## CS362-004

## Assignment-4:

The primary goal of this assignment is to learn how to write a random tester.

**Note:**

- This is NOT part of the team project. Please do it on your own.
- Submit your entire dominion folder with all new files to your repository under projects/youronid/dominion.
- Create a new branch of your repository called "youronid-assignment-4"
- Also, submit a zip file of your repository code.

**Assignment Details:**

1. Use the refactored code you created for assignment-2 to create a new branch for Assignment-4 called "**youronid-assignment-4**".

2. Write an automated random test generator for three Dominion cards, one of them being the adventurer card, and at least one being a card you wrote unit tests for in Assignment-3. Check these testers in as randomtestcard1.c, randomtestcard2.c, and randomtestadventurer.c. (45 points)

3. Submit a pdf file to Canvas called Assignment-4.pdf, that contains the following sections:

   • **Random Testing**: write up the development of your random testers, including improvements in coverage that you gained from random testing. (15 points)

   • **Code Coverage**: discuss how much of adventurer and the other cards' code you managed to cover. Was there code you failed to cover? Why? For at least one card, make your tester achieve 100% statement and branch coverage, and document this and list how long the test must run to achieve this level of coverage. It shouldn't take more than five minutes to achieve the coverage goal (on a reasonable machine, e.g. flip). (15 points)

   • **Unit vs Random**: 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. Which tests had higher coverage – unit or random? Which tests had better fault detection capability? Be detailed and thorough. (15 points)

4. Add rules to the Makefile to produce randomtestcard1.out, randomtestcard2.out, and randomtestadventurer.out, including coverage results (10 points)

Helpful Tips:

- An overview of the random generator structure

1. Identify the method under test
2. Identify all the dependencies (parameters)
3. Write code to generate random inputs for the chosen method
4. Invoke the method (execute the method)
5. Check if stopping criterion (like time or number of loops) is not satisfied, if not go back to step 2.

- Note on inputs:
    - If the input is a primitive data type, generate a random primitive value, etc.
    - If the input is an array, create an array and initialize it with some random values, etc.
    - Try to “stay random” but shift the probability space (e.g., if you choose int numCoppersInDeck = rand() % 20 and int numAdventurersToPlay = rand() % 10. There must be a logical reason in the code specification for choosing 20 and 10.

- You also need to improve your oracles (step 5) (i.e., assertions "if/print in our case") until you feel that all the problems that should be caught are caught! Make sure your Oracle helps you develop a ‘rock solid’ random generator and not just a random generator.