School Of Computer Science

Gujarat University

Certificate

Roll No: **30** Seat No: _____

This is to certify that Mr. /~~Ms.~~ **Rathod Ajinkya** student of MCA Semester **-** IV, has duly completed his / ~~her~~ term work for the semester ending in May 2021, in the subject of **Fundamentals of Networking** towards partial fulfillment of his / ~~her~~ Degree of Masters in Computer Applications.

May 24, 2021
Date of Submission: Internal Faculty

Head of Department

Department Of Computer Science
Rollwala Computer Centre
Gujarat University

MCA – IV

Subject: - Fundamentals of Networking (FON)

Name: - Ajinkya Rathod

Roll No.: - 30 **Exam Seat No.: -**

Sr. No.	Contents	Pg. No	Date	Signature
1	Assignment 1 (Topologies & Devices)		26-03-2021	
2	Assignment 2 (Services & Functions)		26-03-2021	
3	Assignment 3 (Functions and Algorithms)		24-05-2021	
4	Assignment 4 (Network Terminologies)		24-05-2021	

DEPARTMENT OF COMPUTER SCIENCE
ROLLWALA COMPUTER CENTRE
GUJARAT UNIVERSITY
M.C.A. – IV

ROLL NO : **30**
NAME : Ajinkya Rathod
SUBJECT : Fundamentals of Networking (FON)

NO.	TITLE	PAGE NO.	DATE	SIGN
1	**Assignment 1 (TCP & UDP)**		**26-03-2021**	
2	**Assignment 2 (TCP, UDP, Multithreading & Algorithms)**		**24-05-2021**	

Date: / /
Page No.:

Answer in details

Explain different network topologies with its advantages and disadvantages

Ans: the arrangement of a network which compromise of notes and collecting lines were sender and receiver is referred as network topology the various network topologies are

In mesh topology every device connected to another device via a particular channel every device is connected with another via dedicated channel these channel are known as links if suppose a number of devices are connected with each other in mesh technology then total number of ports that is required by each device is n - 1

advantages

It is robust fault
It is diagnosed easily data is reliable because data is transform among the device through dedicated channels all links

It provides security and privacy.

Teacher's Signature

Date : / /
Page No. :

disadvantages

- Installation and configuration is difficult
- Cost of cables are high as bulk writing is required
cost of maintenance is high.

2. star topology

- in Star topology all the devices are connected to a single hub through a cable this hub is central node and all other nodes are connected to the central node the hub can be passive in nature

advantages

- If all devices are connected to each other in Star topology and a number of cables required to connect m is an so it is easy to set up

- Each device requires only one port that is connected to the hub

Problems with this topology

Teacher's Signature

Date : / /
Page No. :

Bus topology

- bus topology is a network type in which every computer and network device is connected to single cable to transmit the data from one end to another in single direction no bi-directional feature is available in bus topology

advantages

- if n devices are connected to each other in bus topology then the number of keywords is equal to one which is known as backbone cable and N chop lines are required

Problem set is this topology

- In the common cable fails then the whole system crashes down if the network traffic is very high it increases collisions in network

Ring Topology

- in ring topology if phone is ordering connecting devices with exactly two neighbouring devices and number of repeaters used for ring topology with a large number of nodes because if someone wants to send some theatre to the last node in the ring topology 100 nodes that the data will have pass 99 nodes hence the prevent of theatre lost repeaters are used in the network

Teacher's Signature

Date : / /
Page No. :

in the connector on which the whole topology relies
failed the whole system crash dump cost for
installation is high performance is based on the
single connector and that is hub.

bus topology

- It is a network type in which every computer
and network device is connected to single cable
to transmit the data from one end to another
in single direction no bi-directional feature is available
in bus topology

advantages

- if n devices are connected to each other in bus
topology then the number of keywords is equal to
one which is known as backbone cable and en chop
lines are required

problem set is thistopology

- in the common cable fails then the whole system
crashes down if the network traffic is very
high it increases collisions in network

Teacher's Signature

Date : / /
Page No. :

ring topology

- In ring topology if phone is of ering connecting devices with exactly two neighbouring devices and number of repeaters used for ring topology with a large number of nodes because if someone wants to send some theatre to the last node in the ring topology 100 nodes that the data will have pass 99 nodes hence the prevent of theatre lost repeaters are used in the network

- The possibility of collision is minimum in this type of topology cheap to install and expand

disadvantages
- Troubleshooting is difficult in this topology
- Addition of statements is between or removal of stations can disturb whole topology

Teacher's Signature

Date : / /

Page No. :

Explain the types of transmission media transmission

- medium is a physical path between the transmitter and the receiver that is the channel through which data is sent from one place to another transmission media is broadly classified into following categories

1. guided media
2. unguided media

Guided media it is also referred to as a wired or bounded transmission media

first twisted pair cable it consists of two separately insulated conductor wires found about each other generally said such pairs are bounded most likely use transmission media

twisted pair is of two types unshielded twisted pair this type of cable has a belated to block interferences does not depend on a physical

Teacher's Signature

Date : / /
Page No. :

shield for this purpose it is used for telephonic.

Advantages

1. least expensive
2. easy to install and
3. high speed capacitor

Disadvantages

- external interface lower capacity
- performance short distance transmission due to attenuation

coaxial cable

it has an outer plastic covering containing two parallel connection each having a separate into insulated protection cover coaxial cable transmits information into two modes basement mode and broadband modem cable TV and analogue communication networks widely used coaxial cables are

the advantages of coaxial cables are hai band with better noise immunity easy to install

Teacher's Signature

Date : / /
Page No. :

and expand and expensive

disadvantages
It is single cable failure can disrupt the entire network

optical fibre cable
it uses the concept of reflection and light through a core made up of glass or plastic tumko is surrounded by less dense glass or plastic covering called the cladding it is used for transmission of large volumes of data.

Advantages
It is of optical fibre cable are
it has increased capacity and bandwidth it is also lightweight and has less signal attention nation it is immunity to electromagnetic interference it is resistance to corrosion of materials

Teacher's Signature

Date : / /
Page No. :

disadvantages
it is difficult to install
and maintain it is very high cost it is a very fragile approach and it is an directional in nature.

Explain OST model in brief

Physical layer
the lowest layer of OSI reference model is the physical layer it is responsible for the actual physical connection between the devices the physical layer contains information in the form of BIT
it is responsible for transmitting individual bits from one node to the next when receiving data the layer will get the signal received and converted into as and send data to data link layer which will put the frame back together the function of physical layer are bit synchronisation with control transmission mode in which synchronisation the physical layer provides the synchronisation of the bridge

Teacher's Signature

Date : / /
Page No. :

by providing a clock the physical layer also defines the transmission rate that is how many bytes for second transmission mode physical layer are define the way which the data flows between the two connected devices the various transmission modes possible her simplex half duplex and full duplex.

2. Data link layer

the data link layer is responsible for the node to node delivery of the message the main function of this layer is to make the use of the data transfer in Adobe form and from one form to another over the physical layer when the packet and arrives in the network it is the responsibility of the dll to transmit in the host2 using m a c address

the packet received from network layer is further divided into 2 subgroups

Teacher's Signature

Date : / /

Page No. :

1. logical link control - LLC

2. media access control m a c

the packet received from network layer is further divided into frames depending on the frame size of LLC also encapsulate sender and receiver IP address in the header

The receivers embassy address is open by placing an aarp request onto the wire asking who has that IP address and the destination host will reply with its m.a.c. address

- The function of the data link layer are framing physical addressing error control flow control access control switch and bridge our data link layer devices.

Teacher's Signature _____

Date : / /

Page No. :

3. Network layer

Network layer vote for the transmission and data from one host to another located in different networks call to take care of packet routing that is selection of the shortest path to transfer the packets from the number of rows available the sender and receiver IP address are placed in the header by the network layer the functions of network layer are routing and logical address.

4. Transport layer

- it provides service application layer and take service from network layer the data in the transport layer is referred as segment it is responsible for the end-to-end delivery of the complete message the transport layer also provide the acknowledgment of the successful data transmission entry transfer the data if the error is bound the function of the transport layer are segmentation and reassembly and service point address

Teacher's Signature _____

Date : / /

Page No. :

Session layer

- this layer is responsible for establishment of connection maintenance patients authentications and also ensure security the function of session layer is session establishment maintenance termination and synchronisation

Presentation layer

- presentation layer is also called the translation layer the data from the application layer is extracted here and manipulated as per the required format to transmit over the network

- Function of presentation layer are translation from ASCII to EBCDIC encryption and decryption compression reduce the number of bits that needs to be transmitted on the network.

Teacher's Signature _____

Date : / /

Page No. :

Application layer

at the very top of the OSI reference model stack of liver be fine application layer which is implemented by the network applications this application procedure the data that has to be transferred over the network the slave also serves as the window for the application services to access the network and for displaying the received information from the user the function of application layer network virtual terminal file transfer axis and management mail services and directory services

Teacher's Signature _____

Date : / /

Page No. :

Explain network application architecture

A network application is an application running on one hos
Android from an occasion to another application running on a different there are two possible structure of application

1 client server architecture
2. peer to peer architecture

in client server architecture there is always host call the server which provide service requested for many to many host call clients in the client server architecture when the client computer science request for data the server through the internet the server accept the request process and delivers the data packet requested to the client do not share any of the resources example client server model are world wide web

Teacher's Signature

Date : / /

Page No. :

Expalin Delay, loss, and throughput in output.

Ans: Network delay is a design and performance characteristics of telecommunication network it specifies the latency for a bit of data to travel across the network from one communication to another

Loss

los packet loss occurs when one or more packets of data traveling across of computer network style to reach the destination packet loss is either caused by errors in data transmission typical across wireless networks or network congestion

Throughput

It is the number of messages successfully transmitted per unit time it is controlled by available bandwidth is available signal-to-noise ratio and hardware limitation the maximum throughput of a network

Teacher's Signature

Date : / /

Page No. :

may be consistently higher than actual throughput achieved in everyday conception throughput is measured by tabulating the amount of data transferred between multiple locations during and specification period of time like bps kpbs to gbps and mbps to gbps.

Teacher's Signature _____

Date : / /

Page No. :

Explain http, POP3, smtp

1. Http

http is a web application layer protocol which defines how web client request with pages from web server and how web server transfer web page to client.

Http uses TCP as its underline transport protocol.

http client first initiate a TCP connection with server once the connection is established the browser and the server process TCP through their socket interface.

2. POP3

post office protocol version 3 is a standard mail protocol used to receive emails from remote server to local incline of three allows you to download message on your local computer and read even when you are offline their advantages of it it provides easy and fast access to the

Teacher's Signature

Date : / /
Page No. :

emails as their local is stored on ABC there is no limit on the size of the email which receivers and it requires less server storage space and all the males are stored on the local machine and it is easy to configure and use.

Disadvantages:

- the demerits of the emails are downloaded rom the server then all the emails are deleted from the server by default and transferring the mail folder from the local machine to another machine can be difficult the email folder which is downloaded from the mail server can also become corrupted

Teacher's Signature

Date : / /

Page No. :

3. smtp

1. Smtp tands for simple mail transfer protocol
2. smtp the set of communication guidelines that allows software to transmit an electronic mail over the internet is called simple mail transfer protocol.
3. it is a program used to send messages to other computer users based on email addresses it provides an e exchange between users on the same and different co and it also supports
4. It can send single message to one or more receptive and sending message can include text voice video or graphics it can also send the messages on network outside the internet
5. The main purpose of Smtp is to setup communication

Teacher's Signature _____

Date : / /

Page No. :

rules between server the server has a way of underlying themselves and announcing what kind of communication they are trying to perform the also have a way of handling the errors that is incorrect email addresses

Teacher's Signature _____

Date : / /

Page No. :

Explain Tcp and udp

TCP

- Connection application process make aur connection before message can be exchange the usage suitable for application that coin higher reliability and transmission time is relatively less critical.

- Reliability guaranteed delivery of application messages with error free and proper data.

- Rearranges data segment in the order specified.

- Segments are acknowledged when received.

UDP

- Application process is exchanges message without creating a connection

- Suitable for application that need fast efficient transmission reliability as critical

Teacher's Signature

Date : / /
Page No. :

multimedia application video online multiplayer games

- The DNS they doesn't guarantee that search the receiving application for the more messages may arise out of order.

- No acknowledgement is seen in UDP.

Explain DNS in detail

An application layer protocol define the application process is running on different system pass the messages on each other DNS stands for domian name system

DNS required for the functioning of the internet ; t node in a tree has a domain name and the ball domain name is a sequence of simple specified by dots

DNS servers to translate the domain to IP

Teacher's Signature

Date : / /
Page No. :

address this allows the user of networks to utilise user-friendly names where are looking for other host instead of remembering the IP address

Q10 Explain peer to peer file distribution in detail

In computer sharing peer-to-peer file sharing technology allowing the users to access mainly the multimedia files like a video of ask ebooks games at sector the individual users in the network refer to the squares the PS request for the file from other pairs were established in TCP or udp connection

new latest how p2p works is p2p network works that p2p network allows computer hardware and software communicate without need of server and client server architecture there is no Central server processing request in p2p the peer directly interacts

Teacher's Signature

Date : / /
Page No. :

with one without requirement of a central server no II peer makes a request it is possible that multiple pairs of the copy of the request object not the problem is how to get the IP address of all these years this is required is decided by the underlying architecture supported by peer-to-peer system by means of one the method the client Kiya can get to know all the pairs have the requested the object file and the transfer takes place directly between those two pairs

Teacher's Signature

Date : / /

Page No. :

Computer network

- It is a system in which multiple computers are connected to each other to share information resources

Unicast

- unicorn refers to 121 transmission from one point in the network to another point that is sender and the receiver is identified by the network address

frequency

- it is the duty ratio of the radio signal to send and receive communication signal measured in hotspot second data used in Wi-Fi communication in 2.4 gigahertz for 5 gz.

Teacher's Signature _____

Date : / /
Page No. :

router

router is a networking device that forwards data packets between computer network router performs directional function on the internet with the traffic

repeater is an electronic device that receives a signal and retransmitted repeaters are used to extend transmission so the signal can cover a longer distance ok and received on the other side of the transaction

switch

- It is a networking hardware that connect devices on a computer network by a packet switching to receive and forward data to destination device

Teacher's Signature _____

Date: / /
Page No.:

Define the following:

1. Unreliable service

The internet network clear protocol has a name IP for internet protocol IP provides logical communication between host the IP service model is a best effort delivery service this means that IP mark is best to deliver segments between communicating hosts but it makes no guarantees it does not guarantee segment delivery it does not guarantee order delivery of segment and it does not get really guaranteed the integrity of the data in the segments for all the reasons it is set to be an unreliable service

2. Logical communication

This means that from an application prospective it is as it is the host running the processes

Date : / /
Page No. :

were directly connected in reality the host may be opposite side of the planet connector while numerous routers and white train of types application processes use.

The logical communication provided by the transparent layer to send message to each other from the worry of the details of physical interest structure used to carry this messages.

