Here are the answers to the questions from the provided document.

Network Theory and Concepts

1. Compare various network topologies in detail.
Network topology refers to the arrangement of the elements (links, nodes, etc.) of a communication network. Here is a detailed comparison of common network topologies:

a) Bus Topology

- **Structure**: All devices are connected to a single central cable, called the bus or backbone.
- **Data Flow**: Data sent by a device is broadcasted along the entire bus. All devices receive the signal, but only the intended recipient accepts and processes the data. Terminators are required at both ends of the bus to prevent signal reflection.
- **Advantages**:
    - Easy to install and inexpensive.
    - Requires less cable length compared to a star topology.
- **Disadvantages**:
    - The entire network shuts down if there is a break in the main cable.
    - Difficult to troubleshoot.
    - Performance degrades as more devices are added due to data collisions.
    - Low security as all computers on the bus can see all data transmissions.

b) Star Topology

- **Structure**: All devices are connected to a central hub or switch.
- **Data Flow**: Devices communicate via the central device. In a hub-based star network, data is broadcast to all devices. In a switch-based star network, the switch intelligently forwards data only to the intended recipient.
- **Advantages**:
    - Easy to install and manage.
    - Failure of a single node or cable does not affect the rest of the network.
    - Easy to add new nodes.
    - Better performance than bus topology.
- **Disadvantages**:
    - If the central hub or switch fails, the entire network goes down.
    - Requires more cable than a bus topology.
    - Can be more expensive due to the cost of the central device.

c) Ring Topology

- **Structure**: Each device is connected to exactly two other devices, forming a single continuous pathway for signals through each node - a ring.
- **Data Flow**: Data travels from node to node in one direction. Each device regenerates the signal before passing it along, which prevents signal degradation.
- **Advantages**:
    - Performs better than a bus topology under heavy network load.
    - Prevents network collisions as only one device can transmit at a time (using a token).
- **Disadvantages**:
    - Failure of one node can affect the entire network.
    - Difficult to troubleshoot.
    - Changes made to the network, like adding or removing a device, disrupt the

network.

d) Mesh Topology

- **Structure**: Every device is connected to every other device on the network. This is known as a full mesh. In a partial mesh, some devices are connected to all others, but some are only connected to those other nodes with which they exchange the most data.
- **Data Flow**: Data can be sent directly from any device to any other device. The network can self-heal by routing data around failed nodes.
- **Advantages**:
    - Extremely fault-tolerant and reliable. A failure of one link does not affect the rest of the network.
    - High security and privacy.
    - Easy to troubleshoot.
- **Disadvantages**:
    - Very expensive due to the large amount of cabling and number of ports required.
    - Complex and difficult to install and manage.

e) Tree Topology

- **Structure**: This is a hybrid topology that combines characteristics of bus and star topologies. It consists of groups of star-configured workstations connected to a linear bus backbone cable.
- **Data Flow**: Data flows through the hierarchy of the tree.
- **Advantages**:
    - Scalable; easy to add new nodes to the network.
    - Point-to-point wiring for individual segments.
    - Easier to manage and troubleshoot than a bus network.
- **Disadvantages**:
    - If the central hub in a segment fails, the devices connected to it are disabled.
    - If the main backbone cable fails, the entire network goes down.
    - More difficult to configure than other topologies.

f) Hybrid Topology

- **Structure**: A combination of two or more different topologies. For example, a star-bus network or a star-ring network.
- **Data Flow**: Depends on the constituent topologies.
- **Advantages**:
    - Flexible and can be designed according to requirements.
    - Very scalable.
- **Disadvantages**:
    - Can be expensive.
    - Complex to design and manage.

2. What is socket? Write down 3 way handshaking mechanism of TCP.

What is a Socket?

A socket is a software endpoint that establishes bidirectional communication between a client program and a server program on a network. A socket is defined by an IP address and a port number. It acts as an interface in a network to send and receive data. When a program wants to communicate with another program over a network, it creates a socket that is bound to a specific port on its machine. The other program can then connect to this socket to establish a communication channel.

TCP 3-Way Handshaking Mechanism

The TCP 3-way handshake is the process used by TCP to establish a reliable connection

between a client and a server. It ensures that both sides are ready to transmit and receive data. The process involves the exchange of three packets:

1. **SYN (Synchronize)**:
    - The client, wishing to establish a connection, sends a segment with the SYN flag set to 1 to the server.
    - This packet includes a random sequence number (e.g., Seq=x) that the client will use as its initial sequence number.
    - Essentially, the client is saying, "I want to connect. My starting sequence number is x."
2. **SYN-ACK (Synchronize-Acknowledge)**:
    - The server, upon receiving the SYN packet, responds with its own segment.
    - This server segment has the SYN flag set to 1 and the ACK flag set to 1.
    - The server acknowledges the client's request by setting the acknowledgment number to one more than the client's sequence number (Ack=x+1).
    - The server also chooses its own initial sequence number (e.g., Seq=y).
    - Essentially, the server is saying, "I got your request and I agree to connect. I'm acknowledging your sequence number x, and my starting sequence number is y."
3. **ACK (Acknowledge)**:
    - Finally, the client receives the SYN-ACK packet from the server and sends a final acknowledgment packet back.
    - This packet has the ACK flag set.
    - The client acknowledges the server's sequence number by setting the acknowledgment number to one more than the server's sequence number (Ack=y+1).
    - Essentially, the client is saying, "I received your acknowledgment. The connection is established."

