Deliverable 4

With all of these test cases, we had the following amounts testing various elements of the code base.

- Parsing test cases: eight test cases
- Parsing a single line: five test cases
- Storing a value: two test cases
- Various pieces of driver execution (see test plan): eight test cases

With the test cases we had, we could adequately test our own testing framework. However, testing our own framework presented some interesting challenges that we may not have expected. One of these challenges, perhaps the greatest, was how to handle executing test cases that we knew would break our code while still allowing our code to continue to execute the rest of the test cases. To overcome this, we decided to create mock instances of the classes that we specified in our test cases. For example, the driver object would create another instance of itself, so that it could execute test cases that would halt execution of the sub-driver while the super-driver could continue to execute. This was also applied to test case objects. A valid test case would create a test case object, but that test case object would contain an invalid test case that would throw the errors we expected.

Another challenge was how to write test cases with bad syntax so we could test parsing, but these, again, should not halt execution of our super-driver. To overcome this, we simply separated the “broken” test cases from the valid test cases that we were actually using. The valid test cases would simply contain a path to the bad test cases. This is not unlike a user would interact with this software, as the path would change for any test case depending on the class and method the user was writing the test case for.

We were able to get all of these test cases successfully implemented after careful work and great determination. This deliverable was likely the most challenging out of all of the deliverables, because it included the entirety of our design and implementation.

Our testing framework was coming along nicely, and working successfully. All we had left to do was inject faults into our code base.