Test Report

Overall my experiences on working with this Dominion code has been rather challenging. Starting out I thought that I was going to be a lot better at this, mainly because I've had plenty of experience trying to work through bad code. (Mainly my own, sometimes classmates). This was an entirely different challenge. I quickly found out that working with someone else's code, especially this someone, requires you to think outside of the box. You cannot just sift through the code and expect to find all of the faults in it. I previously thought that most of debugging required knowing all of the code, for whatever reason I never thought about what testing would be like with programs that have millions of lines of codes across multiple files. This assignment opened me up to the entire field of testing and how it can actually be harder than writing the code in the first place. I had taken software engineering one the term before this, so I had a little experience with unit testing, but I never figured that I would actually use it again. Then I was present with this code, which luckily we were given the whole term to tackle.

For the first part, I simply refactored 5 cards out of the humongous switch statement in the cardEffect function. I didn't put too much time into this and actually was very close to refactoring all of the cards, as it wasn't too big of a deal and I thought it might help me in later assignments. In the end I decided to stick to what I was told to do just so that I wouldn't have to undo my changes later for whatever reason.

For the second part, I had to create 4 unit tests and 4 card tests. The card tests I created were for the Embargo card, the great hall card, the smithy card, and the village card. I decided to stick with easy cards to test so that it would be easy to tell if I was doing the testing correctly. I generally stuck to testing if the cards drew the correct number of cards or gave the correct number of money. For the unit tests I tested simple functions of the dominion game. I ended up getting a total of 25.94% lines covered in the game, which is not too much.

The third part of the project was a little bit harder. I had to create random testers that would each play initialize a game and then play a specific card multiple times. These games would be initialized with a random amount of players and a random seed. In order to work with this code I spend a lot of time using gdb trying to figure out what lines were not being hit. I had to get 100% line coverage of the adventurerCard so I would wait until the card was played, then I would see what my program would do and take note. I would do this a few more times and when it became clear a certain line wasn't being hit, I added another piece of logic to my random tester in order to ensure that line would eventually be hit.

I had a little bit of trouble with the fourth part of the project, specifically the differential tester. We were supposed to create a script that would compile someone else's dominion as well as our own and then compare the results together. Unfortunately I waited too long to start this part of the project so I did not get around to figuring out how to do this in time. I did get to test it against my own code though, which proved that at least my code was acting the same between runs. For the other part of the project we had to create a tester that would play full games of dominion. These games would all have random variables, including the players, seed, and starting cards.

Between the two other classmates' code that I tested there seemed to be little difference in reliability. From looking at the code and comparing it to mine (sadly my diff dominion never saw the light of day) I could tell that the only things that had been changed were the 5 cards that were refactored in the first assignment. Both people, Houpoc and Sullense got in the 80% range for line coverage. I also ended up getting the same coverage so that probably means there's more of a problem with my testing. Unfortunately I do not believe that this means there are fewer errors in either dominion code.

After testing my own classmate's code, I had to test my own code looking for a bug. This provided a rather interesting challenge as it was so open ended. I could find any bug in this giant program and fix it! I knew that finding a bug wasn't going to be hard, as there were plenty. It was all just a matter of finding a bug that I wanted to fix. I spent a lot of time in the gdb debugger and eventually found a consistent bug with one of the cards.

For the project, I ended up using the delta debugging style of testing. I spent a decent amount of time looking into the python scripts, trying to decipher what this code did. Unfortunately I have very little python experience so in the end I was not able to learn everything that was happening. I figured out how to run the code through the instrutions on Prof. Zeller's website, and I got results about failed assertions from one of my previouse card tests.

In the end I'd say my experience testing this program has been rather interesting. The transition from searching through all of the code trying to figure out what it does, to writing tests and having those do the work for me was interesting to see. My coding style, when it comes to debugging, has changed quite a bit in this term. I could definitely still use some work in certain areas, and hopefully I will be able to cement these skills some more. I'm glad that I was subjected to an experience like this, as to me it seems like a realistic project that I may have to undertake someday, I just hope I'll get paid for it next time.