Once this third step is complete, the TCP connection is established, and both the client and server can start exchanging data.

3. Differentiate between hub and switch. Write the advantages and disadvantages of Optical Fiber.

Difference between Hub and Switch

Feature	Hub	Switch
OSI Layer	Operates at the Physical Layer (Layer 1).	Operates at the Data Link Layer (Layer 2).
Function	Acts as a multiport repeater. It receives a signal on one port and broadcasts it to all other ports.	Acts as an intelligent device. It receives a frame, inspects the destination MAC address, and forwards it only to the specific port connected to the destination device.
Intelligence	It is a "dumb" device with no processing capabilities.	It is an "intelligent" device that maintains a MAC address table to direct traffic.
Collision Domain	All ports on a hub belong to a single collision domain. Collisions are frequent if multiple devices transmit	Each port on a switch is its own collision domain. This drastically reduces or eliminates collisions.

Feature	Hub	Switch
	simultaneously.	
Bandwidth	The total network bandwidth is shared among all connected devices.	Each port has dedicated bandwidth. For example, on a 100 Mbps switch, each port can theoretically operate at 100 Mbps.
Transmission Mode	Operates in half-duplex mode (cannot send and receive data at the same time).	Can operate in full-duplex mode (can send and receive data simultaneously).

Advantages and Disadvantages of Optical Fiber

Advantages:

- **Higher Bandwidth**: Optical fiber can carry significantly more data than copper cables of the same size.
- **Longer Distances**: Signals can be transmitted over much longer distances with less signal loss (attenuation).
- **Immunity to Electromagnetic Interference (EMI)**: Since it transmits data using light instead of electricity, it is immune to interference from power lines, motors, and other electrical noise sources.
- **Enhanced Security**: It is very difficult to tap into a fiber optic cable without being detected, making it more secure than copper cable.
- **Smaller Size and Lighter Weight**: Fiber optic cables are much thinner and lighter than copper cables, making them easier to install, especially in tight spaces.

Disadvantages:

- **Higher Cost**: The cost of fiber optic cable, connectors, and the equipment required to install and test it is higher than that of copper cable.
- **Fragility**: Optical fiber is made of glass, which is more fragile and susceptible to damage during installation and handling compared to copper wire.
- **Difficult to Install**: Installation requires specialized skills and tools. Splicing or terminating fiber is a more complex and delicate process than for copper wires.
- **Specialized Equipment**: Requires specialized transceivers and other network hardware that can convert electrical signals to light signals and back.

4. Discuss and compares various types of networks.

Computer networks are often categorized based on their geographical scope. Here is a discussion and comparison of the main types:

a) LAN (Local Area Network)

- **Scope**: A LAN connects network devices over a relatively short distance, such as within a single building, an office, a school campus, or a home.
- **Ownership**: Typically owned, controlled, and managed by a single person or organization.
- **Technology**: Commonly uses Ethernet (wired) and Wi-Fi (wireless) technologies. Data speeds are very high (e.g., 100 Mbps, 1 Gbps, 10 Gbps).
- **Comparison**: LANs have the highest speed and lowest latency among the network types. They are relatively inexpensive to set up and maintain.

b) MAN (Metropolitan Area Network)

- **Scope**: A MAN is a network that spans a larger geographical area than a LAN but smaller than a WAN, such as an entire city or a large campus. It might connect multiple LANs

together.

- **Ownership**: A MAN can be owned and operated by a single organization, but it is often used by many individuals and organizations. It can also be owned and operated by a telecommunications company that provides services to the public.
- **Technology**: Often uses fiber optic cables or other high-speed media to connect the distributed LANs.
- **Comparison**: MANs are larger than LANs but smaller than WANs. They provide a high-speed backbone for connecting geographically separated LANs.

c) WAN (Wide Area Network)

- **Scope**: A WAN spans a very large geographical area, such as across a country, a continent, or even the entire world. The internet is the largest WAN.
- **Ownership**: WANs are rarely owned by a single private organization. They are typically managed and operated by multiple service providers (e.g., telecom companies, ISPs).
- **Technology**: WANs connect LANs and MANs. They use technologies like MPLS, ATM, Frame Relay, and fiber optic links. Speeds are generally lower and latency higher than LANs due to the long distances involved.
- **Comparison**: WANs are the most extensive type of network. They allow for communication over long distances but are more complex and have higher latency than LANs.

d) PAN (Personal Area Network)

- **Scope**: A PAN is a network organized around an individual person. It is used for communication among devices close to one person, such as smartphones, laptops, headphones, and other personal electronics.
- **Ownership**: Owned and used by a single individual.
- **Technology**: Usually uses wireless technologies like Bluetooth or Wi-Fi Direct.
- **Comparison**: PAN is the smallest type of network, with a range of only a few meters.

