Table of Contents

CS362 Software Engineering II Section 400 Fall 2019
Final Project Part A

Team Information:

Member Name	Email	ONID
Alexander Drath	<dratha@oregonstate.edu>	dratha
Eric Morgan	<morganer@oregonstate.edu>	morganer
Serena Tay	<tays@oregonstate.edu>	tays

Bug Reports (45 points)

Use the Piazza post for 'Find/Fix Dominion Bugs' to find all valid bugs. Create a bug report for each of the valid bugs you identify. Use the Bug Report template found at the end of this document to create each bug report. For each bug, make sure your reports include the following:

1. Title – provide titles that could be used to tag each bug in an automated system.
2. Class - use this entry to describe the severity of the bug in terms of valid game play.
3. Description – describe the bug in terms of the code. Show the code and describe what is happening and its impact on gameplay, then describe what should be happening. Justify the bug based on student discussion and game documentation.
4. Steps to reproduce the bug – provide detailed steps on how to reproduce the bug.

Dominion Bug 1 Report

==========

Title: discardCard function call within mine and tribute card does not trash card

==========

Class: ***"Serious Bug"***

==========

Date: 11/22/19
Reported By: Alexander Drath
Email: dratha@oregonstate.edu
Product: Version:
Platform: Version:
Browser: Version:
URL: https://github.com/AneresArsenal/CS362-F2019-FinalProject/tree/master/dominion
Is it reproducible: Yes

Description

==========

"The last discardCard function call within the mine case statement of the cardEffect function is not actually trashing the card it is just putting it into the currentPlayers discard pile. The issue can be seen on line 836. The card requirements that the chosen card is trashed. The correction to the bug is to change the trashFlag value within the discardCard function call from 0 to 1." **bug description by T Grasse extracted from Piazza post "Find and Fix Dominion Bugs"**

The tribute card does not trash the card when the discardCard function is called either but that is not a bug and therefore does not need to be addressed.

What is another bug that is tied to both of these issues is the fact that the discardCard function doesn't even have a trash pile nor does it update the discard pile for the player with the discarded card as well. The discardCard function will need to be altered to actually function properly before any sort of testing can be performed on the mine card function. One shouldn't really be doing testing on the cards at all if basic necessary functions such as discardCard function isn't even written correctly.

Steps to Produce/Reproduce

==========

An example of how to reproduce this error would be:
1) Initialize a new game (gameState G*)
2) CurrentPlayers hand contain two copper pieces (hand[currentPlayer][0] = copper, hand[currentPlayer][1] = copper, hand[currentPlayer] = mine).
3) Set choice2 to a silver piece (choice2 = silver).
4) handPos = 2.
5) choice3 = 0.
6) call playCard(handPos, choice1, choice2, choice3, &G);

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

Expected Results

---

handCount to be -1
discardCount to be +1
discard[thisPlayer][discardCount-1] to equal mine card
hand[thisPlayer] to contain silver card

Actual Results

---

handCount will be -1
discardCount will be the same
discard[thisPlayer][discardCount-1] will not exist
hand[thisPlayer] to contain silver card

Workarounds

Not applicable

Attachments

Not applicable

Other Information

Not applicable

Dominion Bug 2 Report

==========
Title: Mine card cannot handle higher valued cards being exchanged for lower valued cards
==========

Class: *"Serious Bug"*
==========

Date: 11/22/19
Reported By: Alexander Drath
Email: dratha@oregonstate.edu
Product: Version:
Platform: Version:
Browser: Version:
URL: https://github.com/AneresArsenal/CS362-F2019-FinalProject/tree/master/dominion
Is it reproducible: Yes

Description
==========
“For the Mine switch statement in the cardEffect function (around line 821), it seems that it checks the cost of the card they trash against the cost of the card they want to buy incorrectly. It looks like if the cost of the treasure that they choose to trash plus 3, is greater than the cost of the card they want to buy, then it will return -1.” **bug description by Robert Saraceno extracted from Piazza post “Find and Fix Dominion Bugs”**
The statement is saying that if the choice1+2 is greater than choice2 then return -1 but this means that choosing an item that costs less than choice1 will fail when it shouldn’t. If you reverse the statements the proper action will occur.

Steps to Produce/Reproduce
==========
An example of how to reproduce this error would be:
1) Initialize a new game (gameState G*)
2) CurrentPlayers hand contains two silver piece (hand[currentPlayer][0] = silver, hand[currentPlayer][1] = silver, hand[currentPlayer] = mine).

