Name: Haris Sohail
Date: 04/29/2018
Assignment 3
CS362

Bugs

getCost() - (Unit test 1):
No bugs were found for this unit test. All cards correctly returned their proper card values.

isGameOver() - (Unit test 2):
No bugs were found for this unit test. The function properly ends when Province cards reach 0 and does not end when Province cards are above 0. Further the game properly ends when 3 supply piles reach 0 and continues otherwise.

whoseTurn() - (Unit test 3):
No bugs were found for this unit test. The function properly alternates turns between each player and is capable of returning the correct turn each time.

numHandCards() - (Unit test 4):
No bugs were found for this unit test. The function correctly follows incrementations and decrementations for player hand amounts over time.

Smithy - (Card test 1):
A bug concerning card drawing was discovered. This bug was implemented during assignment 2. The Smithy card draws 4 total new cards rather than 3. This was found during the test comparing the expected hand count to the actual. This was further established for the test comparing the remaining deck counts. This bug was traced back to the for loop within callSmithy() where it runs 4 times.

Village - (Card test 2):
A bug concerning total actions was found. This bug was implemented during assignment 2. The Village card should allow the player to gain 2 actions but here the player only gains 1. This was

found in comparing the total actions between the actual game and test game. The player remains with the same number of actions since Village costs 1 to play and they only gain 1. The test game has 1 more action since it is properly functioning. This bug can be traced back to improper incrementation in callVillage().

Sea hag - (Card test 3):
A bug was found here in which player discards cards. This bug was implemented in assignment 2. It appears that only the player who played the card must discard their top deck card. Wheras it should be the reverse where every other player discards their top deck card. This was found when looking at the deck count for each player. The player who played sea hag discards the total amount for each player present. While each other player remain with the same deck count. This is also furthered by seeing the discard pile for each player. Only the player who played Sea Hag has cards in their discard pile.

Adventurer - (Card test 4):
A bug concerning hand count was found. There is 1 more card present in the hand than expected. This card is actually the Adventurer card. The function never calls discard, therefore the card is never removed from the hand. Otherwise the function works as expected and puts two treasure cards into the hand. However, one other bug is missed. Due to a bug implemented in assignment 2, any card drawn is considered a treasure card and is automatically placed into the hand.

Unit Testing

getCost() - (Unit test 1):
Function 'getCost'
Lines executed:100.00% of 30
Branches executed:100.00% of 28
Taken at least once:100.00% of 28
No calls

Looking at the gcov information we can conclude the unit test fully covers this function. All possible inputs and return values are taken into account and tested for and all lines are covered.

isGameOver() - (Unit test 2):
Function 'isGameOver'
Lines executed:100.00% of 10
Branches executed:100.00% of 8
Taken at least once:100.00% of 8
No calls

Looking at the gcov information we can conclude the unit test fully covers this function. All possible inputs and return values are taken into account and tested for and all lines are covered.

whoseTurn() - (Unit test 3):
Function 'whoseTurn'
Lines executed:100.00% of 2
No branches
No calls

Looking at the gcov information we can conclude the unit test fully covers this function. All possible inputs and return values are taken into account and tested for and all lines are covered.

numHandCards() - (Unit test 4):
Function 'numHandCards'
Lines executed:100.00% of 2
No branches
Calls executed:100.00% of 1

Looking at the gcov information we can conclude the unit test fully covers this function. All possible inputs and return values are taken into account and tested for and all lines are covered.

Smithy - (Card test 1):
Function 'callSmithy'
Lines executed:100.00% of 5
Branches executed:100.00% of 2
Taken at least once:100.00% of 2
Calls executed:100.00% of 2

Looking at the gcov information we can conclude the unit test fully covers this function. All possible inputs and return values are taken into account and tested for and all lines are covered.

Village - (Card test 2):
Function 'callVillage'
Lines executed:100.00% of 5
No branches
Calls executed:100.00% of 2

Looking at the gcov information we can conclude the unit test fully covers this function. All possible inputs and return values are taken into account and tested for and all lines are covered.

Sea hag - (Card test 3):
Function 'callSeaHag'
Lines executed:100.00% of 7
Branches executed:100.00% of 4
Taken at least once:100.00% of 4
No calls

Looking at the gcov information we can conclude the unit test fully covers this function. All possible inputs and return values are taken into account and tested for and all lines are covered.

Adventurer - (Card test 4):
Function 'callAdventurer'
Lines executed:76.92% of 13
Branches executed:100.00% of 6
Taken at least once:66.67% of 6
Calls executed:50.00% of 2

The test suite here is lacking in several areas and misses a good portion of the code. As only 76% of the lines were even executed we cannot conclude this to be a valid test as those lines may have errors. The lines in question are those which are activated in case of an empty deck in which the discard pile would be shuffled into the deck. Beyond that the bug implemented from assignment 2 prevents a statement execution from taking place. This one is where a non-treasure card is drawn and must be removed from the hand. The bug automatically assumes all cards to be treasure cards thus it can never be reached. This is not a lack of capability in the test suite but in the bug itself. No matter what test is written, it cannot execute.

Unit Testing Efforts

My main goal during testing was to at the minimum cover all possible inputs for each function and validate all possible outputs. Starting with the getCost() function, it has a range of 0 - 27 for possible input values, as well as 28 possible outputs. My test uses a for loop to feed it each value and validate each return. This resulted in a gcov of full test coverage and demonstrates a reasonable unit test. I further this effort within the isGameOver() function. I set provinces to all possible values and see when an end game is triggered. After that I check the second portion of code to see if setting random supply piles to 0 will trigger and end game. This resulted in full branch coverage as well.

whoseTurn() had a simpler function to run through. I used an alternating for loop to set and check if the correct player had their turn. By keeping a dummy turn function and adjusting it

along the dominion.c function I was able to prove it was functioning along with full coverage. I used a similar tactic of keeping a dummy hand along with a real hand for numHandCards(). Every update to the in game struct hand I did to an outside hand. Then I asserted on each loop their values were equal. All test cases passed with full coverage.

Card testing required a much wider array of tests. As cards can change multiple aspects of the game and are not limited to a few variables. This can range from coins, victory points, hand counts and deck counts. For the Smithy function the main goal was to check that 3 cards were gained. This however branched out into checking that the hand had gained 3 cards, the cards came from the player's deck and that no changes were made to the other player or the supply piles.

This type of testing was further cemented with adventurer. There is a bug that I as of right now, deem uncatchable with this current testing suite. As the adventurer will keep the first two cards it draws, regardless of them being treasure or not. This bug can only be caught if the hand count coins are counted before the call and compared to after. The test would also have to examine exactly which cards were drawn to get an accurate total as they may be copper, silver or gold. Otherwise another bug was found thanks to checking hand counts. It was discovered the adventurer card is never discarded. The hand has 1 extra card than it should after the call. The bug previously mentioned however, prevented me from getting full coverage as the latter lines are never executed.

Overall my goal in testing was to use as many inputs I could to achieve all possible outputs. Coupled with trying to access every line and statement written in the function, which would result in full function coverage.