Summary Comparison Table

Feature	PAN	LAN	MAN	WAN
Geographical Area	A few meters	A room, building, or campus	A city or large campus	A country, continent, or the world
Ownership	Private	Private	Private or Public	Public or Distributed
Speed	Moderate	Very High	High	Lower than LAN/MAN
Common Technology	Bluetooth, Wi-Fi Direct	Ethernet, Wi-Fi	Fiber Optics, ATM	MPLS, Leased Lines, Internet Links
Latency	Low	Very Low	Low to Moderate	High

IP Addressing and Subnetting

1. An ISP is granted a block of addresses starting with 190.100.0.0/16. The ISP needs to distribute these addresses to three groups of customers as follows: i. The first group has 64 customers; each needs 256 addresses. ii. The second group has 128 customers; each needs 128 addresses. iii. The third group has 128 customers; each needs 64 addresses.
Here is a possible design for the address allocation using Variable Length Subnet Masking (VLSM). We will allocate the largest blocks first.

Group 1 & 2 Calculation (same total size)

- **Group 1:** 64 customers × 256 addresses/customer = 16,384 total addresses.
- **Group 2:** 128 customers × 128 addresses/customer = 16,384 total addresses. To accommodate 16,384 addresses, we need a block size of 2^{14} (since 2^{14} = 16,384). This corresponds to a CIDR prefix of 32 - 14 = /18.

Group 3 Calculation

- **Group 3:** 128 customers × 64 addresses/customer = 8,192 total addresses. To accommodate 8,192 addresses, we need a block size of 2^{13} (since 2^{13} = 8,192). This corresponds to a CIDR prefix of 32 - 13 = /19.

Address Allocation

We start with the initial block: **190.100.0.0/16** (Range: 190.100.0.0 to 190.100.255.255).

1. **Allocation for Group 1 (/18 block):**
    - **Subnet Address:** 190.100.0.0/18
    - **Address Range:** 190.100.0.0 to 190.100.63.255. This block provides 16,384 addresses.
2. **Allocation for Group 2 (/18 block):**
    - The next available address is 190.100.64.0.
    - **Subnet Address:** 190.100.64.0/18
    - **Address Range:** 190.100.64.0 to 190.100.127.255. This block provides 16,384 addresses.
3. **Allocation for Group 3 (/19 block):**
    - The next available address is 190.100.128.0.
    - **Subnet Address:** 190.100.128.0/19
    - **Address Range:** 190.100.128.0 to 190.100.159.255. This block provides 8,192 addresses.

Summary of Allocation:

- **Group 1:** 190.100.0.0/18
- **Group 2:** 190.100.64.0/18
- **Group 3:** 190.100.128.0/19

3. A supernet has a first address of 205.16.32.0 and a supernet mask of 255.255.248.0. State the number of blocks are in this supernet and what is the range of addresses?

1. Find the CIDR Prefix: The supernet mask is 255.255.248.0. Let's convert this to binary: 11111111.11111111.11111000.00000000 Counting the number of 1s gives the prefix: 8 + 8 + 5 = 21. So, the mask is /21.

2. Number of Blocks: The first address, 205.16.32.0, belongs to a Class C range (192-223). A standard Class C network has a /24 mask. The supernet mask is /21. The number of combined Class C blocks is calculated as: 2^{(\text{Class C prefix} - \text{Supernet prefix})} = 2^{(24 - 21)} = 2^3 = 8. So, there are **8 blocks** of Class C networks in this supernet.

3. Range of Addresses:

- **First Address (Network Address):** 205.16.32.0
- **Calculate the total number of addresses:** The host portion of the address is 32 - 21 = 11 bits. The total number of addresses is 2^{11} = 2048.
- **Calculate the Last Address (Broadcast Address):**
    - The network address is 205.16.32.0.
    - The last address is found by adding (Total Addresses - 1) to the first address. In the IP address, this means adding 2047 to the host part.
    - The address range covers from 205.16.32.0 to 205.16.(32 + 7).255.
    - The last octet will be 255. The third octet will be 32 + (2048 / 256 - 1) = 32 + 7 = 39.

    - So, the last address is 205.16.39.255.
- **Address Range:** The range of addresses is from **205.16.32.0 to 205.16.39.255**.

5. A company is granted the site address 201.70.64.0. The company needs six subnets design the subnets.

1. Determine Bits to Borrow:

- The given site address 201.70.64.0 is a Class C address, which has a default subnet mask of /24 (255.255.255.0).
- The company needs six subnets. We need to find the number of bits (n) to borrow from the host part such that 2^n \ge 6.
- If n=2, 2^2 = 4 (not enough).
- If n=3, 2^3 = 8 (sufficient).
- So, we need to borrow **3 bits**.

2. Calculate the New Subnet Mask:

- The new subnet mask is the default mask plus the borrowed bits: 24 + 3 = /27.
- In decimal, the mask is **255.255.255.224** (11111111.11111111.11111111.11100000).

3. Calculate Subnet Details:

- **Number of addresses per subnet:** The host part has 32 - 27 = 5 bits. So, each subnet has 2^5 = 32 addresses.
- **Block Size (Increment):** The increment for each subnet is 32 in the fourth octet.

