Test Report 1

Kristen Bartosz

CS 362

Tuesday, May 1, 2012

Approach

I have implemented a little bit of random testing code for the buyCard function. Basically, testBuyCard creates random game states by providing variables with random numbers. This includes random values for the person buying the card, the supply pile being purchase from, the number of coins currently owned, the number of buys the player has available, the number of cards in the supply pile, and the number of cards in the discard pile. The Embargo tokens variable has just been set to 0 for now, since I know I haven't implemented anything with it. Currently, I have it set to run through 50 random states with this test.

```
  struct gameState pre;
  int r;
  memcpy (&pre, post, sizeof(struct gameState));


  r = buyCard (p, post);

  if((pre.numBuys >= 1) && (supplyCount(p, &pre) >=1) && (pre.coins >= getCost(p))){
         pre.supplyCount[p]--;

         pre.hand[ who ][ pre.handCount[who] ] = p;
      pre.handCount[who]++;

         pre.coins = (pre.coins) - (getCost(p));
         pre.numBuys--;
  }

  //assert (r == 0);
  if (r==0)
         printf("card was bought\n");

 // assert(memcmp(&pre, post, sizeof(struct gameState)) == 0);
  if(memcmp(&pre, post, sizeof(struct gameState)) != 0){
         printf("memcmp(&pre, post, sizeof(struct gameState)) == 0 FAILS, You must have a
bug\n\n");
  }
```

Above is the code that is in the checkBuyCard function. It creates a copy of the current state. It runs buy card on the current state, and then goes through the necessary supposed steps on the copy. Then it says whether the card was bought or not. Then it compares the game state of the copy to the current. If everything was done correctly, then the states of both games should be the same. They should be the same, because the only things that should have been changed should have been the same things done to both cases.

I developed this code from the testDrawCard code that we talked about in class. It was difficult because I'm still not entirely sure how everything in Dominion is supposed to work. I haven't tested anything about the phase issues between buy and action phases.

Test Statistics and Coverage

As for statistics and coverage, I don't know how to gather this sort of information. When I ran this test on my code, every time the player actually bought the card, the memcmp failed, meaning that the states of the copy and the current were different after the purchase. I am unsure if this is a fault in my dominion code or in my test code. 14/50 cases resulted in a purchase but also a fail in memory compare. I spent a lot of my time just trying to figure out svn commands to get my code up on beaversource. I am still not sure if I did it correctly. I couldn't figure out how to run the code from command line, and therefore I didn't know how to run this on other student's code. I tried to run it on junkerd's code through visual studio, but his code wasn't set up for visual studio and would have required a lot of time to format.

Bugs

As for bugs, like I said before, I discovered on my own code that when the copied gamestate and the current gamestate are compared after the test/buycard, they are not equal. This must be a bug in my buycard code. I'm unsure of what it exactly is, but in the future I plan to figure out what exactly is going wrong. Also, when I tried to run >>make testdom >>./testdom , a test.out was created, but it was blank. I also plan to figure out why this is happening as well.

Plans for the Future

First of all, what would I have done differently for this first test report? I would have gone to office hours to figure out what I was supposed to be doing really early. I would have figured out how to check in files to beaversource earlier. I would have gotten help to figure out what I should be testing. I basically would have gone to ask for help, because I feel like a lot of knowledge is implied, when in fact I don't know most of it.

For testing plans, I plan to fix the problems that this test has reported. I also plan to figure out how to test other people's code using command line and svn. I also hope to figure out a good way to discover coverage and statistics. I plan to just do everything better and to get an earlier start.