Brian Laccone

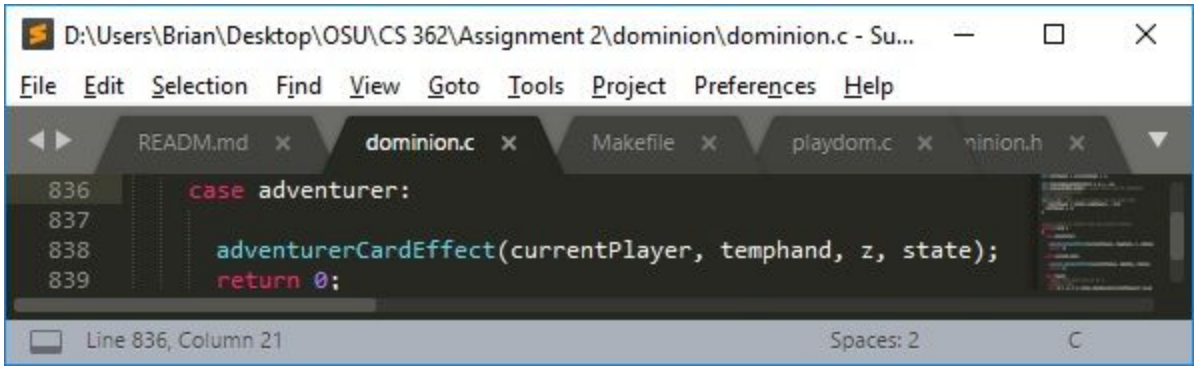
CS 362

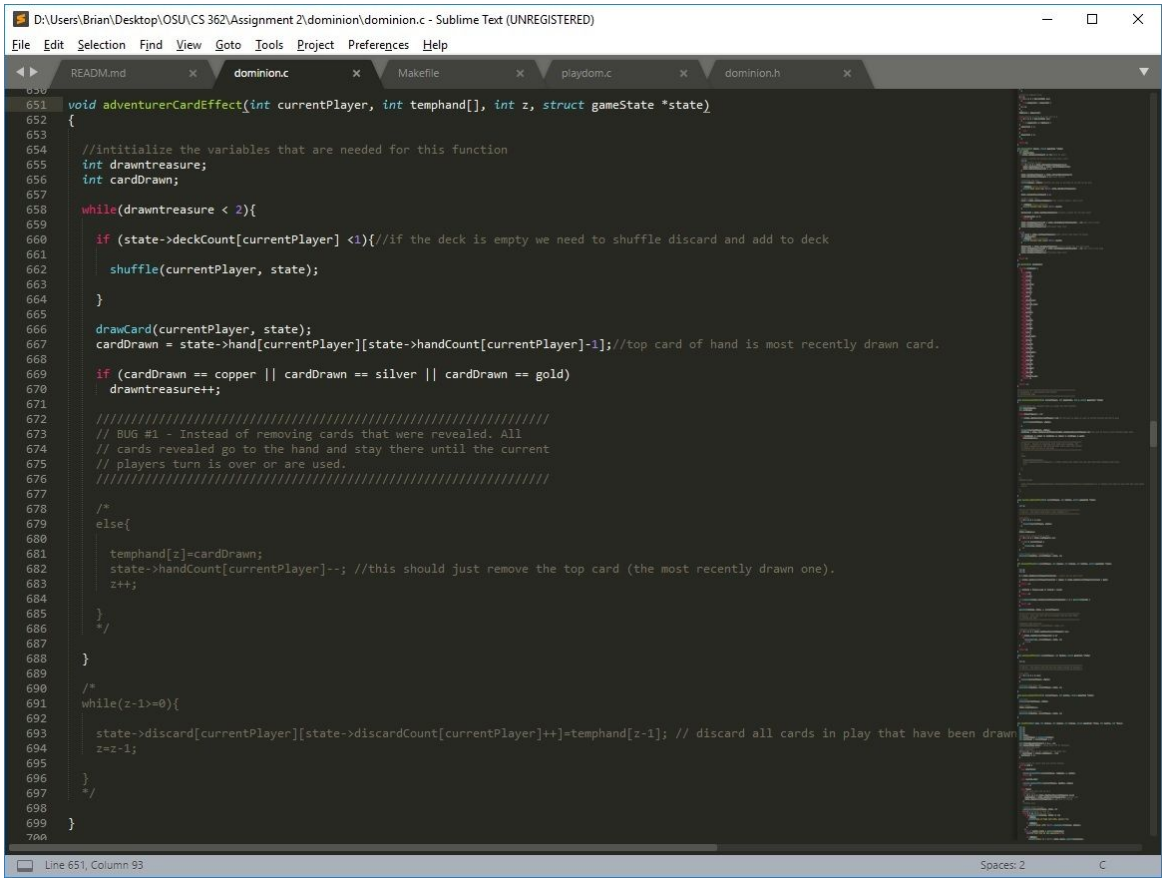
4/14/2018

Assignment-2

Refactor

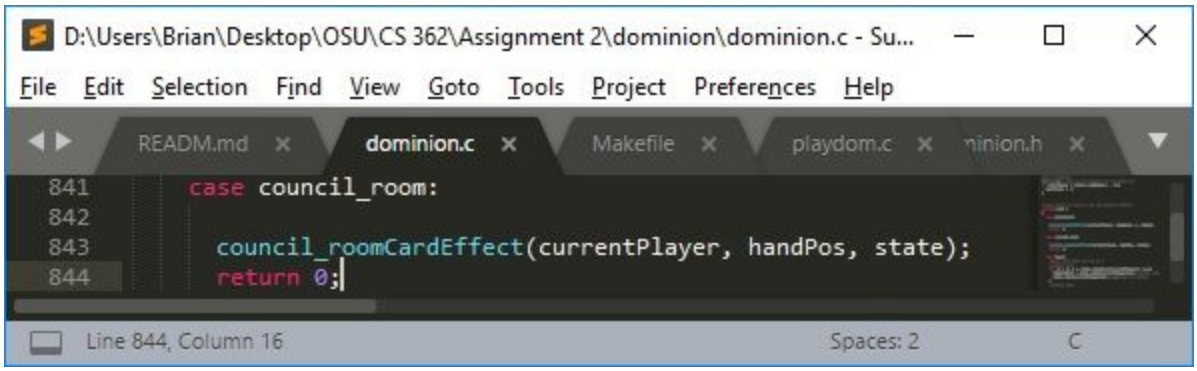
1. Adventurer