4. Design the Subnets: Here are the first six subnets:

- **Subnet 1:**
    - Subnet Address: 201.70.64.0/27
    - Range: 201.70.64.0 - 201.70.64.31
- **Subnet 2:**
    - Subnet Address: 201.70.64.32/27
    - Range: 201.70.64.32 - 201.70.64.63
- **Subnet 3:**
    - Subnet Address: 201.70.64.64/27
    - Range: 201.70.64.64 - 201.70.64.95
- **Subnet 4:**
    - Subnet Address: 201.70.64.96/27
    - Range: 201.70.64.96 - 201.70.64.127
- **Subnet 5:**
    - Subnet Address: 201.70.64.128/27
    - Range: 201.70.64.128 - 201.70.64.159
- **Subnet 6:**
    - Subnet Address: 201.70.64.160/27
    - Range: 201.70.64.160 - 201.70.64.191

6. Write down the subnet work number of a host with an IP address of 172.16.66.0/21?

To find the subnet work number (also known as the subnet address or network ID), we perform a bitwise AND operation between the IP address and the subnet mask.

1. **IP Address:** 172.16.66.0
2. **Subnet Mask:** /21, which is 255.255.248.0 in decimal.

Let's convert the third octet of both the IP and the mask to binary, as this is where the subnet boundary lies.

- IP Address (172.16. **66** .0): 10101100.00010000.01000010.00000000
- Subnet Mask (255.255. **248** .0): 11111111.11111111.11111000.00000000

Now, perform the bitwise AND operation:

```
  10101100.00010000.01000010.00000000  (IP Address: 172.16.66.0)
& 11111111.11111111.11111000.00000000  (Subnet Mask: 255.255.248.0)
-----------------------------------
  10101100.00010000.01000000.00000000  (Result)
```

Finally, convert the resulting binary address back to decimal:

- 10101100 = 172
- 00010000 = 16
- 01000000 = 64
- 00000000 = 0

The subnet work number is **172.16.64.0**.

7. You need to subnet a network that has 5 subnets, each with at least 16 hosts. State classful subnet mask would you use?

Let's analyze the requirements:

1. **Host Requirement:** At least 16 hosts per subnet.
    - The formula for usable hosts is 2^h - 2, where h is the number of host bits.
    - 2^h - 2 \ge 16 \implies 2^h \ge 18.
    - 2^4 = 16 (not enough). 2^5 = 32 (sufficient).
    - So, we need to reserve at least **5 bits for hosts**.
2. **Subnet Requirement:** 5 subnets.
    - The formula for subnets is 2^s, where s is the number of subnet bits.
    - 2^s \ge 5.
    - 2^2 = 4 (not enough). 2^3 = 8 (sufficient).
    - So, we need to borrow at least **3 bits for subnets**.

The most efficient classful network to use would be a **Class C** network. A Class C network has a default /24 mask, leaving 8 bits for hosts.

- We need 3 bits for subnets and 5 bits for hosts.
- Total bits needed from the host portion = 3 + 5 = 8.
- This fits perfectly into the 8 host bits of a Class C address.

Subnet Mask Calculation:

- Start with the default Class C mask: /24 (255.255.255.0).
- Borrow 3 bits for the subnets.
- New subnet mask prefix: 24 + 3 = /27.
- The /27 mask in decimal is **255.255.255.224**.

This mask creates 2^3 = 8 subnets, and each subnet has 2^5 = 32 addresses (30 usable hosts), satisfying both conditions.

8. Your company has the network ID 165.121.0.0. You are responsible for creating subnets on the network, and each subnet must provide at least 900 host IDs. State the subnet mask which meets the requirement for the minimum number of host IDs and provides the greatest number of subnets?

1. **Identify Network Class:** The network ID 165.121.0.0 is a Class B network, which has a default mask of /16 (255.255.0.0).
2. **Calculate Required Host Bits:**
    - Each subnet needs at least 900 host IDs.
    - Using the formula 2^h - 2 \ge 900, where h is the number of host bits.
    - 2^h \ge 902.
    - 2^9 = 512 (too small).
    - 2^{10} = 1024 (sufficient).

    - So, we must reserve **10 bits for the host part**.
3. **Determine the Subnet Mask:**
    - The total number of bits in an IPv4 address is 32.
    - To get 10 host bits, the subnet mask must have 32 - 10 = 22 network bits.
    - The CIDR prefix for the subnet mask is **/22**.
4. **Convert to Decimal Notation:**
    - A /22 mask in binary is 11111111.11111111.11111100.00000000.
    - Converting this to decimal gives **255.255.252.0**.

This subnet mask (255.255.252.0 or /22) meets the requirement of at least 900 hosts (providing 2^{10} - 2 = 1022 usable hosts) and maximizes the number of subnets because it uses the minimum possible number of host bits.

