HW6

1. Persistent HTTP
    a. Source & Destination
        i. Source: valid randomly assigned port number
           Destination: 80
        ii. Source: valid randomly assigned port number
            Destination: 80
        iii. Source: 80
             Destination: valid randomly assigned port number for A
        iv. Source: 80
            Destination: valid randomly assigned port number for B
    b. No. A and B connect to C on different HTTP connections. Further, each connection has a unique socket.
    c. Yes. TCP can still identify them separately via source IP address.
    d. No. TCP would not be able to identify which segment came from which socket because the source port number and IP address would be the same.

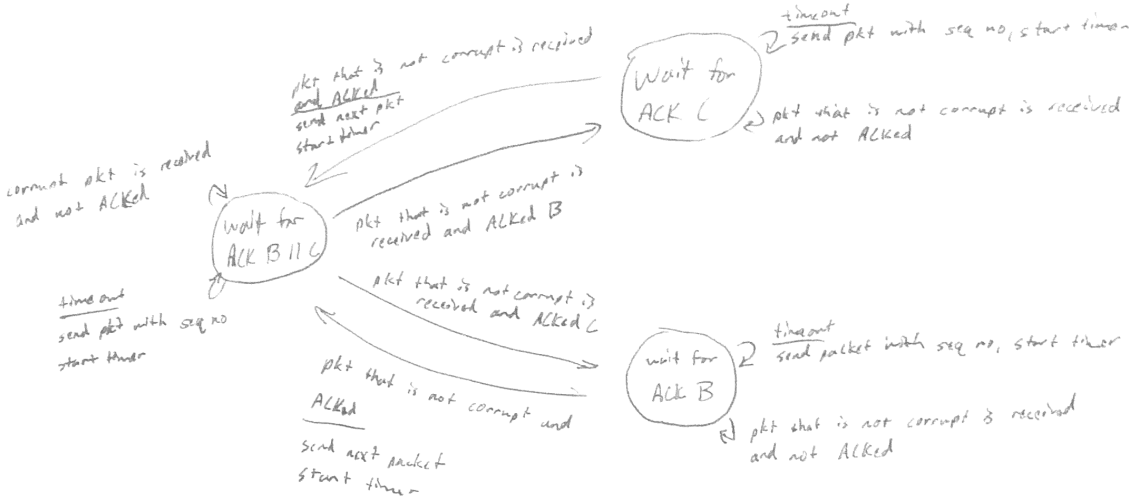
2.
```
   10011001
+  01010010
___________
   11101011
+  01011001
___________
1  01000100
          1 add 9th bit
___________
=  01000101
___________
   10111010 1's comp
```

1-bit errors will always be detected. However, 2-bit errors can go undetected.

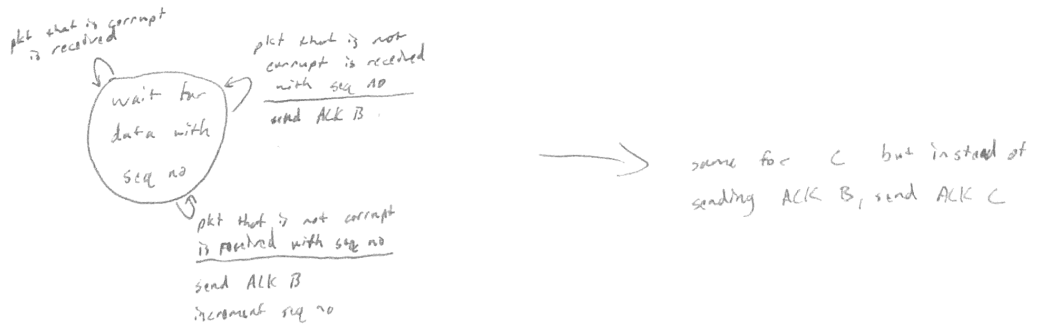
```
   10011000 <- last bit switched (should be 10011001)
+  01010011 <- last bit switched (should be 01010010)
___________
   11101011
+  01011001
___________
1  01000100
          1 add 9th bit
___________
=  01000101
___________
   10111010 1's compl with 2 bit error is the same as previous answer
```

3.