2. multiplexing:

The job of gathering data chunks at the source first from different socket and consulating each data chunk with header information that will later be used in b multiplexing to create segments and passing the segments to the network is called multiplexing

3. Demultiplexing

the transport layer examines the set of field in segments to identify the receiving socket

Teacher's Signature _____

Date : / /
Page No. :

and the derivatives the segment to the socket the job of delivering the data in the transport layer segment in the correct socket is called de multiplexing

- reliable data transfer:
For connected oriented service provided by TCP International security to have a reliable data transfer RDT protocol to ensure delivery of all placards n all enable the receivers in order to application their RDT protocol must use pipelining

- This allows the center to have a large number of etiquette in the pipelining

Q2. Answer the following

i. Explain transport layer multiplexing and demultiplexing.

Ans: gathering data from multiple application processes of sender and following that data with handing them as a whole to the internet receiver is called multiplexing.

Teacher's Signature _____

delivering receive segments at receiver side to the correct app player process is called as Demultiplexing

- Prresenting data from a application at center side to an application and destination side sender must know and IP address of destination and part of the number of applications to which he want to transfer the data

- For example let us consider two messaging apps that are widely used nowadays like hike and WhatsApp suppose center and receiver has this application installed in their system suppose A1 to send the message to be in WhatsApp and hike in both order to do so A must mention the IP address of B and destination part number of WhatsApp while sending the message through WhatsApp application similarly for a hike another message from both the apps will be raped by along the appropriate headers and sent to the single message to the receiver this is called multiplexing at the destination

Date : / /
Page No. :

received messages on web and consistent messages are sent to appropriate application by looking at the destination port number.

Q3. Explain UDP segment structure

Ans UDP header is a 8x fixed and simple header while 40cc it may vary from 20 by 60 by 48 by its content all necessary header information and remaining part consists of data UDP port number fields are 8 16-bit long.

Therefore range for port number defined from 0 to 65536 what number 0 is reserved port number helps to distinguish different requests or process.

Teacher's Signature _____

Q4. Explain TCP segment structure

TCP segment consists of database to send to the header edit is added to the database as shown:

- source port address
16-bit fields that hold support address of the application that is sending the data segment

-destination port address
16-bit fields holds the address of the app in the host latest receiving the data segment

-segment number
can you do it filled at whole sequence number that is the bite number of the first bite that is sent in that particular segment it is used to reassemble the message at the receiving and if the segments are received out of order

Date : / /
Page No. :

Window size :

- This field tells the window size of the sending TCP in bytes.

Urgent pointer:

- This field is used to point to data that is urgently required that needs to reach the receiving process the earliest the value of this field is added to the sequence numberto get the white number of the last urgent byte.

QS Expanded principles of reliable data transfer

- Transport layer protocols are central piece of lead architecture this provides the logical communication between application process this process use theological communication to transfer data from transportation network and this transfer of data should be reliable and secure the data is transferred in the form of the package but the problem occurs in relevant transfer of data

Teacher's Signature

Date : / /
Page No. :

The problem of transferring the data differs not only the transport layer but also the application layer as well as in the link layer the problem occurs when reliable services runs as a and reliable service.

For example :
TCP is a reliable data transfer protocol that is implemented top of unreliable rare that is internet protocol is end-to-end network layer protocol.

in this model we have designed a sender and receiver sites for protocol over reliable channel.

reliable transfer of data that receives the data from the above their breakup message in the form of segments and put the header on each segment entrance for below layer receives

Teacher's Signature _____

the segment and remove the header from each segment and make it packet by adding to header.

- The data which is transferred from the above has no transfer data with corrupted or loss and are delivered in the same sequence in which they were sent to the below layer is reliable data transfer protocol.
- The service model is offered by TCP to internet application that in this transfer of data.

Teacher's Signature

Date : / /
Page No. :

Q6
Explain level data transfer protocol in detail
Same as answer 5.

Q7. Explain go back N protocol
Go back n protocol also called us go back automatic repeat frequencies is a data link there protocol that uses a sliding window method bar reliable and subsequent of data from citizen case of sliding window protocol having ascent window size of an and receiving window size of one.

The sequence number has a number as modulo for example if the sending window sizes for than the sequence number will be 0 1 2 3 0 1 2 3 0 1 and so on

The number of between the sequence number 1 5 2 to generate binary sequence
0 0 0 0 1 1 0 1 1
The size of the receiving window is 1

Teacher's Signature

Date : / /
Page No. :

Q8. Explain case study of talent for sequence and acknowledgment number.

Ans: suppose who stay initiated for Taylor session with hosts a initiatives the session it is able the client and host B labeled the server each character tied by the user will be sent to the remote hose the whole remote host will send back a copy of character which will be displayed on the tennis users when the eco bank is used the insurance with the characters in by the talent user have already been resident to the remote side each character that travels the network twice the time character is displayed on the users monitor.

The second segment is sent from the server to client itself so dual purpose it is an acknowledgment of the data the server is received by putting as an acknowledgment filter server is telling the client that it is successful received everything up through

Teacher's Signature _____

Date : / /
Page No. :

byte and is now waiting or the white onwards
the third segment is sent from the client to the
server
The soul purpose is to acknowledge the data is
received the server the segment has an empty
data this segment has in the acknowledgment
number because the client has received the
stream of byte through byte sequence number 79
and it is now waiting byte 80 onwards

Teacher's Signature

Date: / /

Page No.:

Define the following

1. NAT

Network Address Translationis designed for IP address conservation. It enables private IP networks that use unregistered IP addresses to connect to the Internet. NAT operates on a router, usually connecting two networks together, and translates the private not globally unique addresses in the internal network into legal addresses, before packets are forwarded to another network

2. FDMA

Frequency-division multiple access is a channel access method used in some multiple-access protocols. allows multiple users to send data through a single communication channel, such as a coaxial cable or microwave beam, by dividing the bandwidth of the channel into separate non-overlapping frequency sub-channels and allocating each sub-channel to a separate user. Users can send data through a subchannel by modulating it on a carrier wave at the subchannels frequency. It is used in satellite communication systems and telephone trunklines.

Sig

Teacher's Signature _____

Date: / /

Page No.:

VLAN

VLAN is a custom network which is created from one or more local area networks. It enables a group of devices available in multiple networks to be combined into one logical network. The result becomes a virtual LAN that is administered like a physical LAN. The full form of VLAN is defined as Virtual Local Area Network.

4. MPLS

The thing about JPLS is that its a technique, not a service so it can deliver anything from IP VPNs to metro Ethernet. Its expensive, so with the advent of SD-WAN enterprises are trying to figure how to optimize its use vs. less expensive connections like the internet.

Sig

Teacher's Signature _____

Date : / /

Page No. :

Q Explain in detail

1. Explain Link state routing algoeithm

Link state routing is a technique in which each router shares the knowledge of its neighborhood with every other router in the internetwork.

Step-1: The node is taken and chosen as a root node of the tree, this creates the tree with a single node, and now set the total cost of each node to some value based on the information in Link State Database

Step-2: Now the node selects one node, among all the nodes not in the tree like structure, which is nearest to the root, and adds this to the tree. The shape of the tree gets changed .

Step-3: After this node is added to the tree, the cost of all the nodes not in the tree needs to be updated because the paths may have been changed.

Step-4: The node repeats the Step 2. and Step 3. until all the nodes are added in the tree

Teacher's Signature _____

Date : / /
Page No. :

2. Explain dv routing algorithm

A distance-vector routing DVR protocol requires that a router inform its neighbors of topology changes periodically. Historically known as the old ARPANET routing algorithm or known as Bellman-Ford algorithm.

Bellman Ford Basics - Each router maintains a Distance Vector table containing the distance between itself and ALL possible destination nodes. Distances, based on a chosen metric, are computed using information from the neighbors distance vectors.

A router transmits its distance vector to each of its neighbors in a routing packet.

Each router receives and saves the most recently received distance vector from each of its neighbors.

A router recalculates its distance vector when:

It receives a distance vector from a neighbor containing different information than before.

It discovers that a link to a neighbor has gone down.

Teacher's Signature _____

Date : / /

Page No. :

From time-to-time, each node sends its own distance vector estimate to neighbors.

When a node x receives new DV estimate from any neighbor v, it saves v's distance vector and it updates its own DV using B-F equation

4. ARP and RARP

ARP stands for Address Resolution Protocol.

Whereas RARP stands for Reverse Address Resolution Protocol.

Through ARP, 32-bit IP address mapped into 48-bit MAC address.

Whereas through RARP, 48-bit MAC address of 48 bits mapped into 32-bit IP address.

In ARP, broadcast MAC address is used.

While in RARP, broadcast IP address is used.

Teacher's Signature _____

Date: / /

Page No.:

- In ARP, ARP table is managed or maintained by local host.

While in RARP, RARP table is managed or maintained by RARP server.

- In Address Resolution Protocol, Receiver's JAC address is fetched.

While in RARP, IP address is fetched.

- In ARP, ARP table uses ARP reply for its updation.

While in RARP, RARP table uses RARP reply for configuration of IP addresses.

- Hosts and routers uses ARP for knowing the JAC address of other hosts and routers in the networks.

While RARP is used by small users having less facilities.

Sig

Teacher's Signature _____

Date : / /
Page No. :

Q. Explain Pakcet Switching Algo

Packet switching allows delivery of variable bit rate data streams, realized as sequences of packets, over a computer network which allocates transmission resources as needed using statistical multiplexing or dynamic bandwidth allocation techniques. As they traverse networking hardware, such as switches and routers, packets are received, buffered, queued, and retransmitted (stored and forwarded), resulting in variable latency and throughput depending on the link capacity and the traffic load on the network. Packets are normally forwarded by intermediate network nodes asynchronously using first-in, first-out buffering, but may be forwarded according to some scheduling discipline for fair queuing, traffic shaping, or for differentiated or guaranteed quality of service, such as weighted fair queuing or leaky bucket.

Packet-based communication may be implemented with or without intermediate forwarding nodes switches and routers. In case of a shared physical medium such as radio or 10BASE5, the packets may be delivered according to a multiple access scheme.

Teacher's Signature _____

Date : / /
Page No. :

Packet switching contrasts with another principal networking paradigm, circuit switching, a method which pre-allocates dedicated network bandwidth specifically for each communication session, each having a constant bit rate and latency between nodes. In cases of billable services, such as cellular communication services, circuit switching is characterized by a fee per unit of connection time, even when no data is transferred, while packet switching may be characterized by a fee per unit of information transmitted, such as characters, packets, or messages.

Q. What is Network layer and Data Link Layer ?

The network layer

Located at Layer 3 of the Open Systems Interconnection OSI communications model, the primary function of the network layer is to move data into and through other networks. Network layer protocols accomplish this goal by packaging data with correct network address information, selecting

Teacher's Signature _____

the appropriate network routes and forwarding the packaged data up the stack to the transport layer Layer 4.

Data Link Layer

Data Link Layer is second layer of OSI Layered Jodel. This layer is one of the most complicated layers and has complex functionalities and liabilities. Data link layer hides the details of underlying hardware and represents itself to upper layer as the medium to communicate.

Data link layer works between two hosts which are directly connected in some sense. This direct connection could be point to point or broadcast. Systems on broadcast network are said to be on same link. The work of data link layer tends to get more complex when it is dealing with multiple hosts on single collision domain.

Data link layer is responsible for converting data stream to signals bit by bit and to send that over the underlying hardware.

Teacher's Signature _____

Eng

Date : / /
Page No. :

At the receiving end, Data link layer picks up data from hardware which are in the form of electrical signals, assembles them in a recognizable frame format, and hands over to upper layer.

Q. Error Detection and Correction Techniques

Ans: Error is a condition when the output information does not match with the input information. During transmission, digital signals suffer from noise that can introduce errors in the binary bits travelling from one system to other. That means a 0 bit may change to 1 or a 1 bit may change to 0.

Whenever a message is transmitted, it may get scrambled by noise or data may get corrupted. To avoid this, we use error-detecting codes which are additional data added to a given digital message to help us detect if an error occurred during transmission of the message. A simple example of error-detecting code is parity check.

Teacher's Signature _____

Date : / /
Page No. :

Along with error-detecting code, we can also pass some data to figure out the original message from the corrupt message that we received. This type of code is called an error-correcting code. Error-correcting codes also deploy the same strategy as error-detecting codes but additionally, such codes also detect the exact location of the corrupt bit.

In error-correcting codes, parity check has a simple way to detect errors along with a sophisticated mechanism to determine the corrupt bit location. Once the corrupt bit is located, its value is reverted (from 0 to 1 or 1 to 0) to get the original message.

It is the simplest technique for detecting and correcting errors. The JSB of an 8-bits word is used as the parity bit and the remaining 7 bits are used as data or message bits. The parity of 8-bits transmitted word can be either even parity or odd parity.

Teacher's Signature _____

Date : / /

Page No. :

Parity checking at the receiver can detect the presence of an error if the parity of the receiver signal is different from the expected parity. That means, if it is known that the parity of the transmitted signal is always going to be even and if the received signal has an odd parity, then the receiver can conclude that the received signal is not correct. If an error is detected, then the receiver will ignore the received byte and request for retransmission of the same byte to the transmitter.

2 types are:

Even parity - Even parity means the number of 1s in the given word including the parity bit should be even 2,4,6,... .... .

Odd parity - Odd parity means the number of 1s in the given word including the parity bit should be odd 1,3,5,. . . .

Teacher's Signature _____

Date: / /

Page No.:

Q7. What is Random Access protocol

Random Access Protocols is a Jultiple access protocol that is divided into four categories which are ALOHA, CSMA, CSMA/CD, and CSMA/CA. In this article, we will cover all of these Random Access Protocols in detail.

The random access protocols consist of the following characteristics:

1. There is no time restriction for sending the data - you can talk to your friend without a time restriction.

2. There is a fixed sequence of stations which are transmitting the data.

As in the above diagram you might have observed that the random-access protocol is further divided into four categories, which are:

ALOHA Random Access Protocol

Teacher's Signature _____

Date : / /
Page No. :

The ALOHA protocol or also known as the ALOHA method is a simple communication scheme in which every transmitting station or source in a network will send the data whenever a frame is available for transmission. If we succeed and the frame reaches its destination, then the next frame is lined-up for transmission. But remember, if the data frame is not received by the receiver maybe due to collision then the frame is sent again until it successfully reaches the receiver's end.

Whenever we talk about a wireless broadcast system or a half-duplex two-way link, the ALOHA method works efficiently. But as the network becomes more and more complex e.g. the ethernet. Now here in the ethernet, the system involves multiple sources and destinations which share a common data path or channel, then the conflict occurs because data-frames collide, and the information is lost. Following is the flow chart of Pure ALOHA.

Teacher's Signature _____

Date : / /

Page No. :

CSMA Random Access Protocol

CSMA stands for Carrier Sense Jultiple Access. Till now we have understood that when 2 or more stations start sending data, then a collision occurs, so this CSMA method was developed to decrease the chances of collisions when 2 or more stations start sending their signals over the data link layer. But how do they do it? It makes each station to first check the medium (whether it is busy or not) before sending any data packet.