9. Write down the broadcast address on subnet 32 given a prefix notation of 12.1.0.0/12?

1. **Analyze the Prefix:** The prefix is 12.1.0.0/12.
    - Subnet Mask: /12, which is 255.240.0.0.
    - The subnetting is done in the second octet.
2. **Determine the Subnet Range:**
    - The block size in the second octet is 256 - 240 = 16.
    - This means the valid network addresses for the second octet are multiples of 16: 0, 16, 32, 48, etc.
3. **Identify the Target Subnet:**
    - The question asks for "subnet 32", which refers to the subnet whose network address is **12.32.0.0/12**.
4. **Calculate the Broadcast Address:**
    - The broadcast address is the last address in the subnet's range. To find it, we can find the next subnet's address and subtract one.
    - The current subnet is 12. **32** .0.0.
    - The next subnet would be 12. **(32 + 16)** .0.0 = 12.48.0.0.
    - The broadcast address for the 12.32.0.0 subnet is the address just before 12.48.0.0.
    - Therefore, the broadcast address is **12.47.255.255**.

10. An organization is granted the blocks 130.56.0.0. The administrator wants to create 1024 fixed length subnets. i. Find the subnet mask. ii. Find the number of addresses in each subnet. iii. Find the first and last addresses of the first subnet. iv. Find the first and last addresses of the last subnet.

Initial Information:

- Network Address: 130.56.0.0. This is a Class B address, so the default mask is /16.
- Required Subnets: 1024.

i. Find the subnet mask.

- We need to find the number of bits (s) to borrow such that 2^s \ge 1024.
- 2^{10} = 1024. So, we need to borrow **10 bits**.
- New prefix = Default prefix + borrowed bits = 16 + 10 = /26.
- The subnet mask is **/26**, which in decimal is **255.255.255.192**.

ii. Find the number of addresses in each subnet.

- The number of host bits (h) is 32 - \text{new prefix} = 32 - 26 = 6 bits.
- The number of addresses per subnet is 2^h = 2^6 = 64.

iii. Find the first and last addresses of the first subnet.

- The first subnet is always the original network address with the new mask.
- **First Address (Network ID):** 130.56.0.0

- The subnet has 64 addresses, so the block size is 64. The range is 0-63 in the last octet.
- **Last Address (Broadcast ID):** 130.56.0.63

iv. Find the first and last addresses of the last subnet.

- There are 1024 subnets in total (from subnet 0 to subnet 1023).
- The borrowed bits are 10 (2 from the third octet and 8 from the fourth octet... wait, this is incorrect). The 16 host bits are in the 3rd and 4th octets. We borrow the first 10. This means we borrow all 8 bits from the third octet, and the first 2 bits from the fourth octet.
- For the **last** subnet, all 10 borrowed bits will be '1'.
- The third octet (8 bits) will be 11111111 = 255.
- The first 2 bits of the fourth octet will be 11.
- To find the network ID of this last subnet, we set the host bits to 0. The host bits are the last 6 bits of the fourth octet.
- Fourth octet = 11000000 = 192.
- So, the **First Address (Network ID)** of the last subnet is **130.56.255.192**.
- The block size is 64. The range is 192-255.
- The **Last Address (Broadcast ID)** of the last subnet is **130.56.255.255**.

11. An organization is granted a block of addresses starting with 190.100.0.0/26. The organization needs to have four subnets. Write down the subnet addresses and the range of addresses for each subnets?

1. Analyze the Existing Block:

- The given block is 190.100.0.0/26.
- The mask /26 means the current block size is 2^{(32-26)} = 2^6 = 64 addresses.
- The range of this block is 190.100.0.0 to 190.100.0.63.

2. Determine Bits to Borrow:

- The organization needs four subnets from this block.
- We need to find the number of bits (n) to borrow such that 2^n \ge 4.
- 2^2 = 4. So, we need to borrow **2 more bits**.

3. Calculate the New Subnet Mask:

- The new subnet mask prefix is the old prefix plus the borrowed bits: 26 + 2 = /28.
- The new mask is 255.255.255.240.

4. Calculate New Subnet Details:

- **Number of addresses per new subnet:** 2^{(32-28)} = 2^4 = 16.
- **Block Size (Increment):** The increment is 16 in the fourth octet.

5. List the Subnets and Ranges: We will create four subnets of size 16 within the original 190.100.0.0 - 190.100.0.63 range.

- **Subnet 1:**
  - Subnet Address: 190.100.0.0/28
  - Range of Addresses: 190.100.0.0 - 190.100.0.15
- **Subnet 2:**
  - Subnet Address: 190.100.0.16/28
  - Range of Addresses: 190.100.0.16 - 190.100.0.31
- **Subnet 3:**
  - Subnet Address: 190.100.0.32/28
  - Range of Addresses: 190.100.0.32 - 190.100.0.47
- **Subnet 4:**
  - Subnet Address: 190.100.0.48/28
  - Range of Addresses: 190.100.0.48 - 190.100.0.63

Programming Problems

1. Write client-server programs in java such that the client makes a request to the server for Current date & time and the server respond with the same.

