Braden Ackles
CS362
Final Project Test Report
3/3/16

## Experience:

This past term working with Dominion was difficult at first, but then as I became more familiar with the program and the code it became easier. I found it very difficult to be given someone else's code and get to know it well enough to modify it without creating more bugs. Especially since it wasn't perfect in the first place. But after a week of working with it (and learning how to actually play dominion and how it works), I felt comfortable enough to make changes so that my program would still work.

## Sophia Liu:

Running Gcov on Sophia's implementation of dominion returned a 43.86% coverage score which isn't terrible considering the number of possibilities and edge cases that are possible. This number likely will change dramatically depending on how many different cases happen and if you let the computer play and test randomly or if a human player was making the decisions. After running a full game of dominion on her implementation of dominion it came back with a code coverage of 68.79% code coverage. These are relatively good code coverages, for running general or tests that were meant for other implementations. Sophia's tests and code coverage is much higher closer to 80+% so this shows that the reason the code coverage is lower is because of my tests not being built specifically for her implementation. The different function names and organization of the program make up the difference. My tests on her code do not have a reliability factor that would acceptable in a professional environment however after running her tests on her code it would be more acceptable for reliability tests because of the change in coverage. That is not to say my tests or bad or good, but that our implementations are different and mine doesn't necessarily have the hardcoded information that hers has and expects. When testing Sophia's code for bugs it was easier to run her testDominion.c to find bugs because it was meant to play the entire game on her dominion.c, however running my card testers was able to find bugs on her cards because of the Agan's principle "get a fresh view", my card tests test her code differently and are able to find bugs that she previously wasn't testing. The difference between my testers and hers are we decided to test different things. Her card testers will find a bug if something is wrong, however she for the most part fixed any problems that would occur, where her code wasn't fixed to solve the problems that my cardtests are trying to find.

This is a good reason why having a large test suite will help find more bugs. Especially if they are written by different people because everyone has a different look at the code and how to find bugs. So ideally in a real world environment to create maximum reliability we would run both our tests as well as many other people's tests.

Dominon.c

```
flip3 ~/assignments/CS362/cs362w16_acklesb/Final/SophiaCode 22% gcov dominion.c
dominion.c:source file is newer than graph file 'dominion.gcno'
(the message is only displayed one per source file)
File 'dominion.c'
Lines executed:43.86% of 570
dominion.c:creating 'dominion.c.gcov'
```

TestDominion.c

```
flip2 ~/assignments/CS362/cs362w16_acklesb/Final/SophiaCode 43% gcov testdominion.out
File 'testdominion.c'
Lines executed:68.79% of 173
testdominion.c:creating 'testdominion.c.gcov'
```

## Christian Armatas:

Initially I had much trouble running Christian's code. It would not compile with my code or with his code. I found that the problem actually ended up being the dominion_helpers.h file. He had edited the file which was actually preventing the file from compiling if the correct .h file was not present. After fiddling around with his code for a while and finally being able to run his. It came back with a coverage of 29.67% code coverage. This is not a very good coverage by default. Although it could be contributed to many things. For example, I could not be running it the way it was intended to run. There are no instructions on how to run it so I may be missing information or commands or even files. When running my version of testDominion.c on his code it came back with the same code coverage as when run on Sophia's code. This is likely due to the fact that we all were told to modify the same things over the course of the term. Meaning that theoretically the same things should be different between all our codes. This would cause the same code coverage for both theirs because the same things were different. This is also on top of the edge cases that are unlikely to be fulfilled. This is not a reliable test of the code, and would not be acceptable in a professional environment. After spending half an hour trying to get Christian's tests to run correctly I decided to skip over them. Although I assume that his tests have a much greater code coverage then mine. This would be because they are written specifically for his implementation of dominion and testing the things that he wrote.

Once again just like Sophia's running multiple different tests written by different people on a program will expose more bugs. Because it will not only have different views and test different things but also can open up new ways to write the code as well as test it. This is one reason why I wasn't able to tests his as thoroughly because I couldn't get his tests to run. If I was able to run his tests as well it would be an even more reliable way to test the program. However, given the current situation I do not think the reliability is very good and would by no means say that it is perfect.

Dominion.c

```
flip2 ~/assignments/CS362/cs362w16_acklesb/Final/ChristianCode 100% gcov dominion.c
dominion.c:source file is newer than graph file 'dominion.gcno'
(the message is only displayed one per source file)
File 'dominion.c'
Lines executed:29.67% of 600
dominion.c:creating 'dominion.c.gcov'
```

TestDominion.c

```
flip2 ~/assignments/CS362/cs362w16_acklesb/Final/ChristianCode 69% gcov testdominion.out
File 'testdominion.c'
Lines executed:68.79% of 173
testdominion.c:creating 'testdominion.c.gcov'
```

## Overall Reliability:

As previously stated adding more tests will obviously help with testing reliability. Currently the reliability of the code is not good. But this is because of my test script being written in conjecture with the development of my implementation of dominion. This is not a bad thing though because it would be extremely hard to account for all the possible functions that people can name and make. None of us are perfect and that was the point of the class. To get a good feel about different ways to test programs; from Delta Debugging, Tarantula, to just looking at it. I personally learned a lot, previously the only way I knew to debug was to look at the program and add print statements and logs. Which is extremely inefficient.

The best way to improve the reliability of the testing suite is to add more tests. ALL of the TESTS!!!!! Random testing is good for this program because it has so many possible combinations that can all end up happening in the same game. So tests that randomly tests small parts of the program would be good, also tests written by different people is good because the outlook and understanding they have of the code will change the tests they are able to write. An important thing to remember is even for small programs it is extremely hard to have 100% reliability. Even 10 lines is hard. So over the course of the term we would never be able to achieve this. This is not to say testing is not important, because it obviously is, if we didn't test our programs 99% of us would never have a working program the first, second, or even nth time. It's not practical to not test software.

Last words, I learned a lot this term about testing and techniques to do it better. I know that in other classes I was taking this term, I was already able to use some of the skills I learned in this class. I think this class was very useful and am glad to have the knowledge that I have now going forward. Lastly thank you to the TA for reading this, especially during finals week, I know it probably wasn't very fun to read all of these.