ENGEN103-20A
ASSIGNMENT – PART THREE

Chris Penno - 1519526 & Ethan McKee-Harris - 1536943

Design a Test Plan:

The beginning of our testing design process revolves around the clarification of each function created in the traffic signal simulation. While we would like to believe that we are fully understanding of the intricacies of the controller's code, there will always be elements for which one is more knowledgeable. To eliminate any confusion and allow us to work cohesively during the testing phase and create a testing plan that covers all variables and potential issues. This comes down to deciding what unit tests we will need to create.

We have no way of testing the hardware elements of the intersection, nor how the data integrates itself into the format the controller has been coded to use. This does present a potential issue as engineers. The Middlesbrook City Council (MCC) and Urbano will need to understand that we cannot facilitate testing for these elements of the system. We trust in our testing procedures that these connections work and function as outlined in Urbano's brief.

Due to the technical difficulties involved with testing a few of our functions, we have decided it is more efficient and better to create a new file and put in the same functions. However, we will be modifying the functions, so they are suitable for a testing purpose. I.E. instead of changing the colour of the lights, it returns a tuple with the lights that would be turned on.

We have created a several new functions to help us out in this project, despite testing throughout the coding process. We will still need to create unit tests for all these methods, as this will ensure adequate coverage to the point where Urbano can be satisfied with our test coverage. For all our new functions which do not take input and do set things, we can simply run some simple tests on the values changed. Whereas for our more complex methods where we must pass values, we will conduct further testing to ensure that we do not just test one single value. We have tested enough values to say we have adequate coverage for that function before moving onto the next.

When testing functions, we also considered testing invalid input, as we cannot expect to always work with perfect values, however, the program can handle it. Given there is no user input into the system once it starts, we do not need to extraneously test for outside values as all function calls are hardcoded with expected values.

TEST PLAN (TABLE):

These tables are sorted to near alphabetical order, rather than the order they appear in the code. This is for ease of reading.

Test Name	***test_TLAttr***
Test Description	*Tests initial attributes are set correctly in the TrafficLight class*
Function	*whatson()*
Inputs	*None*
Outputs	*[True, True, False, False, False, False]*
Errors	*None, the supplied code works as expected*