A FSM



B FSM



Pkt data format

Sender → Receiver

seq no | data

Receiver → Sender

ACK no | B | C

4. N=4, range = 1024
    a. Min(send_base) = k-4
        Sender sends packets sequentially: k-4, k-3, k-2, k-1
        All packets are received
        But the ACK of all of the packets is lost
    Send base is still k-4 but the receive_base is k since everything before k is received
    Max(send_base) = k
        If the send_base is > k, the kth packet is sent by the sender, received by the receiver, and then ACKed by the receiver to the sender
        The receive_base can't be k because the kth packet is received
        This contradicts the assumption that receive_base is k >= send_base <= k
    Thus the sets of possible sequence numbers is
        {k-4, k-3, k-2, k-1}
        {k-3, k-2, k-1, k}
        {k-2, k-1, k, k+1}
        {k-1, k, k+1, k+2}
        {k, k+1, k+2, k+3}
    b. The range for the senders sliding window is from k-4 to k+3. Everything in between can be transmitted from the sender and get ACKed by the receiver. However, K cannot be because the receiver has not yet received k
    Thus, the possible ACK value are from k-4 to k+3 excluding k

5.
    a. $EstimatedRTT1 = SampleRTT$
    $EstimatedRTT2 = x \times SampleRTT1 + (1-x) \times SampleRTT2$
    $EstimatedRTT3$
    $= x \times SampleRTT1 + (1-x)[x \times SampleRTT2 + (1-x) \times SampleRTT3]$
    $= x \times SampleRTT1 + (1-x)x \times SampleRTT2 + (1-x)\text{^}2 \times SampleRTT3$
    $EstimatedRTT4 = x \times SampleRTT1 + (1-x) \times EstimatedRTT3$
    $= x \times SampleRTT1 + (1-x)x \times SampleRTT2 + ((1$
    $-x)\text{^}2)\text{x} \times SampleRTT3 + (1-x)\text{^}3 \times SampleRTT4$
    b. $EstimatedRTT\text{n} = \text{x}\sum_{i=1}^{n} n - 1(1-x)^i * SampleRTT\text{i} + (1-\text{x})^{\text{n}} * SampleRTT\text{n}$
    c. $EstimatedRTT\infty = \frac{x}{1-x}\sum_{i=1}^{\infty}(1-x)^i * SampleRTT\text{i} = \frac{1}{9}\sum_{i=1}^{\infty}(.9)^i * SampleRTT\text{i}$
    Comment: 0.9^i deceases exponentially as time goes by which leads the method to be called an exponential moving average

6.
    a. TCP: A sends 6 segments. B sends 5 ACKs. A sends sequence numbers 1, 2, 3, 4, 5 and resends 2.There are 4 ACKS with sequence number 2. There is one ACK with sequence number 6.
    Go-Back-N: A sends 9 segments, B sends 8 ACKs. A sends sequence numbers 1, 2, 3, 4, 5 and resends 2, 3, 4, 5. There are 4 ACKs with seq number 1, and 4 ACKs with sequence numbers 2, 3, 4, 5.
    Selective Repeat: A sends 6 segments, B sends 5 ACKs. A sends sequence numbers 1, 2, 3, 4, 5 and resends 2. There are 4 ACKs with seq no 1, 3, 4, 5. There is one ACK with seq no 2.
    b. TCP because it retransmits without waiting for a timeout.

7.

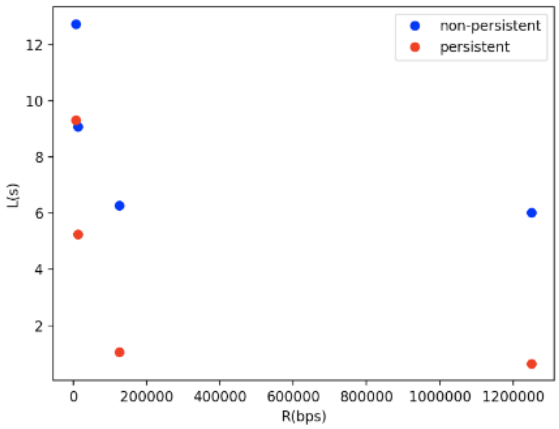
a. [1,6), [23,26)
b. [6, 16), [17, 22)
c. Triple duplicate ACK. The congestion window size would have decreased to 1 if there as a timeout.
d. Timeout, which is why the congestion window's size is 1.
e. 32. At this window size, congestion avoidance begins.
f. 24
g. ~14.5 or 14
h. 7. 1+2+4+8+16+32=63 < 80. After 7, 63+33=96 > 80
i. Congestion window = 8 -> 1 and ssthresh = 8/2 = 4