3) Set choice2 to a silver piece (choice2 = copper).
4) handPos = 2.
5) choice3 = 0.
6) call playCard(handPos, choice1, choice2, choice3, &G);

```
        if ( (getCost(state->hand[currentPlayer][choice1]) + 3) > getCost(choice2) )
        {
            return -1;
        }
```

Expected Results
---
return = 0
handCount to be -1
discardCount to be +1
discard[thisPlayer][discardCount-1] to equal mine card
hand[thisPlayer] to contain copper card

Actual Results
---
return = -1

Workarounds

Not applicable

Attachments

Not applicable

Other Information

Not applicable

Dominion Bug 3 Report

==========
Title: Remodel card cannot handle higher valued cards being exchanged for lower valued cards
==========

Class: ***"Serious Bug"***
==========

Date: 11/22/19

CS362 Software Engineering II Section 400 Fall 2019
Final Project Part A

Reported By: Alexander Drath
Email: dratha@oregonstate.edu
Product: Version:
Platform: Version:
Browser: Version:
URL: https://github.com/AneresArsenal/CS362-F2019-FinalProject/tree/master/dominion
Is it reproducible: Yes

Description
==========
"In the remodel case within the cardEffect function, the if statement that compares the two choice statements needs to be switched (around Line #846)." **bug description by Alexander Goodman extracted from Piazza post "Find and Fix Dominion Bugs"**
The statement is saying that if the choice1+2 is greater than choice2 then return -1 but this means that choosing an item that costs less than choice1 will fail when it shouldn't. If you reverse the statements the proper action will occur.

Steps to Produce/Reproduce
==========
An example of how to reproduce this error would be:
1) Initialize a new game (gameState G*)
2) CurrentPlayers hand contains two silver piece (hand[currentPlayer][0] = silver, hand[currentPlayer][1] = silver, hand[currentPlayer] = remodel).
3) Set choice2 to a silver piece (choice2 = copper).
4) handPos = 2.
5) choice3 = 0.
6) call playCard(handPos, choice1, choice2, choice3, &G);

```
        if ( (getCost(state->hand[currentPlayer][choice1]) + 2) > getCost(choice2) )
        {
            return -1;
        }
```

Expected Results
---
return = 0
handCount to be -2
discardCount to be +2
discard[thisPlayer][discardCount-1] to equal remodel card
hand[thisPlayer] to contain copper card

Actual Results

Return = -1

Workarounds

Not applicable

Attachments

Not applicable

Other Information

Not applicable

Dominion Bug 4 Report

==========
Title: isGameOver function is unable to cycle through all of the supply card piles to see if game is over
==========

Class: *"Serious Bug"*
==========

Date: 11/22/19
Reported By: Alexander Drath
Email: dratha@oregonstate.edu
Product: Version:
Platform: Version:
Browser: Version:
URL: https://github.com/AneresArsenal/CS362-F2019-FinalProject/tree/master/dominion
Is it reproducible: Yes

Description
==========
“There is a bug in the isGameOver function. While checking if there are 3 cards with a card count of 0, it only loops through 25 cards. The problem is that the supply count array has 27 items. treasure map is equal to 26 in the cards enumeration so the loop never properly loops through all the supply cards, it only loops through 25 of the cards (0-24) so two cards are never checked, it should loop through 27 cards (0-26).” **bug description by George Lenz extracted from Piazza post “Find and Fix Dominion Bugs”**

Steps to Produce/Reproduce
==========
An example of how to reproduce this error would be:
1) Initialize a new game (gameState G*)

2) set the current supply for sea_hag, treasure_map and salvager to 0
3) call isGameOver(&G)

```
    //if three supply pile are at 0, the game ends
    j = 0;
    for (i = 0; i < 25; i++)
    {
        if (state->supplyCount[i] == 0)
        {
            j++;
        }
    }
```

Expected Results

Return 1 and game is over

Actual Results

Return 0 and game continues

Workarounds

Not applicable

Attachments

Not applicable

Other Information

Not applicable

Dominion Bug 5 Report

==========
Title: The scoreFor function is using the discardCount to iterate through the deck
==========

Class: *Serious Bug*
==========

Date: 11/24/2019
Reported By: Eric Morgan

CS362 Software Engineering II Section 400 Fall 2019
Final Project Part A

Email: morganer@oregonstate.edu
Product: Version: CS362-F2019-FinalProject
Platform: Version: Ubuntu 18.04.1 LTS
Browser: Version: N/A
URL: N/A
Is it reproducible: Yes / **Occasionally** / One Time / No

Description
==========
In the scoreFor function, the discardCount is being used for the deck count.

Steps to Produce/Reproduce
==========
This bug can be reproduced by setting a game state where victory cards exist in the deck at a location greater than the number of discarded cards. When this occurs these victory cards in the deck will not be included in the score. When the deck contains less cards than the discarded card count, there is a possibility that additional points could be erroneously awarded to a player.

Expected Results

The scoreFor function should could cards in the deck up to the total deck count.

Actual Results

The scoreFor function counts cards in the deck based on the total discarded cards. This could also cause the function to count cards beyond the deck size.

Workarounds

The code should be updated at line 470 in the dominion.c file to count cards based on the deckCount.

Attachments

Snippet showing the erroneous code location:

```
470     for (i = 0; i < state->discardCount[player]; i++)
471     {
472         if (state->deck[player][i] == curse) {
473             score = score - 1;
474         };
```

Other Information

N/A

Dominion Bug 6 Report

==========

Title: Feast erroneously updates the player's coin count when running effect

==========

Class: *Serious Bug*

==========

Date: 11/24/2019
Reported By: Eric Morgan
Email: morganer@oregonstate.edu
Product: Version: CS362-F2019-FinalProject
Platform: Version: Ubuntu 18.04.1 LTS
Browser: Version: N/A
URL: N/A
Is it reproducible: Yes / Occasionally / One Time / No

Description

==========

The logic in the case statement for feast seems to be incorrect. On line 762 Instead of updating the coins tally, it should enter the while loop and allow players to gain a card up to 5 coins without taking into account how much coins the player has. This effect updates the user's coin state, and does not restore it when complete. As such, the user could lose currently earned coins. Additionally, the card is not trashed when it is played.

