

## Bug-Reports

### BUG Report 1 of 2

=====

Title: Smithy Card Effect - Incorrect values for handCount and deckCount

Class: Minor Bug

Date: 2/27/2018

Reported By: Deirdre Moran

Email: morand@oregonstate.edu

Product: dominion.c Version: Assignment-2

URL: <https://github.com/deirdreemoran/CS362-004-W2018/tree/master/projects/morand/elliskenDominion>

Is it reproducible: Yes

#### Description

=====

Ran cardTest2 for Smithy Card and assertions for handCount and deckCount failed test cases for all values. I ran cardtest2.c, checked the gcov files for test results and noticed that the smithyCard failed these two tests. The first one tests the handCount of the player after a smithy Card is played. The handCount of the current player should increase by 2, but instead, the handcount increased by 3. Error message is "\*\*\*\*Did not pass assert test in file cardtest2.c' line 57.\*\*\*\*". The second failure occurred when testing whether the three drawn cards came from the player's deck. There are two problems here - the expected result should be 3, but is not, and the deckCount of the current player is only 1, when it should be 3. The error message reads "\*\*\*\*Did not pass assert test in file 'cardtest2.c' line 63.\*\*\*\*".

#### Steps to Produce/Reproduce

-----

From within the dominion folder, run "make all". Then open "unittestresults.out" file and locate smithy test results. The assertion failures will be listed there.

#### Expected Results

-----

I expected to see matching deck and hand Counts with expected results. The handCount of the current player should increase by 2, and the deckCount of the three drawn cards should come from the player's own deck.

#### Actual Results

-----

*TEST 1: handCount of current player increases by 2  
handCount is 8, expected 7*

Deirdre Moran  
CS362: Assignment 5

\*\*\* Did not pass assert test in file 'cardtest2.c' line 57.\*\*\*

TEST 2: the three drawn cards came from player's deck  
deckCount is 1, expected 2

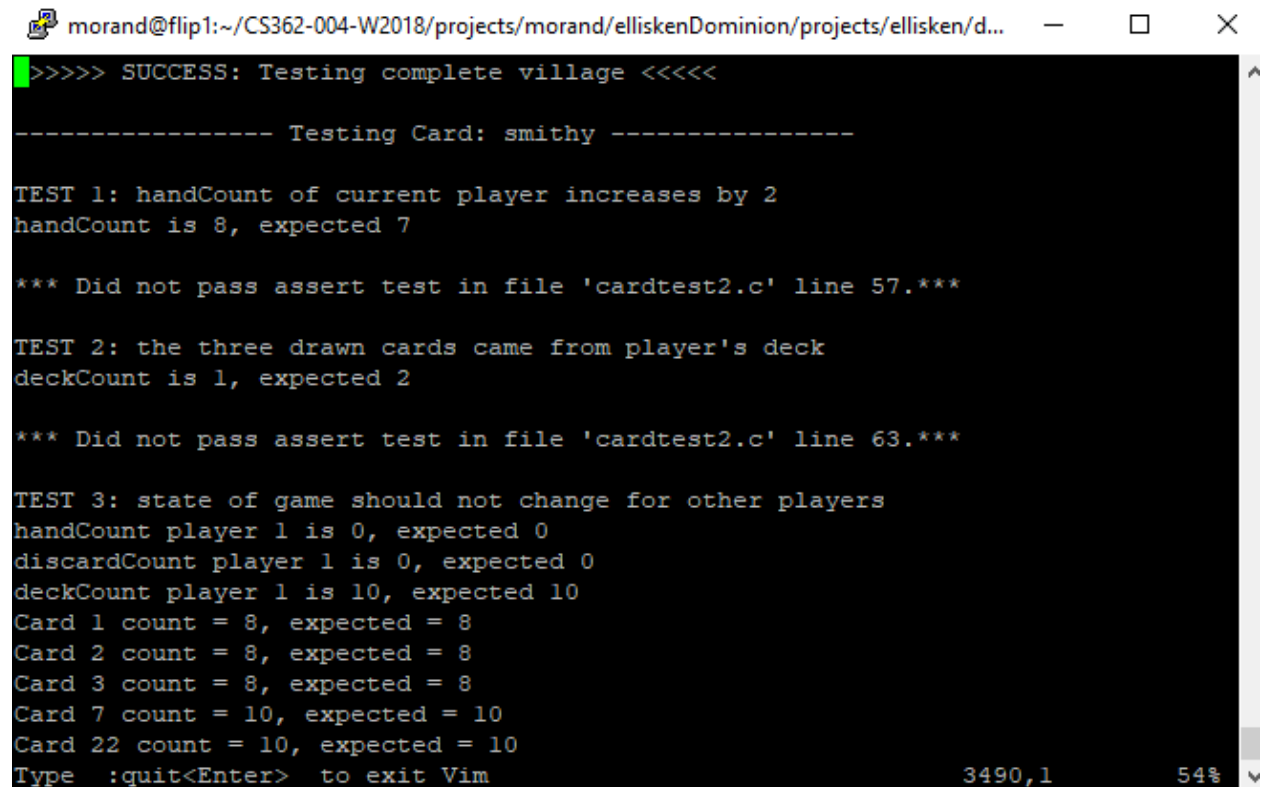
\*\*\* Did not pass assert test in file 'cardtest2.c' line 63.\*\*\*

