Name; Xiaoyong Zheng
Date: 06/04/2016
CS 362 final project test report

This is a test report documenting my experience testing the dominion code. I believed this course is a test study course and start at assignment 1. For assignment 1, there is no directly test in that assignment, but we have to refactor four card functions. If we want to refactor a card, we have to learn about how to use this card in the game. When we want to learn how to use a card in the game, we have to learn how to play this game. It is applied to the first rule of Agan debugging test: understand whole program. Before this class, I had not played Dominion, in order to finish assignment 1, I had to do a whole research about the dominion game, I download this game and played more than ten times. When I refactored card functions, I learned many basic functions, it is very helpful for my test in my next assignments. At last, I changed to values in my refactored functions, it is used to next tests. It applied to the second rule of Agan debugging test: make it fail.

For the second assignment, we learned about unit test. Unit test showed us "divide and conquer" technique. In this assignment, I created four unit tests and four card tests. I believed the purpose of this assignment is that Dr. Groce wanted to show us how divide the dominion code into smaller sections and test smaller sections. It is easy to debug if a large program divide into some smaller parts. I also learned gcov. The gcov let me see what portions of the dominion code that I covered after I run the test. I test my code and my classmates Jiangzh and Wangzhao's code. The coverage report is:

My code, the coverage of cardtest1 is Lines executed:93.33% of 30
The coverage of cardtest2 is Lines executed:92.59% of 27
The coverage of cardtest3 is Lines executed:92.59% of 27
The coverage of cardtest4 is Lines executed:91.67% of 24

The coverage of unittest1 is Lines executed: 45.00% of 20
The coverage of unittest2 is Lines executed: 93.93% of 22
The coverage of unittest3 is Lines executed: 87.75% of 36
The coverage of unittest4 is Lines executed: 88.57% of 13

Wangzhao's code:
The coverage of cardtest1 is Lines executed:86.67% of 30
The coverage of cardtest2 is Lines executed:92.59% of 27
The coverage of cardtest3 is Lines executed: 91.59% of 27
The coverage of cardtest4 is Lines executed: 85.83% of 24

The coverage of unittest1 is Lines executed: 40.00% of 20
The coverage of unittest2 is Lines executed: 90.91% of 22
The coverage of unittest3 is Lines executed: 77.78% of 36

The coverage of unittest4 is Lines executed: 84.62% of 13
Jingzhi's code:
The coverage of cardtest1 is Lines executed:85.27% of 30
The coverage of cardtest2 is Lines executed:88.39% of 27
The coverage of cardtest3 is Lines executed: 89.29% of 27
The coverage of cardtest4 is Lines executed: 91.56% of 24

The coverage of unittest1 is Lines executed: 78.00% of 20
The coverage of unittest2 is Lines executed: 85.41% of 22
The coverage of unittest3 is Lines executed: 74.73% of 36
The coverage of unittest4 is Lines executed: 83.62% of 13

I found if I use unit test, the coverage is difficult to achieve to 100 percent. The average value is 80% to 90%. Because we just test some parts of dominion and setting some values to test. I learned if I wanted to increase the coverage of a program, I need to write a test that use multiple functions. But after assignment 3, I learned the other way to increase the coverage is use random testing.

In the third assignment, I created three random test. In these random test, each value is random setting by program. And I also use gcov to check the coverage. All test can achieve to 100 percent. After I test my classmates' random test, I believed my test is reliability. Everyone can achieve to 100 percent. Hence, I learned random test is better than unit test. But these random test just test a part of dominion, I was not sure the test cover all of the possible scenario. Thus in the fourth assignment, we have more random test.

In the assignment 4, I needed to play the whole dominion game randomly, I created a test code that can test whole dominion randomly. In this test, each value of variables is randomly. For example, player, seed and card. It can make more possible scenario and increase test reliability. After random test, I had a new question, how make sure my test code is correct? That is I learned in assignment 4 part 3.

In the final project and assignment 4 part 3, I had to changed dominion.c and used card test and random test to found least three bugs from my classmates. I mainly used divide and conquer technique to find the bug in my classmates' code, and I had to fix least one bug. Hence, I used gdb to find where the bug is. After I found some bugs, I can communicate my classmates and make sure the bug is correct. Because everyone created some bugs in assignment 2. If the bug is incorrect, my debugger itself had a bug. It is a good way to test my debugger. Thus change test program can debug my debugger.

In conclusion, This is the first I debug a big program, I leaned lots of test and debugging skills from this course. I believed this course is that teaching us how to debug course. Debug is so important, difficult and really unpredictable. Because sometimes I don't know if the program have a bug or my debugger itself have a bug. According to study, I learned there are many ways to debug such as unit test and random test. Each one has their own advantage. Unit test can test more specific function and more easy. Random test can make more scenario and have a high converge. To test debugger, I can

change the test program or use other debugger such as delta debugger or tarantula debugger.