CSMA/CD means CSMA with Collision Detection.

In this, whenever station transmits data-frame it then monitors the channel or the medium to acknowledge the state of the transmission i.e. successfully transmitted or failed. If the transmission succeeds, then it prepares for the next frame otherwise it resends the previously failed data-frame. The point to remember here is, that the frame transmission time should be at least twice the

Teacher's Signature _____

Date : / /
Page No. :

maximum propagation time, which can be deduced when the distance between the two stations involved in a collision is maximum.

CSMA/CA Random Access Protocol

It means CSMA with collision avoidance.

To detect the possible collisions, the sender receives the acknowledgement and if there is only one acknowledgment present its own then this means that the data-frame has been sent successfully. But, if there are 2 or more acknowledgment signals then this indicates that the collision has occurred.

Teacher's Signature _____

Ery

Date : / /

Page No. :

Q 8 - Explain Ethernet, Switch and VLAN

1. Ethernet is a way of connecting computers together in a local area network or LAN. It has been the most widely used method of linking computers together in LANs since the 1990s. The basic idea of its design is that multiple computers have access to it and can send data at any time.

2. A switch is a device in a computer network that connects other devices together. Multiple data cables are plugged into a switch to enable communication between different networked devices. ... Switches may also operate at higher layers of the OSI model, including the network layer and above.

3. VLAN is a custom network which is created from one or more local area networks. It enables a group of devices available in multiple networks to be combined into one logical network. The result becomes a virtual LAN that is administered like a physical LAN. The full form of VLAN is defined as Virtual Local Area Network

Sig

Teacher's Signature _____

Date : / /
Page No. :

What is Wireless Network?

Ans: Wireless networks are computer networks that are not connected by cables of any kind. The use of a wireless network enables enterprises to avoid the costly process of introducing cables into buildings or as a connection between different equipment locations. The basis of wireless systems are radio waves, an implementation that takes place at the physical level of network structure.

Wireless networks use radio waves to connect devices such as laptops to the Internet, the business network and applications. When laptops are connected to Wi-Fi hot spots in public places, the connection is established to that business's wireless network.

2. Explain Cdma in Wireless technology

Teacher's Signature _____

Date : / /

Page No. :

Code Division Multiple Access CDMA is a sort of multiplexing that facilitates various signals to occupy a single transmission channel. It optimizes the use of available bandwidth. The technology is commonly used in ultra-high-frequency UHF cellular telephone systems, bands ranging between the 800-MHz and 1.9-Ghz.

Code Division Multiple Access system is very different from time and frequency multiplexing. In this system, a user has access to the whole bandwidth for the entire duration. The basic principle is that different CDMA codes are used to distinguish among the different users.

Techniques generally used are direct sequence spread spectrum modulation DS-CDMA, frequency hopping or mixed CDMA detection MDCDMA. Here, a signal is generated which extends over a wide bandwidth. A code called spreading code is used to perform this action. Using a group of codes, which are orthogonal to each other, it is possible to select a signal with a given code in the presence of many other signals with different orthogonal codes.

By

Teacher's Signature _____

Date : / /

Page No. :

3. What is PAN

A personal area network PAN is a computer network for interconnecting electronic devices centered on an individual persons workspace. A PAN provides data transmission among devices such as computers, smartphones, tablets and personal digital assistants. PANs can be used for communication among the personal devices themselves, or for connecting to a higher level network and the Internet where one master device takes up the role as gateway. A PAN may be wireless or carried over wired interfaces such as USB.

Body Area Network

It is a mobile network that moves with a persona range for example when person connects his smart phone to the Bluetooth headphone and moves in the market that refers to a body area network.

Offline Network

In this multiple devices are connected through

Teacher's Signature _____

Date : / /
Page No. :

Bluetooth or Wi-Fi. The devices attached to your computer including printers, mouse, speakers, and other appliances are integrated using a Personal Area Network PAN and do not use internet. So a communication network is formed between the devices used in a small single space for example home.

Home Office :

In Home Office setup a separate smaller network is setup for work purpose which is separate from the network used by other home appliances. This network works as a separate body with multiple other devices connected for office work purpose.

Merits:

PAN is relatively flexible and provides high efficiency for short network range.

It needs easy setup and relatively low cost.

It does not require frequent installations and maintenance

Teacher's Signature _____

Date : / /
Page No. :

3g and 4g architecture

3G:

Umts Universal Jobile Telecommunications System 1.
Umts, short for Universal Jobile Telecommunications System, is a 3G networking standard used throughout much of the world as an upgrade to existing Gsm module.

Umts makes use of Wcdma, a technology that shares much with Cdma networks used throughout the world, though it is not compatible with them.

Base level Umts networks are generally capable of downlink speeds as 384 kbps.

The Umts architecture takes advantage of the existing GSJ and GPRS networks which serve as a core network in Umts infrastructure.

4G:

LTE-Uu interface has:

Teacher's Signature _____

Eng

Date: / /
Page No.:

PDCP: The PDCP protocol supports efficient transport of IP packets over the radio link. It performs header compression, Access Stratum security ciphering and integrity protection and packet re-ordering or retransmission during handover.

RLC: In the transmitting side, the RLC protocol constructs RLC PDU and provides the RLC PDU to the mAC layer. The RLC protocol performs segmentation or concatenation of PDCP PDUs during construction of the RLC PDU. In the receiving side, the RLC protocol performs reassembly of the RLC PDU to reconstruct the PDCP PDU. The RLC protocol has three operational modes i.e. transparent mode, acknowledged mode and unacknowledged mode, and each offers different reliability levels. It also performs packet the RLC PDU re-ordering and retransmission.

Sig

Teacher's Signature _____

Date: / /

Page No.:

Q. what is mobility in wireless technology. Explain Direct and indirect approaches.

The uniqueness of wireless networks is their ability to provide dynamic network connectivity for users, devices, services, and applications without being tethered to any wired hardware. As such, wireless technology is expected to cater to different types of mobility characteristics. The following section details the different types of mobility that wireless networks are subjected to, and their characteristics.

A direct routing approach is taken in routing telephone calls to mobile users inseveral mobile telephone network standards, including GSM Lin 2001. An exten-sion of the mobile IP standard to include direct routing is also under consideration

In the indirect routing approach, the correspondent simply addresses the datagram to the mobile node's permanent address, and sends the datagram into the network, blissfully unaware of whether the mobile

Teacher's Signature _____

Date : / /
Page No. :

node is resident in its home network or is visiting a foreign network.

Teacher's Signature _____

Eng

Date : / /

Page No. :

Answer in Short:

1. What is base station

A base station is a radio receiver which may have one or multiple antenna. It was first used in mobile telecommunication networks. The base station is responsible for maintaining communication between the network and the users, and also among users. The equipment works with a mobile switching station which is responsible for connecting cellular calls to the public switched telephone network PSTN. The geographical area covered by a base station is referred to as a cell. A single base station broadly extends the reach of a telecommunication company.

A base station is company specific, but competing telecommunication companies can have their individual base stations on a physical site.

Teacher's Signature _____

Date : / /
Page No. :

COA

A care-of address usually referred to as CoA is a temporary IP address for a mobile device used in Internet routing. This allows a home agent to forward messages to the mobile device. A separate address is required because the IP address of the device that is used as host identification is topologically incorrect - it does not match the network of attachment. The care-of address splits the dual nature of an IP address, that is, its use is to identify the host and the location within the global IP network.

Define following terms :

Home Network:

A home network or home area network (HAN) is a type of computer network that facilitates communication among devices within the close vicinity of a home.

Teacher's Signature _____

Date : / /

Page No. :

Visited Network:

In more technical terms, roaming refers to the ability for a cellular customer to automatically make and receive voice calls, send and receive data, or access other services, including home data services, when travelling outside the geographical coverage area of the home network, by means of using a visited network.

Home Agent:

A home agent is a router on a mobile node's home network which tunnels datagrams for delivery to the mobile node when it is away from home. ... A foreign agent is a router that stores information about mobile nodes visiting its network. Foreign agents also advertise care-of-addresses which are used by mobile IP.

Foreign Agent:

A foreign agent is any person or entity actively carrying out the interests of a foreign country while

Teacher's Signature _____

Date : / /

Page No. :

located in another host country, generally outside the protections offered to those working in their official capacity for a diplomatic mission. Foreign agents may be citizens of the host country.

