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
Robert Burns

The best laid schemes o' mice an' men
Gang aft a-gley, [often go awry]
An' lea' e us nought but grief an' pain,
For promised joy.

blackbox vs whitebox testing
blackbox testing

- Code is treated as a "black box", one which you cannot peek inside.
- Tests are written without regard to **HOW** code is written.
- Tests are meant to capture the intended behavior of the system (the requirements/specifications): **WHAT** the code should do.
- In Test Driven Development (TDD) tests are written before the code is, and so qualifies as black-box testing (though maybe it would be more accurate to call it no-box testing).
- In TDD, think of tests written to capture specifications as **executable specifications**.
whitebox testing

1. Tests are written taking into consideration **HOW** the code is written.
2. Use a code coverage tool to ensure that tests exercise all possible computation paths.
Code coverage

- We will use gcov as our coverage tool.
- Compile with extra flags, as in:

```bash
gcc -g -O0 -Wall -fprofile-arcs -ftest-coverage -L /util/CUnit/lib -I /util/CUnit/include/CUnit/ evaluate.o expression.o expressionOption.o primitiveOption.o stack.o tests.c -o tests -lcunit -lgcov
```
using gcov to verify test coverage

- compile test code with extra flags
  - this instruments code to gather coverage information

- run tests
  - this runs your tests and allows the instrumentation to collect coverage data that shows what parts of the implementation were exercised by the tests

- run gcov on the source file (e.g. source.c) whose coverage you're interested in exploring

- Look at the file produced by gcov (e.g. course.c.gov)
Classroom exercise

- Write tests to achieve 100% code coverage for the LEX03 evaluator.