Test Name	***test_bulkWaitChanges***
Test Description	*Tests both the IncrementAllWaits() and ResetAllWaits() methods from the Controller class, as well as initialized values*
Function	*IncrementAllWaits(), ResetAllWaits()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_carAttr***
Test Description	*Test our Car() initialization works as expected*
Function	*Car() __init__*
Inputs	*Car("Car1", "WEST", "LEFT")*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_carDisplayDirLeft***
Test Description	*Tests the display direction from a car instance, this tests the left value*
Function	*car.displayDirection()*
Inputs	*Car("Car1", "WEST", "LEFT")*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_carDisplayDirRight***
Test Description	*Tests the display direction from a car instance, this tests the right value*
Function	*car.displayDirection()*
Inputs	*Car("Car1", "WEST", "RIGHT")*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_carDisplayDirStraight***
Test Description	*Tests the display direction from a car instance, this tests the straight value*
Function	*car.displayDirection()*
Inputs	*Car("Car1", "WEST", "STRAIGHT")*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_carStr***
Test Description	*Tests the Car class str function*
Function	*str(Car)*
Inputs	*str(Car instance)*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_controllerAttr***
Test Description	*Tests the Controller class init values are correct*
Function	*TrafficLightController("WEST")*
Inputs	*getName() & getTl()*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_controllerDisplay***
Test Description	*Tests the simulator output is correct*
Function	*simulator.controllerDisplay()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_countAllCars***
Test Description	*Used to test the CountAllCars() method and that it can count accurately*
Function	*simulator.CountAllCars(lane)*
Inputs	*1 Lane, (northlane, southlane, eastlane, westlane)*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_countCars***
Test Description	*Tests the direction car counting method*
Function	*simulator.countCars*
Inputs	*(lane, direction)*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_displayCars***
Test Description	*Tests the cars display output is correct*
Function	*simulator.displayCars()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_displayLightOff***
Test Description	*Test light display function*
Function	*trafficLight.displayLight()*
Inputs	*(False, 'T')*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_displayLightOnS***
Test Description	*Test light display function*
Function	*trafficLight.displayLight()*
Inputs	*(True, 'S')*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_displayLightOnT***
Test Description	*Test light display function*
Function	*trafficLight.displayLight()*
Inputs	*(True, 'T')*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_dualWaitIncrement***
Test Description	*Test the method handling incrementation of both Controller class instance wait times*
Function	*controller.IncrementBothWaits()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_dualWaitReset***
Test Description	*Tests the method handling resets for both wait time variables*
Function	*controller.ResetBothWaits()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_generateCars***
Test Description	*Tests simulator car generation*
Function	*simulator.generateCars()*
Inputs	*(number of cars to generate)*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_greenOff***
Test Description	*Tests the traffic light method for turning the green light off*
Function	*trafficLight.greenOff()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_greenOn***
Test Description	*Tests the traffic light method for turning the green light on*
Function	*trafficLight.greenOn()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_greenStr***
Test Description	*Tests the light display functions str method*
Function	*trafficLight.greenStr()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_lightSignalWaitLogic***
Test Description	*Tests the logic behind our wait time logic in lightSignals()*
Function	*testFunctions.lightSignals() -> simulator.lightSignals()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_lightSignalWaitLogicTwo***
Test Description	*Further border value testing for our wait time logic*
Function	*testFunctions.lightSignals() -> simulator.lightSignals()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_lightsBrokenLogic***
Test Description	*Used to test our logic for broken lights*
Function	*simulator.sensor()*
Inputs	*(controller, lane)*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_lightsBrokenVar***
Test Description	*Tests initialized values are correct*
Function	*Variable -> simulator.lightsAreBroken*
Inputs	*simulator.lightsAreBroken*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_orangeOff***
Test Description	*Tests the traffic light method for turning the orange light off*
Function	*trafficLight.orangeOff()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_orangeOn***
Test Description	*Tests the traffic light method for turning the orange light on*
Function	*trafficLight.orangeOn()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_orangeStr***
Test Description	*Tests the light display functions str method*
Function	*trafficLight.orangeStr()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_otherInitWait***
Test Description	*Tests initialized values for our other wait time instance variable*
Function	*controller.GetRightWait(), controller.GetOtherWait()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_partialBrokenCycle***
Test Description	*Test our broken cycle logic, minus the timings*
Function	*trafficlight.whatsOn(), SwitchOff(), orangeOn(), torangeOn()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_partialOtherCycle***
Test Description	*Tests the first set of logic changes in our other cycle*
Function	*switchOff(), greenOn(), tredOn()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_partialOtherCycleTwo***
Test Description	*Tests the second set of logic changes in our other cycle*
Function	*switchOff(), orangeOn(), tredOn()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_partialRightCycle***
Test Description	*Tests the first set of light changes in our right cycle*
Function	*switchOff(), redOn(), tgreenOn()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_partialRightCycleTwo***
Test Description	*Tests the second set of light changes in our right cycle*
Function	*switchOff(), redOn(), torangeOn()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_phaseGreen***
Test Description	*Test the phase green method*
Function	*controller.phaseGreen()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_phaseOrange***
Test Description	*Test the phase orange method*
Function	*controller.phaseOrange()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_phaseStop***
Test Description	*Test the stop phase method*
Function	*controller.phaseStop()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_redOff***
Test Description	*Test the trafflight class's redOff method*
Function	*trafficLight.redOff()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_redOn***
Test Description	*Test the trafflight class's redOn method*
Function	*trafficLight.redOn()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_redOff***
Test Description	*Test the trafflight class's redStr method*
Function	*trafficLight.redStr()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_removeCars***
Test Description	*Test removal of cars from a lane*
Function	*simulator.removeCars()*
Inputs	*(lane, direction) -> (eastLane, "LEFT")*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_sensorAllGo***
Test Description	*Tests the automated sensor for car removal with green lights*
Function	*simulator.sensor()*
Inputs	*(controller, lane)*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_sensorAllOrange***
Test Description	*Tests the automated sensor for car removal with orange lights*
Function	*simulator.sensor()*
Inputs	*(controller, lane)*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_sensorRightGo***
Test Description	*Tests the automated sensor for car removal with right turning traffic*
Function	*simulator.sensor()*
Inputs	*(controller, lane)*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_sensorRightOrange***
Test Description	*Tests the automated sensor for car removal with right turning traffic and an orange light*
Function	*simulator.sensor()*
Inputs	*(controller, lane)*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_sensorStraightGo***
Test Description	*Test the automated sensor for car removal on other traffic*
Function	*simulator.sensor()*
Inputs	*(controller, lane)*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_sensorStraightOrange***
Test Description	*Test the automated sensor for car removal on other traffic with an orange light*
Function	*simulator.sensor()*
Inputs	*(controller, lane)*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_tgreenOff***
Test Description	*Tests the right green light off method*
Function	*trafficLight.tgreenOff()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_tgreenOn***
Test Description	*Tests the right green light on method*
Function	*trafficLight.tgreenOn()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_timer***
Test Description	*Test that our timer is correct at initialization*
Function	*simulator.timer()*
Inputs	*Captured stdout*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_torangeOff***
Test Description	*Tests the right orange light off method*
Function	*trafficLight.torangeOff()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_torangeOn***
Test Description	*Tests the right orange light on method*
Function	*trafficLight.torangeOn()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_tredOff***
Test Description	*Tests the right red light off method*
Function	*trafficLight.tredOff()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_torangeOn***
Test Description	*Tests the right red light on method*
Function	*trafficLight.tredOn()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_wait***
Test Description	*Tests our wait increment methods*
Function	*controller.IncrementRightWait(), controller.ResetRightWait()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Test Name	***test_waitReset***
Test Description	*Test our wait reset methods*
Function	*controller.ResetRightWait(), controller.IncrementRightWait(), controller.ResetOtherWait(), controller.IncrementOtherWait()*
Inputs	*N/A*
Outputs	*Expected values*
Errors	*None, code works as expected*

Considerations:

We did not find many issues after we created our unit tests. This was due to testing throughout development. We did have a couple of errors in testing, one of these errors was differences in ‘stdout’ because I had changed how we were displaying car directions. This was quickly rectified by changing the expected output within the tests.

Another issue was the timer; again, easily fixed. Finally, we had errors in how lanes were printed. This was corrected by altering what the tests were expecting. , Rather than change the code, we changed the tests because we had changed the code itself to improve readability. The tests did not reflect this change accurately, and as a result, we felt happy to alter the test parameters.

Conclusion:

We have now provided Urbano with comprehensive designs, code, and proof testing for our controller. This concludes our involvement with the project. We wish the MCC the best with their implementation for the intersection and their future endeavours.