Maricel Paz
(pazma)
CS-362 W2019

Assignment 5

I worked on the repository of Xiaoqiong Dong (original repository below):
https://github.com/DaphneDD/CS362-W2019/blob/dongxiao-assignment-2/projects/dongxiao/dominion/dominion.c

My repository with my teammate's Dominion code:
https://github.com/maricelpp/CS362-W2019/tree/pazma-assignment-5/projects/pazma/dongxiaoDominion/dominion

Table of Contents

Test-Report

My teammate identified the functions they refactored:

- int adventurerAction(int currentPlayer, struct gameState *state);
- int smithyAction(int currentPlayer, struct gameState *state, int handPos);
- int council_roomAction(int currentPlayer, struct gameState *state, int handPos);
- int feastAction(int currentPlayer, struct gameState *state, int choice1);
- int remodelAction(int currentPlayer, struct gameState *state, int choice1, int choice2, int handPos);

I originally refactored:

- int smithyCard(int currentPlayer, struct gameState *state, int handPos)
- int adventurerCard(int drawntreasure, struct gameState *state, int currentPlayer, int temphand[])
- int villageCard(int currentPlayer, struct gameState *state, int handPos)
- int remodelCard(int currentPlayer, struct gameState *state, int handPos, int choice1, int choice2)
- int mineCard(int currentPlayer, struct gameState *state, int handPos, int choice1, int choice2)

Additionally, I originally wrote unit tests and random tests for:

- int smithyCard(int currentPlayer, struct gameState *state, int handPos)
- int adventurerCard(int drawntreasure, struct gameState *state, int currentPlayer, int temphand[])
- int villageCard(int currentPlayer, struct gameState *state, int handPos)
- int numHandCards(struct gameState *state)
- int whoseTurn(struct gameState *state)
- int fullDeckCount(int player, int card, struct gameState *state)

Based on the overlap, I was only able to test the following with my unit tests and random tests:

- Adventurer Card
- Smithy Card

In order to achieve this, I had to make changes to my Makefile, primarily adding the unit tests and the random tests related to the adventurer card and the smithy card. Additionally, I did not have to make changes to my tests to match the function name and parameters of my teammate's code.

Unit Test: Smithy

The Smithy unit tests executed 8 different tests

- Player's hand
- Player's deck
- Player's action count
- Player's discard deck
- Player's buys count
- Other player's hand
- Other player's deck
- Other player's discard deck

For all the tests above, no bugs were found (all passed).

The "smithyAction" function showed 100% coverage with:

- Lines executed:100.00% of 5
- Branches executed:100.00% of 2
- Taken at least once:100.00% of 2
- No calls

Unit Test: Adventurer

The Adventurer unit tests executed 8 different tests

- Player's hand
- Player's deck
- Player's action count
- Player's discard deck
- Player's buys count
- Other player's hand
- Other player's deck
- Other player's discard deck

For all the tests above, no bugs were found (all passed).

However, the "adventurerAction" function had the following coverage:

- Lines executed:62.50% of 16
- Branches executed:66.67% of 12
- Taken at least once:41.67% of 12
- No calls

When examining the code, we can see that:

- The current player's deck had at least one card. This means the deck shuffle was never executed.
- The card drawn was copper, silver, or gold. This means while drawing cards, no cards went in the discard pile. This also means there was no discarding of draw cards.

The above accounts for the coverage gaps. This also means that my test cases do not reach these sections of the code and therefore cannot find any potential bugs in these lines.

Random Test: Smithy

The random test for the Smithy card had a failed assertion:

- Assertion failed: (preTest.playedCardCount == state.playedCardCount), function main, file randomtestcard2.c, line 141.

Due to our understanding of the smithy card, which when played draws 3 cards into the players hand and then is moved to the played card pile, when can identify that the failed assertion indicates 1 of 2 things:

- The smithy card was not added to the played card pile.
- More than 1 card was added to the played card pile.

In examining the code, we learn that the discard function was called as follows:

- discardCard(handPos, currentPlayer, state, 1);

Taking a look at the discardCard() function, we learn that the flag at the end (called as 1 above) needs to be <1 in order to discard a card instead of trashing it. This is a bug as the discardCard() function was called with the value of 1.

The 'smithyAction' function had 100% coverage:

- Lines executed:100.00% of 5
- Branches executed:100.00% of 2
- Taken at least once:100.00% of 2
- No calls

Random Test: Adventurer

The adventurerAction function immediately led to segmentation fault. Upon adding breakpoints and print statements, I was able to trace the issue to an infinite loop caused during the draw treasure process. In order to resolve this issue, the loop had to be adjusted to take into account the case where there aren't 2 treasure cards for a player to draw into their hand.

New code: while(drawntreasure<2 && (state->deckCount[currentPlayer]>0)

The ‘adventurerAction’ function had the following coverage:

- Lines executed:88.24% of 17
- Branches executed:100.00% of 14
- Taken at least once:78.57% of 14
- No calls

Looking at the missing coverage we find the following:

- Line 1232: An empty deck is never tested

Summary

All in all, my teammate’s code is not currently reliable. With the smithy card, a player loses the card permanently every time it is played. With the adventurer card, it is possible to end up in a segmentation fault. Given these two issues are fixed, I would feel confident about their smithy card implementation but not their adventurer card implementation. This is because with their implementation, my tests do not cover the case where the player starts with an empty deck. Until that case is covered and my testing suite reaches 100% coverage for the adventurer card, I wouldn’t be able to confidently say that my teammate’s code is reliable.