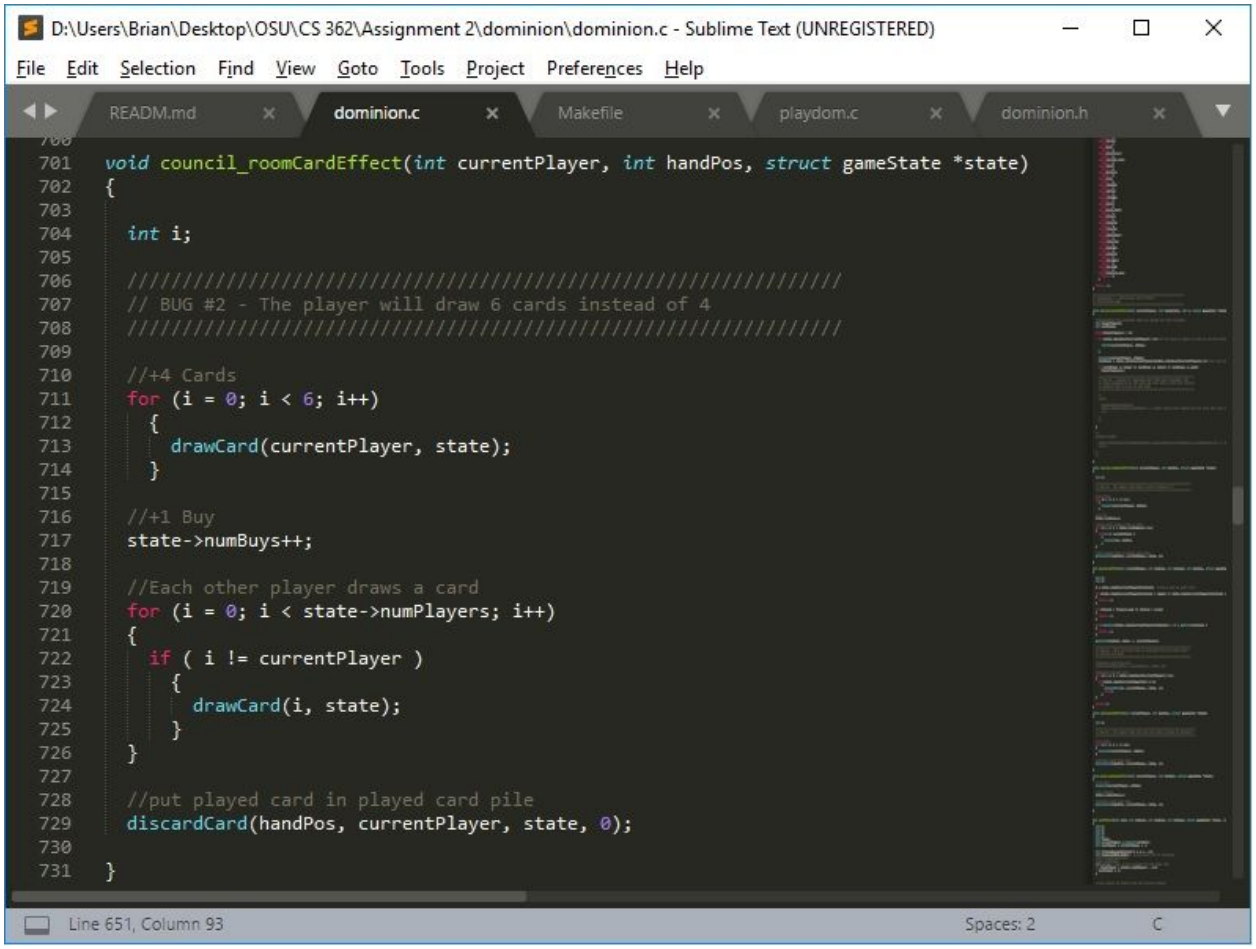




The case adventurer was refactored to the adventurerCardEffect function. The new function contains the parameters: int currentPlayer, int temphand[], int z, struct gameState *state. These parameters are needed to pass information from the cardEffect function to the new function. Also int drawntreasure and int cardDrawn were removed from cardEffect and initialized in the adventureCardEffect function.

