CS 362 Assignment 5

By Josh Hanna

Bug Reports

Bug #1:

For the first bug I found, I was looking at the smithy function in the code from my team member. I found that the bug was that it was supposed to be that at the end of the function, it was supposed to discard a bug but ended up never doing it. I found this bug by testing how many cards were in the total hand before you call the function, and then see how many cards I have at the end of the function.

Bug #2:

For the second bug I found, I was testing the council room action from the choose card. After looking over the function and testing it, I found that the numBuys does not increase whenever the function is called. I found this out by doing basically this same thing with bug #1. I put an expected value for the function, and then I found the that the expected value was always 1 number higher than the value that was returned so knew the bug was there.

Bug #3:

For the third bug, I was looking into the adventure function. I found that the were 2 bugs in this function. For the first bug, I found that the drawntreasure was set to 2 instead of 1. I found this by calling the function and seeing how many cards were drawn at the end of the function.

Bug #4:

For the last bug, I found it at the end of the adventurer function. This bug was tricky to find because it was a really minor one. This just never increased the value of z in if the card drawn was money card or not. If it wasn't a money card, then they needed to add it to the temp hand and incease the z value. This bug would just overwrite whatever was in the temp hand and never increase the value.

Test Report

For the testing of my teammates code, I didn't have to change much of unittest for the code to work on the dominion.c file. I found that after going through the test and getting each test result, that the code that was given to me wasn't very buggy at all. It seemed to run fine, and I never experienced any seg faults or the program crashing. Here are some reports I got while going through the code:

```
File 'unittest1.c'
Lines executed:93.33% of 15
Branches executed:100.00% of 4
Taken at least once:75.00% of 4
No calls
unittest1.c:creating 'unittest1.c.gcov'
```

```
File 'unittest2.c'
Lines executed:90.91% of 11
Branches executed:100.00% of 2
Taken at least once:50.00% of 2
No calls
unittest2.c:creating 'unittest2.c.gcov'
```

```
File 'unittest3.c'
Lines executed:88.89% of 9
Branches executed:100.00% of 4
Taken at least once:75.00% of 4
No calls
unittest3.c:creating 'unittest3.c.gcov'
```

```
File 'unittest4.c'
Lines executed:92.98% of 57
Branches executed:100.00% of 16
Taken at least once:75.00% of 16
No calls
unittest4.c:creating 'unittest4.c.gcov'
```

Debugging

For the debugging part of the project, I found that using your own code and putting in test cases was the easiest way to debug the code. For the first unit test card, I edited the code and made sure that all the code was passing before I moved onto the next part of the program.

```
unittest1
TEST PASSED: Values matched. Expected value: 0   whoseTurn function return value: 0
TEST PASSED: Values matched. Expected value: 1   whoseTurn function return value: 1
TEST PASSED: Values matched. Expected value: 2   whoseTurn function return value: 2
TEST PASSED: Values matched. Expected value: 3   whoseTurn function return value: 3
Function 'main'
```

I did this for unit test 2,3,& 4 too.

```
TEST PASSED: Values matched. Expected value: 4   numHandCards function return value: 4
Function 'main'
Lines executed:90.91% of 11
Branches executed:100.00% of 2
Taken at least once:50.00% of 2
No calls
```

```
unittest3
TEST PASSED: values of the tested arrays match.
TEST PASSED: values of the tested arrays match.
TEST PASSED: values of the tested arrays match.
TEST PASSED: values of the tested arrays match.
TEST PASSED: values of the tested arrays match.
TEST PASSED: values of the tested arrays match.
TEST PASSED: values of the tested arrays match.
TEST PASSED: values of the tested arrays match.
TEST PASSED: values of the tested arrays match.
TEST PASSED: values of the tested arrays match.
```

```
unittest4
TEST PASSED: Values matched. Expected value: 3   updateCoins function return total: 3
TEST PASSED: Values matched. Expected value: 9   updateCoins function return total: 9
TEST PASSED: Values matched. Expected value: 13   updateCoins function return total: 13
TEST PASSED: Values matched. Expected value: 16   updateCoins function return total: 16
Function 'main'
```

I found that most of the functions were really easy to fix and that once you found out that your missing a number then you could go back into the code and fix it easily.