1. True or false? Why? Consider congestion control in TCP. When the timer expires at the sender, the value of ssthresh is set to one-half of its previous value.

False. In congestion window, slow start threshold is set to ½ of the previous value

2. In our rdt protocols, why did we need to introduce sequence numbers and timers?

In case of reliable data transfer, sequence number plays a big role. It communicates to the receiver whether a packet contains new data or old data. On the other hand, timer provides the receiver a mechanism to execute a timeout if a packet never arrives. It handles any loss in channel. If a packet doesn't arrive within timer duration then it is assumed that packet is lost. The packet is then transmitted again.

3. Suppose Host A sends two TCP segments back to back to Host B over a TCP connection. The first segment has sequence number 90; the second has sequence number 110.
   a. How much data is in the first segment?
      110 - 90 = 20 (bytes)
   b. Suppose that the first segment is lost, but the second segment arrives at B. What will be the acknowledgment number in the acknowledgment that Host B sends to Host A?
      Acknowledgment number will be 90

4. Suppose the five measured SampleRTT values are 106 ms, 120 ms, 140 ms, 90 ms, and 115 ms.
   a. Compute the EstimatedRTT after each of these SampleRTT values is obtained, using a value of $\alpha = 0.125$ and assuming that the value of EstimatedRTT was 100 ms just before the first of these five samples were obtained.

      SampleRTT = 106
      EstimatedRTT = α * sampleRTT + (1-a) * EstimatedRTT
      = .125 * 106 + (1-.125) * 100
      = 100.75ms
      SampleRTT = 120
      EstimatedRTT = α * sampleRTT + (1-a) * EstimatedRTT
      = .125 * 120 + (1-.125) * 100.75
      = 103.2ms
      SampleRTT = 140
      EstimatedRTT = α * sampleRTT + (1-a) * EstimatedRTT
      = .125 * 140 + (1-.125) * 103.2
      = 107.8ms

SampleRTT = 90
EstimatedRTT = α * sampleRTT + (1-a) * EstimatedRTT
= .125 * 90 + (1-.125) * 107.8
= 105.5ms
SampleRTT = 115
EstimatedRTT = α * sampleRTT + (1-a) * EstimatedRTT
= .125 * 115 + (1-.125) * 105.5
= 106.7ms

b. Compute also the DevRTT after each sample is obtained, assuming a value of β = 0.25 and assuming the value of DevRTT was 5 ms just before the first of these five samples was obtained.

sampleRTT = 106
DevRTT = β * |SampleRTT - EstimatedRTT| + (1-β) * DevRTT
= .25 * |106-100.75| + (1-.25) * 5
= 5.0625ms
sampleRTT = 120
DevRTT = β * |SampleRTT - EstimatedRTT| + (1-β) * DevRTT
= .25 * |120-103.2| + (1-.25) * 5.06
= 8ms
sampleRTT = 140
DevRTT = β * |SampleRTT - EstimatedRTT| + (1-β) * DevRTT
= .25 * |140-107.8| + (1-.25) * 8
= 14.06ms
sampleRTT = 90
DevRTT = β * |SampleRTT - EstimatedRTT| + (1-β) * DevRTT
= .25 * |90-105.5| + (1-.25) * 14.06
= 14.4ms
sampleRTT = 115
DevRTT = β * |SampleRTT - EstimatedRTT| + (1-β) * DevRTT
= .25 * |115-106.7| + (1-.25) * 14.4
= 12.9ms

c. Compute the TCP TimeoutInterval after each of these samples is obtained
sampleRTT = 106
TimeoutInterval = EstimatedRTT + 4 * DevRTT
= 100.8 + 4*5.063
= 121.05

sampleRTT = 120
TimeoutInterval = EstimatedRTT + 4 * DevRTT
= 103.2 + 4*8
= 135.2
sampleRTT = 140
TimeoutInterval = EstimatedRTT + 4 * DevRTT
= 107.8 + 4*14.06
= 164.04

sampleRTT = 90
TimeoutInterval = EstimatedRTT + 4 * DevRTT
= 105.6 + 4*14.4
= 163.2
sampleRTT = 115
TimeoutInterval = EstimatedRTT + 4 * DevRTT
= 106.7 + 4*12.9
= 158.3

5. Draw the FSM for the receiver side of protocol rdt3.0.

6. Consider transferring an enormous file of L bytes from Host A to Host B. Assume an MSS of 536 bytes
   a. What is the maximum value of L such that TCP sequence numbers are not exhausted? Recall that the TCP sequence number field has 4 bytes.
      Max L = 2^{32}, it is the possible sequence number
   b. MSS = 536bytes
      Segment data = 2^{32}/536 = 8012999
      Number of bytes from 155mbps = 8012999/666 = 528857934
      data transmitted = 2^{32} + 528857934 = $4.8*10^9$
      Time for transmit = $(4.8*10^9)*8/155*10^6$
      = 249s

7. we saw that TCP waits until it has received three duplicate ACKs before performing a fast retransmit. Why do you think the TCP designers chose not to perform a fast retransmit after receiving the first duplicate ACK for a segment?

In some cases, the IP layer sends a packet in the wrong order. When it happens, a duplicate ACK is generated. If retransmission is performed after the first duplicate ACK then the sender will generate many redundant packets in the network.