DateTimeServer.java

```
import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Date;

public class DateTimeServer {
    public static void main(String[] args) {
        try {
            // Server listens on port 9999
            ServerSocket serverSocket = new ServerSocket(9999);
            System.out.println("Server is running and waiting for a
client...");

            // Accept client connection
            Socket clientSocket = serverSocket.accept();
            System.out.println("Client connected: " +
clientSocket.getInetAddress().getHostAddress());

            // Create output stream to send data to the client
            DataOutputStream dos = new
DataOutputStream(clientSocket.getOutputStream());

            // Get current date and time
            Date currentDate = new Date();
            String dateTimeString = currentDate.toString();

            // Write the date and time string to the client
            dos.writeUTF("Current Server Date & Time: " +
dateTimeString);
            dos.flush();

            // Close resources
            dos.close();
            clientSocket.close();
            serverSocket.close();
            System.out.println("Connection closed.");

        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

DateTimeClient.java

```
import java.io.DataInputStream;
import java.net.Socket;

public class DateTimeClient {
    public static void main(String[] args) {
        try {
            // Client connects to the server at localhost on port 9999
            Socket socket = new Socket("localhost", 9999);
            System.out.println("Connected to the server.");

            // Create input stream to receive data from the server
            DataInputStream dis = new
DataInputStream(socket.getInputStream());

            // Read the date and time string from the server
            String dateTimeString = dis.readUTF();
            System.out.println(dateTimeString);

            // Close resources
            dis.close();
            socket.close();

        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

2. Write a program in java to access the IP address of the local machine then find in the class where this address belongs to.

IPClassFinder.java

```
import java.net.InetAddress;
import java.net.UnknownHostException;

public class IPClassFinder {

    public static void main(String[] args) {
        try {
            // Get the local host's InetAddress object
            InetAddress localHost = InetAddress.getLocalHost();
            System.out.println("Local Host Name: " +
localHost.getHostName());
            System.out.println("Local IP Address: " +
localHost.getHostAddress());

            // Get the raw IP address as a byte array
```

```
            byte[] ipAddressBytes = localHost.getAddress();

            // The first byte determines the class
            // Note: Bytes in Java are signed (-128 to 127). We
convert it to an unsigned value (0-255).
            int firstOctet = ipAddressBytes[0] & 0xFF;

            // Determine the class
            String ipClass;
            if (firstOctet >= 1 && firstOctet <= 126) {
                ipClass = "A";
            } else if (firstOctet >= 128 && firstOctet <= 191) {
                ipClass = "B";
            } else if (firstOctet >= 192 && firstOctet <= 223) {
                ipClass = "C";
            } else if (firstOctet >= 224 && firstOctet <= 239) {
                ipClass = "D (Multicast)";
            } else if (firstOctet >= 240 && firstOctet <= 255) {
                ipClass = "E (Experimental)";
            } else if (firstOctet == 127) {
                 ipClass = "Loopback Address";
            } else {
                 ipClass = "Unknown";
            }

            System.out.println("The IP address belongs to Class: " +
ipClass);

        } catch (UnknownHostException e) {
            System.err.println("Could not determine local host IP
address.");
            e.printStackTrace();
        }
    }
}
```

3. Write client-server programs in java such that client sends a string to server & the server. Returns back the string after converting it to uppercase.

UppercaseServer.java

```
import java.io.*;
import java.net.*;

public class UppercaseServer {
    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(8888);
            System.out.println("Server is running and waiting for
connections...");
```

```
            Socket clientSocket = serverSocket.accept();
            System.out.println("Client connected: " +
clientSocket.getInetAddress().getHostAddress());

            // Setup streams
            BufferedReader reader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            PrintWriter writer = new
PrintWriter(clientSocket.getOutputStream(), true);

            // Read string from client
            String clientString = reader.readLine();
            System.out.println("Received from client: " +
clientString);

            // Convert to uppercase and send back
            String uppercaseString = clientString.toUpperCase();
            writer.println(uppercaseString);
            System.out.println("Sent to client: " + uppercaseString);

            // Close resources
            reader.close();
            writer.close();
            clientSocket.close();
            serverSocket.close();
            System.out.println("Connection closed.");

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

UppercaseClient.java

```
import java.io.*;
import java.net.*;

public class UppercaseClient {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("localhost", 8888);
            System.out.println("Connected to the server.");

            // Setup streams
            BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter writer = new
```

```
PrintWriter(socket.getOutputStream(), true);
            BufferedReader consoleReader = new BufferedReader(new
InputStreamReader(System.in));

            // Read string from user
            System.out.print("Enter a string to send to the server:
");
            String stringToSend = consoleReader.readLine();

            // Send string to server
            writer.println(stringToSend);

            // Receive uppercase string from server and print it
            String receivedString = reader.readLine();
            System.out.println("Server response: " + receivedString);

            // Close resources
            reader.close();
            writer.close();
            socket.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

4. Implement Client-Server program for Stop-and-Wait ARQ.

This is a conceptual implementation. A real-world scenario would handle packet loss more robustly. Here, we simulate the logic.

