Assignment 4

Saturday, November 16, 2019 7:26 PM

Ryan Holland
hollryan@oregonstate.edu
11/18/19
CS 362
Assignment 4

Random Testing

I looked through the source code and tried to figure out where to start. I decided to try to get randomtestcard1.c to run the baronCardEffect() function. I spent a while trying to get the makefile working. I set it up so I just run "make restart", and it cleans everything, recompiles, and reruns the tests. I had to feed the function a gameState and a few other things, so I figured out how to generate those. It also needs 10 random cards, which were hardcoded as the same 10 cards in the provided code, so I built a shuffler. I ran into some good old syntax issues and plenty of seg faults trying to shuffle and print array values in c. I generate random but plausible values for most of the variables in state.

Eventually, I got it to run baronCardEffect() with my randomized input. Then I had to test it to see if the before & after values were as expected. I made it declare the variables before and after running the baronCardEffect() function and print errors if it doesn't behave as expected. I was able to reproduce the bugs that I had introduced. One of them was an endless loop, so I replaced it with a print function and return -2 to signal that the part of code with the loop had triggered. I guess another way would be to set a timeout to escape the loop. There are plenty of variables I didn't track and conditions I didn't check for. I guess you have to choose a few, or if you want to be rigorous, write a function that checks every variable before and after, and you tell it the expected changes.

I copied my baron setup code for the other cards, just changed the function call, and got good coverage on them.

Code Coverage

I got 90% coverage on Baron with just 100 test cases.
My randomtestcard1 is calling baronCardEffect from within a function I called "test()". I'm not sure if I'm correctly running gcov on just the target function, since I had it nested in the "test()" function. But if you do gcov on dominion.c, it seems to correctly measure each line. Some branches were only taken about 3% of the time; the hand held an estate only 9% of the time. But I think the rarest cases would be ones like having the draw pile be empty or not having any estates to draw. I'm actually not sure why it didn't say 100% of lines were executed; it's hard to read in Vim..

```
flip3 ~/362/assignment4/dominion 341% gcov randomtestcard1 -f -b
Function 'main'
Lines executed:100.00% of 5
Branches executed:100.00% of 2
Taken at least once:100.00% of 2
Calls executed:100.00% of 3

Function 'test'
Lines executed:93.24% of 74
```

```
Branches executed:100.00% of 38
Taken at least once:78.95% of 38
Calls executed:89.19% of 37

Function 'randInRange'
Lines executed:100.00% of 3
No branches
Calls executed:100.00% of 1

File 'randomtestcard1.c'
Lines executed:93.90% of 82
Branches executed:100.00% of 40
Taken at least once:80.00% of 40
Calls executed:90.24% of 41
Creating 'randomtestcard1.c.gcov'

flip3 ~/362/assignment4/dominion 342% gcov randomtestcard2 -f -b
Function 'main'
Lines executed:100.00% of 5
Branches executed:100.00% of 2
Taken at least once:100.00% of 2
Calls executed:100.00% of 3

Function 'test'
Lines executed:95.56% of 45
Branches executed:100.00% of 12
Taken at least once:83.33% of 12
Calls executed:95.45% of 22

Function 'randInRange'
Lines executed:100.00% of 3
No branches
Calls executed:100.00% of 1

File 'randomtestcard2.c'
Lines executed:96.23% of 53
Branches executed:100.00% of 14
Taken at least once:85.71% of 14
Calls executed:96.15% of 26
Creating 'randomtestcard2.c.gcov'

flip3 ~/362/assignment4/dominion 343% gcov randomtestcard3 -f -b
Function 'main'
Lines executed:100.00% of 5
Branches executed:100.00% of 2
Taken at least once:100.00% of 2
Calls executed:100.00% of 3

Function 'test'
Lines executed:95.56% of 45
Branches executed:100.00% of 12
Taken at least once:83.33% of 12
Calls executed:95.45% of 22

Function 'randInRange'
```

```
Lines executed:100.00% of 3
No branches
Calls executed:100.00% of 1

File 'randomtestcard3.c'
Lines executed:96.23% of 53
Branches executed:100.00% of 14
Taken at least once:85.71% of 14
Calls executed:96.15% of 26
Creating 'randomtestcard3.c.gcov'

flip3 ~/362/assignment4/dominion 344%
```

Unit vs. Random Testing

Unit testing seems better for simple systems, and random testing seems better for complex ones with lots of interacting parts. Random testing has better fault detection capacity if you can approach 100% coverage. You could also do a combination of the two. I didn't have time to finish assignment 3 so I can't compare my results here with those, but I expect it would be tougher to get 100% coverage with unit testing because we'd have had to account for things like the draw deck being empty.