Topics in Testing

Outline of topics:
- Review of material from CS1
- Unit Testing
- White – & Black – Box testing
- Equivalence Classes for input
- Test coverage
- The architecture of a testing system

Types of Testing
- Formal verification
  - Use mathematical and logical assertions to prove the correctness of a program
- Empirical testing
  - Generate test cases to show that errors exist

Review
from CS1 notes on SLC
- Testing of individual program units is performed during the implementation phase.
- This verifies that each unit performs according to the design and functional specification.
- There will also be some limited testing of the overall program during the implementation phase.
- More formal system acceptance testing happens later.

White-Box Testing
- AKA Clear-Box Testing
- Requires access to the code.
- Test cases are based on an understanding of the code.
- Goal is to test "all" program flows; cover all...
  - Statements
  - Decisions
  - Conditions
  - Inputs
Black-Box Testing

- No access to the source code.
- Test cases are based on the requirements.
- Must test valid and invalid data.
- Impossible to test all possible input.
- Must decide on a subset of input that sufficiently covers the input.
  - Be imaginative and devious!
  - Try to break the program.

Some Definitions

- **Error**
  - An improper action of a software developer
- **Fault**
  - Its result: improper logic in the software that could potentially cause ...
- **Failure**
  - An improper action of a software program in execution.

Unit Testing

- It is better to test a single software unit before it is integrated into a system.
- Bugs found earlier are cheaper to fix.
- Easier to generate stimuli that are guaranteed to produce the desired response.
  - Testing individual classes in an integrated system is like what happens in billiards when you try to target the ball that has the most obstacles (other balls) between it and the cue ball.
- Our unit is the class.

Black Box Unit Testing

- Call methods ...
  - With carefully chosen parameter values.
  - In various orders.
- Follow them by calls to query/accessor methods to determine the state of the object.
- Black-box testing tests contracts. Contracts should be well documented!
White Box *Unit* Testing

- The same types of things are done in white-box as in black-box unit testing, except the goals are now
  - complete exercising of all conditional statements
  - complete coverage of statements
  - ...

Test Coverage

- Statement coverage
  - Execute each statement at least once (good).
- Decision coverage
  - Ensures each if/loop/case decision goes every way it possibly can (better).
- Condition coverage
  - Ensure each combination of Boolean outcomes from a single decision is tested (best)

Equivalence Classes

- Testing with all possible stimuli sequences is impossible.
- Choose "groups" of stimuli that you guess would cause the same reaction (proper or faulty), *using your knowledge of how software is built*.
- These groups are *equivalence classes*.
- Choose only one test from each class.
- There are still a phenomenal number of tests that should be done!

Test Scaffolding

- *Skeleton*, or *driver*: emulates the class's client.
- These provide the test cases.
- (most likely not hard coded)
- Skeletons are augmented with queries whose results are reported.
- *Stub*: represents the class's supplier (including hardware devices)
- The calls to the stubs are also reported.
Exercise
(time permitting)

1. Look at the “contract” of the List class in the handout.
2. Under what situations might you expect a certain member function to behave differently than in others? These are the equivalence classes.
3. Devise a unit test (a skeleton program) that would cover those equivalent classes.