Teacher's Signature _____

By

Practical - I

```
/* ============================================================
 *
 * Roll No: 30
 *
 * File:      Client10.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

import java.io.*;
import java.net.*;
import java.nio.ByteBuffer;
import java.util.Arrays;

public class Client10 {
  private static String source = "1";
  private static String destination = "2";

  public static void main(String args[]) throws Exception {
    String sentence;
    BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
    Socket clientSocket = new Socket(InetAddress.getLocalHost(), 6789);
    DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
    DataInputStream inFromServer = new
DataInputStream(clientSocket.getInputStream());

    sentence = inFromUser.readLine();

    byte[] sourceAddress = Arrays.copyOf(source.getBytes(), 10);
    byte[] destinationAddress = Arrays.copyOf(destination.getBytes(), 10);
    byte[] data = Arrays.copyOf(sentence.getBytes(), 50);
    byte[] packet =
ByteBuffer.allocate(70).put(sourceAddress).put(destinationAddress).put(data).arr
ay();

    outToServer.write(packet);

    clientSocket.close();
  }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      ClientHandler.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
```

```
 * ========================================================== */

 package TCP;

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

public class ClientHandler extends Thread {
    private final Socket socket;
    private final TCP7Server server;
    private PrintWriter writer;
    public String username;

    public ClientHandler(Socket socket, TCP7Server server) {
        this.socket = socket;
        this.server = server;
    }

    @Override
    public void run() {
        try {
            BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            writer = new PrintWriter(socket.getOutputStream(), true);
            username = reader.readLine();
            server.addUser(username, this);
            server.listUsers();
            String serverMessage = "New User Connected: " + username;
            server.sendToEveryone(serverMessage, this);
            String clientMessage;
            do {
                clientMessage = reader.readLine().trim();
                serverMessage = "From " + username + ": " + clientMessage;
                System.out.println(serverMessage);
                server.sendToEveryone(serverMessage, this);
            } while (!clientMessage.equalsIgnoreCase("bye"));
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            try {
                server.removeUser(username);
                socket.close();
                String serverMessage = username + " disconnected...";
                server.sendToEveryone(serverMessage, this);
            } catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
    }

    void sendMessage(String message) {
```

```
        writer.println(message);
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      MainServer10.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.*;
import java.net.*;
import java.nio.ByteBuffer;
import java.util.Arrays;

class ms {
    public static void main(String args[]) throws Exception {
        ServerSocket welcomeSocket = new ServerSocket(9999);

        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            DataInputStream inFromClient = new
DataInputStream(connectionSocket.getInputStream());
            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());

            byte[] clientData = new byte[70];
            inFromClient.read(clientData);

            String sSourceAddress = new String(Arrays.copyOfRange(clientData, 0,
10));
            String sDestinationAddress = new
String(Arrays.copyOfRange(clientData, 10, 20));
            String receivedData = new String(Arrays.copyOfRange(clientData, 20,
70));

            System.out.println("-- From Client --");
            System.out.println("Sender Source Address: " + sSourceAddress);
            System.out.println("Sender Destination Address: " +
sDestinationAddress);
            System.out.println("Data: " + receivedData);
            break;
        }
    }
}

 /* ============================================================
 *
 * Roll No: 30
```

```
 *
 * File:      Server10.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ========================================================== */

 import java.io.*;
import java.net.*;
import java.nio.ByteBuffer;
import java.util.Arrays;


public class Server10 {
    private static String source = "2";
    private static String destination = "3";

    public static void main(String args[]) throws Exception {
        ServerSocket welcomeSocket = new ServerSocket(6789);

        String sSourceAddress;
        String sDestinationAddress;
        String receivedData;
        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            DataInputStream inFromClient = new
DataInputStream(connectionSocket.getInputStream());
            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());

            byte[] clientData = new byte[70];
            inFromClient.read(clientData);

            sSourceAddress = new String(Arrays.copyOfRange(clientData, 0, 10));
            sDestinationAddress = new String(Arrays.copyOfRange(clientData, 10,
20));
            receivedData = new String(Arrays.copyOfRange(clientData, 20, 70));

            System.out.println("-- From Client --");
            System.out.println("Sender Source Address: " + sSourceAddress);
            System.out.println("Sender Destination Address: " +
sDestinationAddress);
            System.out.println("Data: " + receivedData);
            break;
        }
        // Server sends to server
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
        Socket clientSocket = new Socket(InetAddress.getLocalHost(), 9999);

        DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
```

```
        byte[] sourceAddress = Arrays.copyOf(sSourceAddress.getBytes(), 10);
        byte[] destinationAddress =
Arrays.copyOf(sDestinationAddress.getBytes(), 10);
        byte[] data = Arrays.copyOf(receivedData.getBytes(), 50);
        byte[] packet =
ByteBuffer.allocate(70).put(sourceAddress).put(destinationAddress).put(data).arr
ay();

        outToServer.write(packet);
        clientSocket.close();
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP1Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP1Client {

    public static void main(String argv[]) throws Exception {
        String sentence;
        while (true) {
            Socket clientSocket = new Socket("localhost", 6789);
            BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
            DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream())
            sentence = inFromUser.readLine();
            outToServer.writeBytes(sentence + '\n');
            if (sentence.equalsIgnoreCase("exit") ||
sentence.equalsIgnoreCase("bye")) {
                clientSocket.close();
                break;
            }
        }
    }
}

 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP1Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
```

```
 *
 * ============================================================ */

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

public class TCP1Server {

    public static void main(String argv[]) throws Exception {
        String clientSentence;
        ServerSocket welcomeSocket = new ServerSocket(6789);
        System.out.println("Searching Client.....");
        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            try (BufferedReader inFromClient = new BufferedReader(
                    new InputStreamReader(connectionSocket.getInputStream()));)
{
                clientSentence = inFromClient.readLine();
                System.out.println("Client String = " + clientSentence);
            }
            if (clientSentence.equalsIgnoreCase("exit") ||
clientSentence.equalsIgnoreCase("bye")) {
                connectionSocket.close();
                break;
            }
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP2Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP2Client {

    public static void main(String argv[]) throws Exception {
        String sentence;
        String upper;

        try (BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
                Socket clientSocket = new Socket(InetAddress.getLocalHost(),
6789);
```

```
                DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
                BufferedReader inFromServer = new BufferedReader(
                        new InputStreamReader(clientSocket.getInputStream()));)
{
            sentence = inFromUser.readLine();

            outToServer.writeBytes(sentence + '\n');

            upper = inFromServer.readLine();

            System.out.println("From Server: " + upper);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP2Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP2Server {

    public static void main(String argv[]) throws Exception {
        String clientSentence;

        ServerSocket welcomeSocket = new ServerSocket(6789);

        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            try (BufferedReader inFromClient = new BufferedReader(
                    new InputStreamReader(connectionSocket.getInputStream()));
                    DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());) {
                clientSentence = inFromClient.readLine();
                String upperCase = clientSentence.toUpperCase();
                System.out.println("Client Request String = " + clientSentence +
" | Response = " + upperCase);
                outToClient.writeBytes(upperCase);
            }
        }
    }
}
 /* ============================================================
 *
```

```
 * Roll No: 30
 *
 * File:      TCP3Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP3Client {

    public static void main(String argv[]) throws Exception {
        String sentence;
        int length;

        try (BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
                Socket clientSocket = new Socket(InetAddress.getLocalHost(),
6789);
                DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
                BufferedReader inFromServer = new BufferedReader(
                        new InputStreamReader(clientSocket.getInputStream()));)
{
            sentence = inFromUser.readLine();

            outToServer.writeBytes(sentence + '\n');

            length = inFromServer.read();

            System.out.println("From Server: " + length);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP3Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP3Server {

    public static void main(String argv[]) throws Exception {
        String clientSentence;
```

```
        int length;

        ServerSocket welcomeSocket = new ServerSocket(6789);

        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            try (BufferedReader inFromClient = new BufferedReader(
                    new InputStreamReader(connectionSocket.getInputStream()));
                    DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());) {
                clientSentence = inFromClient.readLine();
                length = clientSentence.length();
                System.out.println("Client Request String = " + clientSentence +
" | Response = " + length);
                outToClient.writeByte(length);
            }
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP4Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP4Client {

    public static void main(String argv[]) throws Exception {
        String sentence;
        String wish;
        System.out.println("Enter Time as per 24 Hours Format (eg 17:50)");
        try (BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
                Socket clientSocket = new Socket(InetAddress.getLocalHost(),
6789);
                DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
                BufferedReader inFromServer = new BufferedReader(
                        new InputStreamReader(clientSocket.getInputStream()));)
{
            sentence = inFromUser.readLine();

            outToServer.writeBytes(sentence + '\n');

            wish = inFromServer.readLine();
```

```
            System.out.println("From Server: " + wish);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP4Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP4Server {

    public static void main(String argv[]) throws Exception {
        String clientSentence;
        int time = 0;
        int minute = 0;
        ServerSocket welcomeSocket = new ServerSocket(6789);

        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            try (BufferedReader inFromClient = new BufferedReader(
                    new InputStreamReader(connectionSocket.getInputStream()));
                    DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());) {
                clientSentence = inFromClient.readLine();
                try {
                    time = Integer.parseInt(clientSentence.substring(0,
clientSentence.indexOf(":")));
                    minute = Integer.parseInt(
                            clientSentence.substring(clientSentence.indexOf(":")
+ 1, clientSentence.length()));

                    if ((time < 0 || time > 23) || (minute > 59 || minute < 0))
                        outToClient.writeBytes("Invalid Time");
                    else if (time > 19)
                        outToClient.writeBytes("Good Night");
                    else if (time > 15)
                        outToClient.writeBytes("Good Evening");
                    else if (time > 11)
                        outToClient.writeBytes("Good Afernoon");
                    else if (time > 4)
                        outToClient.writeBytes("Good Morning");
                    else
                        outToClient.writeBytes("Good MidNight");
```

```
                } catch (Exception e) {
                    outToClient.writeBytes("Invalid Time");
                }
                // System.out.println("Client Request String = " +
clientSentence + " | Response
                // = " + length);
            }
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP5Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP5Client {

    public static void main(String argv[]) throws Exception {
        String sentence;
        String quote;

        try (BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
                Socket clientSocket = new Socket(InetAddress.getLocalHost(),
1235);
                DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
                BufferedReader inFromServer = new BufferedReader(
                        new InputStreamReader(clientSocket.getInputStream()));)
{
            sentence = inFromUser.readLine();
            outToServer.writeBytes(sentence + '\n');
            quote = inFromServer.readLine();
            System.out.println("From Server: " + quote);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP5Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
```

```
 *
 * ========================================================== */

 import java.io.*;
import java.net.*;
import java.lang.Math;

public class TCP5Server {

    public static void main(String argv[]) throws Exception {
        String clientSentence;
        int random;
        String[] quotes = { "You are nice Person.", "You are good.", "You are
lucky.", "You have good choice.",
                "You can do it.", "You are Awesome.", "You are Smart.", "You are
my friend.", "You are stupid.",
                "Lucky to have you here." };

        ServerSocket welcomeSocket = new ServerSocket(1235);

        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            try (BufferedReader inFromClient = new BufferedReader(
                    new InputStreamReader(connectionSocket.getInputStream()));
                    DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());) {
                clientSentence = inFromClient.readLine();
                if (clientSentence.equalsIgnoreCase("hello")) {
                    random = (int) (Math.random() * 10);
                    outToClient.writeBytes(quotes[random]);
                } else {
                    outToClient.writeBytes("Hi, say hello for quotes.");
                }
            }
        }
    }
}
 /* ==========================================================
 *
 * Roll No: 30
 *
 * File:      TCP6Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ========================================================== */

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

public class TCP6Client {
```

```
    public static void main(String argv[]) throws Exception {
        String sentence;
        String age;

        try (BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
                Socket clientSocket = new Socket(InetAddress.getLocalHost(),
6789);
                DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
                BufferedReader inFromServer = new BufferedReader(
                        new InputStreamReader(clientSocket.getInputStream()));)
{
            System.out.print("Enter Birth Date (eg : day/month/year) : ");
            sentence = inFromUser.readLine();
            outToServer.writeBytes(sentence + '\n');
            age = inFromServer.readLine();
            System.out.println("From Server: " + age);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP6Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.*;
import java.net.*;
import java.util.Calendar;

public class TCP6Server {

    public static void main(String argv[]) throws Exception {
        String clientSentence;
        int year, month, day;

        ServerSocket welcomeSocket = new ServerSocket(6789);

        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            try (BufferedReader inFromClient = new BufferedReader(
                    new InputStreamReader(connectionSocket.getInputStream()));
                    DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());) {
                try {
                    clientSentence = inFromClient.readLine();
```

```
                    day = Integer.parseInt(clientSentence.substring(0,
clientSentence.indexOf("/")));
                    month = Integer.parseInt(
                            clientSentence.substring(clientSentence.indexOf("/")
+ 1, clientSentence.lastIndexOf("/")));
                    year = Integer.parseInt(

clientSentence.substring(clientSentence.lastIndexOf("/") + 1,
clientSentence.length()));

                    Calendar cal = Calendar.getInstance();
                    int tday = cal.get(Calendar.DAY_OF_MONTH);
                    int tmonth = cal.get(Calendar.MONTH) + 1;
                    int tyear = cal.get(Calendar.YEAR);

                    GetAge gage = new GetAge();

                    outToClient.writeBytes(gage.age(tday, tmonth, tyear, day,
month, year));

                } catch (Exception e) {
                    outToClient.writeBytes("Invalid Dates");
                }
            }
        }
    }
}

class GetAge {
    private int finalyear;
    private int finalmonth;
    private int finaldays;

    public String age(int tday, int tmonth, int tyear, int day, int month, int
year) {
        finalyear = tyear - year - 1;
        finaldays = tday;
        if (tmonth >= month && tday >= day) {
            finalyear++;
            finalmonth = tmonth - month;
            finaldays = tday - day;
        } else {
            finalmonth = ((12 - month) + tmonth) - 1;
        }
        if (finalmonth == 0 && finaldays == 0)
            return "Happy Birthday your Age = " + finalyear + " years " +
finalmonth + " months " + finaldays
                    + " days ";
        else
            return "Age = " + finalyear + " years " + finalmonth + " months " +
finaldays + " days ";
```

```
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP7Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP7Client {
    private final String host;
    private final int port;
    private String username;

    public TCP7Client(String host, int port) {
        this.host = host;
        this.port = port;
    }

    public static void main(String[] args) {
        TCP7Client client = new TCP7Client("localhost", 6789);
        client.start();
    }

    public void start() {
        try {
            Socket socket = new Socket(host, port);
            System.out.println("Connected to the server...");
            var writeHandler = new TCP7ClientWriter(socket, this);
            var readHandler = new TCP7ClientReader(socket, this);
            writeHandler.start();
            readHandler.start();
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    void setUserName(String username) {
        this.username = username;
    }

    String getUsername() {
        return this.username;
    }
}
 /* ============================================================
```

```
 *
 * Roll No: 30
 *
 * File:      TCP7ClientReader.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ========================================================== */

 package TCP;

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

public class TCP7ClientReader extends Thread {
    private BufferedReader reader;
    private final Socket socket;
    private final TCP7Client client;

    public TCP7ClientReader(Socket socket, TCP7Client client) {
        this.socket = socket;
        this.client = client;
        try {
            reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    @Override
    public void run() {
        while (true) {
            try {
                String response = reader.readLine();
                System.out.println("\n" + response);
                if (client.getUsername() != null) {
                    System.out.print("From " + client.getUsername() + ":");
                }
            } catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
                break;
            }
        }
    }
}
 /* ==========================================================
 *
 * Roll No: 30
 *
 * File:      TCP7ClientWriter.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
```

```
 *
 * ========================================================== */

 package TCP;

import java.io.*;
