CSE410 aka CSE306
Software Quality

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall

http://www.cse.buffalo.edu/faculty/alphonce/SP17/CSE410
https://piazza.com/class/iybn33z3aro2p
Recall the rules

1. Understand the requirements
2. Make it fail
3. Simplify the test case
4. Read the right error message
5. Check the plug
6. Separate fact from fiction
7. Divide and conquer
8. Match the tool to the bug
9. One change at a time
10. Keep an audit trail
11. Get a fresh view
12. If you didn't fix it, it ain't fixed
13. Cover your bug fix with a regression test
Recall the rules

1. Understand the requirements
2. Make it fail
3. Simplify the test case
4. Read the right error message
5. Check the plug
6. Separate fact from fiction
7. Divide and conquer
8. Match the tool to the bug
9. One change at a time
10. Keep an audit trail
11. Get a fresh view
12. If you didn't fix it, it ain't fixed
13. Cover your bug fix with a regression test
Recall the rules

1. Understand the requirements
2. Make it fail
3. Simplify the test case
4. Read the right error message
5. Check the plug
6. Separate fact from fiction
7. Divide and conquer
8. Match the tool to the bug
9. One change at a time
10. Keep an audit trail
11. Get a fresh view
12. If you didn’t fix it, it ain’t fixed
13. Cover your bug fix with a regression test
Unit testing frameworks

- uniform way of expressing tests
- manage tests through suites
- automate testing process
CUnit

http://cunit.sourceforge.net

http://cunit.sourceforge.net/doc
CUnit is organized like a conventional unit testing framework:

```
Test Registry

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
| Suite '1'                      | Suite 'N'
<p>| |
|                                |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
| Test '11'                      | Test '1M'
|                                |
|                                |
|                                |
|                                |
|                                |
| Test 'N1'                      | Test 'NM'
```

A typical sequence of steps for using the CUnit framework is:

1. Write functions for tests (and suite init/cleanup if necessary).
2. Initialize the test registry - `CU_initialize_registry()`
3. Add suites to the test registry - `CU_add_suite()`
4. Add tests to the suites - `CU_add_test()`
5. Run tests using an appropriate interface, e.g. `CU_console_run_tests`
6. Cleanup the test registry - `CU_cleanup_registry`

a test

- a void -> void method
- test methods must be part of a test suite, which must in turn be registered with registry, before it will be run
Assertions
(the most common ones)

CU_ASSERT_TRUE(x)
CU_ASSERT_EQUAL(x,y)
CU_ASSERT_PTR_EQUAL(x,y)
CU_ASSERT_PTR_NULL(x,y)
CU_ASSERT_STRING_EQUAL(x,y)
CU_ASSERT_DOUBLE_EQUAL(x,y,\varepsilon)
CU_PASS(message)

CU_ASSERT_FALSE(x)
CU_ASSERT_NOT_EQUAL(x,y)
CU_ASSERT_PTR_NOT_EQUAL(x,y)
CU_ASSERT_PTR_NOT_NULL(x,y)
CU_ASSERT_STRING_NOT_EQUAL(x,y)
CU_ASSERT_DOUBLE_NOT_EQUAL(x,y,\varepsilon)
CU_FAIL(message)

http://cunit.sourceforge.net/doc=headers/CUnit.h