William Davies

Assignment 3

Bugs: The implementation of Adventurer had a few bugs. I used the following description from http://wiki.dominionstrategy.com/index.php/Adventurer : “Reveal cards from your deck until you reveal 2 Treasure cards. Put those Treasure cards into your hand and discard the other revealed cards.” One such bug comes from the fact that the adventurer card itself is never discarded after being used. This means the adventurer card will never leave the player’s hand. Additionally, instead of keeping the Treasure cards in the player’s hand (as it should), the Treasure cards are discarded along with the other cards drawn as a result of adventurer. No other bugs were found

Unit Testing:

Unit test 1: kingdomCards(). I got 100% branch and line coverage for this function. Each line was executed 11 times.

Unit test 2: getCost(). I got 100% block coverage and 100% line coverage. Each instruction was executed an equal number of times, which is to be expected.

Unit test 3: isGameOver(). I got 100% block coverage, of which 100% of lines were executed. The number of times each line was executed varied, but did so with every line being executed a number of times that were proportional to what I would have expected.

Unit test 4: fullDeckCount. I got 100% block coverage, of which 100% of lines were executed. Again, the number of times each line was executed varied, but did so in proportion to what I would have expected.

Cardtest1: smithy. I got 100% line coverage in the relevant part of cardEffect. This contributed to significant coverage of the discardCard function, for which I got 100% blocks with 78% executed. The lines that weren’t executed were for special cases when there was a low number of cards in the player’s hand. The suite might be improved by adding cases for when the handCount was low enough for those lines to execute.

Cardtest2: village. Much like smithy, I got 100% line coverage in the relevant part of cardEffect. Again, because discardCard was used several times, use of this card contributed to coverage of that function without covering those special cases where handCount is low.

Cardtest3: great_hall. Here, coverage of the relevant part of cardEffect was again 100% of lines. Additionally, discardCard was used several times but the lines for special cases weren’t used.

Cardtest4: adventurer. Code coverage was near perfect here, with one issue. The special case when the deck is empty, wasn’t tested. The test suite could be improved by adding a test case that would test for this scenario. This would require a unit test for shuffle also, to make sure that function would work correctly within this function.

Other functions: drawcard got 100% blocks but only 38% executed because lines for several special cases were never needed. The suite could be improved by adding a unit test for this function. getWinners, playCard, buyCard, endTurn, gainCard, scoreFor and several smaller functions with one or two lines were not used by the functions I tested, so they got 0% coverage. whoseTurn was used many times within other functions, and got 100% line coverage. initializeGame() got 100% blocks with 92% executed. The lines not executed were the ones that were used when more than 2 players were present. A proper unit test for this function could address this and ensure the function is perfect. Similarly, updateCoins got 100% with 82% executed. A proper unit test for this function would be helpful to the test suite.

Unit Testing Efforts:

getCost(): I tested the function with every card, making sure it returned the value it was supposed to return.

isGameOver(): I tested the function by making sure it correctly returned false when the game was not over, and true for each of the circumstances under which it should be over. Specifically, I the unit test function starts by initializing a game and running isGameOver(). This should (and did) return 0 for false. Then I set the pile for province cards to 0 and ran the function. This should (and did) return 1 for true. Finally, I tested the function for each combination of three empty supply piles.

fullDeckCount(): To test this function, I started by inserting particular numbers of cards into one player's hand, discard, and deck. I tested the function when the card being searched for was only in the hand, when the card being searched for was only in the deck, and when the card being searched for was only in the discard. Then, for each combination of two of those places (i.e. deck and hand, deck and discard, and hand and discard), I tested for cards that were in those two places (and not the third). Then I tested the function for a card that was in all three places. Finally, I compared the results for each type of card in that player's hand, deck, or discard to the results yielded from using the function to count the same type of card in a different player's hand, deck, or discard. This other player had those cards, but in differing amounts. This ensured that the results hadn't been coincidentally correct before.

kingdomCards(): Here I tested that the array of kingdom cards being returned matched the inputs it was given. I did so by using the function with completely different inputs, and comparing the results to make sure that correct results were returned. Comparing different results helped ensure that one usage wasn't returning the expected array simply because the function had coincidentally been hard coded to return an array with those exact cards in those exact places.

Smithy(): Here I tested that three cards had indeed been taken from the player's deck (in the correct order) and placed in their hand. I did so by seeing what the top three cards of the player's deck were prior to the card's use, and then seeing if they were correctly positioned within the player's hand afterward. By correctly positioned I mean that the first two cards drawn were the last two in the hand, and that the third was where the smithy card had been (note that the smithy card was discarded, and the discard card function places the last card in the hand, in this case the last one drawn, in the place occupied by the discarded card). Then the test ensures that the smithy card was placed in the

playedCards array, as it should have been when it was used in discardCard. Then, I compared the values of the other player's deck, hand, and discard pile to what they were before smithy ran. This was to ensure that player wasn't affected. Then I ensured that the supply hadn't been changed, which it shouldn't be since smithy does not have a player interact with it. This ensured that kingdom and victory piles weren't affected.

Village and Great_Hall: These were tested in much the same was as smithy, with the only differences being the number of cards drawn and the change in the number of actions. The change in the number of actions was measured by ensuring that the value had incremented (as appropriate for the card) to the value it should be at after the card was played.

Adventurer: This was a bit different. I carefully created a deck with certain cards in certain places, then tested to make sure adventurer drew the cards it should have and discarded cards as appropriate. The tests were similar to Smithy insofar as I ensured that the hand, deck, discard, and playedCard arrays, along with the integers counting their size, had adjusted appropriately. Then I ensured the cards drawn and discarded were in their proper place. Finally, I ensured that the other players weren't affected. Further, I ensured the supply wasn't affected (so kingdom and victory cards weren't affected.)