import java.net.*;
import java.util.Scanner;

public class TCP7ClientWriter extends Thread {
    private PrintWriter writer;
    private final Socket socket;
    private final TCP7Client client;

    public TCP7ClientWriter(Socket socket, TCP7Client client) {
        this.socket = socket;
        this.client = client;
        try {
            OutputStream output = socket.getOutputStream();
            writer = new PrintWriter(output, true);
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    @Override
    public void run() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter your name: ");
        String userName = sc.nextLine();
        client.setUserName(userName);
        writer.println(userName);
        String text;
        do {
            System.out.print("From " + userName + ": ");
            text = sc.nextLine();
            writer.println(text);
        } while (!text.equalsIgnoreCase("bye"));
        try {
            socket.close();
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
} /* ==========================================================
 *
 * Roll No: 30
 *
 * File:      TCP7Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
```

```
 * ========================================================== */

 package TCP;

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

public class TCP7Server {
    private final Map<String, ClientHandler> users = new HashMap<>();

    public static void main(String[] args) {
        TCP7Server server = new TCP7Server();
        server.start();
    }

    public void start() {
        try (ServerSocket serverSocket = new ServerSocket(6789)) {
            System.out.println("----- Server -----");
            System.out.println("Listening on port 6789...");
            while (true) {
                Socket socket = serverSocket.accept();
                System.out.println("New User Connected...");
                ClientHandler newUser = new ClientHandler(socket, this);
                newUser.start();
            }
        } catch (IOException e) {
            System.out.println("Error in the server: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    void sendToEveryone(String message, ClientHandler notToUser) {
        try {
            for (Map.Entry<String, ClientHandler> aUser : users.entrySet()) {
                if (aUser.getValue() != notToUser) {
                    aUser.getValue().sendMessage(message);
                }
            }
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    void addUser(String userName, ClientHandler userThread) {
        users.put(userName, userThread);
    }

    void removeUser(String userName) {
        users.remove(userName);
```

```
    }

    void listUsers() {
        System.out.println("--- USERS ---");
        users.keySet().stream().forEach(System.out::println);
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP8Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP8Client {
    public static void main(String args[]) throws Exception {
        Socket clientSocket = new Socket(InetAddress.getLocalHost(), 6789);
        DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
        BufferedReader inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
        String response = inFromServer.readLine();
        System.out.println("Response from Server: " + response);
        clientSocket.close();
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP8Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Arrays;

public class TCP8Server {
    public static void main(String[] args) throws IOException {
        ServerSocket welcomeSocket = new ServerSocket(6789);
        while (true) {
```

```
            Socket connectionSocket = welcomeSocket.accept();
            BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
            String ip = connectionSocket.getInetAddress().toString();
            String ipAddress = ip.substring(1, ip.length());
            System.out.println("Request from client: " + ipAddress);
            String response = "Your IP Address is " + ipAddress + ", ";
            response += IPAddress.getIPClass(new IPAddress(ipAddress));
            outToClient.writeBytes(response + "\n");
        }
    }
}

public class IPAddress implements Comparable<IPAddress> {
    int a, b, c, d;

    IPAddress(String ip) {
        if (isValid(ip))
            parse(ip);
        else {
            a = b = c = d = 0;
        }
    }

    private void parse(String ip) {
        int[] data =
Arrays.stream(ip.split("[.]")).mapToInt(Integer::parseInt).toArray();
        a = data[0];
        b = data[1];
        c = data[2];
        d = data[3];
    }

    public static boolean isValid(String ip) {
        String[] data = ip.split("[.]");
        if (data.length < 4 || data.length > 4)
            return false;

        return Arrays.stream(data).allMatch(value -> {
            int val = Integer.parseInt(value);
            return (val >= 0 && val < 256);
        });
    }

    public static String getIPClass(IPAddress ip) {
        if (ip.a >= 0 && ip.a < 128)
            return "Class A";
        if (ip.a >= 128 && ip.a < 192)
            return "Class B";
```

```
        if (ip.a >= 192 && ip.a < 224)
            return "Class C";
        if (ip.a >= 224 && ip.a < 240)
            return "Class D";
        return "Class E";
    }

    public static boolean isSpecial(IPAddress ip) {
        if (ip.compareTo(new IPAddress("0.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("0.255.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("10.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("10.255.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("100.64.0.0")) >= 0 && ip.compareTo(new
IPAddress("100.127.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("127.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("127.255.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("169.254.0.0")) >= 0 && ip.compareTo(new
IPAddress("169.254.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("172.16.0.0")) >= 0 && ip.compareTo(new
IPAddress("172.31.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("192.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("192.0.0.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("192.0.2.0")) >= 0 && ip.compareTo(new
IPAddress("192.0.2.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("192.88.99.0")) >= 0 && ip.compareTo(new
IPAddress("192.88.99.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("192.168.0.0")) >= 0 && ip.compareTo(new
IPAddress("192.168.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("198.18.0.0")) >= 0 && ip.compareTo(new
IPAddress("198.19.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("198.51.100.0")) >= 0 && ip.compareTo(new
IPAddress("198.51.100.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("203.0.113.0")) >= 0 && ip.compareTo(new
IPAddress("203.0.113.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("224.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("239.255.255.255")) <= 0)
            return true;
```

```
        if (ip.compareTo(new IPAddress("240.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("255.255.255.255")) <= 0)
            return true;
        return false;
    }

    @Override
    public int compareTo(IPAddress anotherIP) {
        if (a > anotherIP.a)
            return 1;
        else if (a < anotherIP.a)
            return -1;
        else {
            if (b > anotherIP.b)
                return 1;
            else if (b < anotherIP.b)
                return -1;
            else {
                if (c > anotherIP.c)
                    return 1;
                else if (c < anotherIP.c)
                    return -1;
                else {
                    if (d > anotherIP.d)
                        return 1;
                    else if (d < anotherIP.d)
                        return -1;
                    else
                        return 0;
                }
            }
        }
    }

    @Override
    public String toString() {
        return a + "." + b + "." + c + "." + d;
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP9Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP9Client {
```

```
    public static void main(String args[]) throws Exception {
        Socket clientSocket = new Socket(InetAddress.getLocalHost(), 6789);
        DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
        BufferedReader inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
        String response = inFromServer.readLine();
        System.out.println("From Server: " + response);
        clientSocket.close();
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP9Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Arrays;

public class TCP9Server {
    public static void main(String[] args) throws IOException {
        ServerSocket welcomeSocket = new ServerSocket(6789);
        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
            String ip = connectionSocket.getInetAddress().toString();
            String ipAddress = ip.substring(1, ip.length());
            System.out.println("From Client: " + ipAddress);
            String response = "Your IP Address is " + ipAddress;
            if (IPAddress.isSpecial(new IPAddress(ipAddress)))
                response += ", Your IP is special.";
            else
                response += ", Your IP is not special.";
            outToClient.writeBytes(response + "\n");
        }
    }
}

public class IPAddress implements Comparable<IPAddress> {
```

```
    int a, b, c, d;

    IPAddress(String ip) {
        if (isValid(ip))
            parse(ip);
        else {
            a = b = c = d = 0;
        }
    }

    private void parse(String ip) {
        int[] data =
Arrays.stream(ip.split("[.]")).mapToInt(Integer::parseInt).toArray();
        a = data[0];
        b = data[1];
        c = data[2];
        d = data[3];
    }

    public static boolean isValid(String ip) {
        String[] data = ip.split("[.]");
        if (data.length < 4 || data.length > 4)
            return false;

        return Arrays.stream(data).allMatch(value -> {
            int val = Integer.parseInt(value);
            return (val >= 0 && val < 256);
        });
    }

    public static String getIPClass(IPAddress ip) {
        if (ip.a >= 0 && ip.a < 128)
            return "Class A";
        if (ip.a >= 128 && ip.a < 192)
            return "Class B";
        if (ip.a >= 192 && ip.a < 224)
            return "Class C";
        if (ip.a >= 224 && ip.a < 240)
            return "Class D";
        return "Class E";
    }

    public static boolean isSpecial(IPAddress ip) {
        if (ip.compareTo(new IPAddress("0.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("0.255.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("10.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("10.255.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("100.64.0.0")) >= 0 && ip.compareTo(new
IPAddress("100.127.255.255")) <= 0)
```

```
            return true;
        if (ip.compareTo(new IPAddress("127.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("127.255.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("169.254.0.0")) >= 0 && ip.compareTo(new
IPAddress("169.254.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("172.16.0.0")) >= 0 && ip.compareTo(new
IPAddress("172.31.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("192.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("192.0.0.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("192.0.2.0")) >= 0 && ip.compareTo(new
IPAddress("192.0.2.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("192.88.99.0")) >= 0 && ip.compareTo(new
IPAddress("192.88.99.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("192.168.0.0")) >= 0 && ip.compareTo(new
IPAddress("192.168.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("198.18.0.0")) >= 0 && ip.compareTo(new
IPAddress("198.19.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("198.51.100.0")) >= 0 && ip.compareTo(new
IPAddress("198.51.100.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("203.0.113.0")) >= 0 && ip.compareTo(new
IPAddress("203.0.113.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("224.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("239.255.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("240.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("255.255.255.255")) <= 0)
            return true;
        return false;
    }

    @Override
    public int compareTo(IPAddress anotherIP) {
        if (a > anotherIP.a)
            return 1;
        else if (a < anotherIP.a)
            return -1;
        else {
            if (b > anotherIP.b)
                return 1;
            else if (b < anotherIP.b)
                return -1;
```

```
            else {
                if (c > anotherIP.c)
                    return 1;
                else if (c < anotherIP.c)
                    return -1;
                else {
                    if (d > anotherIP.d)
                        return 1;
                    else if (d < anotherIP.d)
                        return -1;
                    else
                        return 0;
                }
            }
        }
    }

    @Override
    public String toString() {
        return a + "." + b + "." + c + "." + d;
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP10Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.*;
import java.net.*;
import java.nio.ByteBuffer;
import java.util.Arrays;

public class UDP10Client {
    private static String source = "101";
    private static String destination = "102";

    public static void main(String args[]) throws Exception {
        String sentence;
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
        Socket clientSocket = new Socket(InetAddress.getLocalHost(), 6789);
        DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
        DataInputStream inFromServer = new
DataInputStream(clientSocket.getInputStream());
        sentence = inFromUser.readLine();
        byte[] sourceAddress = Arrays.copyOf(source.getBytes(), 10);
        byte[] destinationAddress = Arrays.copyOf(destination.getBytes(), 10);
```

```
        byte[] data = Arrays.copyOf(sentence.getBytes(), 50);
        byte[] packet =
ByteBuffer.allocate(70).put(sourceAddress).put(destinationAddress).put(data).arr
ay();
        outToServer.write(packet);
        byte[] response = new byte[90];
        inFromServer.read(response);
        String rSourceAddress = new String(Arrays.copyOfRange(response, 0, 10));
        String rDestinationAddress = new String(Arrays.copyOfRange(response, 10,
20));
        String sSourceAddress = new String(Arrays.copyOfRange(response, 20,
30));
        String sDestinationAddress = new String(Arrays.copyOfRange(response, 30,
40));
        String receivedData = new String(Arrays.copyOfRange(response, 40, 90));
        System.out.println("Sender Source Address: " + sSourceAddress);
        System.out.println("Sender Destination Address: " +
sDestinationAddress);
        System.out.println("Receiver Source Address: " + rSourceAddress);
        System.out.println("Receiver Destination Address: " +
rDestinationAddress);
        System.out.println("Data: " + receivedData);
        clientSocket.close();
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP10Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.*;
import java.net.*;
import java.nio.ByteBuffer;
import java.util.Arrays;

public class UDP10Server {
    private static String source = "102";
    private static String destination = "103";

    public static void main(String args[]) throws Exception {
        ServerSocket welcomeSocket = new ServerSocket(6789);
        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            DataInputStream inFromClient = new
DataInputStream(connectionSocket.getInputStream());
            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
```

```
            byte[] clientData = new byte[70];
            inFromClient.read(clientData);
            String sSourceAddress = new String(Arrays.copyOfRange(clientData, 0,
10));
            String sDestinationAddress = new
String(Arrays.copyOfRange(clientData, 10, 20));
            String receivedData = new String(Arrays.copyOfRange(clientData, 20,
70));
            System.out.println("_____From Client_____");
            System.out.println("Sender Source Address: " + sSourceAddress);
            System.out.println("Sender Destination Address: " +
sDestinationAddress);
            System.out.println("Client - Data: " + receivedData);
            byte[] sourceAddress = Arrays.copyOf(source.getBytes(), 10);
            byte[] destinationAddress = Arrays.copyOf(destination.getBytes(),
10);
            byte[] newPacket =
ByteBuffer.allocate(90).put(sourceAddress).put(destinationAddress).put(clientDat
a)
                    .array();
            outToClient.write(newPacket);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP1Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class UDP1Client {
    public static void main(String args[]) throws Exception {
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress = InetAddress.getLocalHost();
        byte[] sendData = new byte[1024];
        String clientSentence;

        do {
            clientSentence = inFromUser.readLine();
            sendData = clientSentence.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress, 6789);
            clientSocket.send(sendPacket);
```

```
        } while (!clientSentence.equalsIgnoreCase("bye") && !
clientSentence.equalsIgnoreCase("exit"));
        clientSocket.close();
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP1Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.net.*;

public class UDP1Server {
    public static void main(String args[]) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(6789);
        while (true) {
            byte[] receiveData = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
            serverSocket.receive(receivePacket);
            String clientSentence = (new
String(receivePacket.getData())).trim();
            System.out.println("Client Says : " + clientSentence);
            if (clientSentence.equalsIgnoreCase("bye") ||
clientSentence.equalsIgnoreCase("exit")) {
                serverSocket.close();
                return;
            }
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP2Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class UDP2Client {
    public static void main(String args[]) throws Exception {
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
```

```
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress = InetAddress.getLocalHost();
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];
        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress, 6789);
        clientSocket.send(sendPacket);
        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
        clientSocket.receive(receivePacket);
        String modifiedSentence = (new String(receivePacket.getData())).trim();
        System.out.println("From Server: " + modifiedSentence);
        clientSocket.close();
    }

}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP2Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.net.*;

class UDP2Server {
    public static void main(String args[]) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(6789);
        while (true) {
            byte[] receivedData = new byte[1024];
            byte[] upperCase = new byte[1024];
            DatagramPacket receivedPacket = new DatagramPacket(receivedData,
receivedData.length);
            serverSocket.receive(receivedPacket);
            String sentence = (new String(receivedPacket.getData())).trim();
            // System.out.println("Client Sended : " + sentence);
            InetAddress IPAddress = receivedPacket.getAddress();
            int port = receivedPacket.getPort();
            String capitalizedSentence = sentence.toUpperCase();
            upperCase = capitalizedSentence.getBytes();
            // String upperCase = sentence.toUpperCase();
            System.out.println("Client Request String = " + sentence + " |
Response = " + capitalizedSentence);
            DatagramPacket sendPacket = new DatagramPacket(upperCase,
upperCase.length, IPAddress, port);
            serverSocket.send(sendPacket);
        }
```

```
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP3Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class UDP3Client {
    public static void main(String args[]) throws Exception {
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress = InetAddress.getLocalHost();
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];
        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress, 6789);
        clientSocket.send(sendPacket);
        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
        clientSocket.receive(receivePacket);
        String length = (new String(receivePacket.getData()).trim());
        System.out.println("From Server: " + length);
        clientSocket.close();
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP3Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class UDP3Server {
    public static void main(String argv[]) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(6789);
        while (true) {
```

```
            byte[] receivedData = new byte[1024];
            byte[] len = new byte[1024];
            DatagramPacket receivedPacket = new DatagramPacket(receivedData,
receivedData.length);
            serverSocket.receive(receivedPacket);
            String sentence = (new String(receivedPacket.getData())).trim();
            InetAddress IPAddress = receivedPacket.getAddress();
            int port = receivedPacket.getPort();
            len = ("" + sentence.length()).getBytes();
            System.out.println("Client Request String = " + sentence + " |
Response = " + sentence.length());
            DatagramPacket sendPacket = new DatagramPacket(len, len.length,
IPAddress, port);
            serverSocket.send(sendPacket);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP4Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class UDP4Client {

    public static void main(String[] args) throws Exception {
        String sentence;
        System.out.println("Enter Time as per 24 Hours Format (eg 17:50)");