Steps to Produce/Reproduce

==========

This effect enables a user to gain a card costing up to 5 coins. This effect is run whenever the user plays this card. The card edits the user's coins to enable gaining a card, but it does not restore the original coin count when complete. This can be proven by adding 10 coins to a user's state, then running the effect. The user's coins will be set at 5 - the cost of the chosen card.

Expected Results

The player should be able to gain a card costing up to 5 coins, and their coin state should remain the same as before the effect is run.

Actual Results

The effect sets the user's coins to 5, then allows them to purchase a card. They do not have the same amount of coins as before the card is played.

Workarounds

The code's logic should be edited not to use the player's coin state to facilitate gaining a card. If this is not possible, the code should memorialize the player's coin count and restore it after the card selection is made.

Attachments

Screen snippet of relevant code section:

```
761         //Update Coins for Buy
762         updateCoins(currentPlayer, state, 5);
763         x = 1;//Condition to loop on
764         while( x == 1) {//Buy one card
765             if (supplyCount(choice1, state) <= 0) {
```

Other Information

N/A

Dominion Bug 7 Report

==========
Title: Tribute card does not properly track revealed cards
==========

Class: *Serious Bug*
==========

Date: 11/24/2019
Reported By: Eric Morgan
Email: morganer@oregonstate.edu
Product: Version: CS362-F2019-FinalProject
Platform: Version: Ubuntu 18.04.1 LTS
Browser: Version: N/A
URL: N/A
Is it reproducible: **Yes** / Occasionally / One Time / No

Description
==========
Tribute card bug around line 1075. The for loop overruns the array. The tributeRevealedCards array has two elements and the loop is written to iterate 3 times.

Steps to Produce/Reproduce
==========

This behavior is noticeable whenever a this card is played.

Expected Results

Cards are revealed to the player, and the appropriate bonus is added to the user's state.

Actual Results

The game always grants three bonuses. The last bonus granted is always an additional action bonus even though a third card was not revealed.

Workarounds

The card's code should be updated to to properly iterate across the revealed cards. In addition, the code should be updated not to grant a bonus if the revealed card's value is -1.

Attachments

```
1071        for (i = 0; i <= 2; i ++) {
1072            if (tributeRevealedCards[i] == copper || tributeReveal
1073                state->coins += 2;
1074            }
1075
1076            else if (tributeRevealedCards[i] == estate || tributeR
1077                drawCard(currentPlayer, state);
1078                drawCard(currentPlayer, state);
1079            }
1080            else { //Action Card
1081                state->numActions = state->numActions + 2;
1082            }
1083        }
1084
```

Other Information

N/A

Dominion Bug 8 Report

==========
Title: Bonus coins are inconsistently added throughout the code
==========

Class: *System Error*
==========

Date: 11/24/2019
Reported By: Eric Morgan
Email: morganer@oregonstate.edu
Product: Version: CS362-F2019-FinalProject
Platform: Version: Ubuntu 18.04.1 LTS
Browser: Version: N/A
URL: N/A
Is it reproducible: **Yes** / Occasionally / One Time / No

Description
==========
The number of bonus coins from actions does not appear to be recorded correctly in cardEffect.

Steps to Produce/Reproduce
==========
Certain cards (like Baron, for example), provide coin bonuses. This bug is noticeable when a player plays the Baron card and examines their coin count.

Expected Results

If a player runs a card that provides a coin bonus, it should be successfully added to their coin count.

Actual Results

A coin bonus is not added to the player's coin state after the card has been played.

Workarounds

The addition of the coin bonus is applied inconsistently throughout the code. For the Baron card, it is added directly to the player's coin state. This bonus is then removed when the udpateCoins function is run in the playCard function.

Attachments

Screen snippet of relevant code location:

```
259     //play card
260     if ( cardEffect(card, choice1, choice2, choice3, state, handP
261     {
262         return -1;
263     }
264
265     //reduce number of actions
266     state->numActions--;
267
268     //update coins (Treasure cards may be added with card draws)
269     updateCoins(state->whoseTurn, state, coin_bonus);
```

Other Information

Dominion Bug 9 Report

==========
Title: Way how duplicate "revealed cards" are handled for Tribute card
==========

Class: Feature Request
==========

Date: November 20th 2019
Reported By: Serena Tay
Email: tays@oregonstate.edu
URL: https://github.com/AneresArsenal/CS362-F2019-FinalProject/tree/master/dominion
Is it reproducible: Occasionally (only when duplicates exist for the two top cards in the next player's deck)

Description
==========
"Possible bug with the way duplicate "revealed cards" are handled for the Tribute card. If tributeRevealedCards[0] is the same as tributeRevealedCards[1], the code sets tributeRevealedCards[1] to -1. The loop that follows expects 2 entries in the array, which is fine - there are still two entries: index 0 contains the card and index 1 contains -1.

