Jiawei Liu
CS 362
June 6, 2016

Test Report

Back to the begging of this term, I didn't have a complete debugging experience. Before this class, I usually add some printf() statement to check my program is works fine or not. If my program will work "fine", I will not use different values or situation to test my problem. However, I realized that write tests are very useful for test and debug program. If we have a big project and the logic is complex, using printf() is an awful idea for debugging, that will waste me a lot of time, and some bugs are cannot find out by looking the code directly. Therefore, unit test, random test and mutation test will be a good in this case. In my opinion, dominion is a good project to let student learn how to write tests such as unit test and random test.

The first assignment of this class was not difficult, but it gave me a chance to review the whole project of dominion and understand construction. In addition, I also google the rules and each card effect of dominion. To be honest, assignment 1 let me have a general view of this game and I gain a lot benefits from assignment 1, even assignment is not difficult and only need to make 3 changes for future testing.

The second assignment of this term was write 4 unit tests and 4 card tests for dominion. At first, I was coverage testing my unit tests by using gcov, but the coverage of my test is very low, I remember I only get about 12% coverage on my initial unit test. So I go back to check my tests. Then I realize that I only test the player's draw cards from the deck is correct or not. So I didn't cover a lot in my unit test. And then, I add more test case in my unit test such as coin number, discard card, etc. Finally, I got 27.40% on my unit test. I also learned the output format is very important, because if the result output is garbage, nobody would like to see it. So I spent few hours to let my test result look clear. Here is my coverage and unit test looks like:

```
*************** Unit Test Result *****************
**             Unit Test 1: getCost()            **
**                    Passed                     **
********************** End ************************


*************** Unit Test Result *****************
**             Unit Test 2: fullDeckCount()      **
**                    Passed                     **
********************** End ************************
```

File 'dominion.c'
Lines executed:27.40% of 646
dominion.c:creating 'dominion.c.gcov'

The third assignment of this class is write a random test generator for three Dominion cards. This assignment is more deep than before. Acutely I have never write a random test before. So I learned a lot new stuff after finish this assignment. After my research, I found out the random test is looks similar with unit tests. So I didn't have a lot trouble to implement random test. In my opinion, random test is a good method for debug if we just want to improve test coverage. In dominion, it is very unlikely that a same game be played twice or more. So random test cases such as gamestates, deck counts and player hands are very useful in testing as a small portion of the vast amount of possible games. On the coverage, the situation is same with my assignment 2, I got a low test coverage at first, but I fix this issue by add more test cases. Finally, I got 21.96% on dominion.c file, and 100% coverage on function that I tested, which I very proud.

```
File 'dominion.c'
Lines executed:21.96% of 560
Branches executed:23.74% of 417
Taken at least once:15.59% of 417
Calls executed:11.58% of 95
Creating 'dominion.c.gcov'
```

```
Function 'my_Smithy'
Lines executed:100.00% of 5
Branches executed:100.00% of 2
Taken at least once:100.00% of 2
Calls executed:100.00% of 2
```

The fourth assignment is also write random test, but this time, we need to write tests for whole dominion game. In this assignment, I realized that random test is a good ,ethod to test a big project with a lot of different situation. Because the unit test only provide one situation to test, but random can create many different situation randomly such as the player numer can be 2 to 4, the hand card number can be 1 to 5. If we can write the random test properly, random test should be able to cover alomost all situations. And

found more hidden bugs. However, if the program is not very large, I feel random test is not a good idea since random test is large, and need spend more time to build.

For the view of two classmates, I choose jiangzh and houw code to test. After my unit and random tests, the test coverage is no quite different.

For jiangzh's code:
the unit test result is "Lines executed:27.35% of 563"
the random card test result is "Lines executed:24.16% of 563"
the test dominion result is "Lines executed:77.56% of 563"

For houw's code:
the unit test result is "Lines executed:31.21% of 559"
the random test card result is "Lines executed:27.35% of 559"
the test dominion result is "Lines executed:78.22% of 559"

Based on my test results, houw, jiangzh and my code coverage are almost same. We only have couples of percents differences. Code coverage on houw' code have a tiny lead than jiangzh and my code. I think maybe houw didn't change a lot code on dominion or changed some unpopular cards. So the coverage is a little bit higher.

My test dominion coverage is 78.23%, and my dominion game is playable, and two classmates' test coverage is around this number. So two classmates' dominion game should also be playable. But that doesn't mean their code have no error. Based on my random test, jiangzh's have a error on village card. Jiangzh's village card always return a wrong action number. houw's code also have bug. For example, on the minion card, the test is failed since the handcard was wrong. In general, my classmates code have almost same reliability with my code. All of our code are playable, but still contains few bugs.

In conclusion, this class is a challenging and insteresting experience, I learned a lot new stuff in this course, especially debuging method. As I said on above, my common debuging method was "printf()", but now, I learned unit test, random test and mutation test. And I am more practised on debuging process. Working with dominion has given me an opportunity to learn how to play dominion, writing tests to debug program. I believe dominion is a interesting and valuable project to practice various test method. This class has extend my view of debug position in a software project. I never realized that debug is

so important for a big project before. I gained many benfits from this class and hope this experience can let me have better perform in the future.