Project: Due 8/10/14 (August, 10th)

In the course of assignments 1-4 you will perform testing on a Dominion implementation. Your project comes in four parts:

1. Use the unit and random tests (both card level and "whole game" level) to test code written or modified by your classmates. Find and document at least three bugs, and produce written bug reports (possibly including pointers to code to expose those bugs). Submit these reports to your classmates by adding them to the directory PROJECT/<onid-id>/bugreports. Each bugreport submitted should be in a file whose name starts with your onid ID, followed by a number: e.g. ahmed1.txt.
2. Document the process of identifying and fixing a bug in your own code. You may start with a bug report from a classmate or from your own testing. Show how you used a debugger to understand the problem, and describe how (if) any of Agans' principles applied to the process. Submit this file in your dominion directory as debugging.txt. If any files or logs are discussed in debugging.txt make sure to also submit those!
3. Pick one of the following:
    a. Use delta debugging tools (downloaded from Zeller's website) to minimize a failing test case for Dominion. Describe the process and what you did in deltadebug.txt, and include any relevant code (modifications of the python scripts) or files. One way to go about this is to take your work from Assignment 4 and modify it so that it produces a standalone C file containing calls to play the randomly generated Dominion game.
    b. Implement Tarantula using gcov, and describe the process of using it to localize a bug in Dominion. Write this up as tarantula.txt, and include any relevant code or files.
    c. Organize and moderate an online software inspection of one of your classmate's implementations (with their permission!). Document the process in inspection.txt. Ideally, if you lead an inspection for X, X leads an inspection for you. Make sure you have at least 2 other participants. Describe the level of formality, and any bugs found. This probably works best if groups of four work together, with each person leading 1 inspection and each implementation receiving 1 inspection.
    d. Use CBMC or the SPIN model checker to verify the behavior of a function in your dominion implementation. If you want to do this option, let me know and I can provide some advice and code to get you started.
4. Write a test report, in pdf format, describing your experience testing Dominion. Document in detail, including code coverage information, the status and your view of the reliability of the Dominion code of at least two of your classmates. This file is submitted as testreport.pdf in your dominion directory. Your test report should have a minimum of 1,000 words of text.