What isn't fine is there is no condition to catch the -1. On the first trip through the loop it'll (hopefully correctly) identify the card in index 0 as a Treasure, Victory, or Action card. But on the next iteration, it'll identify that -1 as an Action card, since that falls under the "else" condition." **bug description by Mandi Grant extracted from Piazza post "Find and Fix Dominion Bugs"**

Steps to Produce/Reproduce

==========

- Set top two cards on the next player deck cards to be of the same card (2 action cards, 2 treasury cards or 2 victory cards)

```
if (tributeRevealedCards[0] == tributeRevealedCards[1]) { //If we have a duplicate card, just drop one
            state->playedCards[state->playedCardCount] = tributeRevealedCards[1];
            state->playedCardCount++;
            tributeRevealedCards[1] = -1;
        }

        for (i = 0; i <= 2; i ++) {
            if (tributeRevealedCards[i] == copper || tributeRevealedCards[i] == silver || tributeRevealedCards[i] == gold) { //Treasure cards
                state->coins += 2;
            }

            else if (tributeRevealedCards[i] == estate || tributeRevealedCards[i] == duchy || tributeRevealedCards[i] == province || tributeRevealedCards[i] == gardens || tributeRevealedCards[i] == great_hall) { //Victory Card Found
                drawCard(currentPlayer, state);
                drawCard(currentPlayer, state);
            }
            else { //Action Card
                state->numActions = state->numActions + 2;
            }
        }
```

Expected Results

- When two cards are detected, tribute function should only award once for the corresponding type of cards
- Two action cards - +2 action phases instead of +4
- Two treasure cards - +2 coins in current buy phase instead of +4
- Two victory cards - draw +2 cards instead of +4

Actual Results

- Duplicate card gets marked as -1, this would make it fall in the else catch all statement
- Two action cards - +4 action phases added instead of +2
- Two treasure cards - +2 coins and +2 actions in current buy phase instead of +2 coins only
- Two victory cards - draw +2 cards and +2 actions instead of +2 cards only

Workarounds

Not applicable

Attachments

Not applicable

Other Information

None noted

Dominion Bug 10 Report

==========
Title: Ambassador card comparing position i with card choice
==========

Class: Feature Request
==========

Date: November 20th 2019
Reported By: Serena Tay
Email: tays@oregonstate.edu
URL: https://github.com/AneresArsenal/CS362-F2019-FinalProject/tree/master/dominion
Is it reproducible: Occasionally (need specific situations for the bug to be triggered. See description below)

Description
==========
In the dominion.c cardEffect function, ambassador case, line 1100 to 1106, the program goes through the currentPlayer's hand, and tries to find copies of choice1 card. In this loop, if the i-th card is not "ambassador", the same kind of card as the choice1-th card and "i != choice1", count up "j". But this line compares the position "i" with the card in choice1-th position. [This is incorrect] because we want to compare the card, not the position. **bug description by Akifumi Komori extracted from Piazza post “Find and Fix Dominion Bugs”**

CS362 Software Engineering II Section 400 Fall 2019
Final Project Part A

Steps to Produce/Reproduce

==========

Assuming only valid cases and scenarios where the bug can be triggered

Case 1

- Set choice2 == 1
- Set more than 1 duplicates exist in current hand
- Set choice1 to have an enum that is less than current handCount (i.e i will match the enum once), the bug will be triggered but will not be detected in the original code.

```
state->hand[currentPlayer][choice1]
```

For example: If player choose to discard 1 estate card with choice1 = 7 (i.e. actual estate card is at position 7 in the current hand). The for loop will increase j **by only once** when iterator i equals 1 (enum of estate is 1). Given that choice2 is 1 (only discard one card), the actual result matches the expected results despite the bug being triggered.

Case 2

- Set choice2 == 2
- Set more than 1 duplicates exist in current hand
- Set choice1 to have an enum that is less than current handCount (i.e i will match the enum once), the bug will be triggered and will be detected in the original code.

For example: If player choose to discard 1 estate card with choice1 = 7 (i.e. actual estate card is at position 7 in the current hand). The for loop will increase j **by only once** when iterator i equals 1 (enum of estate is 1). Given that choice2 is 2 (only discard two cards), the actual result discard only 1 whereas in theory, it should discard 2.

Expected Results

Case 1: 1 card matching card type at position choice1 will be discarded
Case 2: 2 cards matching card type at position choice1 will be discarded

Actual Results

Case 1: 1 card matching card type at position choice1 will be discarded
Case 2: 1 card matching card type at position choice1 will be discarded

Workarounds

NA

Attachments

NA

Other Information

NA

Dominion Bug 11 Report

==========
Title: Invalid bug
==========

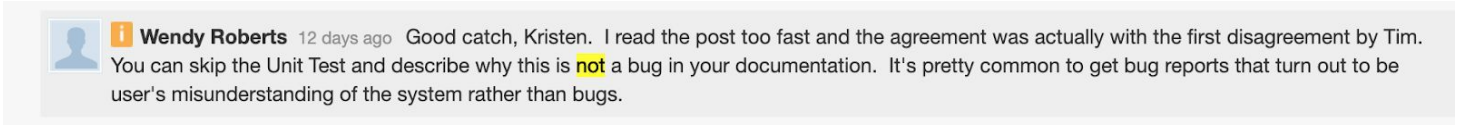
Class: NA
==========

Date:
Reported By:
Email:
Product: Version:
Platform: Version:
Browser: Version:
URL:
Is it reproducible: NA

Description
==========
In the case statement for minion starting around line 955, as the rule says, it will increase action by 1, and then player can make choice either to gain 2 money or draw cards. The code around line 960 calls the function discardCard() before the player makes the choice. And the second issue is the 'if, else if' for choice 1 and choice 2 statement is not right, and should be 'if, else'.