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
        sentence = inFromUser.readLine();
        System.out.println(sendAndReceivePacket(sentence));
    }

    private static String sendAndReceivePacket(String time) throws Exception {
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress ipAddress = InetAddress.getLocalHost();
        byte[] sendData = time.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, ipAddress, 6789);
        clientSocket.send(sendPacket);
        byte[] receiveData = new byte[1024];
        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
        clientSocket.receive(receivePacket);
```

```
        String greeting = (new String(receivePacket.getData())).trim();
        clientSocket.close();
        return greeting;
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP4Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class UDP4Server {
    public static void main(String[] args) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(6789);
        byte[] recieveData = new byte[1024];
        String greeting;
        int time = 0;
        int minute = 0;
        while (true) {
            DatagramPacket receivePacket = new DatagramPacket(recieveData,
recieveData.length);
            serverSocket.receive(receivePacket);
            String clientSentence = new String(receivePacket.getData()).trim();
            try {
                time = Integer.parseInt(clientSentence.substring(0,
clientSentence.indexOf(":")));
                minute = Integer
                        .parseInt(clientSentence.substring(clientSentence.indexO
f(":") + 1, clientSentence.length()));

                if ((time < 0 || time > 23) || (minute > 59 || minute < 0))
                    greeting = "Invalid Time";
                else if (time > 19)
                    greeting = "Good Night";
                else if (time > 15)
                    greeting = "Good Evening";
                else if (time > 11)
                    greeting = "Good Afernoon";
                else if (time > 4)
                    greeting = "Good Morning";
                else
                    greeting = "Good MidNight";

            } catch (Exception e) {
                greeting = "Invalid Time ex";
```

```
            }

            InetAddress ipAddress = receivePacket.getAddress();
            int port = receivePacket.getPort();
            byte[] sendData = greeting.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, ipAddress, port);
            serverSocket.send(sendPacket);
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP5Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class UDP5Client {

    public static void main(String[] args) throws Exception {
        String sentence;
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
        sentence = inFromUser.readLine();
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress ipAddress = InetAddress.getLocalHost();
        byte[] sendData = sentence.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, ipAddress, 6789);
        clientSocket.send(sendPacket);
        byte[] receiveData = new byte[1024];
        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
        clientSocket.receive(receivePacket);
        String greeting = (new String(receivePacket.getData())).trim();
        System.out.println(greeting);
        clientSocket.close();
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP5Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */
```

```java
 import java.net.*;
import java.lang.Math;

public class UDP5Server {
    public static void main(String[] args) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(6789);
        byte[] recieveData = new byte[1024];
        byte[] sendData = new byte[1024];
        String selectedData;
        int random;
        String[] quotes = { "You are nice Person.", "You are good.", "You are
lucky.", "You have good choice.",
                "You can do it.", "You are Awesome.", "You are Smart.", "You are
my friend.", "You are stupid.",
                "Lucky to have you here." };
        while (true) {
            DatagramPacket receivePacket = new DatagramPacket(recieveData,
recieveData.length);
            serverSocket.receive(receivePacket);
            String clientSentence = new String(receivePacket.getData()).trim();
            InetAddress ipAddress = receivePacket.getAddress();
            int port = receivePacket.getPort();
            if (clientSentence.equalsIgnoreCase("hello")) {
                random = (int) (Math.random() * 10);
                selectedData = quotes[random];
            } else {
                selectedData = "Hi, say hello for quotes.";
            }
            sendData = selectedData.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, ipAddress, port);
            serverSocket.send(sendPacket);
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP6Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class UDP6Client {

    public static void main(String[] args) throws Exception {
        String sentence;
```

```
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
        System.out.print("Enter Birth Date (eg : day/month/year) : ");
        sentence = inFromUser.readLine();
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress ipAddress = InetAddress.getLocalHost();
        byte[] sendData = sentence.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, ipAddress, 6789);
        clientSocket.send(sendPacket);
        byte[] receiveData = new byte[1024];
        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
        clientSocket.receive(receivePacket);
        String greeting = (new String(receivePacket.getData())).trim();
        System.out.println(greeting);
        clientSocket.close();
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP6Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.net.*;
import java.util.Calendar;

public class UDP6Server {
    public static void main(String[] args) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(6789);
        byte[] recieveData = new byte[1024];
        byte[] sendData = new byte[1024];
        int year, month, day;
        String selectedData;
        while (true) {
            DatagramPacket receivePacket = new DatagramPacket(recieveData,
recieveData.length);
            serverSocket.receive(receivePacket);
            String clientSentence = new String(receivePacket.getData()).trim();
            InetAddress ipAddress = receivePacket.getAddress();
            int port = receivePacket.getPort();
            try {
                day = Integer.parseInt(clientSentence.substring(0,
clientSentence.indexOf("/")));
                month = Integer.parseInt(
                        clientSentence.substring(clientSentence.indexOf("/") +
1, clientSentence.lastIndexOf("/")));
                year = Integer.parseInt(
```

```
                        clientSentence.substring(clientSentence.lastIndexOf("/")
+ 1, clientSentence.length()));

                Calendar cal = Calendar.getInstance();
                int tday = cal.get(Calendar.DAY_OF_MONTH);
                int tmonth = cal.get(Calendar.MONTH) + 1;
                int tyear = cal.get(Calendar.YEAR);

                GetAge gage = new GetAge();

                selectedData = gage.age(tday, tmonth, tyear, day, month, year);

            } catch (Exception e) {
                selectedData = "Invalid Dates";
            }
            sendData = selectedData.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, ipAddress, port);
            serverSocket.send(sendPacket);
        }
    }
}

class GetAge {
    private int finalyear;
    private int finalmonth;
    private int finaldays;

    public String age(int tday, int tmonth, int tyear, int day, int month, int
year) {
        finalyear = tyear - year - 1;
        finaldays = tday;
        if (tmonth >= month && tday >= day) {
            finalyear++;
            finalmonth = tmonth - month;
            finaldays = tday - day;
        } else {
            finalmonth = ((12 - month) + tmonth) - 1;
        }
        if (finalmonth == 0 && finaldays == 0)
            return "Happy Birthday your Age = " + finalyear + " years " +
finalmonth + " months " + finaldays
                    + " days ";
        else
            return "Age = " + finalyear + " years " + finalmonth + " months " +
finaldays + " days ";
    }
} /* ============================================================


/* ============================================================
```

```
 * ============================================================
 PRACTICAL - II
 * ============================================================
 * ============================================================ */

 *
 * Roll No: 30
 *
 * File:      01-ChatClient.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class ChatClient {

    private final String host;
    private final int port;
    private String username;

    public ChatClient(String host, int port) {
        this.host = host;
        this.port = port;
    }

    public void start() {
        try {
            Socket socket = new Socket(host, port);

            System.out.println("Connected to the server...");

            var writeHandler = new ClientWriteHandler(socket, this);
            writeHandler.start();
            (new ClientReadHandler(socket, this)).start();

        } catch (UnknownHostException e) {
            System.out.println("Server not found: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("I/O Error: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    void setUserName(String username) {
        this.username = username;
    }
```

```
    String getUsername() {
        return this.username;
    }

    public static void main(String[] args) {
        ChatClient client = new ChatClient("127.0.0.1", 5678);
        client.start();
    }
}

// $ java -cp bin/ ChatClient
// Connected to the server...

// No other users connected
// Enter your name: Nirav
// [Nirav]:
// New User Connected: Milind
// [Nirav]:
// [Milind]: Hey
// [Nirav]: Hello
// [Nirav]:
// New User Connected: Pradip
// [Nirav]:
// [Pradip]: Hello Everyone
// [Nirav]:
// [Pradip]: exir
// [Nirav]:
// [Pradip]: exit
// [Nirav]:
// Pradip disconnected...
// [Nirav]:
// [Milind]: Bye
// [Nirav]:
// [Milind]: exit
// [Nirav]:
// Milind disconnected...
// [Nirav]: bye
// [Nirav]: exit

// $ java -cp bin/ ChatClient
// Connected to the server...

// Connected users: [Nirav, Milind]
// Enter your name: Pradip
// [Pradip]: Hello Everyone
// [Pradip]: exir
// [Pradip]: exit

// $ java -cp bin/ ChatClient
// Connected to the server...
```

```
// // Connected users: [Nirav]
// // Enter your name: Milind
// // [Milind]: Hey
// // [Milind]:
// // [Nirav]: Hello
// // [Milind]:
// // New User Connected: Pradip
// // [Milind]:
// // [Pradip]: Hello Everyone
// // [Milind]:
// // [Pradip]: exir
// // [Milind]:
// // [Pradip]: exit
// // [Milind]:
// // Pradip disconnected...
// // [Milind]: Bye
// // [Milind]: exit
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      01-ChatServer.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class ChatServer {
    private final Map<String, ClientHandler> users = new HashMap<>();
    private final int SERVER_PORT;

    public ChatServer(int port) {
        this.SERVER_PORT = port;
    }

    public void start() {
        try (ServerSocket serverSocket = new ServerSocket(SERVER_PORT)) {

            System.out.println("----- Chat Server -----");
            System.out.println("Listening on port " + SERVER_PORT + "...");

            while (true) {
                Socket socket = serverSocket.accept();
                System.out.println("New User Connected...");
                ClientHandler newUser = new ClientHandler(socket, this);
                newUser.start();
            }
        } catch (IOException e) {
```

```
            System.out.println("Error in the server: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        int port;
        try {
            port = 5678;
            ChatServer server = new ChatServer(port);
            server.start();
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    void sendToEveryone(String message, ClientHandler notToUser) {
        for (Map.Entry<String, ClientHandler> aUser : users.entrySet()) {
            if (aUser.getValue() != notToUser) {
                aUser.getValue().sendMessage(message);
            }
        }
    }

    void sendTo(String message, String username) {
        var user = getUser(username);
        if (user != null) {
            user.sendMessage(message);
        }
    }

    private ClientHandler getUser(String username) {
        for (Map.Entry<String, ClientHandler> aUser : users.entrySet()) {
            if (aUser.getKey().equals(username)) {
                return aUser.getValue();
            }
        }
        return null;
    }

    void addUser(String userName, ClientHandler userThread) {
        users.put(userName, userThread);
    }

    void removeUser(String userName) {
        users.remove(userName);
    }

    Set<String> getUserNames() {
        return this.users.keySet();
```

```
    }

    boolean hasUsers() {
        return !this.users.isEmpty();
    }
}

// $ java -d bin/ *.ajav && java -cp bin/ ChatServer
// ----- Chat Server -----
// Listening on port 5678...
// New User Connected...
// New User Connected...
// New User Connected... /*
============================================================
 *
 * Roll No: 30
 *
 * File:      01-ClientHandler.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class ClientHandler extends Thread {
    private final Socket socket;
    private final ChatServer server;
    private PrintWriter writer;
    public String username;

    public ClientHandler(Socket socket, ChatServer server) {
        this.socket = socket;
        this.server = server;
    }

    @Override
    public void run() {
        try {
            BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            writer = new PrintWriter(socket.getOutputStream(), true);

            listUsers();

            username = reader.readLine();
            server.addUser(username, this);

            String serverMessage = "New User Connected: " + username;
            server.sendToEveryone(serverMessage, this);
```

```
            String clientMessage;

            do {
                clientMessage = reader.readLine().trim();
                serverMessage = "[" + username + "]: " + clientMessage;

                if (clientMessage.startsWith("@")) {
                    int indexOfSpace = clientMessage.indexOf(' ');
                    if (indexOfSpace < 2) {
                        server.sendToEveryone(serverMessage, this);
                    } else {
                        System.out.println(clientMessage.substring(1,
indexOfSpace));
                        server.sendTo(serverMessage, clientMessage.substring(1,
indexOfSpace));
                    }
                } else {
                    server.sendToEveryone(serverMessage, this);
                }
            } while (!clientMessage.equals("exit"));
        } catch (IOException e) {
            System.out.println("Error in ClientHandler: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            try {
                server.removeUser(username);
                socket.close();
                String serverMessage = username + " disconnected...";
                server.sendToEveryone(serverMessage, this);
            } catch (IOException e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
    }

    void listUsers() {
        if (server.hasUsers()) {
            writer.println("Connected users: " + server.getUserNames());
        } else {
            writer.println("No other users connected");
        }
    }

    void sendMessage(String message) {
        writer.println(message);
    }
}
 /* ============================================================
 *
 * Roll No: 30
```

```
 *
 * File:      01-ClientReadHandler.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ========================================================== */

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

public class ClientReadHandler extends Thread {

    private BufferedReader reader;
    private final Socket socket;
    private final ChatClient client;

    public ClientReadHandler(Socket socket, ChatClient client) {
        this.socket = socket;
        this.client = client;

        try {
            reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        } catch (IOException e) {
            System.out.println("Error getting input stream: " + e.getMessage());
        }
    }

    @Override
    public void run() {
        while (true) {
            try {
                String response = reader.readLine();
                System.out.println("\n" + response);

                // prints the username after displaying the server's message
                if (client.getUsername() != null) {
                    System.out.print("[" + client.getUsername() + "]: ");
                }
            } catch (IOException e) {
                System.out.println("Error reading from server: " +
e.getMessage());
                break;
            } catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
                break;
            }
        }
    }
} /* ==========================================================
 *
 * Roll No: 30
```

```
 *
 * File:      01-ClientWriteHandler.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.*;
import java.net.*;
import java.util.Scanner;

public class ClientWriteHandler extends Thread {
    private PrintWriter writer;
    private final Socket socket;
    private final ChatClient client;

    public ClientWriteHandler(Socket socket, ChatClient client) {
        this.socket = socket;
        this.client = client;

        try {
            OutputStream output = socket.getOutputStream();
            writer = new PrintWriter(output, true);
        } catch (IOException e) {
            System.out.println("Error getting output stream: " +
e.getMessage());
        }
    }

    @Override
    public void run() {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String userName = sc.nextLine();
        client.setUserName(userName);
        writer.println(userName);

        String text;

        do {
            System.out.print("[" + userName + "]: ");
            text = sc.nextLine();
            writer.println(text);
        } while (!text.equals("exit"));

        try {
            socket.close();
        } catch (IOException e) {
            System.out.println("Error writing to server: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
```

```
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      02-MultiThreadedSocketServer.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.net.*;

public class MultiThreadedSocketServer {
    public static void main(String[] args) throws Exception {
        try {
            ServerSocket server = new ServerSocket(5678);
            int counter = 0;
            System.out.println("Listening on Port 5678...");
            while (true) {
                counter++;
                Socket serverClient = server.accept();
                System.out.println(">>> " + "Client No: " + counter + "
connected!");
                ServerClientThread sct = new ServerClientThread(serverClient,
counter);
                System.out.println(counter + " clients connected.");
                sct.start();
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      02-ServerClientThread.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

class ServerClientThread extends Thread {
    Socket serverClient;
    int clientNo;

    ServerClientThread(Socket inSocket, int counter) {
```

```
        serverClient = inSocket;
        clientNo = counter;
    }

    public void run() {
        try {
            BufferedReader reader = new BufferedReader(new
InputStreamReader(serverClient.getInputStream()));
            PrintWriter writer = new
PrintWriter(serverClient.getOutputStream());
            String clientMessage = "", serverMessage = "";

            while (!clientMessage.equals("bye")) {
                clientMessage = reader.readLine();
                int n = Integer.parseInt(clientMessage);

                System.out.println("From Client - " + clientNo + ": Number is :
" + clientMessage);
                serverMessage = n + " is " + (n % 2 == 0 ? "Even" : "Odd");
                writer.println(serverMessage);
                writer.flush();
            }

            reader.close();
            writer.close();
            serverClient.close();

        } catch (Exception ex) {
            System.out.println(ex);
        } finally {
            System.out.println("Client - " + clientNo + " disconnected!");
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      02-TCPThreadClient.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCPThreadClient {
    public static void main(String[] args) throws Exception {
        try {
            Socket socket = new Socket("127.0.0.1", 5678);
            BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
```

```
            PrintWriter writer = new PrintWriter(socket.getOutputStream());
            BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
            String clientMessage = "", serverMessage = "";

            while (!clientMessage.equals("bye")) {
                System.out.println("Enter number :");
                clientMessage = br.readLine();
                writer.println(clientMessage);
                writer.flush();
                serverMessage = reader.readLine();
                System.out.println(serverMessage);
            }

            writer.close();
            socket.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      03-MultiThreadedSocketServer.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.net.*;

public class MultiThreadedSocketServer {
    public static void main(String[] args) throws Exception {
        try {
            DatagramSocket serverSocket = new DatagramSocket(5678);
            System.out.println("Listening on Port 5678...");
            ServerClientThread sct = new ServerClientThread(serverSocket);
            sct.start();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      03-ServerClientThread.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */
```

