Assignment 5

liam beckman

March 16, 2019

Contents

1 Bugs Report

1.1 Bug 1

- Title: Wrong Smithy draw card value.
- Project: dominion
- Reporter: liam beckman
- Date: 16 March 2019
- Type: Bug
- File Name: cardtest1.c
- Environment: NVIM v0.3.4 (Build type: Release)
- Description:

The Smithy card (implemented in the `play_smithy` function) drew a totoal of four cards, instead of the correct amount of three cards. This is likely due to the incorrect termination expression of `i < 4` in the primary for loop.

```c
int play_smithy(int currentPlayer, int handPos, struct gameState *state)
{
        //+3 Cards
      for (int i = 0; i < 4; i++)
    {
      drawCard(currentPlayer, state);
    }

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

1.2 Bug 2

- Title: Adventurer Card Error
- Project: dominion
- Reporter: liam beckman
- Date: 16 March 2019
- Type: Bug
- File Name: cardtest2.c
- Environment: NVIM v0.3.4 (Build type: Release)
- Description:

  Uninitialized variable `z` in `play_adventurer` function within the `temphand[z]=cardDrawn` statement. Furthermore, the `z` variable is not incremented after the `temphand[z]=cardDrawn` statement, so the first temphand card will continualy be overwritten every iteration of the `while(drawntreasure<2)` statement.

```
while(drawntreasure<2)

//...

    if (cardDrawn == copper || cardDrawn == silver || cardDrawn == gold)
        drawntreasure++;
    else
    {
        temphand[z]=cardDrawn;
        state->
    handCount[currentPlayer]--; //this should just remove the top card (the most
    ↪   recently drawn one).
    }
```

2 Test Report

I chose to use Yan Ming's branch for their second assignment (yanme-assignment-2), as her later branches were such that no prominent bugs could be found. Thus, I am confident in saying that their code is reliable and well executed, as I was forced to analyze code with purposeful bugs implemented.

2.1 Manual Review

2.1.1 cardEffect

Missing end bracket in `cardEffect` function.

```
int cardEffect(int card, int choice1, int choice2, int choice3, struct gameState *state, int
↪   handPos, int *bonus)

// ...
```

Variable misspelling of `drawnteasure` variable in `play_adventurer` function. Initialized as `drawnteasure`.

2.1.2 play_adventurer

```
int play_adventurer(int currentPlayer, int temphand[MAX_HAND],  struct gameState *state)


int drawnteasure = 0;
int cardDrawn;
while(drawntreasure<2)

    // ...
```

Uninitialized variable `z` in `play_adventurer` function within the `temphand[z]=cardDrawn` statement. Furthermore, the `z` variable is not incremented after the `temphand[z]=cardDrawn` statement, so the first temphand card will continualy be overwritten every iteration of the `while(drawntreasure<2)` statement.

```
while(drawntreasure<2)

//...

    if (cardDrawn == copper || cardDrawn == silver || cardDrawn == gold)
        drawntreasure++;
    else
    {
        temphand[z]=cardDrawn;
        state->
    handCount[currentPlayer]--; //this should just remove the top card (the most recently
    ↪  drawn one).
    }
```

2.1.3 play_smithy

Uninitialized variable `i` in `play_smithy` function.

```
int play_smithy(int currentPlayer, int handPos, struct gameState *state)

//+3 Cards
    for (i = 0; i < 4; i++)

        // ...
```

2.1.4 play_mine

Uninitialized variables `j` and `i` in `play_mine` function.

```
int play_mine ( int currentPlayer, int choice1, int choice2, struct gameState *state)

```

```c
    j = state->
    hand[currentPlayer][choice1];  //store card we will trash

// ...

//discard trashed card
for (i = 0; i < state->
    handCount[currentPlayer]; i++)
{
    if (state->
    hand[currentPlayer][i] == j)
    {
        discardCard(i, currentPlayer, state, 0);
        break;
    }
}

return 0;
```

Undeclared variable `handPos` in `play_mine` function.

```c
int play_mine ( int currentPlayer, int choice1, int choice2, struct gameState *state)

// ...

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

2.1.5 play_council

Uninitialized variable `i` in `play_council` function.

