GLASS Test Plan Document

Tests for Functional Requirements:

Unit Tests

- For each major component of the project (the syntax specification reader, lexer, parser, parse tree to XML converter, and macro system), unit tests will be written to ensure our code is written properly and is working as intended.
- Unit tests will be written not just for each component within the program, but in some cases also for each individual method within a component.
- Unit tests will be split into two groups: general use and edge cases. General use tests will simply be tests containing relatively standard or expected input. Edge case tests will contain uncommon or otherwise unexpected inputs to ensure that the program does not crash in the event that the user inputs something unexpected, but rather that the error is correctly handled.

Tests for Interface Requirements:

Command Line Input and Output

- Similar to the unit tests above, a series of test input files will be created to ensure that the command line interface works under a variety of conditions.
- Tests will include input files containing standard/expected inputs, edge case correct inputs, and incorrect inputs. We want to ensure that all inputs, regardless of if they are correct or incorrect, are handled in some way beyond the program simply terminating.
- Test files will be made for both syntax specification files, corresponding source code files, and macro files.

Command Line Automation

- Tests will be made to ensure that the program can complete certain tasks semi-automatically when provided with command line arguments. This includes automatically reading a syntax specification file, source code file, and macro file all without intermediate user intervention.
- In cases where an error may occur, we will ensure that a proper error message is thrown, and the program does not simply crash.

GUI Input and Output

- While more difficult to test (since it cannot be automated easily) we will test our GUI with a variety of inputs.
- We will accumulate a list of inputs to test within our GUI application to ensure that as the GUI is developed that all previously implemented features remain functional and working as expected.

Tests for Performance Requirements:

Time efficiency tests:

- Several tests will be created for the sake of evaluating how the program performs time-wise in several different scenarios. These scenarios may include a large syntax specification with a small source code file, a small syntax specification with a large source code file, a large syntax specification with a large source code file, and a small syntax specification with a small source code file.
- Several runs of different instances of each scenario will be run in order to gather sufficient data for analysis.
- For consistency, all time efficiency tests will be executed on code01.

Memory efficiency tests:

- As with time efficiency, several tests will be created for the sake of evaluating how to program performs in terms of memory consumption in several different scenarios, including the ones described above for time efficiency tests.
- Several runs of different instances of each scenario will be run in order to gather sufficient data for analysis.
- All memory efficiency tests will be run on the same computer to ensure consistency.

Tests for Other Requirements:

Compatibility

- For compatibility, we will make sure to run all of our tests on both Windows and Linux to ensure nothing works unexpectedly on a particular operating system.

Intuitiveness

- To gauge intuitiveness, we will perform user tests to see how quickly users are able to perform certain tasks using our program.
- We will first give users a quick survey to gauge how familiar they are with some of the concepts they may need to work with within our tool (grammars, regex, XML)
- Users will be assigned a set of tasks, which they will have to perform one by one. They will have the program documentation available to them to assist them in completing the tasks.
- Completion times for each task will be grouped based on the familiarity the user had with the concepts necessary for said tasks. Then, the completion times for each task will be analyzed.