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

class ServerClientThread extends Thread {
    DatagramSocket serverClient;

    ServerClientThread(DatagramSocket inSocket) {
        serverClient = inSocket;
    }

    public void run() {
        try {
            String clientSentence = "", serverMessage = "";
            while (!clientSentence.equals("bye")) {
                byte[] recieveData = new byte[1024];
                byte[] sendData = new byte[1024];
                DatagramPacket receivePacket = new DatagramPacket(recieveData,
recieveData.length);
                serverClient.receive(receivePacket);
                InetAddress ipAddress = receivePacket.getAddress();
                int port = receivePacket.getPort();
                clientSentence = new String(receivePacket.getData()).trim();
                int n = Integer.parseInt(clientSentence);
                serverMessage = n + " is " + (isPrime(n) ? "Prime" : "Not
Prime");
                System.out.println("From Client : " + clientSentence + "
Result : " + serverMessage);
                sendData = serverMessage.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, ipAddress, port);
                serverClient.send(sendPacket);
            }

            serverClient.close();

        } catch (Exception ex) {
            System.out.println(ex);
        }
    }

    static boolean isPrime(int n) {
        if (n <= 1)
            return false;

        for (int i = 2; i < n; i++)
            if (n % i == 0)
                return false;

        return true;
    }
```

```java
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      03-UDPThreadClient.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class UDPThreadClient {
    public static void main(String[] args) throws Exception {
        try {
            DatagramSocket clientSocket = new DatagramSocket();
            InetAddress ipAddress = InetAddress.getLocalHost();
            BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
            String clientMessage = "";

            while (!clientMessage.equals("bye")) {
                System.out.println("Enter number :");
                clientMessage = br.readLine();
                byte[] sendData = clientMessage.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, ipAddress, 5678);
                clientSocket.send(sendPacket);
                byte[] receiveData = new byte[1024];
                DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
                clientSocket.receive(receivePacket);
                String result = (new String(receivePacket.getData())).trim();
                System.out.println(result);
            }
            clientSocket.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      04-ChatClient.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

```java
public class ChatClient {

    private final String host;
    private final int port;
    private String username;

    public ChatClient(String host, int port) {
        this.host = host;
        this.port = port;
    }

    public void start() {
        try {
            Socket socket = new Socket(host, port);

            System.out.println("Connected to the server...");

            var writeHandler = new ClientWriteHandler(socket, this);
            writeHandler.start();
            (new ClientReadHandler(socket, this)).start();

        } catch (UnknownHostException e) {
            System.out.println("Server not found: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("I/O Error: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    void setUserName(String username) {
        this.username = username;
    }

    String getUsername() {
        return this.username;
    }

    public static void main(String[] args) {
        ChatClient client = new ChatClient("127.0.0.1", 5678);
        client.start();
    }
}

// $ java -cp bin/ ChatClient
// Connected to the server...

// No other users connected
// Enter your name: Nirav
// [Nirav]:
```

```
// New User Connected: Milind
// [Nirav]:
// [Milind]: Hey
// [Nirav]: Hello
// [Nirav]:
// New User Connected: Pradip
// [Nirav]:
// [Pradip]: Hello Everyone
// [Nirav]:
// [Pradip]: exir
// [Nirav]:
// [Pradip]: exit
// [Nirav]:
// Pradip disconnected...
// [Nirav]:
// [Milind]: Bye
// [Nirav]:
// [Milind]: exit
// [Nirav]:
// Milind disconnected...
// [Nirav]: bye
// [Nirav]: exit

// $ java -cp bin/ ChatClient
// Connected to the server...

// Connected users: [Nirav, Milind]
// Enter your name: Pradip
// [Pradip]: Hello Everyone
// [Pradip]: exir
// [Pradip]: exit

// $ java -cp bin/ ChatClient
// Connected to the server...

// // Connected users: [Nirav]
// // Enter your name: Milind
// // [Milind]: Hey
// // [Milind]:
// // [Nirav]: Hello
// // [Milind]:
// // New User Connected: Pradip
// // [Milind]:
// // [Pradip]: Hello Everyone
// // [Milind]:
// // [Pradip]: exir
// // [Milind]:
// // [Pradip]: exit
// // [Milind]:
// // Pradip disconnected...
// // [Milind]: Bye
```

```java
// // [Milind]: exit /*
============================================================
 *
 * Roll No: 30
 *
 * File:      04-ChatServer.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class ChatServer {
    private final Map<String, ClientHandler> users = new HashMap<>();
    private final int SERVER_PORT;

    public ChatServer(int port) {
        this.SERVER_PORT = port;
    }

    public void start() {
        try (ServerSocket serverSocket = new ServerSocket(SERVER_PORT)) {

            System.out.println("----- Chat Server -----");
            System.out.println("Listening on port " + SERVER_PORT + "...");

            while (true) {
                Socket socket = serverSocket.accept();
                System.out.println("New User Connected...");
                ClientHandler newUser = new ClientHandler(socket, this);
                newUser.start();
            }
        } catch (IOException e) {
            System.out.println("Error in the server: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        int port;
        try {
            port = 5678;
            ChatServer server = new ChatServer(port);
            server.start();
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
```

```
    void sendToEveryone(String message, ClientHandler notToUser) {
        for (Map.Entry<String, ClientHandler> aUser : users.entrySet()) {
            if (aUser.getValue() != notToUser) {
                aUser.getValue().sendMessage(message);
            }
        }
    }

    void sendTo(String message, String username) {
        var user = getUser(username);
        if (user != null) {
            user.sendMessage(message);
        }
    }

    private ClientHandler getUser(String username) {
        for (Map.Entry<String, ClientHandler> aUser : users.entrySet()) {
            if (aUser.getKey().equals(username)) {
                return aUser.getValue();
            }
        }
        return null;
    }

    void addUser(String userName, ClientHandler userThread) {
        users.put(userName, userThread);
    }

    void removeUser(String userName) {
        users.remove(userName);
    }

    Set<String> getUserNames() {
        return this.users.keySet();
    }

    boolean hasUsers() {
        return !this.users.isEmpty();
    }
}

// $ javac *.java && java ChatServer
// ----- Chat Server -----
// Listening on port 5678...
// New User Connected...
// 1 clients connected!
// New User Connected...
// 2 clients connected!
// New User Connected...
// 3 clients connected!
```

```
// $ java ChatClient
// Connected to the server...

// No other users connected
// Enter your name: nirav

// 1 clients connected!
// [nirav]: [nirav]:
// New User Connected: milind
// [nirav]:
// 2 clients connected!
// [nirav]:
// New User Connected: pradip
// [nirav]:
// 3 clients connected!
// [nirav]:
// [milind]: bye
// [nirav]:
// [milind]: exit
// [nirav]:
// milind disconnected...
// [nirav]:
// [pradip]: exit
// [nirav]:
// pradip disconnected...
// [nirav]: exit

// $ java ChatClient
// Connected to the server...

// Connected users: [nirav]
// Enter your name: milind

// 2 clients connected!
// [milind]: bye
// [milind]:
// [milind]: exit
// [milind]:
// milind disconnected...
// [milind]: exit

// $ java ChatClient
// Connected to the server...

// Connected users: [nirav, milind]
// Enter your name: pradip

// 3 clients connected!
// [pradip]: [pradip]:
// [milind]: bye
```

```
// [pradip]:
// [milind]: exit
// [pradip]:
// milind disconnected...
// [pradip]: exit /* ============================================================
 *
 * Roll No: 30
 *
 * File:      04-ClientHandler.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class ClientHandler extends Thread {
    private final Socket socket;
    private final ChatServer server;
    private PrintWriter writer;
    public String username;

    public ClientHandler(Socket socket, ChatServer server) {
        this.socket = socket;
        this.server = server;
    }

    @Override
    public void run() {
        try {
            BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            writer = new PrintWriter(socket.getOutputStream(), true);

            listUsers();

            username = reader.readLine();
            server.addUser(username, this);

            String serverMessage = "New User Connected: " + username;
            server.sendToEveryone(serverMessage, this);
            server.sendToEveryone(server.getUserNames().size() + " clients
connected!", null);
            System.out.println(server.getUserNames().size() + " clients
connected!");

            String clientMessage;

            do {
                clientMessage = reader.readLine().trim();
                serverMessage = "[" + username + "]: " + clientMessage;
```

```
                if (clientMessage.startsWith("@")) {
                    int indexOfSpace = clientMessage.indexOf(' ');
                    if (indexOfSpace < 2) {
                        server.sendToEveryone(serverMessage, this);
                    } else {
                        System.out.println(clientMessage.substring(1,
indexOfSpace));
                        server.sendTo(serverMessage, clientMessage.substring(1,
indexOfSpace));
                    }
                } else {
                    server.sendToEveryone(serverMessage, this);
                }
            } while (!clientMessage.equals("exit"));
        } catch (IOException e) {
            System.out.println("Error in ClientHandler: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            try {
                server.removeUser(username);
                socket.close();
                String serverMessage = username + " disconnected...";
                server.sendToEveryone(serverMessage, this);
            } catch (IOException e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
    }

    void listUsers() {
        if (server.hasUsers()) {
            writer.println("Connected users: " + server.getUserNames());
        } else {
            writer.println("No other users connected");
        }
    }

    void sendMessage(String message) {
        writer.println(message);
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      04-ClientReadHandler.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */
```

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

public class ClientReadHandler extends Thread {

    private BufferedReader reader;
    private final Socket socket;
    private final ChatClient client;

    public ClientReadHandler(Socket socket, ChatClient client) {
        this.socket = socket;
        this.client = client;

        try {
            reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        } catch (IOException e) {
            System.out.println("Error getting input stream: " + e.getMessage());
        }
    }

    @Override
    public void run() {
        while (true) {
            try {
                String response = reader.readLine();
                System.out.println("\n" + response);

                // prints the username after displaying the server's message
                if (client.getUsername() != null) {
                    System.out.print("[" + client.getUsername() + "]: ");
                }
            } catch (IOException e) {
                System.out.println("Error reading from server: " +
e.getMessage());
                break;
            } catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
                break;
            }
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      04-ClientWriteHandler.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */
```

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

public class ClientWriteHandler extends Thread {
    private PrintWriter writer;
    private final Socket socket;
    private final ChatClient client;

    public ClientWriteHandler(Socket socket, ChatClient client) {
        this.socket = socket;
        this.client = client;

        try {
            OutputStream output = socket.getOutputStream();
            writer = new PrintWriter(output, true);
        } catch (IOException e) {
            System.out.println("Error getting output stream: " +
e.getMessage());
        }
    }

    @Override
    public void run() {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String userName = sc.nextLine();
        client.setUserName(userName);
        writer.println(userName);

        String text;

        do {
            System.out.print("[" + userName + "]: ");
            text = sc.nextLine();
            writer.println(text);
        } while (!text.equals("exit"));

        try {
            socket.close();
        } catch (IOException e) {
            System.out.println("Error writing to server: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
} /* ============================================================
 *
 * Roll No: 30
```

```
 *
 * File:      05-Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ========================================================== */

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

public class Client {
    private final String FILENAME = null;
    Connect c = new Connect();
    Socket socket;
    BufferedReader read;
    PrintWriter output;

    public void startClient() throws UnknownHostException, IOException {
        socket = new Socket(c.gethostName(), c.getPort());

        output = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()));
        BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));

        System.out.println("Enter Username : ");
        String username = in.readLine();

        output.println(username);

        System.out.println("Enter Password : ");
        String password = in.readLine();

        output.println(password);
        output.flush();
        read = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        String response = read.readLine();
        System.out.println(response);
    }

    public void fileInfo() {

    }

    public static void main(String args[]) {
        Client client = new Client();
        try {
            client.startClient();
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
```

```
            e.printStackTrace();
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      05-Connect.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 public class Connect {
    private String USERNAME = "java";
    private String PASSWORD = "java";
    private int PORT = 9090;
    private String HOSTNAME = "localhost";

    public String getUsername() {
        return this.USERNAME;
    }

    public String getPassword() {

        return this.PASSWORD;
    }

    public int getPort() {
        return this.PORT;
    }

    public String gethostName() {
        return this.HOSTNAME;
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      05-Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class Server {
    private int currentTot;
    ServerSocket serversocket;
    Socket client;
```

```
    int bytesRead;
    Connect c = new Connect();
    BufferedReader input;
    PrintWriter output;

    public void start() throws IOException {
        System.out.println("Connection Starting on port:" + c.getPort());
        serversocket = new ServerSocket(c.getPort());
        client = serversocket.accept();

        System.out.println("Waiting for connection from client");

        try {
            logInfo();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void logInfo() throws Exception {
        input = new BufferedReader(new
InputStreamReader(client.getInputStream()));

        String username = input.readLine();
        System.out.println("Username : " + username);
        String password = input.readLine();
        System.out.println("Password : " + password);

        output = new PrintWriter(new
OutputStreamWriter(client.getOutputStream()));

        if (username.equals(c.getUsername()) &&
password.equals(c.getPassword())) {
            output.println("Welcome, " + username);
        } else {
            output.println("Login Failed");
        }
        output.flush();
        output.close();
    }

    public static void main(String[] args) {
        Server server = new Server();
        try {
            server.start();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
} /* ============================================================
 *
```

```
 * Roll No: 30
 *
 * File:      06-Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ========================================================== */

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

class Client {
    public static void main(String ar[]) throws Exception {
        Socket s = new Socket("localhost", 1234);
        PrintWriter p = new PrintWriter(s.getOutputStream());
        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        BufferedReader ink = new BufferedReader(new
InputStreamReader(System.in));
        System.out.println("How many numbers to sort? ");
        int num = Integer.parseInt(ink.readLine());
        p.println(num);
        p.flush();
        System.out.println("Enter " + num + " numbers to sort :");
        String sarr[] = new String[num];
        for (int i = 0; i < num; i++) {
            sarr[i] = ink.readLine();
            p.println(sarr[i]);
            p.flush();
        }
        String res;
        System.out.println("\nSorted array::\n");
        while ((res = in.readLine()) != null) {
            System.out.println(res);
        }
        s.close();
    }
}
 /* ==========================================================
 *
 * Roll No: 30
 *
 * File:      06-Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ========================================================== */

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

class Server {
    public static void main(String ar[]) throws Exception {
```

```
        ServerSocket s1 = new ServerSocket(1234);
        System.out.println("Server Started");
        Socket s = s1.accept();
        PrintWriter p = new PrintWriter(s.getOutputStream());
        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        String num = in.readLine();
        int n = Integer.parseInt(num);
        System.out.println("Client want to sort " + n + " numbers");
        String sarr[] = new String[n];
        int arr[] = new int[n];
        int swap, c, d;
        System.out.println("received numbers::\n");
        for (int i = 0; i < n; i++) {
            sarr[i] = in.readLine();
            arr[i] = Integer.parseInt(sarr[i]);
            System.out.println(arr[i]);

        }

        for (c = 0; c < (n - 1); c++) {
            for (d = 0; d < n - c - 1; d++) {
                if (arr[d] > arr[d + 1]) {
                    swap = arr[d];
                    arr[d] = arr[d + 1];
                    arr[d + 1] = swap;
                }
            }
        }

        System.out.println("\nSorted list of numbers");
        String sendarr = new String();
        for (c = 0; c < n; c++) {
            sendarr += arr[c];
            sendarr += " ";
        }
        System.out.println(sendarr);
        p.println(sendarr);
        p.flush();
        s.close();

    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      07-Client2.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */
```

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

public class Client2 {
    private Socket s = null;
    private DataInputStream dis = null;
    private DataOutputStream dos = null;

    public static void main(String args[]) throws IOException {
        InetAddress ip = InetAddress.getLocalHost();
        Client2 cr = new Client2(ip, 1234);
    }

    public Client2(InetAddress ip, int port) throws IOException {

        s = new Socket(ip, port);
        dis = new DataInputStream(s.getInputStream());
        dos = new DataOutputStream(s.getOutputStream());

        while (true) {
            Scanner sc = new Scanner(System.in);
            int i, l, nob, sum = 0, chk_sum;

            l = dis.readInt();
            int c_data[] = new int[l];
            int data[] = new int[l];

            System.out.println("Data received (alond with checksum) is");

            for (i = 0; i < data.length; i++) {
                data[i] = dis.readInt();
                System.out.println(data[i]);

                nob = (int) (Math.floor(Math.log(data[i]) / Math.log(2))) + 1;
                c_data[i] = ((1 << nob) - 1) ^ data[i];

                sum += c_data[i];
            }
            System.out.println("Sum(in ones complement) is : " + sum);

            nob = (int) (Math.floor(Math.log(sum) / Math.log(2))) + 1;
            sum = ((1 << nob) - 1) ^ sum;
            System.out.println("Calculated Checksum is : " + sum);

            if (sum == 0) {
                dos.writeUTF("success");
                break;
            } else {
                dos.writeUTF("failure");
                break;
```

```
            }
        }

        dis.close();
        dos.close();
        s.close();
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      07-Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class Client {
    private int MAX = 100;
    private Socket socket = null;
    private ServerSocket servsock = null;
    private DataInputStream dis = null;
    private DataOutputStream dos = null;

    public static void main(String args[]) throws IOException {
        Client cs = new Client(1234);
    }

    public Client(int port) throws IOException {
        servsock = new ServerSocket(port);
        socket = servsock.accept();

        dis = new DataInputStream(socket.getInputStream());
        dos = new DataOutputStream(socket.getOutputStream());

        while (true) {
            int i, l, sum = 0, nob;
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter data length");
            l = sc.nextInt();
            int data[] = new int[MAX];
            int c_data[] = new int[MAX];

            System.out.println("Enter data to send");

            for (i = 0; i < l; i++) {
                data[i] = sc.nextInt();
                nob = (int) (Math.floor(Math.log(data[i]) / Math.log(2))) + 1;
```