Bug-Reports

Smithy Bug

Title: Smithy Card Not Added to Played Pile After Play
Class: Bug
Is it reproducible: Yes
Description:

> **File:** dominion.c
> **Function:** smithyAction()
> When calling the smithyAction() function, the discardCard() function gets called with the wrong trash flag value.
> **Is currently:** discardCard(handPos, currentPlayer, state, 1);
> **Should be:** discardCard(handPos, currentPlayer, state, 0); <- any number <1

Steps to Produce/Reproduce: Call smithyAction()
Expected Results: Smithy card is moved to current player’s played card pile.
Actual Results: Smithy card is moved to trash pile.
Workarounds: None.
Other Information: My tracing the bug, we can find the root cause is the discardCard() function being called with the wrong flag. The comments in the discardCard() function clarify that if we want a card to be moved to the discard pile, the trash flag for the function should be <1.

Adventurer Bug

Title: Adventurer card looks for treasure infinitely if 2 can't be found
Class: Bug
Is it reproducible: Yes
Description:

File: dominion.c
Function: adventurerAction()
When calling the adventurerAction() function, the while loop that controls the drawing of cards does not account for when the current player does not have 2 treasure cards in their deck and discard pile combined. .
Is currently: while(drawntreasure<2)
Should be: while(drawntreasure<2 && (state->deckCount[currentPlayer]>0))

Steps to Produce/Reproduce: Call adventurerAction()
Expected Results: In the case where the current user does not have 2 treasure cards available in their deck or discard pile, adventurer should draw all cards across deck and discard pile, keeping available treasure cards, and end when all cards have been looked through.
Actual Results: In the case where the current user does not have 2 treasure cards available in their deck or discard pile, adventurer draw all cards across deck and discard pile, keeping available treasure cards, and repeating without end until a segmentation fault is triggered.
Workarounds: None.

Debugging

Smithy Card Bug

When testing the smithy card, I first began with the unit tests. The Smithy unit tests executed 8 different tests

- Player's hand
- Player's deck
- Player's action count
- Player's discard deck
- Player's buys count
- Other player's hand
- Other player's deck
- Other player's discard deck

For all the tests above, no bugs were found, meaning all the actions resulted in the expected behavior. The unit tests were trustworthy as they achieve 100% coverage as well.

I then proceeded to run the random tests on the smithy card. The random tests also achieved 100% coverage. However, upon running the random tests, we see right away that the Smithy card had a failed assertion:

- Assertion failed: (preTest.playedCardCount == state.playedCardCount), function main, file randomtestcard2.c, line 141.

Due to our understanding of the smithy card, which when played draws 3 cards into the players hand and then the smithy card is moved to the played card pile, when can identify that the failed assertion indicates 1 of 2 things:

- The smithy card was not added to the played card pile.
- More than 1 card was added to the played card pile.

In examining the code, we see that the discard function was called as follows:

- discardCard(handPos, currentPlayer, state, 1);

Digging further into the code, we trace the code down to the discardCard() function. Taking a look at the discardCard() function, we learn that the flag at the end (called as 1 above) needs to be <1 in order to discard a card instead of trashing it. This is learned via the comments left in the code. This is a bug as the discardCard() function was called with the value of 1.

New Code: discardCard(handPos, currentPlayer, state, 0);

Upon replacing the code with the correct trash flag, we see all tests pass.

Adventurer Card Bug

When testing the adventurer card, I first began with the unit tests. The Adventurer unit tests executed 8 different tests

- Player's hand
- Player's deck
- Player's action count
- Player's discard deck
- Player's buys count
- Other player's hand
- Other player's deck
- Other player's discard deck

For all the tests above, no bugs were found, meaning all the actions resulted in the expected behavior. However, the "adventurerAction" function did not have 100% coverage with my unit test:

- Lines executed:62.50% of 16
- Branches executed:66.67% of 12
- Taken at least once:41.67% of 12

- No calls

When examining the coverage output, we can see that:

- The shuffle of the deck is never executed. This is due to all use cases having the current player's deck have at least one card.
- The discarding of drawn cards never executed. This is due to the card drawn being a copper, silver, or gold. This means while drawing cards, no cards went in the discard pile.

The above accounts for the coverage gaps. This also means that my test cases do not reach these sections of the code and therefore cannot find any potential bugs in these lines.

I then tested with my random tests and found the adventurerAction function immediately led to segmentation fault. Upon adding breakpoints and print statements and based on my understanding of the game of dominion, I was able to trace the issue to the while loop during the draw treasure process. The loop draws cards until 2 treasure cards are found. However, there is the possibility that the current player has <2 treasure cards in deck + discard. In that case, the while loop runs infinitely. In order to resolve this issue, the loop has to be adjusted to take into account the case where there aren't 2 treasure cards for a player to draw into their hand.

New code: while(drawntreasure<2 && (state->deckCount[currentPlayer]>0)

Upon replacing the conditions of the loop, we see the program execute without the segmentation fault caused by the infinite loop.