As discussed on Piazza, the noted bugs are not valid bugs as the two statements achieve the results the code intends to produce mainly for the following reasons:

- The discardCard call is needed to remove the played card which is a valid move
- Changing the “if else if” to “if, else” does not change the logic flow unless there is intention to create a catch all else block for errors detected (i.e to ensure user chooses choice1 or choice2 as mentioned by Tim Palecek on Piazza)



Steps to Produce/Reproduce
==========
NA - not a valid bug

Expected Results

NA - not a valid bug

Actual Results

NA - not a valid bug

Workarounds

NA - not a valid bug

Attachments

NA - not a valid bug

Other Information

NA - not a valid bug

Test Plans (45 points)

Use the Bug Reports to create a Test Plan for each Bug Report. The Test Plan should consist of a similar format for each test that includes the following information:

1. Description – describe the bug and your basic strategy to target the bug with a Unit Test.
2. Test Setup – describe the variables and states you would need to set up in order to trigger the bug.
3. Functions – describe the functions that need to be tested. Discuss any dependencies between the functions that need to be tested and whether the dependent functions pose a risk to getting clear test results. If there are risks, discuss any mitigating plans to ensure you achieve clear test results.
4. Assertions – describe the assertions you will use to verify that the bug is fixed. Understand that your Unit Tests should fail on the assertions until the bugs are fixed.

Dominion Bug 1 Test Plan

==========
Description
==========
“The last discardCard function call within the mine case statement of the cardEffect function is not actually trashing the card it is just putting it into the currentPlayers discard pile. The issue can be seen on line 836. The card requirements that the chosen card is trashed. The correction to the bug is to change the trashFlag

value within the discardCard function call from 0 to 1." **bug description by T Grasse extracted from Piazza post "Find and Fix Dominion Bugs"**

The tribute card does not trash the card when the discardCard function is called either but that is not a bug and therefore does not need to be addressed.

Test Setup

==========

Will need to declare these variables:
1) Initialize a new game (gameState G*)
2) CurrentPlayers hand (Ex. hand[currentPlayer][0] = copper, hand[currentPlayer][1] = copper, hand[currentPlayer] = mine).
3) Initialize choice2 (Ex. choice2 = silver).
4) Initialize handPos.
5) Initialize choice3.
6) call playCard(handPos, choice1, choice2, choice3, &G);

Functions

==========

Function to be tested is mainly the cardEffect function:

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

Specifically the case mine:

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

        if (state->hand[currentPlayer][choice1] < copper || state->hand[currentPlayer][choice1] > gold)
        {
            return -1;
        }

        if (choice2 > treasure_map || choice2 < curse)
        {
            return -1;
        }

        if ( (getCost(state->hand[currentPlayer][choice1]) + 3) > getCost(choice2) )
        {
            return -1;
        }
```

```
        gainCard(choice2, state, 2, currentPlayer);

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

        //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;
```

Fix the discardCard() function in order to get this to work properly:

```
int discardCard(int handPos, int currentPlayer, struct gameState *state, int
trashFlag)
{

    //if card is not trashed, added to Played pile
    if (trashFlag < 1)
    {
        //add card to played pile
        state->playedCards[state->playedCardCount] =
state->hand[currentPlayer][handPos];
        state->playedCardCount++;
    }

    //set played card to -1
    state->hand[currentPlayer][handPos] = -1;

    //remove card from player's hand
    if ( handPos == (state->handCount[currentPlayer] - 1) )     //last card in hand
array is played
    {
```

```
        //reduce number of cards in hand
        state->handCount[currentPlayer]--;
    }
    else if ( state->handCount[currentPlayer] == 1 ) //only one card in hand
    {
        //reduce number of cards in hand
        state->handCount[currentPlayer]--;
    }
    else
    {
        //replace discarded card with last card in hand
        state->hand[currentPlayer][handPos] = state->hand[currentPlayer][
(state->handCount[currentPlayer] - 1)];
        //set last card to -1
        state->hand[currentPlayer][state->handCount[currentPlayer] - 1] = -1;
        //reduce number of cards in hand
        state->handCount[currentPlayer]--;
    }

