Riley Brandau

CS 362 – Assignment 3

Due 4/30/17

Bugs

Cards

Adventurer (cardtest1)

Upon running my tests for Adventurer, it becomes apparent that there is a critical failure somewhere in the Adventurer code, since the tests never finish. This means that there is some sort of infinite loop causing the program to hang. This is a bug that I introduced into the code, so it was expected. However, since the adventurer tests never complete, I was unable to get any gcov data or additional potential bug finds. In the future, if I ever encountered a bug that caused my tests to hang, I would implement some sort of timer function to break out of the infinite loop bug.

Smithy (cardtest2)

When I ran the tests for Smithy, I had actually forgotten about the bug I introduced (draw 0 instead of 3), so I was puzzled as to why it failed some of the tests. It is clearly evident that the card is broken, as both the hand size and draw pile are not the correct sizes. The player's overall hand should increase by two (+3 from Smithy, then -1 for discarding Smithy), however, the test revealed that NO cards were drawn and that Smithy was discarded, resulting in a hand size that decreased by one. This was mirrored by the second test where the Draw Pile should have decreased by three, but remained untouched.

I was surprised to find a bug that I didn't introduce into the code. This was revealed by testing the victory card piles. At the start of the game, each type of victory card pile should have a count of 8 or 12, depending on the number of players. Estates were correct at 8, as were the Duchys. But for some reason, the Provinces are at 46! Very strange!

Village (cardtest3)

Again, I had forgotten what bug I had introduced to the village card code. But my tests revealed what was wrong with it. Both the hand size and deck size indicated that a card had infact been drawn correctly, but the player's actions and buys were incorrect! The bug I introduced increased the buys instead of the player's actions.

The same province bug, as mentioned in the Smithy description above, was also present here.

Council Room (cardtest4)

After being caught off guard with my previous card tests, I was ready for this card to fail. And it did not disappoint. It is clear that the correct number of cards is not being drawn; both the hand size and the deck size are incorrect. Instead of 4 cards being drawn, only two are drawn.

The same province bug persists here as well.

Functions

fullDeckCount()

I am not sure if this counts as a bug, but I was under the assumption that at the start of the game, each player would have a hand size of 5, and a deck size of 5. But is also reasonable that the cards are drawn at the start of that player's turn and not the start of the game. Either way, player 1 (0) has the correct hand and deck size, while player 2 (1) does not. Both do have the requisite number of total cards though.

I found a bug in my own test suite: at the start of the game, each player should have 3 estates and 7 coppers. However, my tests were showing that while both players had the correct number of estates, they did not have ANY coppers. After inspecting my code, I had made a simple variable error.

scoreFor()

The test for this function was fairly simple. I wanted to check to see if both players had the correct number of victory points at the start of the game: 3. However, Player 1 (0) only has one victory point, and player 2 (1) has zero! So this would warrant some looking into to see what is causing this.

whoseTurn()

As with scoreFor(), I was simply testing to see if it would return the correct player's turn. No bugs were found with this function.

numHandCards()

These tests returned the correct values for player 1 (0), but revealed some flaws (that have yet to be fixed) in my testing methodology. I did not reset the variables that tracked changes after drawing a card, so player 2 (1) displayed incorrect hand and deck size.

Unit Testing

Cards

Adventurer

In the future, when I encounter a bug that gives an infinite loop or takes too long in general, I will implement a timer function to exit the loop and move on to the other tests. This way I would still be able to get test suite information as well as gcov information. As it stands right now, all I know is that there is an infinite loop bug in Adventurer, which, is not a worthless find, but hampered the effectiveness of the rest of the adventurer tests. Although, I would assume it would give similar results for the non-card-specific tests.

Smithy

Smithy passed 3 of the 5 tests that I ran on it. The two tests that it failed were a hand size and deck size check. This implies that the card is not triggering the correct number of cards to be drawn.

It looks like the coverage for Smithy was adequate. It executed 80% of the lines and 100% of the branches. It also touched the updateCoins, discardCard, cardEffect, drawCard, whoseTurn, compare, shuffle and initializeGame functions. I believe that most of these functions that had coverage were due to calling the initializeGame function, which calls whoseTurn and updateCoins, and calling cardEffects to test Smithy. In the future, to improve coverage of the entire dominion code, it might be worth looking into calling the playCard function as it was not called at all, and could probably reveal bugs between playCard and cardEffect since it is the function that would call cardEffect.

Village

Village also passed 3 of the 5 tests that I ran on it. It failed on the tests that checked the player's buys and action totals after playing the card. This indicates that the player was able to draw the correct number of cards, but there was an issue with gaining the correct number of actions.

100% of the lines for Village were called. There are no branches in village, so none were called. It also touched the updateCoins, discardCard, cardEffect, drawCard, whoseTurn, shuffle, initializeGame, and compare; same as with Smithy. Again, calling playCard should probably be implemented to make sure there are no bugs between playCard and cardEffect.

Council Room

Surprise, surprise, Council Room also passed 3 of the 5 tests that were ran on it. It failed the tests that checked the player's hand and deck size. This indicates that the draw function of the card was not functioning properly, but the player still gained the correct amount of actions. For future tests, I would include a test to see if the other player's hand size increased, as council room has each other player draw a card as well.

