Problem 1 Let's assume a RED congestion control, having the following parameters:

- `MaxP`=0.4
- `MinTh` = 4
- `MaxTh` = 10

Calculate the following probabilities:

1. Probability of a segment being dropped with a `AvgLen`=8
2. Probability that 3 consecutive segments enter into the queue, assuming that all of them find the same average queue length (`AvgLen`) of 8
3. Same probability as previous point (2) assuming a modified RED congestion control where the probability of a segment being dropped is $\frac{P}{1-\texttt{compt}\cdot P}$, being P the same probability computed at point (1). `compt` is the number of segments that entered into the queue from the last dropped segment. Assume `compt` = 0 for the first segment entering into the queue.

(0.5 points)

Problem 2 Write a Python script that returns the transmission order sequence of a Weighted Fair Queuing (WFQ) policy based on a list of triplets where each triplet represents:

1. Arrival time (`float`)
2. Packet length (`float`)
3. Flow/stream identifier (`integer` ≥ 1)

As an example,

```
0.1 1.0 1
0.2 2.1 2
...
```

ordered in time.
The script must be provided with two parameters:

- Fraction of the bandwidth assigned to each flow (as a percentage). Comma separated. As an example: `50,10,40`
- File name containing the list of triplets to be scheduled

(1 points)

Problem 3 Build a simulation scenario with the same topology as in cw1.tcl but with the following parameters:

- MSS: 1000 bytes
- `CWMAX`: 10 MSS
- Time resolution: 0.01 s
- Simulation time: 200 s
- UDP traffic activates 20 s after start and ends 20 s before ending simulation
- Links speed:
  - n0-n2: 250 Kbps
  - n1-n2: 250 Kbps
  - n2-n3: 50 Kbps
- Links delay
  - n0-n2: 20 ms
  - n1-n2: 20 ms
  - n2-n3: 0.5 s
- Traffic generator rates for both (CBR and exponential): 50 Kbps
- Node 2 buffer size: 20

You must obtain the transmissions, retransmissions and acknowledgments sent and received at n1. The inputs to be considered are both:

- The simulation trace.
- In order to determine whether a transmission is produced by time out expiration or not, it is required to know the estimated time out. For RFC793 TCP agents you can obtain it directly from the `rto_` variable. As such a variable no longer exists for TCP Reno agents, you have to derived it from `srtt_` and `rttvar_`. Take $\mu = 1$ and $\phi = 4$ for the Jacobson/Karels estimator.

Answer the following questions:

1. What two regular expressions allow to obtain:
   - Segments sent from TCP agent.
   - Acks received at TCP agent.
2. Using three different TCP agents:
   - RFC793 with original congestion control.
   - RFC793 with slow start.
   - Reno.

   Write a python script that plots the congestion window along the simulation time (take as input the simulation trace as well as the provided values for `rto_` or `srtt_` and `rttvar_`). Compare your plot against the value for `cw_` provided by the simulator. Both plots must match.

3. Show an example obtained from your simulation where fast retransmission is produced.

4. For each one of the three agents, explain which is the policy for retransmissions when the congestion window reaches its limit and can no longer budge.

5. Run a fourth simulation with TCP NewReno agent. Measure throughput and number of retranmissions from the simulation trace and compare it against TCP Reno.

6. For TCP New Reno show an example obtained from your simulation where partial Acks were observed.

7. Run a fifth simulation using TCP Reno but, in this case, use RED at n2 with:

   - `MinTh`: 10
   - `MaxTh`: 20

   Compare throughput and congestion window plots with TCP Reno without RED previous simulation.

(3.5 points)