Name: Avnish Sengupta

UCLA ID: 705299746

1. There were a few obstacles that I faced while writing and testing this program:
   1. The nesting of the if-else statements to establish a logical flow of control. Due to the enormous number of choices and permutations that were possible from the input, the if-else statements had to be nested a particular way to work. Initially, I tired accounting for every unique choice that could be inputted by nesting an if statement in another if statement depending on what is picked. For example, if ‘3’ was selected for the model and then “Long Range” was selected for the kind, then there would end up being another nested if statement for every type of different color nested inside kind and another if statement nested inside color for wheels and so on… Such a program deemed unfeasible to code due to the large number of possibilities. So, I studied the dependency of the inputs of the different question to one another. I realized that the prices of every individual question only changed with the model choice. However, once a model choice was picked, every choice had a unique price and the C++ compiler jumps from one if-statement to another. So, I nested the if-else statements for kind, color, wheel, leather, and auto-pilot inside every respective model if-statement but kept them separate inside the model blocks.
   2. The inputs in connection to the error message was hard to format due to the variety in inputs. Other than the normal inputs possible, the user could input whitespaces either between the input or after the input. Coming up with code to tackle both the bugs was hard. Using cin followed by cin.ignore would eat up the whitespace that occurs after the input and still cause the program to compile correctly by registering the input. Using getline would solve the problem of having whitespaces between the input as it would eat the whitespace and register the input as one string. However, neither solves both problems. On asking Professor Stahl, he preferred the getline approach over the cin one and so that was the approach I employed in my code.
2. Test data:
   1. Normal input to see if the program works: (‘X’, “Long Range”, “Red”, 22, “Yes”, “Yes”)
   2. Incorrect color input to test if the error message prints: (‘3’, “Performance”, “Blue”, 18, “Yes”, “Yes)
   3. String input for wheel to check if the program crashes or not. The program crashed.: (‘X’, “Long Range”, “Grey”, “hi”, “Yes”, “Yes”)
   4. Incorrect leather input with spaces between the input to check if an error message is returned: (‘S’, “Performance”, “Black”, 19, “Y e s”, “No”)
   5. Incorrect auto-pilot input to check if the error message works: (‘X’, “Long Range”, “Blue”, 20, “No”, “life”)
   6. Incorrect kind input to check if the error message is printed: (‘3’, “fruit”, “Grey”, 18, “Yes”, “Yes”)
   7. Incorrect model input and color input to check if the appropriate error message is returned: (‘F’, “Long Range”, “Yellow”, 18, “Yes”, “Yes”)