```
                c_data[i] = ((1 << nob) - 1) ^ data[i];
                sum += c_data[i];
            }
            data[i] = sum;
            l += 1;

            System.out.println("Checksum Calculated is : " + sum);
            System.out.println("Data being sent along with Checkum.....");
            dos.writeInt(l);
            for (int j = 0; j < l; j++)
                dos.writeInt(data[j]);

            if (dis.readUTF().equals("success")) {
                System.out.println("Thanks for the feedback!! Message received
Successfully!");
                break;
            }

            else if (dis.readUTF().equals("failure")) {
                System.out.println("Message was not received successfully!");
                break;
            }
        }

        dis.close();
        dos.close();
        socket.close();
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      10-SAWClient.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.*;
import java.net.*;
import java.util.Scanner;

public class SAWClient {
    public static void main(String args[]) {
        int p = 9000, i, q = 8000;
        String h = "localhost";
        try {
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter number of frames : ");
            int number = scanner.nextInt();
            if (number == 0) {
                System.out.println("No frame is sent");
```

```
            } else {
                Socket s2;
                s2 = new Socket(h, q);
                DataOutputStream d1 = new
DataOutputStream(s2.getOutputStream());
                d1.write(number);
                s2.close();
            }
            String str1;
            for (i = 0; i < number; i++) {
                System.out.print("Enter message : ");
                String name = scanner.next();
                System.out.println("Frame " + i + " is sent");
                Socket s1;
                s1 = new Socket(h, p + i);
                DataOutputStream d = new DataOutputStream(s1.getOutputStream());
                d.writeUTF(name);
                DataInputStream dd = new DataInputStream(s1.getInputStream());
                Integer sss1 = dd.read();
                System.out.println("Ack for :" + sss1 + " is  received");
                s1.close();
            }
        } catch (Exception ex) {
            System.out.println("ERROR :" + ex);
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      10-SAWServer.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class SAWServer {
    public static void main(String args[]) {
        String h = "Serverhost";
        int q = 5000;
        int i;
        try {
            ServerSocket ss2;
            ss2 = new ServerSocket(8000);
            Socket s1 = ss2.accept();
            DataInputStream dd1 = new DataInputStream(s1.getInputStream());
            Integer i1 = dd1.read();
            for (i = 0; i < i1; i++) {
```

```
                ServerSocket ss1;
                ss1 = new ServerSocket(9000 + i);
                Socket s = ss1.accept();
                DataInputStream dd = new DataInputStream(s.getInputStream());
                String sss1 = dd.readUTF();
                System.out.println(sss1);
                System.out.println("Frame " + i + " received");
                DataOutputStream d1 = new DataOutputStream(s.getOutputStream());
                d1.write(i);
                System.out.println("ACK sent for " + i);
                ss1.close();
                ss2.close();
            }
        } catch (Exception ex) {
            System.out.println("Error" + ex);
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      10-SWClient.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.net.*;
import java.io.*;
import java.rmi.*;

public class SWClient {
    public static void main(String a[]) throws Exception {
        ServerSocket ser = new ServerSocket(10);
        Socket s = ser.accept();
        BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
        BufferedReader in1 = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        // DataInputStream in1 = new DataInputStream(s.getInputStream());
        String sbuff[] = new String[8];
        PrintStream p;
        int sptr = 0, sws = 8, nf, ano, i;
        String ch;
        do {
            p = new PrintStream(s.getOutputStream());
            System.out.print("Enter the no. of frames : ");
            nf = Integer.parseInt(in.readLine());
            p.println(nf);
            if (nf <= sws - 1) {

                System.out.println("Enter " + nf + " Messages to be send\n");
```

```
                for (i = 1; i <= nf; i++) {
                    sbuff[sptr] = in.readLine();
                    p.println(sbuff[sptr]);
                    sptr = ++sptr % 8;
                }
                sws -= nf;
                System.out.print("Acknowledgment received");
                ano = Integer.parseInt(in1.readLine());
                System.out.println(" for " + ano + " frames");
                sws += nf;
            } else {
                System.out.println("The no. of frames exceeds window size");
                break;
            }
            System.out.print("\nDo you wants to send some more frames : ");
            ch = in.readLine();
            p.println(ch);
        } while (ch.equals("yes"));
        s.close();
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      10-SWServer.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

class SWServer {
    public static void main(String a[]) throws Exception {
        Socket s = new Socket(InetAddress.getLocalHost(), 10);
        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        // DataInputStream in = new DataInputStream(s.getInputStream());
        PrintStream p = new PrintStream(s.getOutputStream());
        int i = 0, rptr = -1, nf, rws = 8;
        String rbuf[] = new String[8];
        String ch;
        System.out.println();
        do {
            nf = Integer.parseInt(in.readLine());
            if (nf <= rws - 1) {
                for (i = 1; i <= nf; i++) {
                    rptr = ++rptr % 8;
                    rbuf[rptr] = in.readLine();
                    System.out.println("The received Frame " + rptr + " is : " +
rbuf[rptr]);
```

```
                }
                rws -= nf;
                System.out.println("\nAcknowledgment sent\n");
                p.println(rptr + 1);
                rws += nf;
            } else
                break;
            ch = in.readLine();
        } while (ch.equals("yes"));
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      11-Dijkstra.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 public class Dijkstra {

    public static void main(String[] args) {
        int graph[][] = new int[][] { { 0, 0, 1, 2, 0, 0, 0 }, { 0, 0, 2, 0, 0,
3, 0 }, { 1, 2, 0, 1, 3, 0, 0 },
                { 2, 0, 1, 0, 0, 0, 1 }, { 0, 0, 3, 0, 0, 2, 0 }, { 0, 3, 0, 0,
2, 0, 1 }, { 0, 0, 0, 1, 0, 1, 0 } };
        dijkstra(graph, 0);
    }

    public static void dijkstra(int[][] graph, int source) {
        int count = graph.length;
        boolean[] visitedVertex = new boolean[count];
        int[] distance = new int[count];
        for (int i = 0; i < count; i++) {
            visitedVertex[i] = false;
            distance[i] = Integer.MAX_VALUE;
        }

        distance[source] = 0;
        for (int i = 0; i < count; i++) {

            int u = findMinDistance(distance, visitedVertex);
            visitedVertex[u] = true;

            for (int v = 0; v < count; v++) {
                if (!visitedVertex[v] && graph[u][v] != 0 && (distance[u] +
graph[u][v] < distance[v])) {
                    distance[v] = distance[u] + graph[u][v];
                }
            }
        }
```

```
        for (int i = 0; i < distance.length; i++) {
            System.out.println(String.format("Distance from %s to %s is %s",
source, i, distance[i]));
        }

    }

    private static int findMinDistance(int[] distance, boolean[] visitedVertex)
{
        int minDistance = Integer.MAX_VALUE;
        int minDistanceVertex = -1;
        for (int i = 0; i < distance.length; i++) {
            if (!visitedVertex[i] && distance[i] < minDistance) {
                minDistance = distance[i];
                minDistanceVertex = i;
            }
        }
        return minDistanceVertex;
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      12-Admin.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.File;
import java.io.IOException;
import java.util.HashSet;
import java.util.Scanner;

public class Admin {

    public static void main(String[] args) {

        HashSet<Integer> hashSetPorts = new HashSet<>();
        if (args.length == 0) {
            System.out.println("Please Include Directory Path");
            return;
        } else if (args.length > 1) {
            System.out.println("Incorrect Input Format.! Only one parameter is
required");
            return;
        }

        String PATH = args[0];
        File directory = new File(PATH);

        if (!directory.isDirectory()) {
```

```
            System.out.println("Directoy Path is Incorrect");
            return;
        }
        File data[] = directory.listFiles();
        int size = data.length;
        int[] ports = new int[size];
        String allNodes = "";

        System.out.println("Initilization of Port Number to " + size + "
Routers");
        Scanner scanner = new Scanner(System.in);

        for (int i = 0; i < size; i++) {
            String val = data[i].getName();
            val = val.substring(0, val.indexOf(".dat"));
            System.out.println("Enter Port No: for Router: " + val);

            boolean status = true;

            while (status) {
                try {
                    int num = Integer.parseInt(scanner.nextLine());

                    if (num <= 1024 || num >= 65536) {
                        throw new NumberFormatException();
                    }
                    if (hashSetPorts.contains(num)) {
                        throw new Exception();
                    }
                    ports[i] = num;
                    hashSetPorts.add(num);
                    status = false;
                } catch (NumberFormatException e) {
                    System.out.println("Enter a valid Port Number > 1024 && <
65536");
                    status = true;
                } catch (Exception e) {
                    System.out.println("Address is Already in Use:");
                    status = true;
                }
            }
            allNodes += " " + val + ":" + ports[i];
        }

        scanner.close();

        for (int i = 0; i < size; i++) {
            ProcessBuilder processBuilder = new ProcessBuilder("cmd.exe", "/c",
"start java MainRouter " + (i + 1)
                    + " \"" + data[i].getParent().replace("\\", "/") + "\" " +
size + allNodes);
```

```
            try {
                processBuilder.start();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        System.out.println("DV Algorithm Started");

    }

} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      12-MainRouter.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.util.Arrays;

public class MainRouter {

    public static int router_DisplayCount = 1;
    public static double[][] router_NetworkVectors;
    public static int[] router_Ports;
    public static String[] router_Nodes;
    public static int router_id;
    public static double[] router_MyVector;
    public static String[] router_MyHopList;
    public static DatagramSocket router_Socket;
    public static File router_File;
    public static String[] router_Neighbours;

    public static void main(String args[]) {
        int total = Integer.parseInt(args[2]);
        int currentNum = Integer.parseInt(args[0]);
        String path = args[1];
        setParameters(total, args, currentNum, path);

        MainThread readThread = new MainThread("r");
        readThread.start();
```

```
        MainThread writeThread = new MainThread("w");
        writeThread.start();

        while (true)
            ;
    }

    public static void setParameters(int len, String[] args, int id, String
parent) {

        router_NetworkVectors = new double[len][len];
        router_Ports = new int[len];
        router_Nodes = new String[len];

        for (int i = 0; i < len; i++) {
            Arrays.fill(router_NetworkVectors[i], Double.MAX_VALUE);
            router_NetworkVectors[i][i] = 0.0;
            String[] temp = args[i + 3].split(":");
            router_Nodes[i] = temp[0];
            router_Ports[i] = Integer.parseInt(temp[1]);
        }

        router_id = id;
        router_MyVector = new double[len];
        router_MyHopList = new String[len];
        Arrays.fill(router_MyVector, Double.MAX_VALUE);
        router_MyVector[router_id - 1] = 0.0;
        router_File = new File(parent + "/" + router_Nodes[router_id - 1] +
".dat");
        try {
            router_Socket = new DatagramSocket(router_Ports[router_id - 1]);
        } catch (SocketException e) {

            e.printStackTrace();
        }
        System.out.println("Router " + router_Nodes[router_id - 1] + " is
Working..!");

    }

    public static void distanceAlgorithm() {

        for (int i = 0; i < router_Neighbours.length; i++) {
            int ind = indexFinder(router_Neighbours[i]);
            for (int j = 0; j < router_MyVector.length; j++) {
                if (j == router_id - 1) {
                    continue;
                } else if (i == 0) {

                    router_MyVector[j] = router_NetworkVectors[router_id - 1]
[ind] + router_NetworkVectors[ind][j];
```

```
                    router_MyHopList[j] = router_Neighbours[i];
                } else {

                    if (router_MyVector[j] > router_NetworkVectors[router_id -
1][ind]
                            + router_NetworkVectors[ind][j]) {
                        router_MyHopList[j] = router_Neighbours[i];
                        router_MyVector[j] = router_NetworkVectors[router_id -
1][ind] + router_NetworkVectors[ind][j];
                    }

                }
            }
        }

    }

    public synchronized static void updateNetworkVectors(String[] vector, int
port) {

        int index = 0;
        int ports_Length = router_Ports.length;
        while (index < ports_Length) {
            if (router_Ports[index] == port) {
                break;
            }
            index++;
        }
        if (index == ports_Length) {
            return;
        }
        for (int i = 0; i < vector.length; i++) {
            router_NetworkVectors[index][i] = Double.parseDouble(vector[i]);
        }
    }

    public static void broadCast() {

        try {
            for (int i = 0; i < router_Neighbours.length; i++) {
                String data = "";
                for (int j = 0; j < router_MyVector.length; j++) {
                    if (router_Neighbours[i].equals(router_MyHopList[j])) {
                        data = data + Double.MAX_VALUE + ":";
                    } else {
                        data += router_MyVector[j] + ":";
                    }
                }
                DatagramPacket packet = new DatagramPacket(data.getBytes(),
data.getBytes().length);
                packet.setAddress(InetAddress.getByName("localhost"));
```

```
                packet.setPort(router_Ports[indexFinder(router_Neighbours[i])]);
                router_Socket.send(packet);

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

    }

    public static void output() {

        System.out.println("> output number " + router_DisplayCount++);
        System.out.println();
        String src = router_Nodes[router_id - 1];
        for (int i = 0; i < router_MyVector.length; i++) {
            if (i != (router_id - 1)) {
                String dest = router_Nodes[i];
                if (router_MyVector[i] == Double.MAX_VALUE) {
                    System.out.println("shortest path " + src + "-" + dest + ":
" + " no route found");
                } else {
                    System.out.println("shortest path " + src + "-" + dest + " :
the next hop is " + router_MyHopList[i]
                            + " and the cost is " + router_MyVector[i]);
                }
            }
        }

    }

    public static void readData() {
        boolean status = true;
        while (status) {
            try {
                String type = "u";
                byte[] data = new byte[1024];
                int size = data.length;
                DatagramPacket packet = new DatagramPacket(data, size);
                router_Socket.receive(packet);
                int length = packet.getLength();
                String vector = new String(packet.getData(), 0, length);
                MainThread updateThread = new MainThread(type, vector,
packet.getPort());
                updateThread.start();

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

```
    public static void read() {
        try {
            Arrays.fill(router_NetworkVectors[router_id - 1], Double.MAX_VALUE);
            router_NetworkVectors[router_id - 1][router_id - 1] = 0.0;
            BufferedReader br = new BufferedReader(new FileReader(router_File));
            int length = Integer.parseInt(br.readLine());
            router_Neighbours = new String[length];
            for (int i = 0; i < length; i++) {

                String[] temp = br.readLine().split(" ");
                int ind = indexFinder(temp[0]);
                router_Neighbours[i] = temp[0];

                if (router_DisplayCount == 1) {
                    router_MyHopList[ind] = temp[0];
                    router_MyVector[ind] = Double.parseDouble(temp[1]);
                } else {

                }
                router_NetworkVectors[router_id - 1][ind] =
Double.parseDouble(temp[1]);

            }
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static int indexFinder(String temp) {
        int pos = -1;
        for (int i = 0; i < router_Nodes.length; i++) {
            if (router_Nodes[i].equals(temp)) {
                pos = i;
                break;
            }
        }
        return pos;
    }

}
```