Erin Alltop
Assignment 3
1-23-18
CS362 – Winter 2018

Bugs

For Assignment 3, I tested four different cards and four functions of the dominion code. They are:

- Smithy
- Adventurer
- Village
- Council_Room
- isGameOver()
- UpdateCoins()
- fullDeckCount()
- gainCard()

After completing all of the tests, I found two bugs that I implemented in Assignment two. One was in the Smithy card, which works as expected until discard when it puts the discarded card into the next player's hand. This was successfully caught in test 4 of my card test.

The second bug I caught was in the Village card which I also implemented for Assignment two. Instead of three actions being gained with this card, two were gained instead. This was successfully detected in test three of the village card test.

I was expecting to catch a bug in the Adventurer card which I placed in Assignment two, but I had trouble getting that one to work correctly. In this bug, the discarded card goes incorrectly into the next player's hand. I did do a test for this (test 4), but it passes and I was unable to determine why. More testing (and probably debugging) is needed for this case.

Unit Testing

For all of my functions, I found that I achieved 100% branch coverage, though not 100% line coverage. This seems to be because I was not using asserts and instead using if/else statements as suggested for more clear testing. Since not at least one of the if/else statements was not accessed in each of the tests, 100% line coverage was not possible.

Smithy:

```
File 'cardtest1.c'
Lines executed:87.88% of 66
Branches executed:100.00% of 22
```

```
Taken at least once:63.64% of 22
Calls executed:78.95% of 38
Creating 'cardtest1.c.gcov'
```

Adventurer:

```
File 'cardtest2.c'
Lines executed:87.88% of 66
Branches executed:100.00% of 22
Taken at least once:63.64% of 22
Calls executed:78.95% of 38
Creating 'cardtest2.c.gcov'
```

Village:

```
File 'cardtest3.c'
Lines executed:86.84% of 76
Branches executed:100.00% of 26
Taken at least once:61.54% of 26
Calls executed:78.26% of 46
Creating 'cardtest3.c.gcov'
```

Council_Room

```
File 'cardtest4.c'
Lines executed:87.84% of 74
Branches executed:100.00% of 24
Taken at least once:62.50% of 24
Calls executed:80.00% of 45
Creating 'cardtest4.c.gcov'
```

isGameOver():

```
File 'unittest1.c'
Lines executed:89.74% of 39
Branches executed:100.00% of 10
Taken at least once:50.00% of 10
Calls executed:83.33% of 24
Creating 'unittest1.c.gcov'
```

updateCoins():

```
File 'unittest2.c'
Lines executed:92.94% of 85
Branches executed:100.00% of 12
```

```
Taken at least once:50.00% of 12
Calls executed:81.82% of 33
Creating 'unittest2.c.gcov'
```

fullDeckCount():

```
File 'unittest3.c'
Lines executed:90.77% of 65
Branches executed:100.00% of 18
Taken at least once:66.67% of 18
Calls executed:80.65% of 31
Creating 'unittest3.c.gcov'
```

gainCard():

```
File 'unittest4.c'
Lines executed:89.06% of 64
Branches executed:100.00% of 20
Taken at least once:65.00% of 20
Calls executed:80.00% of 35
Creating 'unittest4.c.gcov'
```

Unit Test Efforts

This assignment took considerably longer to complete that I initially expected. Thankfully I started the assignment early so that I did not have trouble finishing in time. After watching the lectures and videos, I found the best way to isolate a function from other functions as much as possible is to find simpler functions that had fewer additional functions in them (to unit test and true as possible). I started by looking through the code and finding shorter functions that don't call many other functions. The isGameOver() function was the best candidate for the unit tests in this regard. After this, I wrote out what was expected of each card or function. For cards this was easier as you can follow the general rules of the card (add one card, discard one card, etc.) while adding other less-obvious rules (e.g. no state change to other players, supply decks unchanged). After the initial planning stages it was just a matter of writing out the code in a way that made sense and worked.