P1. Suppose Client A initiates a Telnet session with Server S. At about the same time, Client B also initiates a Telnet session with Server S. Provide possible source and destination port numbers for
a. The segments sent from A to S.
b. The segments sent from B to S.
c. The segments sent from S to A.
d. The segments sent from S to B.
e. If A and B are different hosts, is it possible that the source port number in the segments from A to S is the same as that from B to S?
f. How about if they are the same host?

a) 23 is the port number of the Server S.
467 is the port number of the Client A.
b) 23 is the port number of the Server S.
513 is the port number of the Client B.
c) 23 is the port number of the Server S.
467 is the port number of the Client A.
d) 23 is the port number of the Server S.
513 is the port number of the Client B.
e) Yes, it is possible that the source port number in the segments from A to S is the same as that from B to S even when A and B are different hosts. This is because IP addresses are also included in the segments to recognize the correct host.
f) No, the source ports would need to be different. It is not possible that the source port number in the segments from A to S is the same as that from B to S when they are the same hosts. The server to distinguish the hosts uses IP addresses.

P2. Consider Figure 3.5. What are the source and destination port values in the segments flowing from the server back to the clients' processes? What are the IP addresses in the network-layer datagrams carrying the transport-layer segments?

To host A: Source port =80, source IP address = b, destination port = 26145, destination IP address = a
To host C, left process: Source port =80, source IP address = b, destination port = 7532, destination IP address = c
To host C, right process: Source port =80, source IP address = b, destination port = 26145, destination IP address = c

P3. UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit bytes: 01010011, 01100110, 01110100. What is the 1s complement of the sum of these 8-bit bytes? (Note that although UDP and TCP use 16-bit words in computing the checksum, for this problem you are being asked to consider 8-bit sums.) Show all work. Why is it that UDP takes the 1s complement of the sum; that is, why not just use the sum? With the 1s complement scheme, how does the receiver detect errors? Is it possible that a 1-bit error will go undetected? How about a 2-

bit error?

01010011 + 01100110 = 10111001
10111001 + 01110100 = 00101110
1s complement = 11010001
To detect errors, the receiver adds the four words (the three original words and the checksum).
If the sum contains a zero, the receiver knows there has been an error.
All 1-bit errors will be detected.
Two-bit errors can be undetected.

P5. Suppose that the UDP receiver computes the Internet checksum for the received UDP segment and finds that it matches the value carried in the checksum field. Can the receiver be absolutely certain that no bit errors have occurred? Explain.

No, the receiver can't be confident there haven't been any bit errors. The checksum for the packet is determined in such a way that this is the case. If the corresponding bits (that would be added together) of two 16-bit words in the packet were 0 and 1, the sum would remain the same even if they were flipped to 1 and 0 correspondingly. As a result, the 1s complement calculated by the receiver will be the same. This ensures that even if there is a transmission fault, the checksum will be valid.

P7. In protocol rdt3.0, the ACK packets flowing from the receiver to the sender do not have sequence numbers (although they do have an ACK field that contains the sequence number of the packet they are acknowledging). Why is it that our ACK packets do not require sequence numbers?

- If a sender transfers the packet to the receiver, then receiver will receive and send ACK to the sender for conformation.
- If sender received ACK then go to the next level.
- In this process, needs sequence number to the sender for finding duplicate packets data or ACK data.
- If the sender finds any duplicate ACK, then ignore it. In this process ACK packets does not require sequence number.