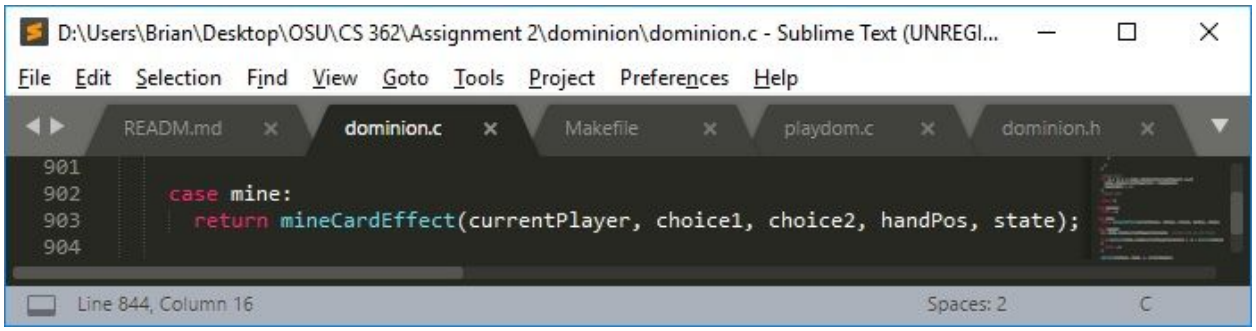
2. Council Room

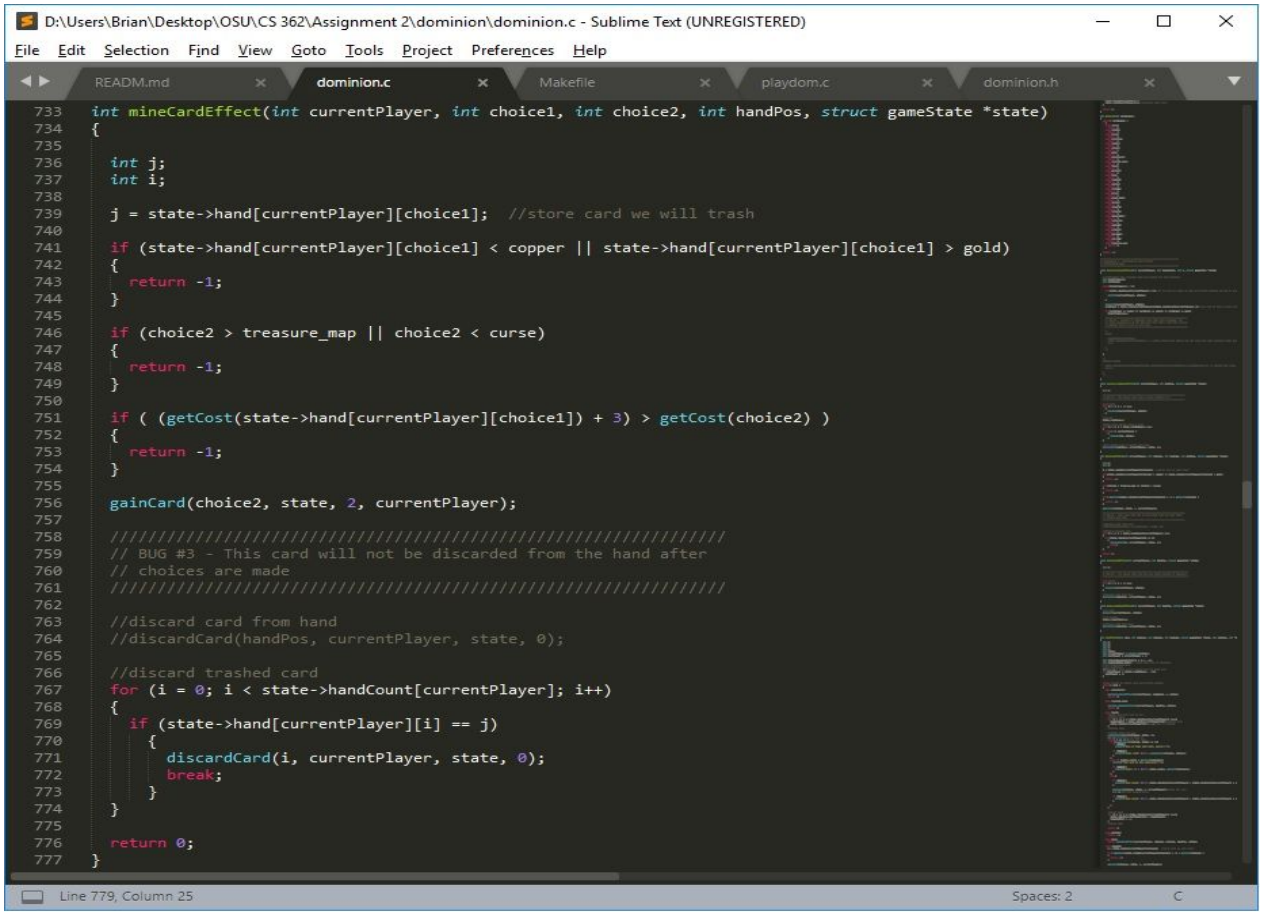




The Council Room card was refactored and moved from case council_room to its own separate function called council_roomCardEffect. The parameters include: int currentPlayer, int handPos, and struct gameState *state. Int i had to be initialized to use the same code from “case council_room.” Case council_room currently calls council_roomCardEffect and returns 0 after the function completes.

