CS 362: Assignment 3

James Palmer

Inputs-

** Unit test 1: Coins **

Copper Test:

We should have 10 copper coins: 0

Test Failed

Silver Test:

We should have 10 silver coins: 0

Test Failed

Gold Test:

We should have 10 gold coins: 0

Test Failed

Test for zero existing Coins:

We should have zero existing Coins: 0

Test Passed

** End of Unit Test 1: Coins **

** Unit test 2: Buying Cards **

Do we lack Coins in our hand?

Test Passed

Attempting to buy desired card!

Testing Failed

Can we check amount?

Test Failed

Check to see if user still has existing Buys:

Test Passed

** End of Unit Test 2: Buying Cards **

** Unit Test 3: Turn Over? **

Test Passed

** End of Unit Test 3: Turn over? **

** Unit Test 4: Who's turn is it? **

Running test 3 times because base game is set to 4 players

Checking to see if other three players are still legal targets

Test Passed

Test Passed

Test Passed

** End of Unit Test 4: Who's turn is it? **

** Card Test 1: Adventure **

Expected value: 0 Result: 0

Expected card count: 0, Result: 0

Final expected treasure count: 0, Result: 0

** End Card Test 1: Adventure **

** Card Test 2: Smithy **

Expected value: 0 Result: 0

Expected card count: 0, Result: 0

Final expected treasure count: 0, Result: 0

** End Card Test 2: Smithy **

** Card Test 3: Village **

Expected value: 0 Result: 0

Expected card count: 0, Result: 0

Final expected action count: 0, Result: 0

** End Card Test 3: Village **

** Card Test 4: Great Hall **

Expected value: 0 Result: 0

Expected card count: 10, Result: 0

Final expected action count: 1, Result: 1

** End Card Test 4: Great Hall **

choice of the inputs.

I choose the following inputs to keep my testing as clear and basic as possible. I found when trying to attempt multiple layers of testing, my code would become convoluted and difficult. At times the use of simple take the value of the array variable, would give a cleaner testing base. I found out very late that it is needed in adventure unit tests and card tests to declare all units of the array. I would like to note my comments above. The variables, as seen above, use each testing criteria at a basic level to prevent overlap (see comments in bugs). They are taken directly from my testing. I kept both unittest and cardtests similar in style and output for clarity rather than how good it looks.

Bugs-

There were a few bugs I found with my testing. One of the bugs was the fact that fullDeckCount does not accurately count the deck. It only looks for a certain card in the deck. In my initial testing, the adventure card was difficult to weigh its coin bonus (after buy values). It is unlikely it has a lasting effect on other cards. However, it shows that when the value based of fullDeckCount is equal to zero, Smithy has a ripple effect in all other unitcard testings. One piazza file said to not change the base code only work around the bug. I believe this bug comes from the gainCard function that is called from other places and only when another card (i.e. village) was put in the wrong pile (illegal move) or was not in play when called upon.

As seen above, adventure had the same bug as smithy. I believe this could be because adventure is the man base for where we pull our variables from (or treasure values/counts etc.) There are cases when the adventurer did not have treasure at all or one treasure. To get around this, one had to implement both cases by either removing all cards and forcing a hand draw back to 5 or removing all but one instances. Attempted to add a single treasure card (randomly chosen). However, instead of a new bug, only the previously detected bug remained. The card mechanics do not detect the care where the player deck is empty and needs to be reshuffled. This can be seen above section, yet again, the program defaults to zero (can claim this as a fail?).

Coverage-

As seen at the top of unitestresults.out I have the gcov measured the code time:

File 'dominion.c'

Lines executed:29.73% of 565

dominion.c:creating 'dominion.c.gcov 'dominion.c:creating 'dominion.c.gcov'

This above is one pulled Gcov test during my testing. This is subject to change based off my bug changes, which compiler I used and how much of the program was in use, from card values or calls. Above only 29:73% of lines executed indicates a few possible outcomes. First, there could be some lines that are not supported by the Gcov line examination output. Second, there could beis an overlap between two card functions when tested randomly by my code. I can also conclude that some of my functions calls are being used in multiple areas (hence the overlap) or not all lines are code are being executed. This would be because I ended the game very early, some .c files or function calls are not being reached or my code is just terrible. Could be any option of the three.