CS362: Test Report 1

Devlin Junker 931636867

May 1st, 2012

1 Test Approach

I have implemented a basic random test case for *buyCard*. The random test case generates a gameState struct with random values for a majority of the data. Most of the struct is generated completely randomly except for the *whoseTurn* variable. *whoseTurn* is randomized with an upper limit of the maximum players allowed, this is to stay inside of the gameState struct's array bounds and avoid segFaults. The random tests call the following function:

```
int checkBuyCard(int card, struct gameState* post){
struct gameState pre;
memcpy(&pre, post, sizeof(struct gameState));

int who = pre.whoseTurn;

int r = buyCard(card, post);

if(pre.supplyCount[card] > 0 && pre.coins > getCost(card) && pre.numBuys > 0){
pre.supplyCount[card]--;
pre.discardCount[who]++;
pre.coins -= getCost(card);
pre.numBuys--;
pre.phase = 1;
}else{
if(r != -1){
printf("Illegal Buy");
return -1;
}else{
return 0;
}
}
if(memcmp(&pre, post, sizeof(struct gameState)) != 0){
printf("Error");
return -1;
}

return 0;
}
```

This test will check that the expected actions in BuyCard will happen and nothing unexpected changed in the gameState. This is effective if the implementation of others code is similar to mine, however this may not work if someone else decides that they will do something in buycard that I decide should be in a different piece of the code (For Example: When to change the phase that the gameState is in).

I implemented because this type of testing because it is what we have been covering in class mostly so far. For the next report I would like to implement tests that will work more definitively for other implementations, rather than ones that are based off the same ways that I implemented buyCard.

2 Test Statistics

During initial testing I ran the test case that includes 2000 random tests of my own implementation of buyCard 3 different times. The initial two runs diagnosed two bugs. I have not had time to test other classmates code yet, although I plan on testing others by the next test report. I spent most of this time setting up the basic random testing which turned out a lot harder than I expected.

3 Bugs

The first bug that was caught by my test case was caused by placing the bought card directly into the players deck, this was caught when the discardCount was not incremented but the deckCount was. The second bug discovered was discovered due to segfaulting during testing, if the discardCount is greater than the maximum deck size, buyCard will try to read past the arrays limit.

4 Quality of Code

On initially inheriting the code, there were a lot of bugs and redundant pieces of code, after starting testing I have been able to clean up a few of the most obvious errors with the code.