

Jennifer Wolfe
CS 362 Final Report
June 11, 2012

Approach: I performed a random variable test 2000 times. I declared the necessary variables to perform the test on the buyCard:

```
p = floor(Random() * 2);  
s = floor(Random() * 16);  
G.whoseTurn = p;  
G.numBuys = floor(Random() * 5);  
G.coins = floor(Random() * 10);  
G.discardCount[p] = floor(Random() * MAX_DECK);  
checkBuyCard(s, &G);
```

I printed out the comparisons of before and after of what the card enum was, the number of buys, how many coins and the cost of the card to be bought. If a value was returned and not equal to 0, then I made the notation that nothing was bought. (Fortunately, I used the code if(r != 0) because one of the programmers used 999 instead of -1.)

Random testing was done on the Adventurer card as well. A variable was declared for treasureCardCount and a loop happened until there were 2 treasure cards or there were no more deck cards and discard cards left in their piles.

Test Statistics:

In reviewing my own code in comparison to others, I did not check to see if there were any cards left in the supplyCount. This was not included in my testing and will be there for the next time. When gcov was ran, 12.41% of the 556 lines of code is executed. I did have a seg fault when I first ran my testBuyCard since I had not initialized the discardCount. I did have problems with seg faults with my testAdv and was able to resolve them by debugging with gdb. Once that was resolved I had no problems running my test code. I did not have any problems compiling. Although I was able to get complete coverage with the

testBuyCard and the testDrawCard, I was not able to figure out why my testAdv file is not being read properly. It has the same problems being read like my testdom converting the 2 test.out files. So the gcov does not show any results in testing the adventurer case in dominion.c.

The coverage of my buyCard in dominion shows that out of 2000 random tests, almost 400 of them did not have any numBuys and did nothing. About half of the remaining test cases had enough coins to buy the card and the other half did not and didn't buy the card.

```
2000: 248: int buyCard(int supplyPos, struct gameState *state) {
-: 249: if(DEBUG){
-: 250:   printf("Entering buyCard...\n");
-: 251: }
-: 252:
2000: 253: if(state->numBuys > 0){
1612: 254:   if(state->coins > getCost(supplyPos)){
822: 255:     state->coins = state->coins - getCost(supplyPos);
822: 256:     state->numBuys--;
-: 257:   }
-: 258:   else{
790: 259:     state->numBuys--;
790: 260:     return -1;
-: 261:   }
-: 262: }
-: 263:
1210: 264: int who = state->whoseTurn;
1210: 265: state->discard[who][state->discardCount[who]] = supplyPos;
1210: 266: state->discardCount[who]++;
-: 267:
1210: 268: return 0;
-: 269: }
```

I realized that my test.out file somehow either splits into 2 different files and won't remove the information from the previous making of the file or just becomes a 'test.out?' file. I'm not sure how to correct the split files. I did modify the clean all statement in the Makefile to remove all test.* files to get rid of the file problems for each time the tests are ran.

Test Process and Bugs:

Dunhame – The code written already states a to do list of making sure the players turn ends if they are out of buys and also to decrement the money spent. The code compiles without any problems. When testing the buyCard, the numBuys doesn't usually decrease. Also, the coins are not being decreased after a card is bought, or potentially could be bought. (This confirms the comments made by the programmer.) The problem happens with putting the code:

```
"if(supplyCount(supplyPos, state)<1){  
    return -1;"
```

Almost half the time this is being used. When gcov was ran, 13.08% of the 520 lines of code is executed. This would mean almost all the buyCard and drawCard was being utilized.

Randb – The code compiles for the main code, but the testdom had an error with the drawCard. The assert r==0 failed. The programmer changed the code in dominion.c to return -1 or 1 unless it the deckCounter==0. So it would fail every time unless the deck was empty. The drawCard runs, but the numBuys does not decrement like it is supposed to. When gcov was ran, 4.95% of the 545 lines of code is executed.

Wandlins – The code does not have any new information for the buyCard in dominion.c. The adventurer card has some code, but I do not have testing for it this time. I am not going to analyze this programmers code any further at this time.

Olsojeff – The code compile and looked mostly good. There is a problem that if a card is not purchased, then the numBuys does not decrement. The numBuys should decrement if there is 1 available. When gcov was ran, 13% of the 523 lines of code is executed. This would mean almost all the buyCard and drawCard was being utilized.

Westbyb – In looking at the code, I can see a bug with the following code:

```
state->coins -= getCost(supplyPos);  
  
int who = state->whoseTurn;  
  
state->discard[who][state->discardCount[who]] = supplyPos;  
  
state->discardCount[who]++;  
  
(state->numBuys)--;  
  
return 0;
```

The coins are going to be decreased by the cost of the card, whether there is enough money to do so.

Also, there is no test as to whether there is enough numBuys to decrement or not. This is going to cause a problem and cause negative numbers in both the numBuys and the coins. No matter what, a card is bought. When gcov was ran, 12.36% of the 526 lines of code is executed. This would mean almost all the buyCard and drawCard was being utilized.

Parkan – The code is noisy, but otherwise I was unable to locate any problems with the buyCard. When gcov was ran, 16.10% of the 528 lines of code is executed. This would mean almost all the buyCard and drawCard was being utilized.

Milleand – Just by looking at their code, I can tell that the numBuys does not decrease. The code compiled, but running it confirmed by observation that numBuys is not decrementing. Also, the return value when invalid is supposed to be -1 and this programmer has a return value of 999. When gcov was ran, 12.74% of the 526 lines of code is executed. This would mean almost all the buyCard and drawCard was being utilized.

Alarkaz – no files found

I have submitted bug reports to all of these programmers (at least the ones that I ran code on and found bugs with) by uploading a text file to their folder in the bug reports folder.

For all them I was unable to get the adventurer card to work. They all showed segmentation faults and when looking in gdb, it showed the problem was located within the dominion.c code of their adventurer card.

Opinions for Future Testers:

If you don't know already, quickly learn Cygwin, how to do makefiles, gcov, gdb, and any other tool you may need to debug. It is much more difficult to get started if you have never done any of this before.

Once you learn it, it does get easier.

Some quick glances at code can seem like it will work, but it comes down to testing it out before you realized where the bug is. Automated testing makes it so much easier to get an idea if your code is even being used as well as finding what is being used wrong.

Sometimes your code has bugs you can't figure out how they happened and how to correct them to be able to proceed. For example, the test.out file(s) created by the testdom makefile has an error that can't be located which caused it to create 2 files or not read one of the files to be tested. Another example is the testAdv.c file, as previously mentioned, can't be read by the testdom makefile and therefor makes it difficult to debug or determine if the testing is even working.