int welcome, new_soc, fd, n;
    char buffer[1024], fname[50];
    struct sockaddr_in addr;
    welcome = socket(PF_INET, SOCK_STREAM, 0);
    addr.sin_family = AF_INET;
    addr.sin_port = htons(7891);
    addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    bind(welcome, (struct sockaddr *) &addr, sizeof(addr));
    printf("\nServer is Online");
    /*  listen for connections from the socket */
    listen(welcome, 5);
    /*  accept a connection, we get a file descriptor */
    new_soc = accept(welcome, NULL, NULL);
    /*  receive the filename */
    recv(new_soc, fname, 50, 0);
    printf("\nRequesting for file: %s\n", fname);
    /*  open the file and send its contents */
    fd = open(fname, O_RDONLY);
    if (fd < 0)
        send(new_soc, "\nFile not found\n", 15, 0);
    else
        while ((n = read(fd, buffer, sizeof(buffer))) > 0)
            send(new_soc, buffer, n, 0);
    printf("\nRequest sent\n");
    close(fd);
    return 0;
}
```