Question 1
After having the agent choose randomly from the set of possible actions [None, 'forward', 'left', 'right'], I can observe that my agent (the red car) is now moving. Its choice of action being random, it only manages to achieve a performance of around 20% (when enforce_deadline = True), meaning it reaches the destination only around 20% of the time. This will be our baseline performance.

Question 2
The agent has access to these information at each intersection :
1) *nextway_point location, heading, current location*
2) *light, left, right, oncoming*

We are building a model of a cab that follows traffic rules and gets to a destination before a certain deadline; for that we need to pick a state so that it helps with:

- the navigation in order to learn how to get to the destination : the next_waypoint variable
- obeying the traffic rules : traffic lights, traffic from left and oncoming traffic (the traffic coming from the right is excluded because of US right of way rules)
- getting to destination before a certain deadline: for this I need to include the deadline variable but since it has up to 40 different values(which is huge and would blow up the state space if included), I chose to categorize it through a variable 'behavior' seperating the values of deadline into 3 zones : more than 38 (behavior = Patient) more than 14 (behavior = Normal) and less than 14 (behavior = Urgent)

Our state variable is then defined as : State(light, next_waypoint, left, oncoming, behavior)

Optional
We have a total of `2*4*3*3*3 = 108` possible states here, since :

- the light variable has 2 states,
- the nextway_point has 4 states,
- the left, the oncoming and the behavior variables all have 3 states each.

Question 3
After implementing the Q-Learning Algorithm, the performance of the agent has significantly increased as compared to the baseline performance: from around 20% it is now around 95%. This is due to the fact that the agent is now learning from its previous actions and trying to maximize the total Q value.
One particularly noticeable pattern in the agent's behavior while learning is that it keeps repeating the same action until the reward ceases to increase for that action and it seems to be finding a policy that works consistently only after three trials.
The vehicle now also seems to stop at red lights and gives right of way.

Question 4
As printed out in the stats section, the net reward with the basic Q-Learning is always a very large and positive number and the average of all remaining time after the agent reaches destination is also positive and around 18.

Alpha	Gamma	Epsilon	Net Reward	Performance

Question 5

After tuning the parameters we obtain the following results:

Alpha	Gamma	Epsilon	Net Reward	Performance
0.5	0.2	0.3	2100+	97 to 99%
0.7	0.3	0.7	2100+	94 to 96%
0.9	0.9	0.9	2100+	75 to 85%

Decreasing the different values of Alpha, Gamma, Epsilon seems to be increasing the performance up to a certain point. After many trials I've reported the three representatives of the three classes of performance values I seem to be getting. There certainly is an algorithm to find the best combination of values, in order to perform a much more credible tuning of parameters.

I have printed out the sum of all negative rewards and none of my trials ever keeps that value at 0 even when I obtain a Performance of 100%. An optimal policy must be one which application doesn't incur any penalty to the agent, one that keeps the total negative reward's value at 0. For example in my implementation, even when the agent seems to have found a working policy it stills chooses the wrong move sometimes; with an optimal policy it would not.