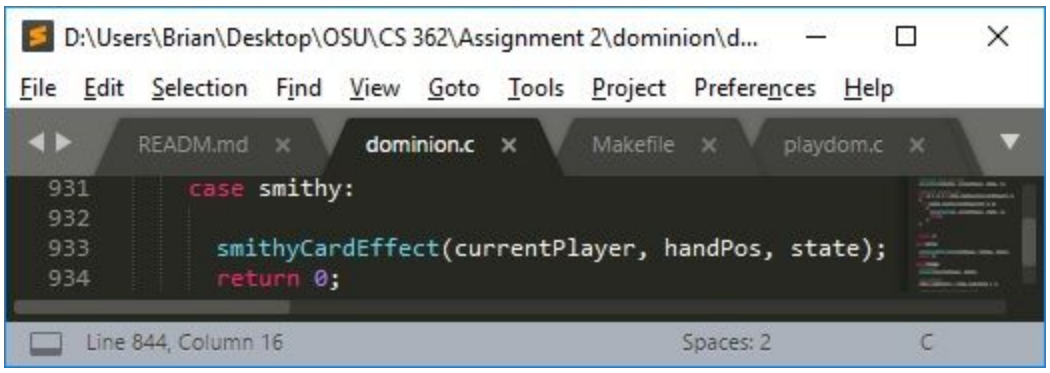
3. Mine

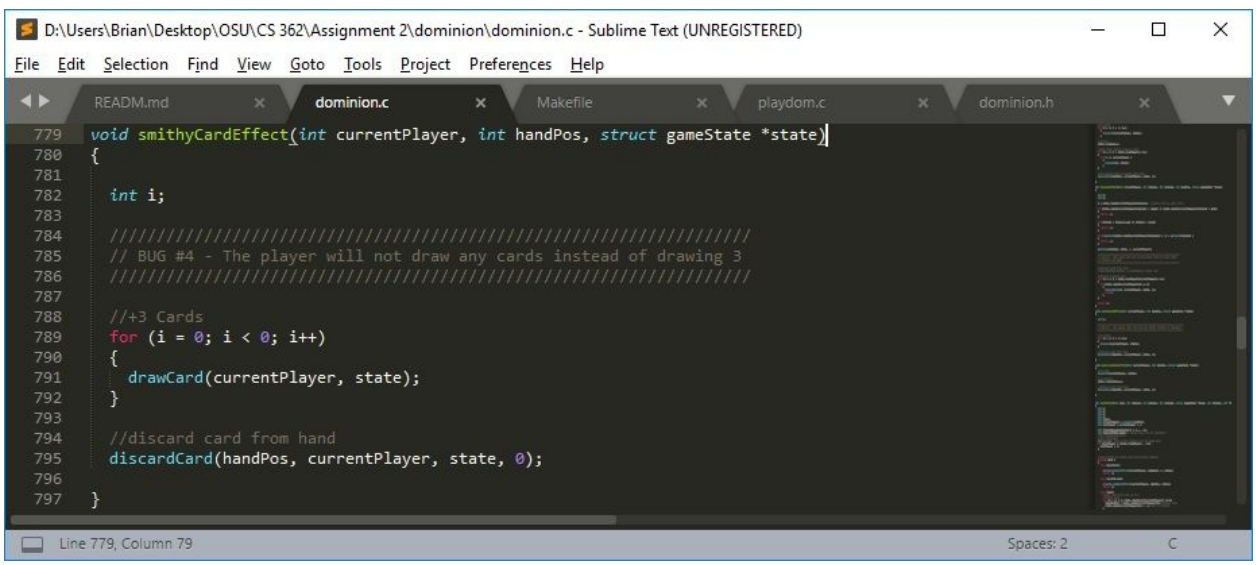




The Mine card case was refactored and moved to a new function called mineCardEffect. Unlike the first two card, mineCardEffect needs a return type of int instead of void because the function needs to return -1 if the choices were invalid. The code is almost the exact same as when it was in case mine except that int i and int j needed to be initialized in the new function. The parameters for mineCardEffect include: int currentPlayer, int choice1, int choice2, int handPos, and struct gameState *state. Case mine changed to simply return the result of mineCardEffect function.

