CS362, Software Engineering II

Assignment 4

Assignment 4 - Learn how to create a Random Tester

1. **Github**
    a. Use the Assignment 2 branch to create and check out a new youronid-assignment-4 branch.
2. **Random Test Generator** (**45 points**)
    a. Use your refactored functions from Assignment 2 to write automated random test generators for three Dominion cards; 1) baron, 2) minion, and 3) tribute. Check these testers in as randomtestcard1.c, randomtestcard2.c, and randomtestcard3.c.
    b. For the baron card, make your tester achieve 90% statement and branch coverage. For minion and tribute, make your tester achieve at least 70%. You will document this and note how long the test must run to achieve this level of coverage. Use the -b and -f options when you generate your coverage. Try to write your test to reach the coverage goal in less than five minutes on a reasonable machine**.**
3. **MakeFile**
    a. Add a rule in Makefile named **randomtestresults** that will generate and execute all the tests and append complete testing results, including % coverage, into a file called **randomtestresults.out**. (**10 points**)
4. **Documentation**
    a. Describe your random tests in detail in a section called **Random Testing**. (**15 points**)
    b. Describe in detail how much code you managed to cover in a section called **Code Coverage**. Was there code you failed to cover? Why? For the baron card, describe how long the test must run to achieve the specified level of coverage. (**15 points**)
    c. Compare your coverage to that of your unit tests that you created in Assignment-3 and discuss how the tests differ in ability to detect faults in a section called **Unit Vs. Random Testing**. Which tests had higher coverage – unit or random? Which tests had better fault detection capability? Be detailed and thorough. (**15 points**).

5. **Notes**

- If the input is a primitive data type, generate a random primitive value.
- If the input is an array, create an array and initialize it with some random values.
- If the input is an object, create the object and initialize member variables to random values. You may be investigating a subset of member variables so you would generate random values for the member variable(s) under investigation.
- Try to "stay random", but you may need to shift the probability space. For example, you might choose to generate a random number within a range but there would be a logical reason in your code for limiting the range.
- Improve your oracles (assertions) until you feel that all the problems that should be caught are caught.

Deliverables:

1. Push your youonid-assignment-4 branch to github. This branch must be created before the due date to receive credit. This branch will contain your Makefile and random tests.
2. Upload your Assignment-4.pdf document to Canvas. This document will contain three sections; **Random Testing, Code Coverage, and Unit vs. Random Testing**.
3. When you submit your PDF to Canvas, add a Comment in the Comments Box and provide the URL for your GitHub repository**.** (-10 points for missing it).