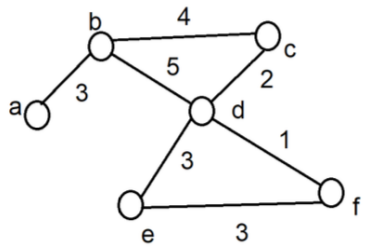
Student: Taylor Tang
Student ID: 1323782



Question 1:

A) We1

a	
Seq.	
Age	
b	3

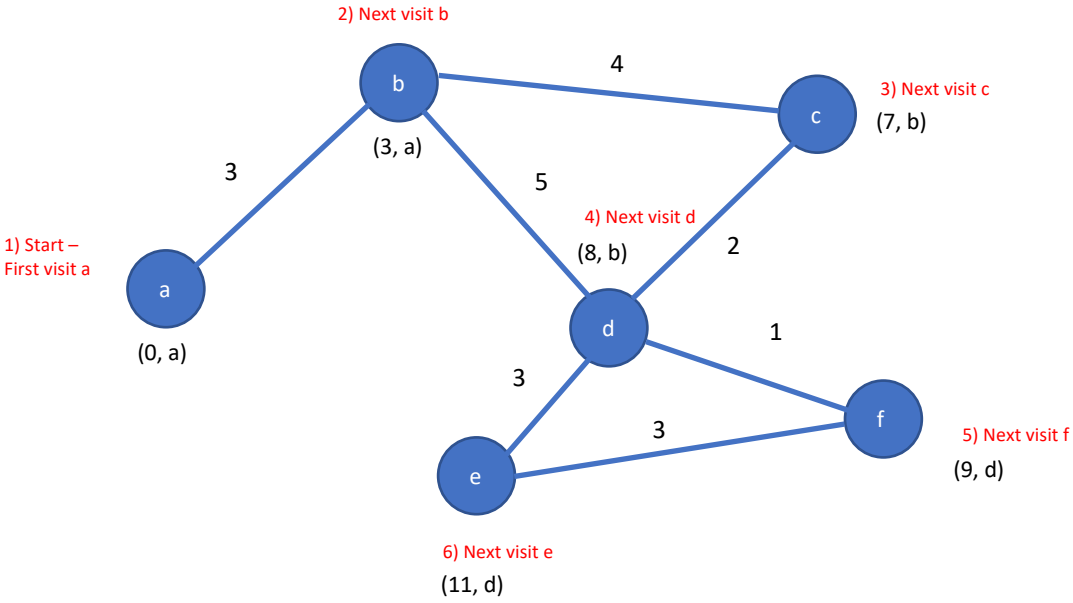
b	
Seq.	
Age	
a	3
c	4
d	5

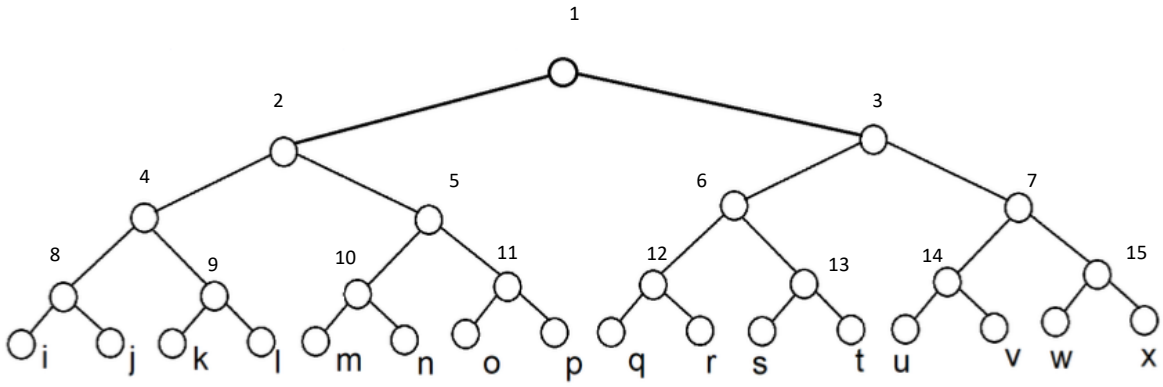
c	
Seq.	
Age	
b	4
d	2

d	
Seq.	
Age	
b	5
c	2
e	3
f	1

e	
Seq.	
Age	
d	3
f	3

f	
Seq.	
Age	
d	1
e	3





Question 2

- Please see the diagram above of each node and its number. Since p, s and v all are stations under node 1, and all trying to send something now. Hence, in slot 1, we have a collision of p, s and v.