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

        // ...
```

2.2 Unit Testing

2.2.1 Test Results

1. getCost Function

```
1  unittest1.c:17 TEST SUCCESSFULLY COMPLETED ->
2       getCost(0) == 0
3  unittest1.c:20 TEST SUCCESSFULLY COMPLETED ->
```

```
     getCost(1) == 2
unittest1.c:23 TEST SUCCESSFULLY COMPLETED ->
     getCost(2) == 5
unittest1.c:26 TEST SUCCESSFULLY COMPLETED ->
     getCost(3) == 8
unittest1.c:29 TEST SUCCESSFULLY COMPLETED ->
     getCost(4) == 0
```

2. isGameOver Function

```
unittest2.c:28 TEST SUCCESSFULLY COMPLETED ->
     isGameOver(&testGame) == 0
unittest2.c:35 TEST SUCCESSFULLY COMPLETED ->
     isGameOver(&testGame) == 1
unittest2.c:43 TEST SUCCESSFULLY COMPLETED ->
     isGameOver(&testGame) == 0
unittest2.c:46 TEST SUCCESSFULLY COMPLETED ->
     isGameOver(&testGame) == 1
```

3. compare Function

```
unittest3.c:14 TEST SUCCESSFULLY COMPLETED ->
     compare(b, a) == 1
unittest3.c:15 TEST SUCCESSFULLY COMPLETED ->
     compare(a, b) == -1
unittest3.c:16 TEST SUCCESSFULLY COMPLETED ->
     compare(a, a) == 0
unittest3.c:17 TEST SUCCESSFULLY COMPLETED ->
     compare(b, b) == 0
```

4. updateCoins Function

```
unittest4.c:31 TEST SUCCESSFULLY COMPLETED ->
     testGame->coins == 9
unittest4.c:43 TEST SUCCESSFULLY COMPLETED ->
     testGame->coins == 8
```

5. Smithy Card

   (a) Failed Tests

```
cardtest1.c:14 TEST SUCCESSFULLY COMPLETED ->
     play_smithy(player, handPos, testGame) == 0
cardtest1.c:15 TEST FAILED: ->
     testGame->deckCount[player] == 2
cardtest1.c:16 TEST FAILED: ->
     testGame->handCount[player] == 7
cardtest1.c:19 TEST SUCCESSFULLY COMPLETED ->
     testGame->deckCount[player] == 0
cardtest1.c:20 TEST SUCCESSFULLY COMPLETED ->
     testGame->handCount[player] == 8
```

   (b) Successful Tests

```
cardtest1.c:14 TEST SUCCESSFULLY COMPLETED ->
     play_smithy(player, handPos, testGame) == 0
cardtest1.c:15 TEST SUCCESSFULLY COMPLETED ->
     testGame->deckCount[player] == 2
cardtest1.c:16 TEST SUCCESSFULLY COMPLETED ->
     testGame->handCount[player] == 7
deck count: 2
hand count: 7
cardtest1.c:21 TEST SUCCESSFULLY COMPLETED ->
     testGame->deckCount[player] == 0
cardtest1.c:22 TEST SUCCESSFULLY COMPLETED ->
     testGame->handCount[player] == 8
File 'cardtest1.c'
Lines executed:100.00% of 14
Branches executed:100.00% of 10
Taken at least once:50.00% of 10
Calls executed:66.67% of 15
Creating 'cardtest1.c.gcov'
```

(c) Code Changed

```c
int play_smithy(int currentPlayer, int handPos, struct gameState *state)
{
        //+3 Cards
        // changed "i < 4" to "i < 3"
      for (int i = 0; i < 3; i++)
    {
      drawCard(currentPlayer, state);
    }

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

6. Adventurer Card

```
cardtest2.c:19 TEST SUCCESSFULLY COMPLETED ->
     play_adventurer(player, temphand, testGame) == 0
cardtest2.c:22 TEST SUCCESSFULLY COMPLETED ->
     lastCard == copper || lastCard == silver || lastCard == gold
```

7. Village Card

```
cardtest3.c:31 TEST SUCCESSFULLY COMPLETED ->
     cardEffect(card, choice1, choice2, choice3, testGame, handPos, bonus) == 0
cardtest3.c:35 TEST SUCCESSFULLY COMPLETED ->
     actions == actionsOld + 2
```