Workarounds

-----  
In the 'cardSmithy' function of dominion.c, the for loop should be for values less than 3 instead of less than or equal to 3. Too many cards are being drawn, which is throwing off the deck and hand counts.

Attachments

-----



```
>>>> SUCCESS: Testing complete village <<<<

----- Testing Card: smithy -----

TEST 1: handCount of current player increases by 2
handCount is 8, expected 7

*** Did not pass assert test in file 'cardtest2.c' line 57.***

TEST 2: the three drawn cards came from player's deck
deckCount is 1, expected 2

*** Did not pass assert test in file 'cardtest2.c' line 63.***

TEST 3: state of game should not change for other players
handCount player 1 is 0, expected 0
discardCount player 1 is 0, expected 0
deckCount player 1 is 10, expected 10
Card 1 count = 8, expected = 8
Card 2 count = 8, expected = 8
Card 3 count = 8, expected = 8
Card 7 count = 10, expected = 10
Card 22 count = 10, expected = 10
Type :quit<Enter> to exit Vim                                     3490,1      54%
```

```
int cardSmithy(int currentPlayer, struct gameState *state, int handPos)
{
    int i;

    //+3 Cards
    for (i = 0; i <= 3; i++)
    {
        drawCard(currentPlayer, state);
    }

    //discard card from hand
    discardCard(handPos, currentPlayer, state, 0);
    return 0;
}
```

## BUG Report 2 of 2

---

Title: Adventurer Card Effect - Incorrect values for handCount and extra cards

Class: Minor Bug

Date: 2/27/2018

Reported By: Deirdre Moran

Email: morand@oregonstate.edu

Product: dominion.c Version: Assignment-2

URL: <https://github.com/deirdreemoran/CS362-004-W2018/tree/master/projects/morand/elliskenDominion>

Is it reproducible: Yes

### Description

=====

Ran cardTest4 for Adventurer Card and assertions for handCount and extra card failed test cases for all values. I ran cardtest4.c, checked the gcov files for test results and noticed that the AdventurerCard failed these two tests. The first one tests if the two treasures are removed from the deck and placed into the hand. The handCount of the current player should increase by 2, but instead, the handcount increased by is a negative value. Error message is "\*\*\*\*Did not pass assert test in file cardtest4.c' line 53.\*\*\*\*". The second failure occurred when testing whether the 2 extra cards came from the discard or deck piles. The expected value is 2, but the test returned 5. Three extra cards are being added. The error message reads "\*\*\*\*Did not pass assert test in file 'cardtest4.c' line 62.\*\*\*\*".

#### Steps to Produce/Reproduce

-----  
From within the dominion folder, run "make all". Then open "unittestresults.out" file and locate adventurer test results. The assertion failures will be listed there.

#### Expected Results

-----  
I expected to see a matching deck count and 2 extra cards from either the discard or deck piles.