4. Smithy

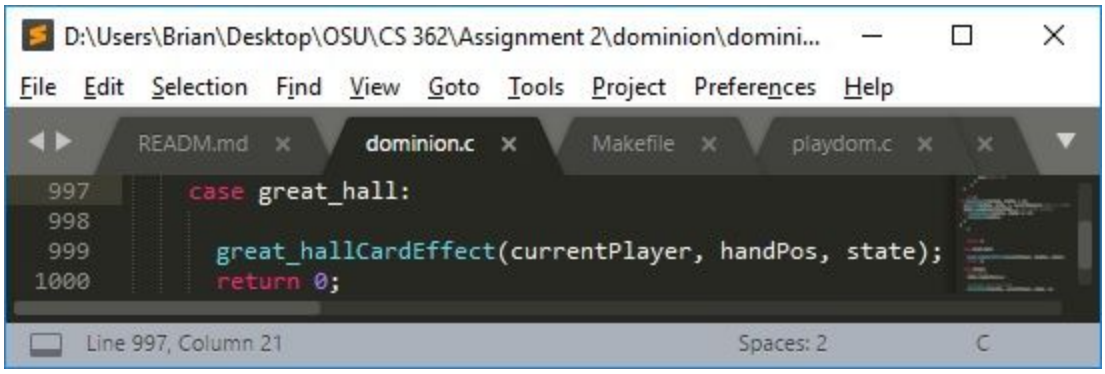


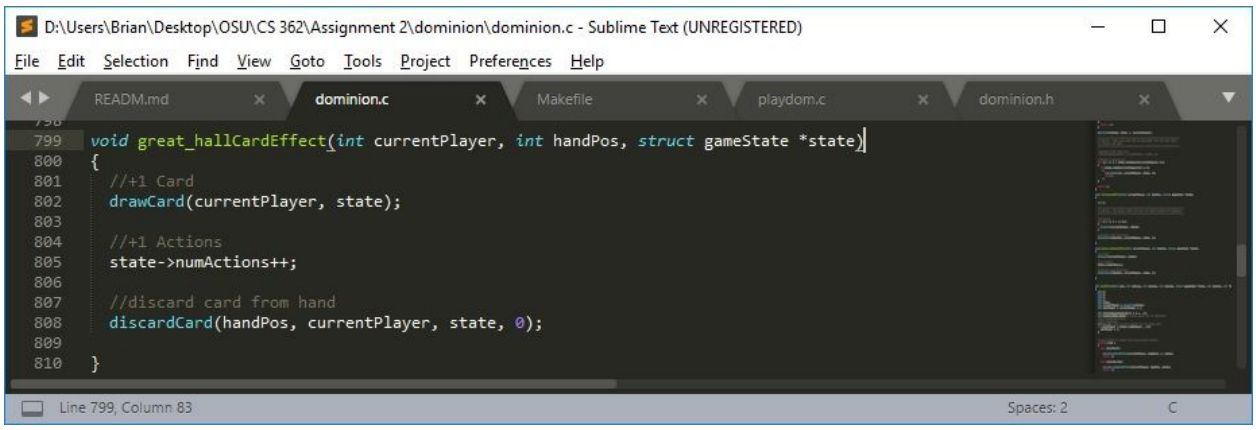


The Smithy card case was refactored and moved to a new function called smithyCardEffect. This function contains the exact code found in the old “case smithy” version except that int i needed to be initialized so the for function can work properly in the new function. The parameters of smithyCardEffect include: int currentPlayer, int handPos, and struct

gameState *state. The new version of case smithy calls the smithyCardEffect function then returns 0.

5. Great Hall





The Great Hall card was refactored and moved to a new function called great_hallCardEffect. This function contains the same exact code found in the old “case great_hall” version. The parameters for great_hallCardEffect contain: int currentPlayer, int handPos, and struct gameState *state. The updated case great_hall simply calls greate_hallCardEffect and returns 0.

Bugs

All the bugs are displayed in the new refactored functions displayed above so I won’t be inserting the same pictures again to save space.

1. Adventurer

The adventurer card is designed to reveal cards from the players deck until they reveal 2 treasure cards. Those 2 treasure cards that were reveal go to the players hand and the other cards, that were revealed, are discarded. To introduce a subtle bug, I commented out the code that discards the other revealed cards that weren't treasure cards. This means that the player could possibly have a gigantic hand that they can then play after adventurer assuming that they have more action moves to use. I feel like this might be hard for a general test to detect because it may need a specific test case to identify that the card wasn't discarding properly.

2. Council Room

The Council Room card allows the player to draw 4 cards and increases there buy by 1 but, as a drawback, each other player draws a card. To introduce a bug, I changed the for loop in the council_roomCardEffect function to stop drawing after 6 cards have been drawn instead of 4. This is a small bug but would drastically change the value of this card and might be a bug that a test might not detect.

3. Mine

The Mine card is an action card that allows the player to trash a treasure from their hand and then gain another treasure that costs 3 more coins to buy. The new card also goes straight to the hand and can be used towards the buy phase that turn. To introduce a bug, I commented out the call to the discard function so this card would not be discarded after it was used. If another action that allows the player to use more than one action cards that turn was played before Mine, then Mine could be played multiple times with one copy of the card because it will never discard until the end of the turn. This seems like a hard thing for a test case to discover unless the test case was specifically designed to look at this card.

4. Smithy

The Smithy card allows the player to draw three cards that turn. To introduce a bug, I changed the for loop so that it would never loop through and draw cards. The bug makes the card do nothing but waste an action and then discard smithy without drawing cards. I feel like this might be an obvious catch from a test case but I wanted to include it because I don't have much experience created tests and I am curious how easily this flaw in the card would come to light.