Daniel C Stewart
Computer Science 362
June 6, 2016

Final Report

Over the course of this class I've spent a lot of time testing the dominion code that we were provided. When I first looked at the code and ran a 2 player game against a computer player, I didn't think there were actually that many problems with it, but delving in further I found out there were a ton of problems.

Throughout this course we made 4 unit tests, 4 card tests, a random tester for the adventurer card, random testers for 2 more cards, and a random tester that plays a game of dominion with 2-4 computer players. My 4 unit tests test the functions whoseTurn, drawCard, fullDeckCount, and buyCard. This seemed like a good mix of functions to start with, as whoseTurn is used in almost every other function, as is drawCard, and all 4 of these functions are vital to the game so they should all be functioning properly. For the 4 card tests I chose the smithy card, the minion card, the council room card and the village card. My 2 random testers that weren't for the adventurer card were for the salvager card and the council room card.

I'll first choose to use my tests (the random tests, unit tests, card tests) to examine the dominion code for Nic Desilets.

The results of my unit tests are as follows-

whoseTurn: 1 test passed 0 failed
drawCard: 3 tests passed 0 failed
fulldeckCount: 2 tests passed 0 failed
buyCard: 1 test passed 0 failed

Nic's dominion code passes all of my unit tests, which isn't surprising, as most of the tests pass without editing much of the original code.

The results of my card tests are as follows-

smithy: 1 test passed 0 failed
minion: 2 tests passed 1 failed
council room: 2 tests passed 1 failed
village: 2 tests passed 0 failed

Here some of the tests failed for the minion and council room cards. All of these test pass in my own code, and the tests the fail here are the same ones i found were failing because of the way endTurn and intalizeGame are set up. In this version of dominion, at the end of a player's turn the player discards their hand, having no cards in hand, and then the state passes the turn on to the next player, leaving the former player with 0 cards in hand. They don't draw 5 cards until it comes back around to their turn, which isn't how it should be to play a proper game of dominion.

The results of my random tests are as follows-

randomtestcard1 for salvager: 18 tests passed, 2 failed
randomtestcard2 for council room: 20 tests passed, 0 failed

Council room passed each of the 20 random generated tests, I didn't have any problems with council room either in my implementation of dominion without making any changes to it. Salvager has some failing tests. I ran both of these tests with the same seed that i used on my own, and randomtestcard1 failed the same 2 tests that i found it was failing in my tarantula.txt file. These tests i found failed because of how the salvager function is written, if the card chosen to be salvaged is the first card in the player's hand, it has index number 0 and it doesn't enter the if statement that actually salvages the card and gives the player the gold coins, that's probably what is happening here as well.

Running testdominion.c doesn't tell me much, as it really just plays a full game with computers, and if it doesn't cause an infinite loop or seg fault I don't get a lot of feedback from it. When I run it on this code the game exits completely, but only 1 player has a hand at the end, which is related to the problem found by the card tests for council room and minion.
Overall I think this dominion code is reliable for the state the game is in right now. Like my own code, it doesn't let players interrupt other players turn with a response play like they normally can during a game of dominion, so the endTurn and initializeGame problems aren't a huge deal, as the way the game runs right now, which players only being able to do things during their own turns, it just means that the players will always start a turn with 5 cards and won't have any extra cards they get to draw from other players card effects during their own turn, which isn't a severe problem.

Next I'll examine the dominion code from jiangzh, whose repository I made some bug reports for

The results of my unit tests are as follows-

whoseTurn: 1 test passed 0 failed
drawCard: 3 tests passed 0 failed
fulldeckCount: 2 tests passed 0 failed
buyCard: 1 test passed 0 failed

All of these tests pass, which as i stated before is to be expected for most implementations of dominion, unless the person broke one of the functions.

The results of my card tests are as follows-

smithy: 1 test passed 0 failed
minion: 2 tests passed 1 failed
council room: 1 tests passed 2 failed
village: 1 tests passed 1 failed

There are more errors here than there were before. I was expecting the council room error as i found it before for a bug report, and it can be fixed by simply increasing the number of actions and adding card draw. The minion error was also expected, as I don't think too many people will think to examine how many cards other players have when it isn't their turn, or to look at the endTurn function. One of the tests for the village card also fail, because it's giving the player the wrong number of actions.

The results of my random tests are as follows-

randomtestcard1 for salvager: 18 tests passed, 2 failed
randomtestcard2 for council room: 0 tests passed, 20 failed

Here randomtestcard1 is failing for the same reason as it was for Nic's code and my own originally, because players have 0 cards in their hand in between turns and only draw 5 at the start of their turn instead of at the end like they should. For randomtestcard2 all 20 of the tests fail, because this tests the council room card, which isn't drawing the right number of cards for the player who plays it.

Again, running testdominion.c doesn't tell me a whole lot, as it doesn't seg fault or enter an infinite loop. However as i stated before, it only prints out the hand for the player whose current turn it is when the game ends, because every other player has no cards in hand. Like i stated for the other implementation of dominion, this isn't a huge problem considering everything, the problems with council room and village are fairly large though as they don't operate how they are expected to at all.

In conclusion I learned a lot about debugging in this class. I came into it thinking that debugging was what we did in 261 and passed classes, where the code would seg fault or error and we would sit around for hours on end racking our heads and changing variables until we figured out what was wrong. Mutation testing and using tarantula both seem very useful to me, of course there are times when tarantula may not give you any useful information but for problems for things like this dominion code, where I didn't write any of it initially and don't even know where to begin to look for problems, it provides a good starting point of functions that are suspicious. Using mutation testing is a good way to tell whether or not unit tests that are created are good or not. I don't think that the full dominion random tester that i made is very useful, because it doesn't provide a lot of feedback after it’s been ran, I think it would’ve been more useful in hindsight to examine the return values of every call to playcard and to examine whether each card gave the proper output, and that the values at the end of the game matched up with expected values given the sequence of action cards. I don’t think my dominion code or the other two that i examined are even close to being able to play *real* games of dominion, as that would require changing the interface quite a bit, but they’re better than they were at the start of the class.