    return 0;
}
```

Assertions

==========

Assertions will be testing these parameters:

1) handCount[thisPlayer]
2) hand[thisPlayer][]
3) discardCount[thisPlayer]
4) discard[thisPlayer][discardCount-1]
5) trashCount[thisPlayer]
6) trashCount[thisPlayer][discardCount-1]

Dominion Bug 2 Test Plan

==========

Description

==========

“For the Mine switch statement in the cardEffect function (around line 821), it seems that it checks the cost of the card they trash against the cost of the card they want to buy incorrectly. It looks like if the cost of the treasure that they choose to trash plus 3, is greater than the cost of the card they want to buy, then it will

return -1.” **bug description by Robert Saraceno extracted from Piazza post “Find and Fix Dominion Bugs”**
The statement is saying that if the choice1+2 is greater than choice2 then return -1 but this means that choosing an item that costs less than choice1 will fail when it shouldn’t. If you reverse the statements the proper action will occur.

Test Setup

==========

Will need to declare these variables:
1) Initialize a new game (gameState G*)
2) CurrentPlayers hand (Ex. hand[currentPlayer][0] = copper, hand[currentPlayer][1] = copper, hand[currentPlayer] = mine).
3) Initialize choice2 (Ex. choice2 = silver).
4) Initialize handPos.
5) Initialize choice3.
6) call playCard(handPos, choice1, choice2, choice3, &G);

Functions

==========

Function to be tested is mainly the cardEffect function:

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

Specifically the case mine:

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

        if (state->hand[currentPlayer][choice1] < copper || state->hand[currentPlayer][choice1] > gold)
        {
            return -1;
        }

        if (choice2 > treasure_map || choice2 < curse)
        {
            return -1;
        }

        if ( (getCost(state->hand[currentPlayer][choice1]) + 3) > getCost(choice2) )
        {
            return -1;
        }
```

```
        gainCard(choice2, state, 2, currentPlayer);

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

        //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;
```

Assertions
==========
Assertions will be testing these parameters:
1) handCount[thisPlayer]
2) hand[thisPlayer][]
3) discardCount[thisPlayer]
4) discard[thisPlayer][discardCount-1]
5) return status

Dominion Bug 3 Test Plan

==========
Description
==========
“In the remodel case within the cardEffect function, the if statement that compares the two choice statements needs to be switched (around Line #846).” **bug description by Alexander Goodman extracted from Piazza post “Find and Fix Dominion Bugs”**
The statement is saying that if the choice1+2 is greater than choice2 then return -1 but this means that choosing an item that costs less than choice1 will fail when it shouldn’t. If you reverse the statements the proper action will occur.

Test Setup
==========
Will need to declare these variables:

CS362 Software Engineering II Section 400 Fall 2019
Final Project Part A

1) Initialize a new game (gameState G*)
2) CurrentPlayers hand (Ex. hand[currentPlayer][0] = copper, hand[currentPlayer][1] = copper, hand[currentPlayer] = mine).
3) Initialize choice2 (Ex. choice2 = silver).
4) Initialize handPos.
5) Initialize choice3.
6) call playCard(handPos, choice1, choice2, choice3, &G);

Functions

==========

Function to be tested is mainly the cardEffect function:

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

Specifically the case remodel:

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

        if ( (getCost(state->hand[currentPlayer][choice1]) + 2) > getCost(choice2) )
        {
            return -1;
        }

        gainCard(choice2, state, 0, currentPlayer);

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

        //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;
```

Assertions
==========
Assertions will be testing these parameters:
1) handCount[thisPlayer]
2) hand[thisPlayer][]
3) discardCount[thisPlayer]
4) discard[thisPlayer][discardCount-1]
5) return status

Dominion Bug 4 Test Plan

==========
Description
==========
"There is a bug in the isGameOver function. While checking if there are 3 cards with a card count of 0, it only loops through 25 cards. The problem is that the supply count array has 27 items. treasure map is equal to 26 in the cards enumeration so the loop never properly loops through all the supply cards, it only loops through 25 of the cards (0-24) so two cards are never checked, it should loop through 27 cards (0-26)." **bug description by George Lenz extracted from Piazza post "Find and Fix Dominion Bugs"**

Test Setup
==========
Will need to declare these variables:
1) Initialize a new game (gameState G*)
2) set the current supply for sea_hag, treasure_map and salvager to 0
3) call isGameOver(&G)

Functions
==========
Function to be tested is mainly the isGameOver function:

```
int isGameOver(struct gameState *state) {
    int i;
    int j;

    //if stack of Province cards is empty, the game ends
    if (state->supplyCount[province] == 0)
    {
        return 1;
    }

    //if three supply pile are at 0, the game ends
    j = 0;
```

```
    for (i = 0; i < 25; i++)
    {
        if (state->supplyCount[i] == 0)
        {
            j++;
        }
    }
    if ( j >= 3)
    {
        return 1;
    }

