1.

a. Internet Checksum
    i. 0110001010111000
    ii. 1001110101000111
b. Reliable data transfer rdt22
    i. Wait for ACK 0
    ii. Wait for 0 from below
    iii. 0
    iv. Wait for ACK 0
    v. Wait for 1 from below
    vi. 0
    vii. Wait for ACK 1
    viii. Wait for 1 from below
    ix. 1
    x. Wait for ACK 1
    xi. Wait for 0 from below
    xii. 1
    xiii. 2
c. Reliable data transfer rdt30
    i. S3
d. TCP sequence and ACK numbers, with segment loss
    i. 373,1020,1667
    ii. x,373,373
e. TCP RTT and Timeout
    i. 295
    ii. 26.5
    iii. 401
    iv. 290.63
    v. 28.63
    vi. 405.13
    vii. 281.80
    viii. 39.13
    ix. 438.3
f. TCP retransmissions
    i. 58
    ii. 834
    iii. 1610
    iv. 2386
    v. 3162
    vi. 834

vii. 1610
viii. x
ix. x
x. x
xi. 3938
xii. 4714
xiii. x
xiv. x
xv. X

2.

a. Introduction and Transport Layer Services
    i. Transport layer functions are implemented primarily at the hosts at the "edge" of the network.
    ii. True
    iii. **Services provided by TCP include**: Reliable data delivery, a congestion control service to ensure that multiple senders do not overload network links, a byte stream abstraction, that does not preserve boundaries between message data sent in different socket send calls at the sender, a flow-control service that ensures that a sender will not send at such a high rate so as to overflow receiving host buffers, and in-order data delivery.
    iv. **Services provided by UDP include**: A message abstraction, that preserves boundaries between message data sent in different socket send calls at the sender.
    v. Correct! The network layer's best effort service doesn't really provide much service at all, does it?
b. Multiplexing and De-Multiplexing
    i. Receiving a transport-layer segment from the network layer, extracting the payload (data) and delivering the data to the correct socket.
    ii. Taking data from one socket (one of possibly many sockets), encapsulating a data chuck with header information – thereby creating a transport layer segment – and eventually passing this segment to the network layer.
    iii. True
    iv. False
    v. True
    vi. True
c. Connectionless Transport: UDP
    i. True
    ii. **UDP Segment header fields**: Source port number, Destination port number, Length (of UDP header plus payload), and Internet checksum.

iii. Because the payload section can be of variable length, and this lets UDP know where the segment ends.
iv. The entire UDP segment, except the checksum field itself, and the IP sender and receive address fields.
v. **True statements about checksum**: A checksum is computed at a sender by considering each byte within a packet as a number, and then adding these numbers (each number representing a bytes) together to compute a sum (which is known as a checksum), the sender-computed checksum value is often included in a checksum field within a packet header, and the receiver of a packet with a checksum field will add up the received bytes, just as the sender did, and compare this locally-computed checksum with the checksum value in the packet header. If these two values are different then the receiver knows that one of the bits in the received packet has been changed during transmission from sender to receiver.
vi. 0101011011100111
vii. 1001111000010000
viii. True
ix. False
x. D, A, C, G

d. Principles of reliable data transfer
i. D, C, B, A, E
ii. S1, R1, S2, R2,S3 – S1, R2, S3 – S1, R1, S2
iii. E, C, A, B
iv. C, G, F, D, E
v. C, A, D, F, E
vi. A cumulative ACK(n) acks all packets with a sequence number up to and including n as being received.
vii. 0.1
viii. 1.0
ix. 10
x. **True statements about pipelining**: A pipelined sender can have transmitted multiple packets for which the sender has yet to receive an ACK from the receiver and with a pipelined sender, there may be transmitted packets "in flight" – propagating through the channel – packets that the sender has sent but that the receiver has not yet received.
xi. **Some reasons for discarding received-but- out-of-sequence packets at the receiver in GBN**: The implementation at the receiver is simpler and the sender will resend that packet in any case.
xii. **Some reasons for not discarding received-but- out-of-sequence packets at the receiver in GBN**: Even though that packet will be

retransmitted, its next retransmission could be corrupted, so don't discard a perfectly well-received packet, silly!

xiii. There is a packet with a lower sequence number than any of the red packets that has yet to be received, so in-order delivery of data in the red packets up to the application layer is not possible.

xiv. Because the sender may not have received an ACK for that packet yet.

e. Connection-oriented Transport: TCP
   i. False
   ii. F, B, H, A, D, G, E, C
   iii. Because the send-to receiver segment carries only one byte of data, and after that segment is received, the next expected byte of data is just the next byte (i.e., has an index that is one larger) in the data stream.
   iv. E, C, B, D, A
   v. False
   vi. Cancels any running timers.
   vii. True
   viii. E, D, C, B, A
   ix. If the channel can reorder messages, a triple duplicate ACK can occur even though a message is not lost; since it's possible that a message has just been reordered and has not yet arrived when the three duplicate ACKs were generated and if the channel cannot reorder messages, a triple duplicate ACK indicates to the sender that a segment loss has happened for sure. Actually (again assuming the channel cannot corrupt or reorder messages), even a single duplicate ACK would indicate that a segment loss has happened for sure.

3.
   a. estimatedRTT = 232.5, devRTT = 69.75, TCP timeout = 511.5
   b. estimatedRTT = 250.94, devRTT = 89.19, TCP timeout = 607.69
   c. estimatedRTT = 264.57, devRTT = 94.16, TCP timeout = 641.20

4.
   a. Total time = 14 seconds, Total transmissions = 12
   b. Total time = 4 seconds, Total transmissions = 15
   c. Total time = 3.5 seconds, Total transmissions = 12

5. TCP flow control is used to ensure that a sender does not overwhelm a receiver by sending more data than the receiver can handle. This is achieved primarily on the receiver's side, as the receiver advertises a window of free buffer space in which the sender limits the size of its sent data too. As acknowledgments arrive this window is adjusted dynamically to account for the next batch of data being sent.

6. TCP fast retransmission helps to improve performance by detecting and retransmitting lost packets faster. This is achieved by resending the unacknowledged segment with the smallest

sequence number if the sender receives three additional acknowledgements for the same piece of data.

7. TCP establishes connection via the three way handshake, as this system ensures that both sides (client and server) have acknowledged each other's sequence numbers and that they are ready to begin sending/receiving. This is done with the client first sending a request to connect to the server to connect, followed by the server sending back an acknowledgment of this request. Finally the client acknowledges that it received the response from the server, thus completing the three way handshake.