Reinforcement Learning Project Report
Nathan Mayer

Question 1: ***Observe what you see with the agent's behavior as it takes random actions. Does the smartcab eventually make it to the destination? Are there any other interesting observations to note?***

The smartcab behaves very erratically, as all of its decisions are being made randomly. Because of this random process, the smartcab does eventually find its way to the destination, not because of any intelligence, but because it is randomly navigating through a relatively small grid. Eventually it just happens to run into the destination by accident.

The car is frequently penalized for running into other cars and for not following the directions, but it also occasionally receives rewards for moving in the right direction. The car never learns from these rewards or penalties, however.

Question 2: ***What states have you identified that are appropriate for modeling the smartcab and environment? Why do you believe each of these states to be appropriate for this problem?***

I identified all of the input as appropriate for modeling the smartcab and its environment. In addition to the input provided, I also used the direction of the next waypoint to indicate which way the car should travel. While that multiplies the size of the state space by 3, it is necessary to help the car navigate to the destination.

I believe these states are appropriate for solving the problem because they all communicate useful information about the system in ways that add relatively little complexity to the state for predictions.

I considered adding the deadline to the state, but chose not to because it would multiply the size of the state space by around 40, while not communicating much useful information. The high weight of the reward for reaching the destination and the punishment for going the wrong way will implicitly encourage the agent to take the shortest path to the destination, without needing to include the deadline.

Question 3: ***What changes do you notice in the agent's behavior when compared to the basic driving agent when random actions were always taken? Why is this behavior occurring?***

When the simulation begins, the agent's behavior is very similar to the random actions taken by the previous agent. This is because all decisions are equally weighted and there is no reason to choose one direction over another. There are many punishments given and the agent cannot find the destination before the deadline.

As the simulation progresses, there are fewer and fewer punishments given and more and more rewards given. The agent can find the destination sometimes, and its driving pattern is not as erratic. This is because the agent is learning weights for its various transitions and is figuring out what transitions result in the biggest rewards (reaching the destination).
By the end of the 100-trial experiment, the agent performs much better than it did at the start, but often gets stuck at intersections, choosing not to make any moves. This is because it has weighted all other directions lower than the value of just sitting still. It doesn't always find the destination, but does find it fairly regularly. This is because the agent has learned the value of finding the destination, but isn't optimistic enough to find better ways of getting there.

Question 4: ***Report the different values for the parameters tuned in your basic implementation of Q-Learning. For which set of parameters does the agent perform best? How well does the final driving agent perform?***

I tried every combination of the following variables on 100-trial tests of the simulation:

- Initialization values = [0, 2, 5, 10, 20]
- alphas = [0.1, 0.3, 0.5, 0.7, 0.9]
- discounts = [0.1, 0.3, 0.5, 0.7, 0.9]

Sample Results:

Inputs			Outputs		
Initialization	**Alpha**	**Discounts**	**Wins**	**Reward**	**Punishment**
2	0.5	0.7	99	2304	-41
0	0.3	0.5	100	2210	-23
2	0.5	0.3	100	2296	-35
5	0.5	0.7	99	2250	-63.5

I chose to discount the estimate of maximum value for the next state by a factor of 0.5.
I use a learning factor alpha of 0.3.
I initialize the system neutrally, with every transition having a value of 0.
This set of parameters was the most performant set that I could derive. It produced results of:

- 100/100 destinations reached
- -23 points of punishment
- 2210 points of reward

There were other options which were close to this level of performance, but this was the combination of parameters which maximized success rate (# destinations reached) and minimized punishment (fewest wrong turns/infractions). There were other highly successful algorithms that earned more rewards, but rewards gained is more influenced by average distance to destination than the other parameters, and can actually be increased by taking a wrong turn (making the route longer). As such, I chose not to optimize for points earned.

Near the end of the simulation, the agent has learned weights for most or all of the possible states of the system. It makes it to the destination nearly every time and makes very few invalid moves. At this point, the agent has a good model of the system, and can find its way to the destination safely and reliably.

Question 5: ***Does your agent get close to finding an optimal policy, i.e. reach the destination in the minimum possible time, and not incur any penalties? How would you describe an optimal policy for this problem?***

An optimal policy for this problem would be a policy which does not result in any traffic violations, takes the shortest route possible (according to the route planner), and always gets to the destination on time. Because of the large reward for getting to the destination on time, there would sometimes be a case where it is most optimal to incur a penalty, if there is little time remaining and there is another car that is preventing you from taking the shortest path.

My agent comes very close to finding an optimal policy in the 100-trial runs. It very rarely incurs a traffic penalty and always reaches the destination. It almost always takes the shortest path in the last 50% of trials.