Brian Laccone
CS 362
5/13/2018

Assignment-4

Random Testing

The first thing that I made sure was included in randomtestadventurer.c was randomize as many primitives as I could before initializing the game and after. I randomized the number of players in the game from 2 to the max amount of players. I made sure that there was at least 2 players in the game because two players are required to play the game and testing with only one player seemed counterintuitive. I tried to randomize the seed number but I keep running into problems where the program would randomly stop and ask for a different seed value because it was too high. The code that stops the game is located in rngs.c.I set the number of iterations to 50000 because I felt that this was enough iterations to find any problems in the program. I tried more but it took a long time and didn't result in any more coverage. After initializing the game, I randomized the deck and hand counts of the player using a minimum of 3 for the deck count because I felt that I needed the deck to have at least 3 treasure cards to actually test the adventurer card. I made sure to shuffle the cards after changing the deck size and contents of the deck to make sure that the deck is randomized. I created 5 tests for the adventurer card that all compare the state of the game after the function is called to the state of the game before the function is called. The five tests are: total card count, hand count, treasure count, victory supply pile count, and kingdom card supply pile count. I knew this card would fail in the hand count test because I commented out the part where the extra cards, that aren't treasure cards, were discarded.

For randomtestcard1.c, I chose to test the Great Hall Card. I had the same exact variable set up as the randomtestadventurer.c except that I set the minimum card to 1 instead of three and didn't require at least three cards to be treasure cards. For the Great Hall card I created 6 tests: total card count, hand count, deck count, action count, victory supply pile count, and kingdom card supply pile count. This card passed every test because I didn't change the function for this card in assignment 2.

For randomtestcard2.c, I chose the Smithy card. I had the same exact variable set up as the randomtestadventurer.c except that I set the minimum card to 1 instead of three and didn't require at least three cards to be treasure cards. For the Smithy card, I created 5 tests: total card count, hand count, deck count, victory supply pile count, and kingdom card supply pile count. Smithy did not pass the test because the deck and hand count are always off. This bug is because I made the for loop, which draws three cards, not loop at all by making the variables to 0.

Code Coverage

randomtestadventurer.c

File 'randomtestadventurer.c'
Lines executed:89.02% of 82
Branches executed:100.00% of 46
Taken at least once:82.61% of 46
Calls executed:83.33% of 24
Creating 'randomtestadventurer.c.gcov'

File 'dominion.c'
Lines executed:21.08% of 555
Branches executed:24.34% of 415
Taken at least once:17.11% of 415
Calls executed:10.64% of 94
Creating 'dominion.c.gcov'

For the adventurer card, I completed 90% block coverage and completely missed one if statement in the branch coverage. The coverage for the whole dominion.c file was 21.08% statement coverage and 24.34% branch coverage. The reason that I didn't achieve 100% statement and branch coverage for the adventurer card because I failed to hit the if statement which checks to see if the player's deck count is below one and then proceeds to shuffle the cards. I didn't hit this branch because I didn't allow the randomized deck count for the players

go below 1 card because I wanted the deck to contain a certain amount of treasure cards to be able to test the card properly. This may have been an oversight and if I were to recreate this test I would allow the randomized deck count to hit zero instead of giving it a minimum amount.

randomtestcard1.c

File 'randomtestcard1.c'
Lines executed:76.47% of 68
Branches executed:100.00% of 26
Taken at least once:61.54% of 26

Calls executed:64.00% of 25
Creating 'randomtestcard1.c.gcov'

File 'dominion.c'
Lines executed:22.52% of 555
Branches executed:23.37% of 415
Taken at least once:15.90% of 415
Calls executed:11.70% of 94
Creating 'dominion.c.gcov'

For randomtestcard1.c, I tested the Great Hall card. I received 100% statement and branch coverage for the Great Hall card function. The Great Hall card is a very basic card function that only has one branch and I didn't introduce a bug to this function in assignment 2, so I am not surprised that I achieved 100% statement and branch coverage. It only took 1 iteration to complete 100% coverage. It would have been very hard to not hit 100% coverage for this function. All the tests that I created passed for this function.

randomtestcard2.c

File 'randomtestcard2.c'
Lines executed:87.50% of 64
Branches executed:100.00% of 24
Taken at least once:62.50% of 24
Calls executed:83.33% of 24
Creating 'randomtestcard2.c.gcov'

File 'dominion.c'
Lines executed:22.34% of 555
Branches executed:23.86% of 415
Taken at least once:16.14% of 415
Calls executed:10.64% of 94
Creating 'dominion.c.gcov'

For randomtestcard2.c, I tested the Smithy card. I only achieved 67% statement coverage and completely missed a branch in this card's function. The coverage for the whole dominion.c file was 22.34% statement coverage and 23.86% branch coverage. The reason that my function statement and branch coverage was not 100%, is because I introduced a bug during assignment 2 which caused the player to not draw any cards instead of drawing three cards. This bug made it impossible to hit the if statement which caused the player to draw three cards. If it was not for this bug, I would have achieved 100% statement and branch coverage.

Unit vs Random

Luckily, I choose three cards that I completed unit tests for in assignment 3, but unfortunately the coverage for both the unit tests and random tests were exactly the same. Although the adventurer card in assignment 3 had less statement and branch coverage compared to my random test coverage is because I unknowingly introduced a bug in the last assignment which I didn't realize until this assignment. I initialized int drawntreasure during the unit test but for some reason it didn't set the value to 0 so a whole branch was never covered. If that bug was fixed in the last assignment then the coverage would be the same between both assignments.

The main difference I see between random testing and unit testing is that random testing can find faults that you might not even be able to think off. Unit tests seem to be more linear in the way that the tester has to think of ways that might cause an error and manually set variables and parameters to a certain value to try and trip up the program. With random testing, The randomized numbers may create a combination of numbers that throws off the program in a

way that you might not have been able to predict. If I had set the adventurer card test's random deck count to be able to reach 0 then I would have reached 100% statement and branch coverage eventually. But the same could be said for the unit test if I had thought about making the deck count to 0 before calling that function. I think the functions that we are testing are too minor to fully grasp the benefits and drawbacks of either unit or random tests. I can see how in functions that have many parts that random testing might have more benefits in finding faults because it would be very hard to think of how bugs might occur with unit testing.