    return 0;
}
```

Assertions
==========
Return status
supplyCount[all cards]

Dominion Bug 5 Test Plan

==========
Description
==========
In the scoreFor function, the discardCount is being used for the deck count. This causes the score to be incorrectly counted. A unit test should be setup that places a known number of victory cards in both the discard pile, deck, and hand and checks to ensure the proper count is returned by the function.

Test Setup
==========
This bug is encountered when the number of cards in the deck exceeds the number number of cards in the discard pile, and when victory cards are in a position greater than discard count. The test, we will place estate cards, which are worth a single point, in the player's hand, discard pile, and deck. 1 estate card will be placed in the player's hand, 2 cards will be placed in the player's discard pile, and 8 cards will be placed in the deck. This should result in a final score of 8. As the discard pile will only contain two cards, the current scoreFor functionality will not count the full range of cards in the deck.

Functions
==========

This unit test will target the scoreFor function. This function returns the current score based on the cards held in the player's hand, discard pile, and deck. There are no additional dependencies in the functional that could affect the results of this unit test.

Assertions
==========
We will assert that the result returned by the scoreFor function for the state described above returns a score of 11.

Dominion Bug 6 Test Plan

==========
Description
==========
The Feast card allows a player to trash the card to gain a card worth up to 5. On line 762 Instead of updating the coins tally, it should enter the while loop and allow players to gain a card up to 5 coins without taking into account how much coins the player has. This effect updates the user's coin state, and does not restore it when complete. As such, the user could lose currently earned coins. Additionally, the card is not trashed when it is played.

Test Setup
==========
Players can earn additional coins when they play actions. These coins are bonus coins above those earned by holding treasure in their hand. To implement this card's effect the code changes the player's coins, and allows them to select a card. Unfortunately, the current implementation erases any bonus the player may have earned.

To trigger this bug and to ensure we have caught it, we will setup a game state that contains a known number of coins. We will then play the feast card and examine the number of coins after the card is played. The number of coins in the player's hand should remain unchanged after playing this card.

Functions
==========
We will be testing the Feast card effect starting at line 752. Our concern with these tests is primarily around coin management. The updateCoins function is utilized as part of the card's effect. This could be problematic are it directly edits the coin state. To ensure we receive a clear test, we will need to make sure we track the coins prior to the effect being ran, and that it is properly restored after the selected card is gained.

Assertions
==========
To ensure this function properly ran, we need to ensure that the coin state remained unchanged. We will assert that the beginning coin count equals the coin count after the card has been played. In addition, we will need to ensure the card was trashed. We will examine the hand count to ensure it was properly reduced.

Dominion Bug 7 Test Plan

==========

Description

==========

Tribute card bug around line 1075. The for loop overruns the array. The tributeRevealedCards array has two elements and the loop is written to iterate 3 times. This currently awards the player three bonuses.

Test Setup

==========

This bug is currently triggered anytime the card effect is run. To test for specific bonuses we will add two different action cards to the top of the next player's deck. This should grant the player a +4 action bonus only.

Functions

==========

We will be testing the Tribute card effect in the cardEffect function starting at line 1028 in the dominion.c file. There are minimal dependencies that could be considered blockers for testing this card effect. There are a number of functions that directly manipulate the player's deck, so care should be given in designing the tests to ensure cards are being placed in known positions within the deck.

Assertions

==========

The two cards inserted at the top of the deck are two unique action cards. This should result in a +2 bonus being granted for each card, for a total bonus of +4 actions only. As such, we will assert that the proper bonus has been granted for actions, and that no bonus has been granted for coins or additional cards.

Dominion Bug 8 Test Plan

==========

Description

==========

The number of bonus coins from actions does not appear to be recorded correctly. Originally this bug was reported as being isolated to card effect, though this bug exists in any place within the code where a bonus is being granted as part of a card effect. The following effects are affected:

1) Baron
2) Cutpurse
3) Embargo
4) Minion
5) Salvager
6) Steward

The current workflow invokes a card through the playCard function. This function then invokes the selected card through the cardEffect function passing in a coin_bonus argument that tracks coin bonuses added from

played cards. The above detailed cards to not utilize the bonus parameter when added coins, but instead mutate the coin state directly. When code execution returns to the playCard function, we immediately run the updateCoins function. This has the effect of removing any coin bonuses that have been added to the coin state. In addition, as players can earn bonus across multiple actions, this flow removes all bonuses from the player's coin state.

Test Setup
==========
Six separate unit tests will be setup for each of the cards above. Each test will initialize a game state and add the applicable card as the first in the player's hand, and will then call the playCard function to play the card with the relevant choices made to trigger the card's coin bonus. We will then check that the applicable coin bonus has been added.

Functions
==========
As detailed above this error is occuring in the workflow detailed in the playCard function. This workflow has significant downstream dependencies on coin_bonus being properly utilized to grant the proper bonus. If this variable is not used, the bonus will not be properly granted. Additionally, the updateCoins function further has the potential to mutate the game state tracking coins. Care should be given to ensuring all places the game state tracking coins is mutated is identified and understood.

Assertions
==========
For each of the six unit tests we will be asserting that the final coin count after the cardEffect has run is equal to the beginning coin count plus the applicable card bonus.

Dominion Bug 9 Test Plan

==========
Description
==========
"Possible bug with the way duplicate "revealed cards" are handled for the Tribute card. If tributeRevealedCards[0] is the same as tributeRevealedCards[1], the code sets tributeRevealedCards[1] to -1. The loop that follows expects 2 entries in the array, which is fine - there are still two entries: index 0 contains the card and index 1 contains -1.

What isn't fine is there is no condition to catch the -1. On the first trip through the loop it'll (hopefully correctly) identify the card in index 0 as a Treasure, Victory, or Action card. But on the next iteration, it'll identify that -1 as an Action card, since that falls under the "else" condition." **bug description by Mandi Grant extracted from Piazza post "Find and Fix Dominion Bugs"**

Test Setup
==========

CS362 Software Engineering II Section 400 Fall 2019
Final Project Part A

To trigger the bug, we need to make sure that the top two cards from the next player's deck are of the same card type. This can be done by setting the two variables to cards of the same type (does not need to be the exact same card). Unit test will be done by setting two duplicate cards(victory, action and treasure)

