Richard Wheeler

CS 362

Test Report #1

Approach Taken:

*testDrawCard:*

For testing the DrawCard function in class we decided in class to fill a gameState struct with randomly generated numbers so we could test out all the cases that could possibly take place. And then ran out test code a bunch of times. The test code itself takes the card we want to test and then the randomly filled gameState struct. We then make a duplicate of it and call the DrawCard method on the struct we pass in. We then do the steps the draw card method should be doing to the copy. After all these steps have been followed we check to see if the hands have been increased on both structs to make sure the card has been drawn.

int checkDrawCard(int p, struct gameState \*post) {

struct gameState pre;

memcpy (&pre, post, sizeof(struct gameState));

int r = drawCard (p, post);

if (pre.deckCount[p] > 0) {

pre.handCount[p]++;

pre.hand[p][pre.handCount[p]-1] = pre.deck[p][pre.deckCount[p]-1];

pre.deckCount[p]--;

} else if (pre.discardCount[p] > 0) {

memcpy(pre.deck[p], post->deck[p], sizeof(int) \* pre.discardCount[p]);

memcpy(pre.discard[p], post->discard[p], sizeof(int)\*pre.discardCount[p]);

pre.hand[p][post->handCount[p]-1] = post->hand[p][post->handCount[p]-1];

pre.handCount[p]++;

pre.deckCount[p] = pre.discardCount[p]-1;

pre.discardCount[p] = 0;

}

assert (r == 0);

assert(memcmp(&pre, post, sizeof(struct gameState)) == 0);

}

*testBuyCard:*

For testing the BuyCard function I decided the best method would just be to make a new gameState struct and then set some arbitrary gold values and then check to make sure the BuyCard command worked correctly. The command randomly buys a different card every time and checks to make sure that we actually bought a card by checking to see if the BuyCard command returns a 0, if it doesn’t we assume that we weren’t about to purchase a card.

int checkBuyCard(int p, struct gameState \*post)

{

int randomCard = floor(Random() \* 10);

printf("%d\n", randomCard);

int x = buyCard(randomCard, post);

assert(x == 0)

}

....  
initializeGame(4, k, 4, &G);  
G.coins = 10;  
checkBuyCard(4, &G);  
G.coins = 0;  
checkBuyCard(5, &G);  
G.coins = 5;  
checkBuyCard(silver, &G);  
checkBuyCard(silver, &G);  
....

*Opinions for future testing:*

For future Testing I would want to buff out the BuyCard testing as well as write some testing for all the other functions in the program just to make sure they are all working correctly. Also possibly add some more testing to the DrawCard function just for the sake of adding more. I’d like to focus more on random testing because that seems like a good way of just flat out brute forcing your code to make sure it works correctly.