Sender.java (Client)

```
import java.io.*;
import java.net.*;

public class Sender {
    public static void main(String[] args) throws Exception {
        String[] data = {"Frame1", "Frame2", "Frame3", "Frame4",
"end"};
        Socket socket = new Socket("localhost", 9999);
        System.out.println("Connected to receiver.");

        DataInputStream in = new
DataInputStream(socket.getInputStream());
        DataOutputStream out = new
DataOutputStream(socket.getOutputStream());

        int sequenceNumber = 0;
```

```
        for (String frame : data) {
            boolean ackReceived = false;
            while (!ackReceived) {
                // Send the frame with sequence number
                String packet = sequenceNumber + ":" + frame;
                out.writeUTF(packet);
                System.out.println("Sent: " + packet);

                // Set a timeout for the acknowledgment
                socket.setSoTimeout(2000); // 2 seconds timeout

                try {
                    // Wait for ACK
                    String ackString = in.readUTF();
                    int ackNum = Integer.parseInt(ackString);
                    System.out.println("Received ACK: " + ackNum);

                    if (ackNum == sequenceNumber + 1) {
                        ackReceived = true;
                        sequenceNumber++; // Move to the next sequence
number
                    }
                } catch (SocketTimeoutException e) {
                    System.out.println("Timeout! Resending the
frame...");
                }
            }
        }
        socket.close();
    }
}
```

Receiver.java (Server)

```
import java.io.*;
import java.net.*;
import java.util.Random;

public class Receiver {
    public static void main(String[] args) throws Exception {
        ServerSocket serverSocket = new ServerSocket(9999);
        System.out.println("Receiver is ready.");
        Socket socket = serverSocket.accept();
        System.out.println("Sender connected.");

        DataInputStream in = new
DataInputStream(socket.getInputStream());
        DataOutputStream out = new
DataOutputStream(socket.getOutputStream());
```

```
        int expectedSequenceNumber = 0;
        Random random = new Random();

        while (true) {
            String receivedPacket = in.readUTF();
            String[] parts = receivedPacket.split(":", 2);
            int receivedSeqNum = Integer.parseInt(parts[0]);
            String data = parts[1];

            System.out.println("Received: " + receivedPacket);

            if (data.equals("end")) {
                break;
            }

            // Simulate packet loss occasionally (e.g., 25% chance to
"lose" packet)
            if (random.nextInt(4) == 0) {
                 System.out.println("SIMULATING PACKET LOSS. NOT
SENDING ACK.");
                 continue; // Don't send ACK, sender will time out.
            }

            if (receivedSeqNum == expectedSequenceNumber) {
                System.out.println("Data received successfully: " +
data);
                // Send ACK for the next expected frame
                int ackNum = expectedSequenceNumber + 1;
                out.writeUTF(String.valueOf(ackNum));
                System.out.println("Sent ACK: " + ackNum);
                expectedSequenceNumber++;
            } else {
                System.out.println("Duplicate or out-of-order frame
received. Resending last ACK.");
                // Resend ACK for the frame it's still waiting for
                int ackNum = expectedSequenceNumber;
                out.writeUTF(String.valueOf(ackNum));
                 System.out.println("Sent duplicate ACK: " + ackNum);
            }
        }

        socket.close();
        serverSocket.close();
        System.out.println("Connection closed.");
    }
}
```

5. Write a program in java that takes a site address and the number of subnets (that may or may not be power 2) as input and print the subnet mask as well as subnetwork addresses of each subnet.

SubnetCalculator.java

```
import java.util.Scanner;

public class SubnetCalculator {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter site address (e.g., 201.70.64.0): ");
        String siteAddress = scanner.nextLine();

        System.out.print("Enter the number of subnets required: ");
        int numSubnets = scanner.nextInt();

        // --- 1. Determine bits to borrow ---
        int bitsToBorrow = 0;
        // Smallest 'n' such that 2^n >= numSubnets
        while (Math.pow(2, bitsToBorrow) < numSubnets) {
            bitsToBorrow++;
        }
        System.out.println("\nBits to borrow: " + bitsToBorrow);

        // --- 2. Determine default mask ---
        String[] octetsStr = siteAddress.split("\\.");
        int firstOctet = Integer.parseInt(octetsStr[0]);
        int defaultMaskPrefix = 0;
        if (firstOctet >= 1 && firstOctet <= 126) defaultMaskPrefix =
8;
        else if (firstOctet >= 128 && firstOctet <= 191)
defaultMaskPrefix = 16;
        else if (firstOctet >= 192 && firstOctet <= 223)
defaultMaskPrefix = 24;
        else {
            System.out.println("Invalid or unsupported IP address
class.");
            return;
        }

        // --- 3. Calculate new subnet mask ---
        int newMaskPrefix = defaultMaskPrefix + bitsToBorrow;
        System.out.println("New Subnet Mask Prefix: /" +
newMaskPrefix);

        // Convert prefix to decimal mask
        long mask = 0xFFFFFFFFL << (32 - newMaskPrefix);
```

```
        String subnetMask = String.format("%d.%d.%d.%d",
                (mask >> 24) & 0xFF, (mask >> 16) & 0xFF, (mask >> 8)
& 0xFF, mask & 0xFF);
        System.out.println("New Subnet Mask: " + subnetMask);

        // --- 4. Calculate and print subnet addresses ---
        long ipAsLong = 0;
        for (String octet : octetsStr) {
            ipAsLong = (ipAsLong << 8) | Integer.parseInt(octet);
        }

        long baseNetworkAddress = ipAsLong & mask;
        long blockSize = 1L << (32 - newMaskPrefix);
        long numberOfPossibleSubnets = 1L << bitsToBorrow;

        System.out.println("\n--- Subnetwork Addresses ---");
        for (int i = 0; i < numberOfPossibleSubnets; i++) {
            long currentSubnetAddress = baseNetworkAddress + (i *
blockSize);
             String subnetAddrStr = String.format("%d.%d.%d.%d",
                (currentSubnetAddress >> 24) & 0xFF,
                (currentSubnetAddress >> 16) & 0xFF,
                (currentSubnetAddress >> 8) & 0xFF,
                currentSubnetAddress & 0xFF);
            System.out.println("Subnet " + (i+1) + ": " +
subnetAddrStr + "/" + newMaskPrefix);
        }

        scanner.close();
    }
}
```

