Informatics 115 – Fall 2018
Professor Iftekhar Ahmed
Final project
Assigned: Tuesday, November 27th, 2018
Due: Wednesday, December 12th, 2018 at 11:59PM

This project builds upon your work from Homework 4. The same Defects4J data set will be used for this assignment. Everyone is assigned a specific bug (same as assignment 4).

The goal of this project is to use Tarantula Approach for fault localization. To apply the Tarantula fault localization we need three information.

• failed(e) = # of tests covering e that fail
• passed(e) = # of tests covering e that pass
• totalfailed, totalpassed = what you'd expect

The following commands will help to get started towards collecting these information.

1. *defects4j checkout -p Lang -v 1b -w /home/iftekha/lang_1_buggy* (Assuming that you checked out in /home/iftekha/lang_1_buggy)
2. *cd /home/iftekha/lang_1_buggy*
3. *defects4j info -p Lang -b 1*
4. *defects4j export -p tests.all* ***[prints all tests]***
5. *defects4j test*

Following is the output

```
###############################################################################
Running ant (compile.tests)..... OK
Running ant (run.dev.tests)..... OK
Failing tests: 4
  - org.apache.commons.lang3.LocaleUtilsTest::testParseAllLocales
  - org.apache.commons.lang3.math.NumberUtilsTest::TestLang747
  - org.apache.commons.lang3.time.FastDateFormat_ParserTest::testParseZone
  - org.apache.commons.lang3.time.FastDateParserTest::testParseZone
###############################################################################
```

6. By this point we have all tests and failing test. By taking the difference between the two lists we can identify the passing test cases. So the next step is to collect coverage information after executing each of the test case. The way to execute an individual test case is by following:

6a. defects4j coverage -t org.apache.commons.lang3.LocaleUtilsTest::testParseAllLocales
6b. cat coverage.xml ***[this file will give coverage information]***
6c. Extract the coverage information for each statement to use in calculating suspiciousness score
6d. > coverage.xml ***[Empty the coverage file]***

The same approach can be used for executing the other failing test cases and collect information:
7a. defects4j coverage -t org.apache.commons.lang3.math.NumberUtilsTest::TestLang747
7b. cat coverage.xml [this file will give coverage information]
7c. Extract the coverage information for each statement to use in calculating suspiciousness score
7d. > coverage.xml [Empty the coverage file]

8a. defects4j coverage -t org.apache.commons.lang3.time.FastDateFormat_ParserTest::testParseZone
8b. cat coverage.xml [this file will give coverage information]
8c. Extract the coverage information for each statement to use in calculating suspiciousness score
8d. > coverage.xml [Empty the coverage file]

9a. defects4j coverage -t org.apache.commons.lang3.time.FastDateParserTest::testParseZone
9b. cat coverage.xml [this file will give coverage information]
9c. Extract the coverage information for each statement to use in calculating suspiciousness score

9d. > coverage.xml [Empty the coverage file]

The tests.all command provides the output including the test method name
(i.e. org.apache.commons.lang3.tuple.TripleTest), so to execute passing test cases, you have to change the format (i.e. org.apache.commons.lang3.tuple::TripleTest) and repeat the above procedure for all passing tests.

10a. defects4j coverage -t org.apache.commons.lang3.tuple::TripleTest
10b. cat coverage.xml [this file will give coverage information]
10c. Extract the coverage information for each statement to use in calculating suspiciousness score
10d. > coverage.xml [Empty the coverage file]

11. Calculate suspiciousness score.

Upload Instructions
Turn in the following zip files and regular files combined in one zip file:
1. 1 zip file with all coverage.xml for passing test cases named: PassingXML_"**Projectname"_"bugnumber".zip**
2. 1 zip file with all coverage.xml for failing test cases named: FailingXML_"**Projectname"_"bugnumber".zip**
3. 1 zip file with all scripts (any language) used throughout the whole process (parsing the xml file, calculating the suspiciousness score etc.) named: Scripts_"**Projectname"_"bugnumber".zip**
4. A csv file with line numbers and suspiciousness scores for the buggy class in the following format.

Project	Bug id	Line number	Suspiciousness score

5. Compare the highly suspicious lines with the actual lines where the bug existed and report that in Summary.txt (Does the highly suspicious lines contain the buggy lines etc.)

Note: We will be grading based on the successful completion of steps. In addition to that, we will also look for exhaustiveness (have you executed all passing test cases, all failing test cases etc.)