While I spent much longer than 80 minutes on Rush Hour, I was ultimately unable to get it to work. I created a car class in order to keep track of the different pieces that would have been in the given input, which had attributes such as a label, X coordinate, Y coordinate, direction the car is facing, and whether the car was a 1x2 or a 1x3 piece. I also had methods that would change the current X or Y coordinate of the car. Dependent on what direction it was facing, it would either move the object horizontally or vertically. However, thinking about the class structure again, I am now unsure if that method was really necessary, as I was also planning to simply create a new list of cars that would track all the positions of the cars in the children state. That would make it unnecessary to manually update the position, even though it would definitely take less memory. Regardless, as I was unable to get to that step, I can not really predetermine what would be necessary or not, as original plans can change very easily. I was also able to create a format where cars could be identified through a label, or a one digit number. This method is definitely not ideal, though, as it relies on the fact that there has to be less than or equal to 10 cars on the given input board in order for the code to work. If I ever continue working in this lab later, I think it would be a viable option to use letters of the alphabet to label car objects as well. This would allow for much more car pieces on the input board, as well as a simple way to distinguish each car object from each other. I would also have to slightly change my current code to find all the car objects from the given board, but I think it would be a worthwhile next step.

Some successes that I was also able to encounter, despite not being able to finish this lab, were creating a method to show the puzzle, which was easily adapted from my code from Sliding Puzzles 1 and 2. I implemented keyword arguments for the first time, too, which were interesting to use. I was also able to create the correct children states of cars; at least, for all cars that were facing an horizontal direction. I had much more trouble with creating children states of vertically facing cars, which was also where I chose to stop coding. I tried to use a similar algorithm for getting the children states of horizontally facing cars for vertically facing cars, but I was unable to do so. It would simply output strings with more characters than the original string had, or would return strings with characters in the incorrect order. I think it is not working because my current method, while it is able to correctly move the position of the car, doesn't change the rest of the characters. This would definitely result in additional characters being added to the string. I would need to change my current code so it would take into account other cases when modifying the parent state. Once I get my get_children method working, it would be very simple to solve the lab, as the rest of the lab is based off the code I had done for Sliding Puzzles 1 and 2.