8. Mine Card

```
cardtest4.c:21 TEST SUCCESSFULLY COMPLETED ->
     play_mine(player, choice1, choice2, testGame, handPos) == 0
```

2.2.2 Code Coverage

1. getCost Function

```
File 'unittest1.c'
Lines executed:100.00% of 6
Branches executed:100.00% of 10
Taken at least once:50.00% of 10
Calls executed:66.67% of 15
Creating 'unittest1.c.gcov'

File 'dominion.c'
Lines executed:2.00% of 599
Branches executed:6.85% of 409
Taken at least once:1.22% of 409
Calls executed:0.00% of 92
Creating 'dominion.c.gcov'
```

2. isGameOver Function

```
File 'unittest2.c'
Lines executed:100.00% of 17
Branches executed:100.00% of 12
Taken at least once:66.67% of 12
Calls executed:71.43% of 14
Creating 'unittest2.c.gcov'

File 'dominion.c'
Lines executed:1.67% of 599
Branches executed:1.96% of 409
Taken at least once:1.96% of 409
Calls executed:0.00% of 92
Creating 'dominion.c.gcov'
```

3. compare Function

```
File 'unittest3.c'
Lines executed:100.00% of 9
Branches executed:100.00% of 8
Taken at least once:50.00% of 8
Calls executed:66.67% of 12
Creating 'unittest3.c.gcov'

File 'dominion.c'
Lines executed:1.00% of 599
Branches executed:0.98% of 409
Taken at least once:0.98% of 409
Calls executed:0.00% of 92
Creating 'dominion.c.gcov'
```

4. updateCoins Function

```
File 'unittest4.c'
```

```
Lines executed:100.00% of 25
Branches executed:100.00% of 4
Taken at least once:50.00% of 4
Calls executed:77.78% of 9
Creating 'unittest4.c.gcov'

File 'dominion.c'
Lines executed:2.34% of 599
Branches executed:1.96% of 409
Taken at least once:1.71% of 409
Calls executed:0.00% of 92
Creating 'dominion.c.gcov'
```

5. Smithy Card

```
File 'cardtest1.c'
Lines executed:100.00% of 12
Branches executed:100.00% of 10
Taken at least once:50.00% of 10
Calls executed:61.54% of 13
Creating 'cardtest1.c.gcov'
```

6. Adventurer Card

```
File 'cardtest2.c'
Lines executed:100.00% of 13
Branches executed:50.00% of 8
Taken at least once:25.00% of 8
Calls executed:71.43% of 7
Creating 'cardtest2.c.gcov'
```

7. Village Card

```
File 'cardtest3.c'
Lines executed:100.00% of 19
Branches executed:100.00% of 4
Taken at least once:50.00% of 4
Calls executed:71.43% of 7
Creating 'cardtest3.c.gcov'
```

8. Mine Card

```
File 'cardtest4.c'
Lines executed:100.00% of 11
Branches executed:100.00% of 2
Taken at least once:50.00% of 2
Calls executed:80.00% of 5
Creating 'cardtest4.c.gcov'
```

2.3 Random Testing

2.3.1 Test Results

1. Smithy

```
numPlayers: 4
thisPlayer: 3
----------------- Testing Card: smithy ----------------
TEST 1: random test
randomtestcard1.c:65 TEST SUCCESSFULLY COMPLETED ->
     testG.handCount[thisPlayer] == G.handCount[thisPlayer] + 2
randomtestcard1.c:66 TEST SUCCESSFULLY COMPLETED ->
     testG.deckCount[thisPlayer] == G.deckCount[thisPlayer] - 3

 >>>>> SUCCESS: Testing complete smithy <<<<<
```

2. Mine

```
----------------- Testing Card: mine ----------------
TEST 1: random test
randomtestcard2.c:56 TEST SUCCESSFULLY COMPLETED ->
     getCost(testG.hand[thisPlayer][choice1]) + 3 <= getCost(choice2)
randomtestcard2.c:65 TEST SUCCESSFULLY COMPLETED ->
     testG.handCount[thisPlayer] == G.handCount[thisPlayer]
randomtestcard2.c:71 TEST SUCCESSFULLY COMPLETED ->
     testG.supplyCount[choice2] == G.supplyCount[choice2]

 >>>>> SUCCESS: Testing complete mine <<<<<
```