#### Actual Results

----- Testing Card: adventurer -----  
  
*TEST 1: two treasures are removed from deck and placed in hand  
handCount is -79, expected 7*  
  
*\*\*\* Did not pass assert test in file 'cardtest4.c' line 53.\*\*\**  
  
*TEST 2: the 2 extra cards should come from discard or deck  
Difference is 5, expected 2*  
  
*\*\*\* Did not pass assert test in file 'cardtest4.c' line 62.\*\*\**

#### Workarounds

-----  
In the 'cardAdventurer' function of dominion.c, in the while loop for drawn treasure < 2 (see attachment below), the if statement for if(cardDrawn == gold) only increases the treasure count for gold coins, allowing the player to repeatedly draw coins that are silver and copper. The player could essentially draw as many copper and silver coins as they want, so long as gold coins are not drawn more than twice. This allows the player to loop through the while loop for drawntreasure many more times than just twice. To fix this, the if statement should be changed to include copper and silver coins, as such:

```
if (cardDrawn == gold || cardDrawn == silver || cardDrawn == copper)
```

#### Attachments

```
morand@flip1:~/CS362-004-W2018/projects/morand/elliskenDominion/projects/ellisken/d...  
----- Testing Card: adventurer -----  
TEST 1: two treasures are removed from deck and placed in hand  
handCount is -79, expected 7  
  
*** Did not pass assert test in file 'cardtest4.c' line 53.***  
  
TEST 2: the 2 extra cards should come from discard or deck  
Difference is 5, expected 2  
  
*** Did not pass assert test in file 'cardtest4.c' line 62.***  
  
TEST 3: 2 drawn cards should be treasure cards  
treasure card is 6, expected 4 or 5 or 6  
treasure card is 5, expected 4 or 5 or 6  
  
TEST 4: state of game should not change for other players  
handCount player 1 is 0, expected 0  
discardCount player 1 is 0, expected 0  
deckCount player 1 is 10, expected 10  
Card 1 count = 8, expected = 8  
Card 2 count = 8, expected = 8  
3556,0-1 55%
```

```
int cardAdventurer(int drawntreasure, struct gameState *state, int currentPlayer, int cardDrawn, int temphand[], int z)
{
    while(drawntreasure<2){
        if (state->deckCount[currentPlayer] <1){//if the deck is empty we need to shuffle discard and add to deck
            shuffle(currentPlayer, state);
        }
        drawCard(currentPlayer, state);
        cardDrawn = state->hand[currentPlayer][state->handCount[currentPlayer]-1];//top card of hand is most recently drawn card.
        if (cardDrawn == gold)
            drawntreasure++;
        else{
            temphand[z]=cardDrawn;
            state->handCount[currentPlayer]--; //this should just remove the top card (the most recently drawn one).
            //z++;
        }
    }
    while(z-1>=0){
        state->discard[currentPlayer][state->discardCount[currentPlayer]++]=temphand[z-1]; // discard all cards in play that have b
        z=z-1;
    }
    return 0;
}
```

## Debugging

I chose to debug the bug observed in unit and random testing for the *council\_room* card. I first started by running "gdb cardtest3" in my command line. This is a GNU debugger which allows one to step through code and specify variables to watch and functions to pause on. I chose to insert breakpoints for the *drawCard()* and *discardCard()* functions, as well as watch variables *state->handCount[player]* and *state->deckCount[player]*. The combination of these variables and watch points allowed me to step through my code and identify the environment where discrepancies of assertions occurred. The first thing I noticed was that *drawCard()* was being executed 9 times, instead 4, and all for the current player. Ideally, the *council\_room* card should have the current player draw 4 cards, and then each other player draws a card as well.

To check the number of players, I added another variable to watch, *state->numPlayers*, while running gdb. Because this variable was local to the *council\_roomEffect()* function, I had to continually enter 'watch state->numPlayers' during the gdb run. The number of players remained 2 throughout the execution, so this turned out to not be the source of the problem. The bug had something to do with the other players not being allowed to draw cards. Examining the *council\_roomEffect* function further, I identified the bug in the second for loop, where instead of allowing any player but the *currentPlayer* to draw a card, the if statement instead allows only the *currentPlayer* to draw a card and repeats this for the number of players in the game. Below is the bug report for the *council\_roomEffect()* function.

