Sprint 1: QA Language, By Dmitry Vasin and Ben Ni

File locations

The parser file is a CUP file located at src/Parser/qatest.cup

The scanner file is a JFlex file located at src/Scanner/qatest.flex

Test inputs are src/TestInputs

Build script is buildV2.bat

Intermediate Representation

Syntax tree represented in XML format.

Language structures not parsed

All of the language parts are parsed except non-primitive data types. Every other operation is tested in test case 15 in the example inputs. We did not parse non-primitive data types because the syntax for them is not yet set in stone.

Example inputs provided

The test case are written so that they separately test an independent function of the language and whether it is being parsed correctly.

Test case 0: Simplest test case, one action regarding object with one descriptor

Test case 1: Step with results, result and should statement checking

Test case 2: Combining action on object with result checking

Test case 3: Adding together numbers from multiple sources

Test case 4: Actions requiring arguments, like editing a textbox

Test case 5: Multi step programs

Test case 6: If statement

Test case 7: If statement with an otherwise clause

Test case 8: Go to statements

Test case 9: Loops

Test case 10: Exit statement

Test case 11: References to earlier steps and multi step result checking

Test case 12: Testing contain statement in should clause

Test case 13: Functions

Test case 14: Concatenating strings acquired from objects

Test case 15: Big program that combines everything

Running the script and verifying output files

The script is a .bat file so it has to be run on a Windows system. Simply run the script called buildV2.bat and the outputs for the test cases will be located in the outputs folder. There is an html representation numbered accordingly to the test case and an xml representation of the output as well. To verify the output files you can look at the html which is a human readable representation.

Expressiveness:

There are no warnings when the code is compiled and no ambiguity is there.

Correctness:

The code implements the input language as test case 15 implements every single aspect of the language that we wish to include, other than non-primitive types. Since it is being parsed correctly, the language is highly likely to be parsed correctly. Test case 15 adheres to our description in the language specification.

Verifiability

Correctness can be verified as all the output files are human readable. The examples provided are complete as they individually test every aspect of the language, and test case 15 tests them all combined.