Alec Haagenson
931647160

Testing dominion was interesting project for me because of my complete lack of knowledge about creating test functions as well as my complete inability to understand deck building games. A good portion of the course was spent just trying to figure out how to play the game for me. I found the rules a bit unintuitive, and I still don't really have the game figured out completely. Luckily, there were some online resources like the dominion wiki to help me to get a grip on the basics of the game. Due to this inability, along with my previously aforementioned lack of experience creating code tests lead to my first few assignments being downright awful tests. The results seemed to be inconsistent even given a a shared random seed because I was dumb and didn't realize that I needed to feed my dominion seed to my random number generator seed so that I could produce consistent results. As the course progressed, I definitely feel like my ability to write test cases improved, along with my understanding of the play mechanics. Once we moved onto the big test suite of testDominion.c, I definitely started to get a bit lost again. Even after writing the test, pinpointing down why something would go wrong, or what would cause the program to hang became a lesson in frustration. This was in comparison to the unit tests, which I felt were far easier to pinpoint down issues in implementation. I think this is partly because the existing code was so broken to begin with, and I really didn't want to pour any work in fixing any of that awful code, since it almost seemed broken beyond repair. I definitely feel like if I would have written test cases for all the individual functions and then moved onto the testDominion.c, I'd have a far easier time of pinpointing bugs in implementation.

Gcov was an interesting tool that I had never heard of before. It was actually really interesting to see how much code my tests were hitting. On my initial couple of unit tests, it made sense that I was usually hitting around 16%-22% of the code, since a good portion of the codebase is just the initialization functions. The interesting part came from running gcov on my testdominion.c code, which, depending on the length of game would hit anywhere from 52.23% to 58.35%. This surprised me as I had anticipated to hit more of the code base. Although, I had to comment out the use of Sea Hag, since this would cause my game to spiral into an infinite loop where Sea Hag would get continually played over and over, never properly discarded, or played correctly. In fact, looking at my .gcov file, it seemed that the majority of the missed areas were testing cards that I had not implemented into my tester. Which, in retrospect, is a pretty good reason to not have it hit such a large percentage of the code base. It also came in handy for making sure that I hit all my tests in each of my unit tests. After running a piece of test code, I'd run gcov on my test code to make sure that I was hitting all of the code that I needed to, and that the only unused pieces of code were print statements for various errors that never ended up arising, like those in the initialize game function. It also allowed me to see where my test would stop if I get stuck in an infinite loop, since I could just see what line of code was last hit before the incident. Overall, it was definitely one of the most interesting and useful parts of the course for me to learn about.

Going over the code results, Sea Hag would always cause my code to stall or crash altogether. I never experienced any seg faults while doing the run, but the variance in stall versus infinite loop I was never able to discern what had caused it. As well, sometimes tribute would trip and hold everything up. Everything else seemed to at least be working relatively well, although some gold values were not being properly doled out, as all the values ended up just being one on every round.

Going over Khue Trank's code, I ran into similar issues with mine. Although gold values were being correctly given out, it still ran into multiple issues with Sea Hag. It would do a similar thing to my code where it would infinitely spin out, but I never noticed a hanging with the card. On top of this, gold values were being given out correctly. Digging through the code, there seemed to be some issues with the way that the discard move interfered with the way that Sea Hag interacted with the rest of the game. As well, there were a few bugs in the actual cards themselves, especially adventurer which didn't seem to actually tally on points. The gcov results showed anywhere between 46.55% and 53.27% which were slightly lower than my own, which means some cards may not have hit their conditionals correctly.

It was a similar story with Anita Chow's code as well. The sea hag issues were just as prevalent as with Khue's. The infinite sea hag was definitely a problem that was seen again. I believe that just as with Khue's code the issue was related to how the discard was performed, which lead to a never-ending loop. As well, it seems some of her gold values were also a little off. The gcov results for her code were more in line with mine, hitting between 49.76% and capping up to my final value of 57.35% which hit more code than Khue's.

Overall though, the dominion code itself is a buggy mess and should be completely scrapped and re-written, considering that it really wouldn't take too much time to re-write in a manner that was more consistent and programmatically made more sense.