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 our range is 1024, we have 2 scenarios:

   Case 1: we are receiving packet number k and the receiver ACKed all packets before. Then our next range would be [k, k + 1, k + 2, k + 3]

   Case 2: if the sender's window is [k - 4, k - 3, k - 2, k - 1] after the receiver doesn't successfully ACK, then our possible range would be [k - 4, k - 3, k - 2, k - 1, k]

b) The ACK field would be [k - 4, k - 3, k - 2, k - 1], thus making all the possible values range between k - N - 1 and k - 1.

Problem 3)

a) ReceiveWindow should be large enough to keep the pipe in 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 = 25 MB. ceil($\log_2$(25MB)) is 25 bits.

   For SequenceNum, we know that in the 10 second segment lifetime we can send 10 Gb of data. Since SequenceNum counts packets and we assume the minimum packet size is 40 bytes (size of IP + TCP header), we have 250 million minimum sized packets if we divide 10 Gb by 40 bytes. ceil($\log_2$(250 million min. sized packets)) = 28 bits

b) A 16 bit receive window limits the max. window size to approximately 64 KB. 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) = (64KB / 0.2s) = 26 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 55555559

b) TCP because it uses fast retransmit without waiting time.

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	Start the connection	1	4
60	Send segment 1	1	4
100	Receive ACK for segment 1	2	4
160	Send segments 2 and 3	2	4
200	Receive ACK for segment 2	3	4
240	Receive ACK for segment 3 (THRESHOLD CROSSED)	4	4
300	Send segments 4, 5, 6, and 7	4	4
340	Receive ACK for segment 4	5	4
380	Receive ACK for segment 5	6	4
420	Receive ACK for segment 7	7	4
480	Send segments 8, 9, 10, 11, 12, 13, 14, and 15	7	4
520	Receive ACK for segment 8	8	4
560	Receive ACK for segment 9	9	4
600	Receive ACK for segment 10	10	4
640	Receive ACK for segment 11	11	4
680	Receive ACK for segment 12	12	4

720	Receive ACK for segment 13	13	4
760	Receive ACK for segment 14	14	4
800	Receive ACK for segment 15	15	4

Takes around 800 ms.