

Assignment 5

liam beckman

March 16, 2019

1 Bug-Reports

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 total 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.

```
1  int play_smithy(int currentPlayer, int handPos, struct gameState *state)
2  {
3      //+3 Cards
4      for (int i = 0; i < 4; i++)
5      {
6          drawCard(currentPlayer, state);
7      }
8
9      //discard card from hand
10     discardCard(handPos, currentPlayer, state, 0);
11     return 0;
12 }
```

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 continually be overwritten every iteration of the `while(drawntreasure<2)` statement.

```

1  while(drawntreasure<2)
2
3  //...
4
5      if (cardDrawn == copper || cardDrawn == silver || cardDrawn == gold)
6          drawntreasure++;
7      else
8      {
9          temphand[z]=cardDrawn;
10         state->
11         handCount[currentPlayer]--; //this should just remove the top card (the most
           ↳ recently drawn one).
12     }
```

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.

```

1  int cardEffect(int card, int choice1, int choice2, int choice3, struct gameState *state, int
   ↳ handPos, int *bonus)
2
3  // ...
```

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

2.1.2 play_adventurer

```
1  int play_adventurer(int currentPlayer, int temphand[MAX_HAND], struct gameState *state)
2
3
4  int drawntreasure = 0;
5  int cardDrawn;
6  while(drawntreasure<2)
7
8      // ...
```

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 continually be overwritten every iteration of the `while(drawntreasure<2)` statement.

```
1  while(drawntreasure<2)
2
3  //...
4
5      if (cardDrawn == copper || cardDrawn == silver || cardDrawn == gold)
6          drawntreasure++;
7      else
8      {
9          temphand[z]=cardDrawn;
10         state->
11         handCount[currentPlayer]--; //this should just remove the top card (the most recently
12         ↪ drawn one).
13     }
```

2.1.3 play_smithy

Uninitialized variable `i` in `play_smithy` function.

```
1  int play_smithy(int currentPlayer, int handPos, struct gameState *state)
2
3  //+3 Cards
4      for (i = 0; i < 4; i++)
5
6          // ...
```

2.1.4 play_mine

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

```
1  int play_mine ( int currentPlayer, int choice1, int choice2, struct gameState *state)
2
3      j = state->
4      hand[currentPlayer][choice1]; //store card we will trash
```

```

5
6 // ...
7
8 //discard trashed card
9 for (i = 0; i < state->
10     handCount[currentPlayer]; i++)
11 {
12     if (state->
13         hand[currentPlayer][i] == j)
14     {
15         discardCard(i, currentPlayer, state, 0);
16         break;
17     }
18 }
19
20 return 0;

```

Undeclared variable handPos in play_mine function.

```

1 int play_mine ( int currentPlayer, int choice1, int choice2, struct gameState *state)
2
3 // ...
4
5 //discard card from hand
6     discardCard(handPos, currentPlayer, state, 0);

```

2.1.5 play_council

Uninitialized variable i in play_council function.

```

1 int play_council(int currentPlayer, int handPos, struct gameState *state)
2 {
3     //+4 Cards
4     for (i = 0; i < 4; i++)
5
6         // ...

```

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 ->
4     getCost(1) == 2
5 unittest1.c:23 TEST SUCCESSFULLY COMPLETED ->

```

```

6      getCost(2) == 5
7  unittest1.c:26 TEST SUCCESSFULLY COMPLETED ->
8      getCost(3) == 8
9  unittest1.c:29 TEST SUCCESSFULLY COMPLETED ->
10     getCost(4) == 0

```

2. isGameOver Function

```

1  unittest2.c:28 TEST SUCCESSFULLY COMPLETED ->
2      isGameOver(&testGame) == 0
3  unittest2.c:35 TEST SUCCESSFULLY COMPLETED ->
4      isGameOver(&testGame) == 1
5  unittest2.c:43 TEST SUCCESSFULLY COMPLETED ->
6      isGameOver(&testGame) == 0
7  unittest2.c:46 TEST SUCCESSFULLY COMPLETED ->
8      isGameOver(&testGame) == 1

```

3. compare Function

```

1  unittest3.c:14 TEST SUCCESSFULLY COMPLETED ->
2      compare(b, a) == 1
3  unittest3.c:15 TEST SUCCESSFULLY COMPLETED ->
4      compare(a, b) == -1
5  unittest3.c:16 TEST SUCCESSFULLY COMPLETED ->
6      compare(a, a) == 0
7  unittest3.c:17 TEST SUCCESSFULLY COMPLETED ->
8      compare(b, b) == 0

```

4. updateCoins Function

```

1  unittest4.c:31 TEST SUCCESSFULLY COMPLETED ->
2      testGame->coins == 9
3  unittest4.c:43 TEST SUCCESSFULLY COMPLETED ->
4      testGame->coins == 8

```

5. Smithy Card

(a) Failed Tests

