Mark Moore
CS362-HW3

Bugs

The first bug I found was in the scoreFor(). The third of four loop should you use "state→deck[player]" for the limit of the loop instead of "state->discardCount[player]". This error will have the loop cycle only the amount of discard cards that player has at that time. So if one player has few of their cards in their discount pile at the end of a game, they are more likely to lose the game.

The second bug that I found was in testing the council_room function. I found that the discardCount is not incremented in the discardCard() function.

Hey separate thing that is not actually a bug is that there are two arrays declared for the cards used it during the game to be held in. The one declared in dominion.c is never used.

Unit Testing

All of my Unit and Card Testing fell in a range of 19-27% or each test. Untested cards predictably did not have any coverage.

Some interesting stats:

Smithy - 2 calls - 100% executed,
Adventurer - 2 calls - 100% executed,
discard() - 4 calls 100% executed,
cardEffect - 2 calls - 100% executed,
drawCard() - called 51 times - 100% covered -41% executed,
numHandCards - 11 calls - 100% executed,
shuffle() - 20 calls - 100% executed,
initializeGame() - 8 calls - 87% executed,
compare() - 537 calls - 100% executed,

cards inside cardEffect did not have statistics for their individual sections but they had commensurate coverage.

The most interesting thing I found in these statistics was that The initializeGame() function only had 87% execution. That means that there are lines in there that are doing nothing through multiple initialization's.

Then there are some functions that I had not thought about, like scoreFor(), that was called si6 times and had 100% of the blocks covered with 34% executed.

What I gather from these statistics is that I can definitely write some better tests to evaluate all of the actions that these functions take. As I progressed in this assignment I also became more adept at writing tests that evaluated specifically the target function.

Unit Testing Efforts

I spent the better part of 2 days just reading and understanding the dominion.c code. This code is very difficult to understand without commentary! I started to think that the major point of this assignment is to convince us students to verbosely comment on any and all code we write in the future. I put some new commentary into my branch of the code.

The first function I tested was the initializeGame() function. It took a while to figure out how the code worked together but, finally, I was able to organize a test sufficiently rigorous to display the efficacy of the initializeGame() function.

My test checks all the cards in the hand and deck of the player to sum the total amounts and arrive at the final number.

UnitTest2.c proves the scoreFor() function. My test incrementally adds cards which are worth a specific amount and records that amount with each addition.

UnitTest3.c tests the shuffle() function. I create an array of 30 elements containing various cards. Then, I copy that array into the array for Player 1. I then shuffle() that array and compare it with its previous version.

I repeat this process three times, showing that Random() function sufficiently shuffles the deck as to make it completely unpredictable.

Last, in UnitTest4.c, I prove the numHandCards() function. I manually change the size of the deck recorded for P1 and P2, then check the number though the numHandCards() function to see if it gives the correct number of cards for that player.

cardTest1.c tests the Smithy function. I introduced a bug in the Smithy function for homework two that sends the hand position Number into the draw card function instead of the Player number(0 or 1). The call to the drawCard function happens at the beginning of the Smithy function. As expected, when more than one is sent into the funcSmithy function as the hand position, the program has a Seg fault inside the drawCard function. I have now left the Seg fault in the test. You can delete the comment('//') out of the Second and third FuncSmithy calls in cardtest1.c to run the program with the Seg vault.

cardTest2.c tests the Adventurer function. Adventurer should reveal cards until 2 treasures are found, then add the treasure to the players hand and discard the other revealed cards.In HW2, I introduced a bug into funcAdventurer() that reversed the comparison or treasures drawn from 'drawntreasure<2' to 'drawntreasure>2'. This will cause the while statement to never excecute unless more than 2 is sent in for the drawn treasure. When more than 2 is sent in as cardsDrawn, the while loop cycles infinitely because the point of the loop is to draw cards until there are 2 in the players hand. cardsDrawn is incremented with each loop so the condition will never not be met when more than 2 is sent into the function.

cardTest3.c will test the council_room(). Council_room should allow the player to get 4 cards from his deck and all other players get 1 from their decks. I test that there is 1 card left in P1's deck, that numBuys for P1 is 1 after calling the function, that P2's hand has one more and that his deck has one fewer. I also check that P1's discard pile increased by 1 and that that new card is 'council_room'. In testing this function I found that the discardCount is not incremented in the discardCard() function. Therefore, the number of cards discarded by each player stays the same. I also found that the number of buys is incremented incorrectly.

cardTest4.c will test the outpost card function. The number of outposts played should be incremented by one and that card should be discarded from the players hand, leaving one fewer. Of course, the discardCard() function will not increment The number of cards in the discard pile because of the previously mentioned a bug.