Title: council\_roomEffect - Incorrect values for handCount and deckCount

Class: Minor Bug

Date: 2/27/2018

Reported By: Deirdre Moran

Email: morand@oregonstate.edu

Product: dominion.c Version: Assignment-2 (buggy)

URL: <https://github.com/deirdreemoran/CS362-004-W2018/tree/master/projects/morand/dominion>

Is it reproducible: Yes

Description

=====

Ran cardTest3 for council\_roomEffect card. Assertions for handCount and deckCount failed test cases for all values. I ran cardtest3.c with a debugger (gdb) and identified that the currentPlayer was invoking the drawCard function too many times (it should only be 4). Error message is "\*\*\*\*Did not pass assert test in file cardtest3.c' line 52.\*\*\*\*".

Steps to Produce/Reproduce

-----

From within the dominion folder, run "make all". Then run "gdb cardtest3". Enter "break council\_roomEffect", enter "break drawCard". Enter "run". Watch the function calls and enter "continue" to progress through the program. The assertion failures will be listed towards the end.

Deirdre Moran  
CS362: Assignment 5

Expected Results  
-----

I expected to see matching deck and hand Counts with expected results. The handCount of the current player should increase by 3, and the deckCount should decrease by 4.

Actual Results  
-----

```
----- Testing Card: council_room -----  
TEST 1: Handcount is 3 (4 cards drawn, 1 discarded)  
handCount is 9, expected 8  
  
*** Did not pass assert test in file 'cardtest3.c' line 52.***  
  
TEST 2: Deckcount decreases by 4  
deckCount is 0, expected 1  
  
*** Did not pass assert test in file 'cardtest3.c' line 58.***
```

Workarounds  
-----

In the council\_roomEffect() function of dominion.c, in the second for loop where each other player draws a card, by changing the if statement "if(I == currentPlayer)" to read instead "if(i != currentPlayer)", this rectifies assertion errors. This simple change excludes the currentPlayer from drawing cards in this loop and instead allows each other player to draw a card, as is correct to the game of Dominion rules. Listed below in attachments is the workaround code.

Attachments  
-----

Workaround code for dominion.c council\_roomEffect function (next page)

Deirdre Moran  
CS362: Assignment 5

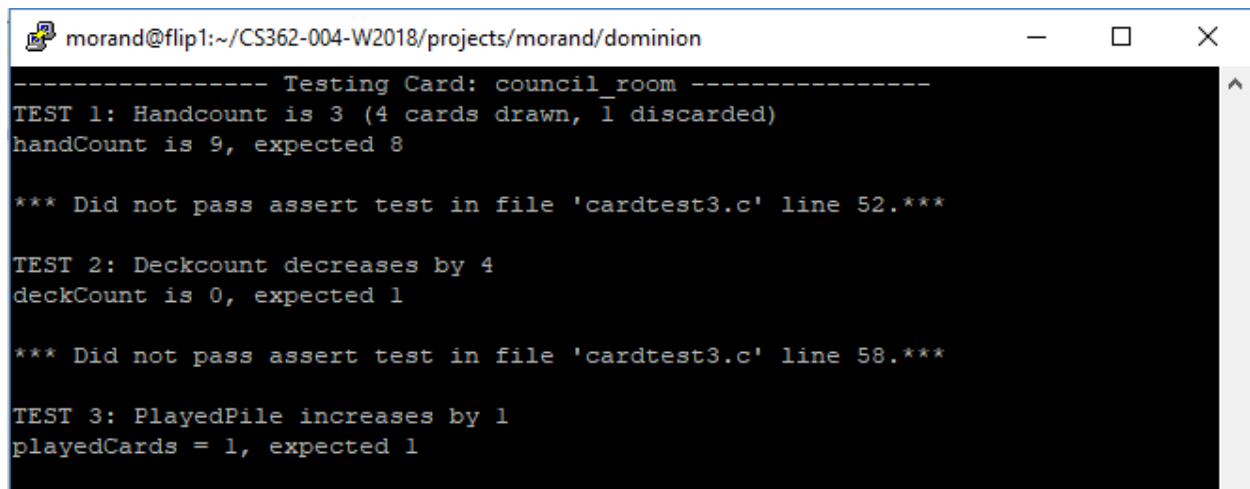
```
int council_roomEffect(int currentPlayer, struct gameState * state, int handPos)
{
    int i = 0;
    //+4 Cards
    for (i = 0; i < 4; i++)
    {
        drawCard(currentPlayer, state);
    }

    //+1 Buy
    state->numBuys++;

    //Each other player draws a card
    for (i = 0; i < state->numPlayers; i++)
    {
        if ( i != currentPlayer ) // FIXED BUG: changed i == to read i !=
        {
            drawCard(i, state);
        }
    }

    //put played card in played card pile
    discardCard(handPos, currentPlayer, state, 0);
    return 0;
}
```

