1. Rubric
    1. The Model - Student describes their model in detail. This includes the state, actuators and update equations.
        1. Model Predictive Control (MPC) involves simulating actuator inputs. This process is done several times resulting in several trajectories. The trajectory associated with the lowest cost is chosen. The simulated trajectories are based off the initial state and the model's dynamics, constraints and, of course, the cost function
        2. This model is bit better than PID and LQR controllers which do not have this predictive ability.
        3. This project makes use of Kinematic and dynamic vehicle model for predictive control.
        4. Kinematic model is simpler but ignores gravity, longitudinal and lateral forces, mass, tire forces, inertia, air resistance, drag, as well as mass and geometry of vehicle. As such it is good first step to under the framework for MPC to control the car speed and steering. The model is tested in the SDC simulator (V 1.45) to visualize both the car reference path and the MPC trajectory path.
        5. Output from the simulator is the Json data containing waypoints (x, y), car position (x, y) car angle and speed.
        6. polyfit() to determine the coefficient. Invoked polyeval() to determine CTE, steering error and car velocity.
        7. The MPC implementation is in Solve()/solve() functions, which takes the state vector and coeffs along with constraints (steering angle can be between 30 and - 30 deg, and acceleration of 1 and -1) to determine the next steering angle and throttle. The steering and throttle values are passed back to the simulator via Json data to drive the car.
        8. Additionally, as part of visualization, codes were added to display the connected point paths in the simulator by sending a list of optional x and y values to the mpc_x, mpc_y (yellow line), next_x, and next_y (green line) fields
        9. equations for state, actuator and update are:
            - Xt+1 = Xt + Vt * cos (PSIt) * dt
            - Yt+1 = Yt + Vt * sin (PSIt) * dt
            - PSIt+1 - PSIt + Vt/Lf * Deltat * dt
            - Vt+1 - Vt + At * dt
            - CTEt+1 = f(Xt) - Yt + (Vt * sin(ePSIt) * dt)
            - ESPIt+1+1 = PSIt -PSIdest + (*Vt/Lf * Deltat * dt)* dt
    2. Timestep Length and Elapsed Duration (N & dt) - Student discusses the reasoning behind the chosen N (timestep length) and dt (elapsed duration between timesteps) values. Additionally the student details the previous values tried.
        1. The SDC QA session suggested N of 10 and dt of 0.1 (prediction make every sec) are optimal.
        2. N = 10 was chosen to avoid taking too long to calc thetrajectory path. Value of 20 takes longer to compute and tend to be off track.
        3. If N 20/dt 0.1 , overcorrected and went off track on first turn.

    4. If N 5/dt 0.1 - resulted in off track upon starting.
    5. If N 10/ dt 0.2 car wandered more and almost went off track off the shoulders on tight turns. But finish the lap.
    6. If N 10/dt 0.05 car wandered off track after second turn
    7. If N 20/dt 0.05 = car went off track on first turn
3. Polynomial Fitting and MPC Preprocessing - A polynomial is fitted to waypoints. If the student preprocesses waypoints, the vehicle state, and/or actuators prior to the MPC procedure it is described.
    1. Transform waypoints to be from car's perspective/ (main.cpp lines 107-116) This means we can consider px = 0, py = 0, and psi = 0 , this greatly simplifying future calculations
    2. Also shift reference angle to 90 deg. Help polynomial fit (same as horizontal line) And MPC eval and CTE. Else will have to do some other additional math.
4. Model Predictive Control with Latency - The student implements Model Predictive Control that handles a 100 millisecond latency. Student provides details on how they deal with latency.
    1. At time great than 0, use previous actuation (to account for the latency). The latency accommodation is performed on both time t > 0 second (MPC.cpp lines 128-131), and on speed based on projected path angle (MPC.cpp line 78).