  Slot 1 → p, s, v – collision

- Next, using binary tree depth first search, we investigate node 2 and it shows that p is the only station under node 2 trying to send. Hence, in slot 2, we can put station p with no collision.

  Slot 2 → p

- Next, under the investigation of node 3, we have s and v trying to send. Hence, slot 3 has collision.

  Slot 3 → s, v – collision

- Investigate node 6, no collision.

  Slot 4 → s

- No collision left, Hence, slot 5 can be filled.

  Slot 5 → v

Question 3

On transport layer, UDP and TCP are the two main protocols. UDP provides features like transmitting encapsulated datagram without a connection establishment. If the receiver detected error or loss from the data of sender, it rejects the data. TCP, on the other hand, provide features like connection-orientated, reliable data communication. It first establishes the connection with 3 hand-shake and disconnect with 4 hand-wave. The data transmitted under TCP protocol is to be in correct order and content. If the receiver detected error or loss, it notifies the sender to resend the lost data.

Therefore, for video and audio in a bidirectional video conferencing (e.g., zoom). UDP is more suitable.

- In real-life data transmitting scenario, error in data or packet loss is inevitable. If using TCP protocol, it means the sender needs to resend the data after being notified something went wrong by receiver. In such a case, the data transmitting pace is affected. So, TCP is not suitable for bidirectional video conferencing, in which live data (or instant data transmission) is critical.

- Some data dropping and missing is acceptable in the presentation of video and audio. In some cases, a bit of sound muffing, image fading, or colour inaccuracy do not affect audience to understand the meaning of the message. Moreover, there are modern technologies (e.g., AI to correct or predict error or missing data). Moreover, for video and audio, it is critical to keep the presentation in a static pace. Meaning, for video and audio, the smoothness is important. You would not enjoy a video call with a lot of breaking and pausing. Therefore, it is good to trade quality over speed. Using UDP the sender always sends the current data to receiver, without caring too much about data loss. In such a way, it provides a better real-time video or audio transmission quality.

- For on-demand video or audio, TCP is better since we can have a buffer area. In such a way,

For text communication, TCP is more suitable.

- The text message must be accurate to present its content. Therefore, a reliable connection is a must, to make sure the receiver receives the original data in correct format in full. TCP can be used to ensure of this, if any data during transition got wrong or lost, the send will resend the data.

- Moreover, in text communication, the factor of ‘‘real-time’’ is not so important comparing to audio and video. Even a few seconds delay on text message won’t become an issue in many cases. Besides due to the data size difference between text message and video feed, there are not much gain to trade quality over speed. So, TCP is more suitable for text communication.

Question 4

1) The transport layer adds end-to-end communication for processes running on hosts services. While network layer, provide logical communication between hosts.
In other words, network layer can communicate with another's computer's network layer and exchange information. However, it sees the information (to or from) another hosts as one big unit and does not care about separation of the content. Transport layer, (combine or distribute) the information (to or from) network layer, depends on the application or processes that needed the information.

2) First, transport layer and network layer have different responsibilities. Network layer is to provides connection between different hosts on networks, to make the data package be able to transmit from one host to another. While transport layer is to server the application layer by combining or distribute the data from or to network layer.

   The core reason of separating the whole network system into different layers is to breakdown big problems to smaller ones, so it is easier to solve.

   Network layer solved the issues such as deciding each host's unique IP address, which router to choose and what is the shortest path between two hosts etc. In general, it solves the problem of transmitting data from one host to another host.

   Transport layer solved the issues such as when receiving or sending data, what kind of service is needed (whether it is reliable or not, connectionless, or connecting-orientated), which port to use for which applications, and which data received at network layer will be forward to which application at application layer. In general, it solves the issue of end-to-end communication.

   So, if we combine these two layers together, we will face much more complex problems. Therefore, it is important two separate these layers.

