Problem 1

The sender side of *rdt3.0* simply ignores (that is, takes no action on) all received packets that are either in error or have the wrong value in the acknum field of an acknowledged packet. Suppose that in such circumstances, *rdt3.0* were simply to retransmit the current data packet. Would the protocol still work? (Hint: Consider what would happen if there were only bit errors; there are no packet losses but premature timeouts can occur. Consider how many times the *nth* packet is sent, in the limit as n approaches infinity).

The protocol would still work, since a retransmission would be what would happen if the packet received with errors has actually been lost (and from the receiver standpoint, it never knows which of these events, if either, will occur).
To get at the more subtle issue behind this question, one has to allow for premature timeouts to occur. In this case, if each extra copy of the packet is ACKed and each received extra ACK causes another extra copy of the current packet to be sent, the number of times packet n is sent will increase without bound as n approaches infinity.

Problem 2

Consider a reliable data transfer protocol that uses only negative acknowledgments. Suppose the sender sends data only infrequently. Would a NAK-only protocol be preferable to a protocol that uses ACKs? Why? Now suppose the sender has a lot of data to send and the end-to-end connection experiences few losses. In this second case, would a NAK-only protocol be preferable to a protocol that uses ACKs? Why?

In a NAK only protocol, the loss of packet x is only detected by the receiver when packet *x+1* is received. That is, the receivers receives *x-1* and then *x+1*, only when *x+1* is received does the receiver realize that x was missed. If there is a long delay between the transmission of x and the transmission of *x+1*, then it will be a long time until x can be recovered, under a NAK only protocol.
On the other hand, if data is being sent often, then recovery under a NAK-only scheme could happen quickly. Moreover, if errors are infrequent, then NAKs are only occasionally sent (when needed), and ACK are never sent – a significant reduction in feedback in the NAK-only case over the ACK-only case.

Problem 3

Consider the GBN protocol with a sender window size of 6 and a sequence number range of 1,024. Suppose that at time t, the next in-order packet that the receiver is expecting has a sequence number of k. Assume that the medium does not reorder messages. Answer the following questions:

(a) What are the possible sets of sequence numbers inside the senders window at time t? Justify your answer.

(b) What are all possible values of the ACK field in all possible messages currently propagating back to the sender at time t? Justify your answer.

(a) Here we have a window size of N=4. Suppose the receiver has received packet k-1, and has ACKed that and all other preceding packets. If all of these ACK's have been received by sender, then sender's window is [k, k+N-1]. Suppose next that none of the ACKs have been received at the sender. In this second case, the sender's window contains k-1 and the N packets up to and including k-1. The sender's window is thus [k-N,k-1]. By these arguments, the senders window is of size 4 and begins somewhere in the range [k-N,k].

(b) If the receiver is waiting for packet k, then it has received (and ACKed) packet k-1 and the N-1 packets before that. If none of those N ACKs have been yet received by the sender, then ACK messages with values of [k-N,k-1] may still be propagating back.Because the sender has sent packets [k-N, k-1], it must be the case that the sender has already received an ACK for k-N-1. Once the receiver has sent an ACK for k-N-1 it will never send an ACK that is less that k-N-1. Thus the range of in-flight ACK values can range from k-N to k-1.

Note: In very extreme case where RTT vary from much lower than RTO to much higher than RTO, the ACK k-N-1 could also happen to be propagating back. We accept both answers.

Problem 4

Follow the same problem setting in Page 62 of Slides Chapter3-2020.ppt. Suppose packet size is 4KB (*i.e.* 4000 bytes), bandwidth is 8Mbps, and one-way propagation delay is 20 msec. Assume there is no packet corruption and packet loss.

(a) Suppose sener window size is 5, will the sender be kept busy? If yes, explain why. If not, What is the effective throughput?

(b) What is the minimum sender window size to achieve full utilization? Then how many bits would be needed for the sequence number field?

(a) No. RTT is 40 ms. Transmission delay is 4ms. It takes 4ms * 5 = 20ms to transmit 5 segments. Therefore, the link is not fully utilized.
The effective throughput is (8Mbps * 4 * 5) / 44.

(b) 11. Calculated by (RTT + transmission delay) / transmission delay. 5 bits since $2^5 \geq 11 * 2$ for selective repeat or 4 bits for go back N.

Problem 5

Answer True or False to the following questions and briefly justify your answer:

(a) With the Selective Repeat protocol, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.

(b) With Go-Back-N, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.

(c) The Stop-and-Wait protocol is the same as the SR protocol with a sender and receiver window size of 1.

(d) Selective Repeat can buffer out-of-order delivered packets, while GBN cannot. Therefore, SR saves network communication cost (by transmitting less) at the cost of additional memory.

(a) True. It could happen if premature timeout is triggered, so that the sender resends the packets, and then receives the ACKs for the original packets. Thus, it would move on to another window; however, the ACKs for duplicated packets will be outside of its current window.

(b) True. By essentially the same scenario as in (a).

(c) True. With a window size of 1, SR, GBN, and the alternating bit protocol are functionally equivalent.

(d) True. Reason same as (c).