8. The reanalysis should say that the connection has to be established and then that packets travelling over the links can be divided accordingly to maximize throughput. Thus 2RTT to establish the connection and (O/T)RTT to send the data.

9. Assume the persistent connection is pipelined
Non-persistent: $L = (2RTT + O/R + P(S/R + RTT) – (2^P – 1)S/R) * (M + 1)$
Persistent: $L = 2RTT + O/R + P(S/R + RTT) – (2^P – 1)S/R + RTT + M * O/R$
$= 3RTT + (M + 1) * O/R + P(S/R+RTT) – (2^P – 1)S/R$
$P=min(q,k-1)$
$q=log2(1+RTT/S*R)+1$
$k=log2(1+O/S)$
$M = 10$
$O = 5kbytes = 5000 bytes$
$S = 536 bytes$

a. RTT = 100ms = .1s

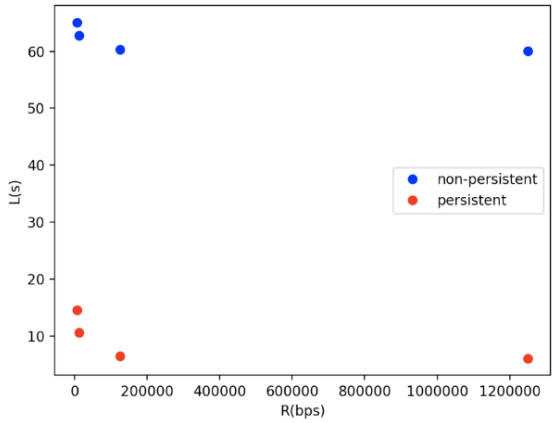
R(Kbps)	P	L(non-persistent)	L(persistent
54	2	12.74	9.31
100	2	9.09	5.26
1000	2	6.27	1.06
10000	2	6.03	.65



b. RTT = 1s

R(Kbps)	P	L(non-persistent)	L(persistent

54	3	65.08	14.57
100	3	62.74	10.63
1000	3	60.27	6.46
10000	3	60.03	6.05



c. L = (M + 1)O/R + 2(M/x + 1)RTT + SSL
M + 2 objects with size O sent on the link with capacity R -> (M + 1)O/R
Get base page in 2 RTTs, then open x parallel connections -> 2(M/x + 1)RTT
Add the latency of sow-start -> SSL

10.

a. K = min{I : $\sum_{i=0}^{n-1} 4^i \geq \frac{O}{S}\} = log4(\frac{2*O}{S}+1)$
b. Q = max{I : $RTT + \frac{S}{R} \geq 4^{(i-1)}\frac{S}{R}\} = log4(\frac{RTT}{S/R}+1)+1$
c. P=min{Q,K-1}
Stall time = P$(\frac{S}{R} + RTT) - \frac{4^P-1}{2} * \frac{S}{R}$
Total Latency = 2RTT + $\frac{O}{R} + P(\frac{S}{R} + RTT) - \frac{4^P-1}{2} * \frac{S}{R}$

11. There are 2 cases:
a. Both connections decrease by the same amount
Since both connections operate together, the increase and decrease the window size comparatively. The throughput of the connections lies between 2 points at any given time.
b. One connection decreases more than the other
If connection 1 decreases its window size by 2 whereas connection 1 decreases its window size by 1: After each loss connection 2 occupies more bandwidth. Eventually connection 1 will have a throughput of 0 and the entire bandwidth will be consumed by connection 2.

The additive increase additive decrease function will not retain the equal share property. My answer does not depend on the initial rates of the two connections.