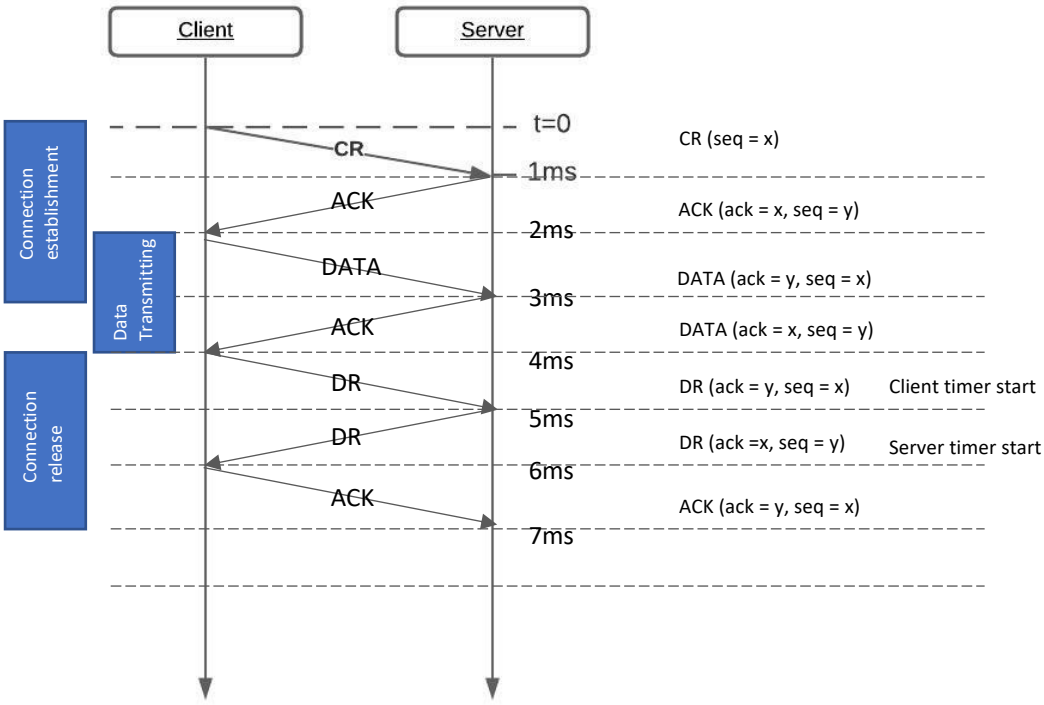
3) If network layer and transport layer are merged. When applications or processes are trying to send data to other applications on other hosts, a lot more effort will be spent on how to establish connection between them.

   For example, when one application needs to send data to another application on another host, each of the data packet need to include not only the IP address, but also the application ID or port number. This certainly increase the complexity at the merged transport and network layer which will dramatically decrease the transferring efficiency.

   Another example is when we need a reliable connection between two applications, and the hosts are far away from each other, how to check the data and make sure the data is lossless during transmitting becomes very difficult. If the two layers are merged, then each router will need to check the actual data correctness instead of just the header or checksum. It will increase the stress of routers and decrease the transferring efficiency.

   Depends on the use case, different network uses different network protocol, such as reliable or unreliable. Different application may require different protocol in transport layer, such as connectionless or connection orientated. Therefore, if we merge two layers together, it will be difficult to satisfy both.

Question 5



1) According to Tom's clarification on this question, I use the simplified version in this question (details such as SYN or FIN packet, SEQ number are not showing)

   During connection establishment stage, the client and server establish a connection by three-way-handshake:

   - Client first sends connection request with a random chosen sequence number x.
   - Server then replies to the CR with an ACK segment, acknowledge the x from client, and a random chosen sequence number y,
   - Client then acknowledges server's sequence number y and sending data to server with its own sequence number x.

   During the data transmitting stage:

   - (As the last step from the connection establishment stage above) Client acknowledges server's sequence number y and sending data to server with its own sequence number x.
   - Server acknowledges client's sequence number x, and sending its own sequence number y.

   During the connection stage:

   - After receiving the last acknowledgement from server that acknowledging the last data is received, the client sends the disconnection request to server with acknowledging the server's seq y, and client's seq x. Start its own timer.
   - Server acknowledging client's disconnection request, acknowledging client's seq x, send DR with its own seq y. Start its own timer.
   - Client acknowledging server's disconnection request, acknowledging server's seq y, with its own seq x. Client's connection released.
   - When server receive client's acknowledgment, server's connection released.
   - If any of above got lost during transmitting, when client or server's timer timeout, it releases the connection.

2) Total 7ms
3) UDP is connect less, which means when a UDP segment send from one to another, it doesn't request the acknowledgement from the other side. In other words, UDP segment is fired and forgot. In terms of speed, UDP is faster than TCP as it doesn't require to establish a connection first, and acknowledgement of each segment. It just sends. In the case above, for 1 packet of data, the whole process of TCP takes 8ms. It takes only 1ms for UDP under the same circumstance.