```

1  cardtest1.c:14 TEST SUCCESSFULLY COMPLETED ->
2      play_smithy(player, handPos, testGame) == 0
3  cardtest1.c:15 TEST FAILED: ->
4      testGame->deckCount[player] == 2
5  cardtest1.c:16 TEST FAILED: ->
6      testGame->handCount[player] == 7
7  cardtest1.c:19 TEST SUCCESSFULLY COMPLETED ->
8      testGame->deckCount[player] == 0
9  cardtest1.c:20 TEST SUCCESSFULLY COMPLETED ->
10     testGame->handCount[player] == 8

```

(b) Successful Tests

```

1  cardtest1.c:14 TEST SUCCESSFULLY COMPLETED ->
2      play_smithy(player, handPos, testGame) == 0
3  cardtest1.c:15 TEST SUCCESSFULLY COMPLETED ->

```

```

4      testGame->deckCount[player] == 2
5 cardtest1.c:16 TEST SUCCESSFULLY COMPLETED ->
6      testGame->handCount[player] == 7
7 deck count: 2
8 hand count: 7
9 cardtest1.c:21 TEST SUCCESSFULLY COMPLETED ->
10     testGame->deckCount[player] == 0
11 cardtest1.c:22 TEST SUCCESSFULLY COMPLETED ->
12     testGame->handCount[player] == 8
13 File 'cardtest1.c'
14 Lines executed:100.00% of 14
15 Branches executed:100.00% of 10
16 Taken at least once:50.00% of 10
17 Calls executed:66.67% of 15
18 Creating 'cardtest1.c.gcov'

```

(c) Code Changed

```

1  int play_smithy(int currentPlayer, int handPos, struct gameState *state)
2  {
3      //+3 Cards
4      // changed "i < 4" to "i < 3"
5      for (int i = 0; i < 3; i++)
6      {
7          drawCard(currentPlayer, state);
8      }
9
10     //discard card from hand
11     discardCard(handPos, currentPlayer, state, 0);
12     return 0;
13 }

```

6. Adventurer Card

```

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

```

7. Village Card

```

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

```

8. Mine Card

```

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

```

2.2.2 Code Coverage

1. getCost Function

```
1 File 'unittest1.c'
2 Lines executed:100.00% of 6
3 Branches executed:100.00% of 10
4 Taken at least once:50.00% of 10
5 Calls executed:66.67% of 15
6 Creating 'unittest1.c.gcov'
7
8 File 'dominion.c'
9 Lines executed:2.00% of 599
10 Branches executed:6.85% of 409
11 Taken at least once:1.22% of 409
12 Calls executed:0.00% of 92
13 Creating 'dominion.c.gcov'
```

2. isGameOver Function

```
1 File 'unittest2.c'
2 Lines executed:100.00% of 17
3 Branches executed:100.00% of 12
4 Taken at least once:66.67% of 12
5 Calls executed:71.43% of 14
6 Creating 'unittest2.c.gcov'
7
8 File 'dominion.c'
9 Lines executed:1.67% of 599
10 Branches executed:1.96% of 409
11 Taken at least once:1.96% of 409
12 Calls executed:0.00% of 92
13 Creating 'dominion.c.gcov'
```

3. compare Function

```
1 File 'unittest3.c'
2 Lines executed:100.00% of 9
3 Branches executed:100.00% of 8
4 Taken at least once:50.00% of 8
5 Calls executed:66.67% of 12
6 Creating 'unittest3.c.gcov'
7
8 File 'dominion.c'
9 Lines executed:1.00% of 599
10 Branches executed:0.98% of 409
11 Taken at least once:0.98% of 409
12 Calls executed:0.00% of 92
13 Creating 'dominion.c.gcov'
```

4. updateCoins Function

```
1 File 'unittest4.c'
2 Lines executed:100.00% of 25
3 Branches executed:100.00% of 4
4 Taken at least once:50.00% of 4
5 Calls executed:77.78% of 9
```

```
6 Creating 'unittest4.c.gcov'
7
8 File 'dominion.c'
9 Lines executed:2.34% of 599
10 Branches executed:1.96% of 409
11 Taken at least once:1.71% of 409
12 Calls executed:0.00% of 92
13 Creating 'dominion.c.gcov'
```

5. Smithy Card

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

6. Adventurer Card

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

7. Village Card

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

8. Mine Card

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

2.3 Random Testing

2.3.1 Test Results

1. Smithy


```

1 numPlayers: 4
2 thisPlayer: 3
3 ----- Testing Card: smithy -----
4 TEST 1: random test
5 randomtestcard1.c:65 TEST SUCCESSFULLY COMPLETED ->
6     testG.handCount[thisPlayer] == G.handCount[thisPlayer] + 2
7 randomtestcard1.c:66 TEST SUCCESSFULLY COMPLETED ->
8     testG.deckCount[thisPlayer] == G.deckCount[thisPlayer] - 3
9
10 >>>> SUCCESS: Testing complete smithy <<<<

```

2. Mine

