Summary of the analysis_pcap_tcp

Uses the PCAP file.
The program analyzes the TCP packets between the specified sender and receiver. The sender and receiver are specified by the user by their IP addresses. The program goes through each packet and uses the dpkt library to parse the values for each packet. Afterwards, the packet's information can be analyzed. The program outputs the following information to the terminal: The number of TCP flows initiated from the sender, and for each flow: the first 2 packets sent with their corresponding ack packets - sequence number, ack number, and receive window size, flow's throughput - total amount of data sent by the sender.
A TCP flow starts with a TCP "SYN" and ends at TCP "FIN". A TCP flow also starts with a 3 way handshake (SYN, SYN ACK, ACK) to set up the connection.
The program uses the dpkt library, Ethernet that parses the packets, to look up packet's information.

Part A
The number of TCP flows initiated from the sender
In the program, a new TCP flow is created when the packet's source IP matches the specified sender, and when the TCP flag contains SYN. The flow is added to a list of TCP flows, and the number of TCP flows increases by 1.

For each TCP flow
(a) Getting first 2 packets and corresponding ack packets, after the TCP connection is set up (from sender to receiver)
The first 2 packets of the flow are found after the flow's connection has been set up (after the 3 way handshake). As the program reads the PCAP file packet by packet, the current packet is called "curr". If the packet's flag contains ACK, it goes through the TCP flow list. If the packet's TCP source port or destination port matches curr, then it is known which flow this packet belongs to. The number of packets sent for each flow is kept track of, so if it is the first two packets, the number of packets in the flow would be <= 2.
The first 2 packet's corresponding acks would be the first two acks with the packet's IP source that matches the specified receiver. If the packet's flag contains ACK, it goes through the TCP flow list. If the packet's TCP source port or destination port matches curr, then it is known which flow this packet belongs to. The number of acks received for each flow is kept track of, so if it is the first two packets, it is the corresponding ack packets.
(a) packets … Sequence number, Ack number, Receive Window size
The sequence number of the packet is gotten using the dpkt library, Ethernet, calling the packet's tcp.seq.
The ack number of the packet is gotten using the dpkt library, Ethernet, calling the packet's tcp.ack.
The receive window size is calculated:
receive window size = window size * window size scaling factor

The ack number of the packet is gotten using the dpkt library, Ethernet, calling the packet's tcp.win.
The window size scaling factor is calculated by getting the packet's TCP options, with the use of the dpkt library. In the list of options (list of tuples), the window scaling size factor is the tuple, with the first number as 3. This value is in byte format, so it is converted to an int. Then the value must be left shifted with 1.

(b) Sender throughput (TCP level)
Total amount of data sent by the sender, between sending the first byte to receiving the last acknowledgement. A flow's sender throughput is calculated with packets that have the IP source the same as the specified sender, getting the total number of bytes sent divided by seconds. The total number of bytes sent is calculated by adding the packet's length of TCP.data + length of the TCP header data to the flow's throughput value. The TCP data and header is found using the dpkt library. To calculate the seconds that past, the timestamp of when a flow is first created, SYN to the last acknowledgement, FIN. End time - start time = number of seconds past.

Part B
Congestion Control
(1) Print first 5 congestion window sizes
Calculating the congestion window size is done by keeping track of the number of packets in flight - packets that have been sent but not have been acknowledged yet. This is because the cwnd is the number of packets that the sender can send before receiving an ACK.
The icwnd, initial cwnd, is calculated after the SYN ACK in the TCP 3 way handshake set up of the flow. This is because the last ACK in the 3 way handshake can have data in it. Then the following ACK packets sent are counted until the first ACK packet is received. The number of packets that were sent before the ACK packet received is the icwnd. The following cwnds are calculated similarly. The number of packets sent is counted before the number of previous cwnd value ACK packets are received.
The program shows that the first 5 congestion sizes are roughly doubling. It seems to be in the slow start phase, with the cwnd increasing greatly.

(2) The number of times a retransmission occured due to triple duplicate ACK and the number of times a retransmission occurred due to timeout.
The total number of retransmissions is calculated, which is done if the previous packet's sequence number is greater than the current packet's sequence number. This means that the packets are sent out of order and there is a retransmission. Triple duplicate ACKS are calculated by checking if the last three packets have the same ACK value, with a counter. If the previous packet's ACK == current packet's ACK. When the number of triple duplicate ACK is calculated, then the number of times a retransmission occured due to a timeout is calculated by total_number_of_retransmissions - triple_diplicate_acks.