QUESTION: 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?
When it's totally random, it seems to make its way to the destination more than half the time, but it hits the hard time limit of -100 sometimes as well. As it's entirely random, it does not care about traffic or the lights, making it an incredibly dangerous driver. Unless every other driver on the road is incredibly skilled, this car is not likely to make it to its destination in one piece!

QUESTION: 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?
Almost all of the input is logically necessary for the state to function, except for the deadline variable. Because it changes depending on the number of moves made, it makes for a poor teacher as far as state goes. The learner is very likely to encounter the same combination of the other variables (oncoming, left, right, and light), but rarely will it encounter the same combination of the others *and* the deadline, making for an exceedingly poor learner. Kind of an equivalent to overfitting. However, adding on one more variable, next_waypoint, should be a good thing, because we want the qlearner to associate good things with getting closer to the target position.

QUESTION: How many states in total exist for the smartcab in this environment? Does this number seem reasonable given that the goal of Q-Learning is to learn and make informed decisions about each state? Why or why not?
The 1st, 3rd, 4th, and 5th variables all have 4 possible states. The 2nd variable has 2 possible states. So, the number of possible states should be 4^4 * 2^2, or 1024 possible states.

QUESTION: 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?
I notice that it fails a lot less. On top of that, it gets better over time; on the 1st run, it's not much better than a random actor, but as it goes on, it improves until it's near perfect. This behaviour is occurring because it's learning the appropriate action for every possible scenario, and takes into account the next_waypoint, which will always points to the best way to get to the destination.

QUESTION: 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?
See the tally folder for more details, but the 2 sets of parameters that performed best were:
gamma = 0.0001
alpha = 0.0001
epsilon = 1.0001

avg success = 68.95%
and
gamma = 0.1
alpha = 0.1
epsilon = 1.1
avg success = 67.5%
In the tally folder, you'll find output from 64 runs of each set of parameters. The 64 lines of numbers in a row are the number of times a given run succeeded out of 100 trials. At the end, I took the average, and got the percentages you see above. The learner performs not as well as I had hoped it would; at only ~70% success, this is not fit for the road, although further tweaking failed to make it better.

QUESTION: 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?
It does get closer than any random policy, but it's not near 100%. I would describe an optimal policy as a series of if statements, described below:

```
if traffic left, right, forward is clear, and light is green:
   next_waypoint
elif traffic left and right are clear, and light is green:
   next_waypoint, excluding left
elif traffic left is clear, and light is red:
   next_waypoint, if right
```

and so on.