Stephen Townsend
CS 362-400
11/19/17

Assignment 5

Bug-Reports

Using my cardtest4, I noticed that the Great Hall's benefit of adding an extra action failed. As a result, I then ran my randomtester1 because this also tests the Great Hall. After running 10,000 passes, all of the runs that I could see had the same result. The add one to action failed each and every time. Looking at the source code, it looks like there was a bug that decreased the action by 1 instead of increasing it by one.

Using my cardtest3, I noticed that the Village's benefit of two extra action failed. As a result, I then ran my randomtester2 because this also tests the Village. After running 50 passes, all of the runs had the same result. The add two to actions failed each and every time. Looking at the source code, it looks like there was a bug that added 20 actions instead of two.

Using cardtest1, I noticed that the card count, for the hand count, after playing smithy, was not correct because it failed that specific test. It looks like there was a bug where the code should be adding three cards to the hand, but only one is added. There was also an issue with how many cards were being discarded. There should be one card discarded, which is the smithy, but there were three cards being discarded instead.

Using cardtest 2, I noticed that there was a problem with how the code was counting down. As a result, I looked at the code and noticed that instead of decreasing by 1 it is decreasing by 10. This made it so that all the cards were not being discarded from the players hand. Thus, making the hand count throw a failed test.

Test Report

Testing my partners dominion code went very well. It was luck of the draw where Alex refactored the same functions as I did. As a result, the cardtests and random tests that I created for my dominion were ready to test against his.

Below is a chart of the code coverage running all of my cardtests, unittests, and randomtests.

Test	Coverage
Cardtest1	84.62 % of 39
Cardtest2	83.33 % of 42
Cardtest3	84.00 % of 25
Cardtest4	84.00 % of 25
Unittest1	86.96 % of 23
Unittest2	76.72 % of 116
Unittest3	87.88 % of 33
Unittest4	83.33 % of 12
Randomtestadvetnurer	88.64 % of 44
Randomtestcard1	96.30 % of 27
Randomtestcard2	88.89 % of 27

After running the tests, a lot of my tests were throwing flags. There was no part of the code that passed every test. As a result, I feel that there are a lot of bugs in the code and it is not to reliable.

There were 6 bugs that I found with the existing tests I had ran.

Bug1:

Smithy

Expected:

Add 3 cards

Results:

Only adds 1

Bug2:

Smithy

Expected

Discard 1 Card

Results:

Discards 3 cards

Bug3:

Great Hall

Expected:

Add 1 to action

Results:

Action decreased by 1

Bug4:

Village

Expected:

Add 2 to action

Results:

Action added by 20

Bug5

Council Room

Expected:

Add 4 cards

Results:

Added 2 cards

Bug6

Adventurer

Expected:

Counter to decrease by 1

Results:

Counter decreased by 10

Stephen Townsend
CS 362-400
11/19/17

Debugging

During the process of creating the unit tests and random tests, I was having a seg fault in my adventurer code. The tests would run one output statement and then seg fault. For debugging, I put my code into visual studios to debug it and I found that it would not compile. I was getting an error that one of my variables were uninitialized.

Adventurer (VS Showing when compiling)

Severity	Code	Description	Project	File	Line
Error	C4700	uninitialized local variable 'cardDrawn' used	Project1	d:\users\stephen\desktop\project1\project1\dominion\dominion.c	835

I though that maybe this was the problem all along. As a result, when I got VS to compile the code, I placed a breakpoint at the beginning of my adventurerRefactor function so that I could step through the entire function. I found that the code still crashed, but it was at the end of the function when using my testcard2. I looked at the crash and it was a read access violation error. There seemed to be an issue with the temphand array in adventurer and where the program was attempting to read corrupted memory, or a memory location that was not valid. Looking at the contents of `temphand[z-1]`, it contained trash. Moreover, I noticed that I had passed a copy of temphand into adventurer and did not pass the memory address. I modified to code to pass the address from cardeffect to adventurerRefactor.

When I ran the code again, I was still having a memory crash at the same location where:

```
while(z-1>=0){
 state->discard[currentPlayer][state->discardCount[currentPlayer]++]=temphand[z-1];
 // discard all cards in play that have been drawn
 z=z-1;
}
```

Looking at the snippet above, I decided to see how z was being handled. Originally, I passed a pointer for z to pass the address. However, I found that this was the problem. I placed a breakpoint at the function call in cardeffect. From there I noticed that before z was passed it had a value of 0, but when it went into adventurerRefactor it contained trash (-8524162). This showed me that the variable was pulling trash from memory and not keeping 0 as its value.

I had two choices:

1. I could keep z passed into adventurerRefactor and reinitialize the int variable to 0.
2. I could remove the passing and just create a local variable of z initialized to 0.

I decided to go with option two because it would be a cleaner option. Once I made this change, I did not have a read access violation error in visual studios. I then reran my cardtest2 with the dominion code on the flip server and behold, I did not have a seg fault.

This was an error that was not intentional and I could not figure out why this was happening when running the code on the flip server. I did not write any code, or test, in Visual Studios as I normally would have for this class. However, the debugging aspect of VS is very good and allowed me to find the seg fault quick.