Jon Raleigh
CS362
7/22/18

BUGS

One of the bugs encountered involved the use of the discard and discardCount arrays within the gameState struct and the discardCard function in dominion.c. discardCard either adds cards to the played cards pile and then removes them or it simply removes them from the game altogether. The playedCards array holds a record of all discarded cards and the discard array is not used.

Additionally, cards did not universally use the discardCard function to remove them after being played or they did not contain any code within their switch statement or function to remove them from play. This would keep a card within the hand of the player to be purchased repeatedly rather than added to the discard pile. Remodel and Smithy both exhibited this error.

The Smithy card also caused the error that I created in Assignment 2. The card added 4 cards to the player's hand rather than the expected 3.

Otherwise, all tested functions and cards behaved as expected within the limits of the test.

Unit Testing

gcov coverage amounts:

unittest1.c: Lines executed:16.84% of 564

unittest2.c: Lines executed:17.20% of 564

unittest3.c: Lines executed:18.09% of 564

unittest4.c: Lines executed:17.55% of 564

cardtest1.c: Lines executed:20.04% of 564

cardtest2.c: Lines executed:19.15% of 564

cardtest3.c: Lines executed:19.86% of 564

cardtest4.c: Lines executed:18.09% of 564

There was a considerable amount of code not covered by the testing suite due to the fact that it was outside of the scope of the tests. Viewing the unittestresults.out file showed a number of areas that were not addressed by the tests, which was expected.

Additionally, several of the variables used in dominion.c were declared and defined within the testing programs for the purpose of testing, so they did not see coverage within gcov.

Several parts of the code are inaccessible due to being only accessible if a the DEBUG value is set to 1.

To increase coverage, the tests can be altered to cover those functions. Increasing the number of players from 2 to the maximum will also provide more coverage. Functions can also be called via other functions, such as playcard(), to test for behavior within these functions when we move outside of unit testing (as this is integration testing).

Unit Testing Efforts

I focused on path testing for these unit tests. I looked at each of the if statements, fed it different values based on what it could expect to receive in a normal playthrough, and made sure that the output was as expected.

I originally only used “TEST SUCCESSFULLY COMPLETED” and “TEST FAILED” as the only messages from my test, but this did not provide me with adequate information for the failed tests.

I added error messages for the functions to 1) determine whose fault the bug was (mine in the test creation or the program’s), and 2) determine the nature of the bug. Upon failed tests, I went through and added error messages for every possible failure state.

There were a few issues with determining card counts on my part, but those were quickly fixed.

In future test suites, I need to include more non-error messages to determine if I am receiving false positives caused by errors in my coding. Regular counts for each player for each turn, the composition of the various piles, and randomized play will help to make the debugging more thorough and insightful.