Initial assertion failures



The screenshot shows a terminal window with the title bar "morand@flip1:~/CS362-004-W2018/projects/morand/dominion". The terminal output displays three test cases, each failing an assertion. The first test, "TEST 1: Handcount is 3 (4 cards drawn, 1 discarded)", fails because the handCount is 9 instead of the expected 8. The second test, "TEST 2: Deckcount decreases by 4", fails because the deckCount is 0 instead of the expected 1. The third test, "TEST 3: PlayedPile increases by 1", passes as the playedCards value is 1, which matches the expected value. Each failure is preceded by the message "\*\*\* Did not pass assert test in file 'cardtest3.c' line 52.\*\*\*" (for the first two) and line 58 (for the third).

```
----- Testing Card: council_room -----
TEST 1: Handcount is 3 (4 cards drawn, 1 discarded)
handCount is 9, expected 8

*** Did not pass assert test in file 'cardtest3.c' line 52.***

TEST 2: Deckcount decreases by 4
deckCount is 0, expected 1

*** Did not pass assert test in file 'cardtest3.c' line 58.***

TEST 3: PlayedPile increases by 1
playedCards = 1, expected 1
```



## Test-Report

Testing for the *cardAdventurer* function in my teammates code went relatively smoothly. My unit test assertions did not need to be reworked, as my teammate implemented the introduction of bugs according to class guidelines. I first created a separate folder to store my teammate's code and then added my *Makefile* and all unit, random, and card tests to it as well. Then I ran *make all* and *make unittestresults.out*. I examined my *unittestsresults.out* for the smithy and adventurer bugs specifically, as those two cards were guaranteed to be in my test suite as was required for past assignments.

For the *cardAdventurer* unit test, my coverage results came back as 85.71% for 14 lines, and 92% of the blocks were executed. Upon further examination of the *cardAdventurer* function itself, it became clearer as to why there was such a large negative value for the current player's handcount after playing the *cardAdventurer* card. The error here was due to the repeated decrementing of the player's hand in the else statement of the first while loop. This error is connected to the if statement that increments the drawn treasure counter, which in turn, determines how many times the out while loop is executed. Because this value is so large, it was easier to locate the problem in the handCount decrementing. If the value had not been negative, the decrementing error would have been less visible.

Testing or the *cardSmithy* function in my teammates code also went smoothly due to her correct implementation of introducing bugs into *dominion.c*. Upon examining my *unittestsresults.out* file, my coverage results for this function came back as 100% for 5 lines and 100% of the blocks were executed. After analyzing the *cardSmithy* function itself, it became clear that the for loop was being executed too many times due to the incorrect test statement operator, namely i should be less than 3, and not less than or equal to 3, as this cause the loop to execute one extra time. Looping an extra time results in the player being able to draw an extra card, which is what ultimately throws off the hand and deck counts.

While these two bugs are present in my teammate's code, they are easily resolvable and do not cause the entire program to produce a segmentation fault. With a couple fixes, as mentioned above, my teammate's code would be quite functional. The organization, modularization, and commenting in her code helped me to identify errors more easily and find solutions.