100% of both the lines and branches were executed for this card. The tests also touched updateCoins, discardCard, cardEffect, drawCard, whoseTurn, shuffle, initializeGame, and compare; again the same as smithy and village before it.

Overall, the coverage of the dominion.c (lines and branches executed in dominion.c) file were not so great. Only about 20-25% of the lines and branches were executed with any given card. This shows that these tests are lacking in generating coverage for a majority of the functions. However, the coverage of the cards was very high, so I believe they were tested thoroughly.

Functions

fullDeckCount()

FullDeckCount passed 10 of the 12 tests that I threw at it. It failed the tests associated with the 2nd player's hand and deck size. Depending on how initializeGame functions, this is understandable, as player 2 might not draw until the start of their first turn. The tests were built assuming that both players start with a hand size of 5 and a deck size of 5.

88.89% of the lines and 83.33% of the branches were executed for fullDeckCount. This is a good amount of coverage for this function. The tests also touched updateCoins, drawCard, shuffle, initializeGame and compare.

If I were to develop additional tests for fullDeckCount, I would probably include calls to the supplyCount function for testing the different victory and kingdom card piles.

scoreFor()

ScoreFor failed both of the tests I ran on it. I am unsure of exactly what the bug is. The tests were simply seeing if the correct score value would be returned for each player at the start of the game: 3. However, player 1 only had a score of 1, while player 2 had a score of 0. Both should have had a score of 3, since they both have 3 estates in their hand/deck.

In the future, I would include tests that tested the score for all players at different stages of the game. Testing the victory card piles would also be of benefit in tracking down missing or lost points.

Only 50% of the lines, and 42.86% of the branches were executed for scoreFor. This shows that I did not test as thoroughly as I could have, and additional tests are required to fully test scoreFor. updateCoins, drawCard, shuffle, and compare were touched.

whoseTurn()

whoseTurn passed both of the tests I ran on it. The tests were simply to see if the function would return the correct player's turn. In order to capture potential bugs, I would implement more tests that would check for the correct player past the initial start of the game and turn 2.

100% of whoseTurn's lines were executed with these tests. updateCoins, drawCard, endTurn, shuffle, initializeGame, and compare were touched while running these tests on whoseTurn. Since whoseTurn is a very simple function, any bugs that would be found would most likely indicate either an improper whoseTurn call, or a different dominion function altering the turn state.

numHandCards()

numHandCards passed 9 of the 12 tests that I performed on it. I believe this is most likely due to how the initializeGame function works (player 2 does not gain a hand of cards until the start of their first turn, thus their hand size and deck size will be incorrect when tested for before their first turn). However, it was interesting to see that something was amiss when player 2 was asked to draw cards, as their hand and deck size were completely off.

There was 100% line coverage (no branches) for this function. updateCoins, drawCard, shuffle, initializeGame, and compare were touched during the tests ran for numHandCards. Initially, gcov was reporting that there was 0% coverage for numHandCards, which alerted me to the fact that I never actually called this function in it's test suite. After rewriting the tests and including the call to numHandCards, the bugs that were present before (obtained through getting the hand count and deck size through the game state) were still present.

Overall, only 16-19% of the dominion code lines were executed, and 15-20% of the branches were executed. This indicates that these tests did not test a whole lot of the entire dominion code. However, I still belive they were helpful in locating some bugs in the code; ones that I introduced with the 2nd assignment, as well as ones that were in the dominion code all along. It also helped in seeing some of the initialization quirks of the initializeGame function.

Unit Testing Efforts

Card Tests

Cards Tested: Adventurer, Smithy, Village, Council Room

Unique Tests (card specific)

Adventurer

Test: Did the card increase the amount of coins the player has? Increase should be between 2 and 6, depending on what treasure cards were drawn.

Smithy

Test: Did the player's hand size increase by 3?

Village

Test: Did the player's hand size increase by 1?

Test: Did the player's actions increase by 2?

Council Room

Test: Did the player's hand size increase by 4?

Test: Did the player’s buys increase by 1?

General Tests (all cards)

Was the player’s hand size affected?

Was the player’s draw pile affected?

Were the player’s actions affected?

Were the player’s buys affected?

Were the victory card piles affected?

Function Tests

Functions tested: fullDeckCount(), scoreFor(), whoseTurn(), numHandCards()

fullDeckCount()

At the start of the game was:

- The hand size correct for player 1 & 2?
- The deck size correct for player 1 & 2?
- The discard size correct for player 1 & 2?
- The total card count correct for player 1 & 2?
- The number of estates correct for player 1 & 2?
- The number of coppers correct for player 1 & 2?

scoreFor()

At the start of the game, was:

- The score for player 1 correct?
- The score for player 2 correct?

whoseTurn()

Does the function return the correct player’s turn?

- At the start of the game, is it player 1’s turn?
- After an endTurn() call, is it now player 2’s turn?

numHandCards()

At the start of the game, was:

- The hand size correct for player 1 & 2?
- The deck size correct for player 1 & 2?

- The discard size correct for player 1 & 2?

After calling drawCard(), was:

- The hand size correct for player 1 & 2?
- The deck size correct for player 1 & 2?
- The discard size correct for player 1 & 2?