```

1 ----- Testing Card: mine -----
2 TEST 1: random test
3 randomtestcard2.c:56 TEST SUCCESSFULLY COMPLETED ->
4     getCost(testG.hand[thisPlayer][choice1]) + 3 <= getCost(choice2)
5 randomtestcard2.c:65 TEST SUCCESSFULLY COMPLETED ->
6     testG.handCount[thisPlayer] == G.handCount[thisPlayer]
7 randomtestcard2.c:71 TEST SUCCESSFULLY COMPLETED ->
8     testG.supplyCount[choice2] == G.supplyCount[choice2]
9
10 >>>> SUCCESS: Testing complete mine <<<<

```

3. Adventurer

```

1 numplayers: 4
2 thisplayer: 3
3 ----- Testing Card: adventurer -----
4 TEST 1: random test
5 randomtestcardadventurer.c:88 TEST SUCCESSFULLY COMPLETED ->
6     drawntreasure == drawntreasurePre + 2
7
8 >>>> SUCCESS: Testing complete adventurer <<<<
9

```

2.3.2 Code Coverage

1. Smithy

```

1 File 'randomtestcard1.c'
2 Lines executed:100.00% of 23
3 Branches executed:100.00% of 4
4 Taken at least once:50.00% of 4
5 Calls executed:88.24% of 17
6 Creating 'randomtestcard1.c.gcov'
7
8 File 'myAssert.c'
9 Lines executed:35.71% of 14
10 Branches executed:50.00% of 8
11 Taken at least once:25.00% of 8
12 Calls executed:16.67% of 6
13 Creating 'myAssert.c.gcov'

```

2. Mine

```
1 File 'randomtestcard2.c'
2 Lines executed:100.00% of 23
3 Branches executed:100.00% of 8
4 Taken at least once:62.50% of 8
5 Calls executed:86.36% of 22
6 Creating 'randomtestcard2.c.gcov'
7
8 File 'myAssert.c'
9 Lines executed:71.43% of 14
10 Branches executed:100.00% of 8
11 Taken at least once:50.00% of 8
12 Calls executed:33.33% of 6
13 Creating 'myAssert.c.gcov'
14
```

3. Adventurer

```
1 File 'randomtestcardadventurer.c'
2 Lines executed:81.82% of 33
3 Branches executed:33.33% of 18
4 Taken at least once:16.67% of 18
5 Calls executed:92.31% of 13
6 Creating 'randomtestcardadventurer.c.gcov'
7
8 File 'myAssert.c'
9 Lines executed:35.71% of 14
10 Branches executed:50.00% of 8
11 Taken at least once:25.00% of 8
12 Calls executed:16.67% of 6
13 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 break point 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).

```
1 $ gdb cardtest1
2 GNU gdb (GDB) 8.2.1
3 Copyright (C) 2018 Free Software Foundation, Inc.
4 License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
5 This is free software: you are free to change and redistribute it.
6 There is NO WARRANTY, to the extent permitted by law.
```

```

7  Type "show copying" and "show warranty" for details.
8  This GDB was configured as "x86_64-pc-linux-gnu".
9  Type "show configuration" for configuration details.
10 For bug reporting instructions, please see:
11 <http://www.gnu.org/software/gdb/bugs/>.
12 Find the GDB manual and other documentation resources online at:
13 For help, type "help".
14 Type "apropos word" to search for commands related to "word"...
15 Reading symbols from cardtest1...done.
16 (gdb) b 20
17 Breakpoint 1 at 0x2471: file cardtest1.c, line 20.
18 (gdb) b 21
19 Breakpoint 2 at 0x2484: file cardtest1.c, line 21.
20 (gdb) r
21 Starting program: /home/liam/Documents/code/osu/2019winter/cs362-software/CS362-W2019/projects/ya
22 cardtest1.c:14 TEST SUCCESSFULLY COMPLETED -> play_smithy(player, handPos, testGame) == 0
23 cardtest1.c:15 TEST SUCCESSFULLY COMPLETED -> testGame->deckCount[player] == 2
24 cardtest1.c:16 TEST SUCCESSFULLY COMPLETED -> testGame->handCount[player] == 7
25 deck count: 2
26 hand count: 7
27
28 Breakpoint 1, main (argc=1, argv=0x7ffffffffffe0d8) at cardtest1.c:20
29 20      play_smithy(player, handPos, testGame);
30 (gdb) p testGame->deckCount[player]
31 $1 = 2
32 (gdb) p testGame->handCount[player]
33 $2 = 7
34 (gdb) s
35 play_smithy (currentPlayer=0, handPos=0, state=0x555555569260)
36   at dominion.c:681
37 681      for (int i = 0; i < 4; i++)
38 (gdb) s
39 683      drawCard(currentPlayer, state);
40 (gdb) c
41 Continuing.
42 Breakpoint 2, main (argc=1, argv=0x7ffffffffffe0d8) at cardtest1.c:21
43 21      myAssert(testGame->deckCount[player] == 0);
44 (gdb) p testGame->deckCount[player]
45 $3 = 0
46 (gdb) p testGame->handCount[player]
47 $4 = 8
48 (gdb) q
49 A debugging session is active.
50
51      Inferior 1 [process 2873] will be killed.
52
53 Quit anyway? (y or n) y

```