Holly Switzer
CS 362
Assignment-5
2/28/18

Bug-Reports:

- Bugs from unit testing
    - isGameOver()
        - doesn't account for accidental negative provinces
        - doesn't account for accidental negative supply counts
    - updateCoins()
        - doesn't check for unknown players
        - doesn't check that hand count is correct
    - smithy()
        - hand count not correct
- Bugs from manual debugging
    - cardEffectAdventurer()
        - draws 3 treasure cards instead of 2
    - cardEffectMine()
        - invalid flag, which is taken as flag 0 and will add to discard instead of added to hand
- There was nothing too different from the testing done on my own individual code compared to my group members code. Though when manually debugging my group members code, I was able to see where the bugs were introduced based on what the card functions were actually supposed to do.

Test Report:

- My experience with testing the Dominion code has been very interesting and overwhelming.
    - When using the tests against my own code, I had a better idea on what the outcome should be, but when testing my group members code, I was not as sure on the outcome and if I am catching all of the bugs. I was able to manually locate bugs in the code, but I cannot confidently say that I have found all of the bugs that I have tested for, which is intimidating.
- Testing someone else's code is quite scary as well
    - I can barely remember my own codes and each little part to it, let alone someone elses much larger program
    - I have also tried playing Dominion countless times and have not even come close to winning. So sad.
        - I feel as though I have a decent understanding of the game, but I cannot even win one game, therefore I am not confident in my knowledge of the program

        - If I am not confident in my knowledge of the program, I am more than likely not testing it to the full capacity.
- My tests show only a few bugs, which would make you think that the program is fairly put together well
    - But I found a few more bugs just by looking at my group members code and that makes me lose confidence in how well this program is written, especially since my testing is only actually testing a small portion.
- The code coverage information has been a fascinating topic to me
    - The fact that a program just magically counts and tracks the coverage kills me
        - I would like to see the step by step of this process and how it actually works
    - It is very handy to see the coverage so that you can have and strive for goals to cover 100%
    - Without knowing the coverage, you may think that you have a very well put together test when it only covers a small amount
    - It is also hard to accept that 100% coverage does not mean that you have a perfect test, it just means that you have perfect coverage
        - But it is getting you one step closer to that non existent perfect coverage you are going for

Debugging:

- I use Xcode on the Mac for my testing and debugging, but the gdb process will be very similar
    - I find the best way debug my codes is to throw in a ton of breakpoints in the beginning
    - I will then walk through the code step by step and watch the variables update after each line
    - This will help to locate the bugs to see when, where and why the bug happened
    - Because it has already been narrowed down, you can change the code right there to fix the bug and run again (adding more breakpoints if needed)
    - Once that bug has been fixed, you can keep moving through the code until you find another bug and work to fix it
    - Without breakpoints, the program will just run and it will be difficult to locate the bugs and changes