Eddie Christopher Fox

February 7, 2017

CS 362:

Assignment 3 Testing Document

Choice of the Inputs:

Unit Test 1: Testing the getCost() function did not require much input choice. I simply created hard coded arrays of the proper names and costs of every card, making sure the indexes would match up. Then, using a for loop, I called the function manually while passing it every single card, and then asserted it equal to the correct cost of the matching index.

Unit Test 2: Tested the numHandCards() function. Since it returns the state.handCount[currentPlayer], I thought it would be a good idea to create two four loops, one for each player, and run various comparisons, remembering to end the turn in between players to test if the function was able to change player state properly. If it couldn't this would lead to incorrect results. I used similar variables as I did in the card tests, initializing the bare minimum I would need to initialize the game (game state struct, random seed, kingdom array, current player, number of players.) Inside the for loop, I actively changed the value of the hand of the current player, and did assert true's for all three variables with each other in every way: handCount, the loop control variable, and numHandCards. At every stage after the initial assignment, and no matter how it is written, they should be equal.

Unit Test *3:* Did not finish in time.

Unit Test 4: Did not finish in time.

Card Test 1: Smithy card. For this testing, I used the example method of creating two game states, an original and a test, and copying the original. After modification from the tests, we compare the test state to the original with the modifications that should have occurred via alternative variables. Variables initialized were necessary for several purposes.

1. Initializing the game state (such as numPlayers, seed, kingdom array)
2. In order to call cardEffect() (choice1, choice2, choice3, handpos, bonus)
3. To test something relevant within the function by providing baseline modifications to the original state to compare with the tested state later (newCards, extraCoins, discarded, shuffled).

Card Test 2: Adventurer card. For this one, I eased off on variables defined before the game was initialized. I was only doing a single game state for this card, so initializing too much before the game didn't serve a purpose of alternative comparison. I just initialized current player, the kingdom array, the game state, and the seed before calling initialize game. Afterwards, I manually put adventurer in the current players hand and created a deck manually so I could test for specific outcomes. The tests aren't exhaustive, but they can be instructive, and because I set it

up manually, someone can have more control. I set deck count, discard count, played count, and hand count accordingly. After asserting that playAdventurer() worked, I compared several values such as hand count, discard count, and action count to their expected values.

Card Test 3: While mostly like the testing of the adventurer card, I needed to add choice1, choice2, choice3, handpos, and bonus back into the variables declared before the game because I didn't have a refactored playVillage function ready. Everything else is similar to adventurer testing, except I tweaked some of the values, like having 6 cards in the deck instead of 5.

Card Test 4: Testing council room was similar to the others with a bit of difference. I needed handPos but not the other predeclared variables (like choice1, choice2, and bonus) because I already had a refactored playCouncil_Room() function. I decided to start the current player with 1 buy point to see if the card would give them another, and started the other player off with a card so I could test if they would have 2 cards after Council Room was played.

Bugs:

Unit Test 1: No bugs found.

Unit Test 2: No bugs found.

Unit Test 3: Didn't finish in time.

Unit Test 4: Didn't finish in time.

Card Test 1: Hand and deck were off by 1. The player drew one more card than intended, as the hand had one more card than expected and the deck had one less card than expected. This bug was actually expected as it was the bug I put into Smithy last assignment.

Card Test 2: While the function executes, it does not create correct results. The player ended up with 3 cards in their hand, so it is likely that Adventurer was not discarded after drawing the 2 treasures. Bug is to be expected as I messed up the deck in the last assignment, but I wasn't exactly sure what the effect would be.

Card Test 3: Village should not have had a bug, as I didn't refactor it last assignment. The function failed to execute, so it's possible that I messed up the hand position for the card in the testing, and therefore the card wasn't even played. It does have the line "Village card not registered as played." Interestingly, one test did pass: the hand count being 1. What this likely means is the Villager card I gave the player is still in their hand, and it was never played, seeing as the function failed to execute.

Card Test 4: I expected to find bugs in Council Room as I bugged it in Assignment 2. I made it so that the player who played the card receives all the cards that are supposed to be drawn to other players. The player having 5 cards in their hand instead of 4 supports my theory that the

intentional bug was executed. Unfortunately, it seems that discarding still doesn't work properly, which is something shared by every card test. That might be a problem with my test suite. Encouragingly, the number of buys the player had properly incremented from 1 to 2, and the card registered as properly played, adding further confidence to the systemic discard problem I've been having in this test suite.

Coverage:

To be fair with these coverage tests, each test only covers a small section of dominion.c, so it would be unexpected for any 1 test to cover a significant amount. Note: After writing down all the coverage, I realized that it is very likely that the coverage after each test is cumulative.

Unit Test 1:

Lines executed:5.18% of 560

Branches executed:6.73% of 416

Taken at least once:6.49% of 416

Calls executed:0.00% of 93

Unit Test 2:

Lines executed:25.18% of 560

Branches executed:24.04% of 416

Taken at least once:20.67% of 416

Calls executed:11.83% of 93

Unit Test 3: Didn't finish in time.

Unit Test 4: Didn't finish in time.

Card Test 1:

Lines executed:29.64% of 560

Branches executed:31.25% of 416

Taken at least once:22.36% of 416

Calls executed:16.13% of 93

Card Test 2:

Lines executed:31.96% of 560

Branches executed:34.13% of 416

Taken at least once:24.52% of 416

Calls executed:18.28% of 93

Card Test 3:

Lines executed:32.14% of 560

Branches executed:34.13% of 416

Taken at least once:24.76% of 416

Calls executed:18.28% of 93

Card Test 4:

Lines executed:33.75% of 560

Branches executed:35.58% of 416

Taken at least once:26.20% of 416

Calls executed:21.51% of 93

Dominion.c also has extensive coverage notes below all the tests spanning for thousands of lines.