Chido Nguyen

CS362 Assignment 5

Nguychid / 931506965

Lucky for me, on the first run there were a page of errors which turned out was an issue when I was transferring files the content became corrupted. After re-doing the files everything compiled and ran without a hitch on the first (second run). I did not have to edit my unittest_suite at all. From the gcov output, my group mate refactored the cards into functions called:

- runEmbargo
- runGreat_hall
- runVillage
- runAdventurer
- runSmithy

This makes me think that my test codes are moderately maintainable. My card tests invoke dominion's built-in enumeration of card type which by design calls the appropriate function refactored by my group-mate. Therefor I do not have to account for the differences in function names between my code and his code. As for the unit test themselves, they were built from the function prototypes present in the dominion code. Unless my group mate went out of his way to break those, there was no reason for my code not to work. See unittestresults.out in assignment-5 folder submission.

Bug-Report:

ALEXJ_001_CARD_SMITHY

- Description: Card effects are not implemented correctly, 3 cards are not drawn.
- Possible Function in Question: runSmithy(currentPlayer,handPosition, gameState)
- Test Suite Report: The basic understanding behind this test is that we are expecting 3 cards to be drawn. Due to the nature of how hands are played, you are suppose to get +2 card yield. Discarding smithy then drawing 3 , one of the 3 fills the slot of old smithy hence the net gain of 2. The test function checks for this and prints it out (pre_edit_unittestresults.out). The results show that we have 5 cards in hand, and expect 7 afterwards but end up with 8.

```
TESTING SMITHY +3 CARDS(NET GAIN OF 2 EXPECTED):

CURRENT HAND COUNT:                5

EXPECTED HAND COUNT AFTER DRAW:                7

AFTER DRAWING HAND COUNT:                8

FAILED
```

ALEXJ_002_CARD_VILLAGE:

- Description: Card effects are not implemented correctly for increasing action count.
- Function: runVillage();
- TSR: The test results printed out that action increase was not in the expected range. The before card effect state shows one, and the after state shows 4. If we read the description of the card, we're expecting +2 or 3, but it seems that the code adds more than what is expected.
- Side note: I found out my test suit code was also semi-broken here. The initial check for cards in hand accounts for the +1 but not for the card that needs to be discarded. Net gain of 0. Which now that I think about it would be difficult to test to see whether or not an actual card is drawn if we only check for 0. Something to fix down the line.
- TESTING VILLAGE CARD (+2 ACTION, +1 CARD) :
- DRAWING A CARD: HAND BEFORE: 5   HAND AFTER: 5   FAIL
- ACTION +2 BEFORE: 1   HAND AFTER: 4   FAIL
- DISCARD +1 BEFORE: 0   HAND AFTER: 0   FAIL

Test-Report:

My experience testing dominion has been mixed. Personally, I thought the code was super messy originally. The formatting was very disorganized with bad tabbing and organization, I had to redo all the indentation to make code tracing a lot easier. It got progressively easier to under stand the code as I got more use to the game/program. For instance, some of the functions or card we needed to test had no prototype listed in the header files. This made it difficult to test initially; mainly because I wasn't sure when a function would have a prototype or not and would always have to look it up (if it was a new function I haven't worked with before). I originally started out making instances of the game state and manually filling in starting values then running the card effect to see how it changed. Down the line learned that there were functions built in to initialize everything then we can call card effect and compare for expected values.

Back onto the assignment at hand. I think the reliability of my partner's code is not reliable at all. This isn't to say that he is a bad coder but I don't have enough evidence to back my statement if I said I were to have full confidence.:

```
Lines executed:39.04% of 584

Branches executed:45.54% of 415

Taken at least once:36.39% of 415

Calls executed:24.47% of 94
```

From the gcov results my partner's code gets decent coverage 40%-line coverage, 46% branch coverage I would say things get ran quite a bit. Except my "tests" don't actually verify everything. Of the 15(?) or how many cards built in we test 4-5 at most, and then of the 10 or more function we examine 4 of them. When we do examine them, we are also looking at them in a very narrow point of view. How does this card look when its ran, how does this function look when its ran? It was not until the random testing that we start testing how the system operates rather than the independent function. Ultimately, I am comfortable with how my partner coded, but I do not feel that my test suite may be robust enough to say that dominion was tested properly. However, I am comfortable saying that at assignment-5 status the game is still broken, and probably was still broken at the start of the term.

Debugging:

ALEXJ_001_CARD_SMITHY:

When we set breakpoints for runSmithy the first line that prints out after the function signature is a for loop which we can identify the problem right away.

```
Breakpoint 1, runSmithy (currentPlayer=0, handPos=0, state=0x60f010) at
dominion.c:1308
1308        for (int i = 0; i < 4; i++)
```

The for loop is set to run till I is equal to or greater than 4, but since I starts at 0 we get 4 run thought's, 0 1 2 3 4. If we swap 4 out for 3 we should be able to achieve passing results on our test runs.

ALEXJ_002_CARD_VILLAGE:

We set breakpoint for runVillage

I step through the function and notice that draw card is called then we reach our action modifiers to trace it I printed out the action value pre and post effect. From the code printout I can already see that the +3 is wrong and it should be +2. But to confirm this is what I see:

```
runVillage (currentPlayer=0, handPos=0, state=0x60f010) at dominion.c:1349
1349        state->numActions = state->numActions + 3;
(gdb) print state->numActions
$1 = 1
(gdb) step
1352        discardCard(handPos, currentPlayer, state, 0);
(gdb) print state->numActions
$2 = 4
```

If we check back with our test results we now know how the 1 and 4 results came about. To fix we edit +3 to +2. While I'm at it I also fixed my cardtest function to account for the net gain rather than the action gain of the card.

Post_edit_unittestresults.out:

TESTING SMITHY +3 CARDS(NET GAIN OF 2 EXPECTED):

```
CURRENT HAND COUNT:            5

EXPECTED HAND COUNT AFTER DRAW:              7

AFTER DRAWING HAND COUNT:             7

PASS
```

```
TESTING VILLAGE CARD (+2 ACTION, +1 CARD) :

DRAWING A CARD: HAND BEFORE: 5   HAND AFTER: 5            PASS

ACTION +2 BEFORE: 1      HAND AFTER: 3               PASS
```