3. Adventurer

```
numplayers: 4
thisplayer: 3
----------------- Testing Card: adventurer ----------------
TEST 1: random test
randomtestcardadventurer.c:88 TEST SUCCESSFULLY COMPLETED ->
     drawntreasure == drawntreasurePre + 2

 >>>>> SUCCESS: Testing complete adventurer <<<<<

```

2.3.2 Code Coverage

1. Smithy

```
File 'randomtestcard1.c'
Lines executed:100.00% of 23
Branches executed:100.00% of 4
Taken at least once:50.00% of 4
Calls executed:88.24% of 17
Creating 'randomtestcard1.c.gcov'

File 'myAssert.c'
Lines executed:35.71% of 14
Branches executed:50.00% of 8
Taken at least once:25.00% of 8
Calls executed:16.67% of 6
Creating 'myAssert.c.gcov'
```

2. Mine

```
File 'randomtestcard2.c'
Lines executed:100.00% of 23
Branches executed:100.00% of 8
Taken at least once:62.50% of 8
Calls executed:86.36% of 22
Creating 'randomtestcard2.c.gcov'

File 'myAssert.c'
Lines executed:71.43% of 14
Branches executed:100.00% of 8
Taken at least once:50.00% of 8
Calls executed:33.33% of 6
Creating 'myAssert.c.gcov'

```

3. Adventurer

```
File 'randomtestcardadventurer.c'
Lines executed:81.82% of 33
Branches executed:33.33% of 18
Taken at least once:16.67% of 18
Calls executed:92.31% of 13
Creating 'randomtestcardadventurer.c.gcov'

File 'myAssert.c'
Lines executed:35.71% of 14
Branches executed:50.00% of 8
Taken at least once:25.00% of 8
Calls executed:16.67% of 6
Creating 'myAssert.c.gcov'
```

3 Debugging

3.1 gdb

The GNU Debugger was used to track down a bug in the Smithy card in dominion.c (implemented in the `play_smithy` function). By creating a breakpoint at the `play_smithy` function, we were able to first confirm the initial values of the variables (hand count and deck count), and then step into the `play_smithy` function. In this case the example was readily apparent, as the `for (int i = 0; i < 4; i++)` statement should be `for (int i = 0; i < 3; i++)` for the Smithy card to draw the correct amount of three cards (and not four as the previous version had it doing).

```
$ gdb cardtest1
GNU gdb (GDB) 8.2.1
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

```
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from cardtest1...done.
(gdb) b 20
Breakpoint 1 at 0x2471: file cardtest1.c, line 20.
(gdb) b 21
Breakpoint 2 at 0x2484: file cardtest1.c, line 21.
(gdb) r
Starting program: /home/liam/Documents/code/osu/2019winter/cs362-software/CS362-W2019/projects/ya
cardtest1.c:14 TEST SUCCESSFULLY COMPLETED -> play_smithy(player, handPos, testGame) == 0
cardtest1.c:15 TEST SUCCESSFULLY COMPLETED -> testGame->deckCount[player] == 2
cardtest1.c:16 TEST SUCCESSFULLY COMPLETED -> testGame->handCount[player] == 7
deck count: 2
hand count: 7

Breakpoint 1, main (argc=1, argv=0x7fffffffe0d8) at cardtest1.c:20
20          play_smithy(player, handPos, testGame);
(gdb) p testGame->deckCount[player]
$1 = 2
(gdb) p testGame->handCount[player]
$2 = 7
(gdb) s
play_smithy (currentPlayer=0, handPos=0, state=0x555555569260)
    at dominion.c:681
681           for (int i = 0; i < 4; i++)
(gdb) s
683               drawCard(currentPlayer, state);
(gdb) c
Continuing.
Breakpoint 2, main (argc=1, argv=0x7fffffffe0d8) at cardtest1.c:21
21          myAssert(testGame->deckCount[player] == 0);
(gdb) p testGame->deckCount[player]
$3 = 0
(gdb) p testGame->handCount[player]
$4 = 8
(gdb) q
A debugging session is active.

        Inferior 1 [process 2873] will be killed.

Quit anyway? (y or n) y
```