6. Write client-server programs in Java to implement echo-client & echo-server.
EchoServer.java

```
import java.io.*;
import java.net.*;

public class EchoServer {
    public static void main(String[] args) {
        System.out.println("Echo Server started. Waiting for
clients...");
        try (ServerSocket serverSocket = new ServerSocket(7777)) {
            Socket clientSocket = serverSocket.accept();
            System.out.println("Client connected: " +
clientSocket.getInetAddress());

            PrintWriter out = new
PrintWriter(clientSocket.getOutputStream(), true);
```

```
            BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));

            String inputLine;
            while ((inputLine = in.readLine()) != null) {
                System.out.println("Received from client: " +
inputLine);
                out.println(inputLine); // Echo the line back to the
client
            }
            System.out.println("Client disconnected.");

        } catch (IOException e) {
            System.out.println("Exception caught when trying to listen
on port 7777 or listening for a connection");
            System.out.println(e.getMessage());
        }
    }
}
```

EchoClient.java

```
import java.io.*;
import java.net.*;

public class EchoClient {
    public static void main(String[] args) {
        String hostname = "localhost";
        int port = 7777;

        try (Socket echoSocket = new Socket(hostname, port);
             PrintWriter out = new
PrintWriter(echoSocket.getOutputStream(), true);
             BufferedReader in = new BufferedReader(new
InputStreamReader(echoSocket.getInputStream()));
             BufferedReader stdIn = new BufferedReader(new
InputStreamReader(System.in))) {

            System.out.println("Connected to Echo Server. Type
anything and press Enter.");
            System.out.println("Type 'exit' to quit.");

            String userInput;
            while ((userInput = stdIn.readLine()) != null) {
                if ("exit".equalsIgnoreCase(userInput)) {
                    break;
                }
                out.println(userInput); // Send user input to the
server
```

```
                System.out.println("Echo from server: " +
in.readLine()); // Print server's response
            }
        } catch (UnknownHostException e) {
            System.err.println("Don't know about host " + hostname);
            System.exit(1);
        } catch (IOException e) {
            System.err.println("Couldn't get I/O for the connection to
" + hostname);
            System.exit(1);
        }
    }
}
```

7. Write a C program for generating CRC of a binary string.

crc_generator.c

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

#define MAX_LEN 100

// Function to perform XOR operation
void xorOperation(char *result, const char *divisor, int len) {
    for (int i = 0; i < len; i++) {
        // If bits are the same, result is 0; otherwise, 1
        result[i] = (result[i] == divisor[i]) ? '0' : '1';
    }
}

// Function to generate CRC
void generateCRC(const char *data, const char *generator, char *crc) {
    int dataLen = strlen(data);
    int genLen = strlen(generator);

    // Create a temporary dividend by appending n-1 zeros to data
    char dividend[MAX_LEN];
    strcpy(dividend, data);
    for (int i = 0; i < genLen - 1; i++) {
        strcat(dividend, "0");
    }
    int dividendLen = strlen(dividend);

    // Perform modulo-2 division
    char tempDivisor[MAX_LEN];
    for (int i = 0; i <= dividendLen - genLen; ) {
        if (dividend[i] == '1') {
            // Perform XOR only if the leading bit is 1
            xorOperation(&dividend[i], generator, genLen);
```

```
        }
        i++;
    }

    // The remainder is the last (genLen - 1) bits of the dividend
    strncpy(crc, &dividend[dataLen], genLen - 1);
    crc[genLen - 1] = '\0';
}

int main() {
    char data[MAX_LEN];
    char generator[MAX_LEN];
    char crc[MAX_LEN];

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

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

    // Check for valid binary inputs (simple check)
    for(int i=0; i<strlen(data); i++){
        if(data[i] != '0' && data[i] != '1'){
            printf("Invalid data string.\n");
            return 1;
        }
    }
     for(int i=0; i<strlen(generator); i++){
        if(generator[i] != '0' && generator[i] != '1'){
            printf("Invalid generator string.\n");
            return 1;
        }
    }

    generateCRC(data, generator, crc);

    printf("\nData: %s\n", data);
    printf("Generator: %s\n", generator);
    printf("Generated CRC (Remainder): %s\n", crc);

    // The codeword to be transmitted
    char codeword[MAX_LEN];
    strcpy(codeword, data);
    strcat(codeword, crc);
    printf("Codeword to be transmitted: %s\n", codeword);

    return 0;
}
```