Problem 1)

No, because while a checksum can detect one bit error or even multiple bit errors, we run the risk that two-bit errors or bit flips can actually cancel each other out and while still showing up as correct in the checksum. So while the data can all be intact we can't guarantee that no bit errors have occurred in the transmission process.

Problem 2)

a) Given that the window size is 4 and the receiver expects k, we have two scenarios. We don't lose any packets but the ACKs are still in propagation for scenario 2:

   Scenario 1: the sender receives all the ACKs before k:
   Window = [k, k + 1, k + 2, k + 3]

   Scenario 2: sender doesn't receive all ACKs before k:
   Windows = [k - 4, k - 3, k - 2, k - 1] (original sending window)
   [k - 3, k - 2, k - 1, k] (only k - 4 ACK missing)
   [k - 2, k - 1, k, k + 1] (up to k - 3 ACK missing)
   [k - 1, k, k + 1, k + 2] (up to k - 2 ACK missing)

b) The ACK numbers possible would be k - 4, k - 3, k - 2, k - 1 since the receiver has received the respective packets. k - 5 is already processed by sender and ack k couldn't have been sent because segment k wasn't received yet.

Problem 3)

a) ReceiveWindow should be large enough to keep the pipe full at all times, meaning that we will have a one-delay bandwidth package in-flight at a time. To achieve this we do 200 ms * 1 Gbps = 2 * 10^8 bits = 200 Mb = 25MB. ceil($\log_2$(25 MB)) is 25 bits.

   For SequenceNum, we know that in the 10 second segment lifetime we can send 10 Gb of data. To find the number of bytes sent we divide 10 Gb by 8 getting us 1.25 GB. ceil($\log_2$(1.25GB)) = 31 bits

b) A 16 bit receive window limits the max. window size to 65536 bytes. This is the largest amount of unacknowledged in-flight data the sender is allowed to transmit at a time. If we plug in the numbers: (window size in bits / RTT) = (65536 bytes / 0.2s) = 327,680 bytes per second = 2.6 Mbps which severely limits the transport despite the bandwidth being 1 Gbps

Problem 4)

a) Go-Back-N: A sends 8 segments and later re-sends 4 more, so A sends a total of 12 segments. B sends 11 ACKS:

   A sends 123456785678

B sends ACKS for 12344445678

Selective Repeat: A sends 8 segments and later resends segment number 5, so A sends 9. B sends 8 ACKS:

A sends 123456785

B sends ACKS for 12346785

TCP: A sends 8 and later only resends segment number 5 so A sends a total of 9. B sends 8 ACKS:

A sends 123456785

B sends ACKS for 23455559

b) TCP because it uses fast retransmit without waiting on the timeout.

Problem 5) To calculate the time for the sender to receive the ACK for segment 15, we gotta consider the 60 ms from sender to receiver and the 40ms from receiver to sender. We also need to remember that we lose sequence 6 once. We also know RTO is 500 ms the whole time so we don't have to write it down

Time (ms)	**Event Description**	**cwnd**	**ssthresh**
0	Connection Start	1	6
0	Send segment 1	1	6
100	ACK 2 received (cwnd + 1)	2	6
100	Send segment 2 and 3	2	6
200	ACK 3 and 4 received (cwnd + 2)	4	6
200	Send segment 4, 5, 6, 7	4	6
300	Segment 6 lost, segment 7 is buffered. ACK 5 received for segment 4, and we should receive a duplicate ACK 6 for segment 5	6	6
300	Send segments 8, 9, 10, 11. Segment 6 is the LOWEST outstanding packet since it is never ACKed. As a result our current window is [6, 7, 8, 9, 10, 11]	6	6
400	Receive 3 duplicate ACK 6 for segments 7, 8, and 9. You also receive ACK 6 for segments 10 and 11. (you technically have 5 dupes) This triggers fast retransmit	6	6

400	Fast recovery, ssthresh = cwnd/2 = 3, cwnd = ssthresh + 3 (we inflate by 3 every time)	6	3
400	Retransmit segment 6	6	3
500	ACK 12 is received for segment 6, cwnd is set back to 3 (which would be segments 12, 13, and 14)	3	3
500	We are at congestion avoidance right now. Send segments 12, 13, and 14	3	3
600	ACK is received for 13, 14, and 15 for segments 12, 13, and 14 respectively. This triggers cwnd = cwnd + 1 for congestion avoidance.	4	3

We've received ACK 15 for segment number 14. At this point cwnd 4 and ssthresh is 3. This is our final answer.

Cwnd = 4

Ssthresh = 3