```
tributeRevealedCards[0]
```

```
tributeRevealedCards[1]
```

Functions

==========

Function to be tested is mainly the card effect function:

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

Specifically the case tribute:

```
if ((state->discardCount[nextPlayer] + state->deckCount[nextPlayer]) <= 1) {
            if (state->deckCount[nextPlayer] > 0) {
                tributeRevealedCards[0] = state->deck[nextPlayer][state->deckCount[nextPlayer]-1];
                state->deckCount[nextPlayer]--;
            }
            else if (state->discardCount[nextPlayer] > 0) {
                tributeRevealedCards[0] = state->discard[nextPlayer][state->discardCount[nextPlayer]-1];
                state->discardCount[nextPlayer]--;
            }
            else {
                //No Card to Reveal
                if (DEBUG) {
                    printf("No cards to reveal\n");
                }
            }
        }

        else {
            if (state->deckCount[nextPlayer] == 0) {
                for (i = 0; i < state->discardCount[nextPlayer]; i++) {
                    state->deck[nextPlayer][i] = state->discard[nextPlayer][i];//Move to deck
                    state->deckCount[nextPlayer]++;
                    state->discard[nextPlayer][i] = -1;
                    state->discardCount[nextPlayer]--;
                }
```

```
            shuffle(nextPlayer,state);//Shuffle the deck
        }
        tributeRevealedCards[0] = state->deck[nextPlayer][state->deckCount[nextPlayer]-1];
        state->deck[nextPlayer][state->deckCount[nextPlayer]--] = -1;
        state->deckCount[nextPlayer]--;
        tributeRevealedCards[1] = state->deck[nextPlayer][state->deckCount[nextPlayer]-1];
        state->deck[nextPlayer][state->deckCount[nextPlayer]--] = -1;
        state->deckCount[nextPlayer]--;
    }

    if (tributeRevealedCards[0] == tributeRevealedCards[1]) { //If we have a duplicate card,
just drop one
        state->playedCards[state->playedCardCount] = tributeRevealedCards[1];
        state->playedCardCount++;
        tributeRevealedCards[1] = -1;
    }

    for (i = 0; i <= 2; i ++) {
        if (tributeRevealedCards[i] == copper || tributeRevealedCards[i] == silver ||
tributeRevealedCards[i] == gold) { //Treasure cards
            state->coins += 2;
        }

        else if (tributeRevealedCards[i] == estate || tributeRevealedCards[i] == duchy ||
tributeRevealedCards[i] == province || tributeRevealedCards[i] == gardens ||
tributeRevealedCards[i] == great_hall) { //Victory Card Found
            drawCard(currentPlayer, state);
            drawCard(currentPlayer, state);
        }
        else { //Action Card
            state->numActions = state->numActions + 2;
        }
    }

    return 0;
```

- There is a risk that should we call the cardEffect function, the chances that it would trigger another case statement exist. To reduce the risk of that occurring, I have refactored the code to call the playTribute function directly. This ensures all variables can be set manually to trigger the bug occurrence.

- The case statement also calls on the drawCard statement on line 1077 and 1078 should a victory card is detected, to ensure function is performing correctly, the tally of cards will be recorded and printed for observation purposes

```
drawCard(currentPlayer, state)
```

Assertions
==========

- To ensure the bug is fixed, need to ensure that duplicate cards in top 2 cards are only activated once, hence, in theory, all potential bonus coins/action phase and/or cards drawn are a maximum of +2.
- Thus, we would create the following assertions for each of the duplicate cards set:
- Treasure cards: Post coin tally <= pre coin tally + 2
- Victory cards: Post handCount tally <= pre handCount tally + 2
- Action cards: Post numActions tally <= pre numActions tally + 2

Dominion Bug 10 Test Plan

==========

Description
==========

In the dominion.c cardEffect function, ambassador case, line 1100 to 1106, the program goes through the currentPlayer's hand, and tries to find copies of choice1 card. In this loop, if the i-th card is not "ambassador", the same kind of card as the choice1-th card and "i != choice1", count up "j". But this line compares the position "i" with the card in choice1-th position. [This is incorrect] because we want to compare the card, not the position. **bug description by Akifumi Komori extracted from Piazza post "Find and Fix Dominion Bugs"**

Test Setup
==========

Test Setup
==========

Case 1

- Set choice2 == 1
- Set more than 1 duplicates exist in current hand
- Set choice1 that points to a card with an enum that is less than current handCount (i.e iterator i will match the enum once), the bug will be triggered but will not be detected in the original code.

```
    state->hand[currentPlayer][choice1]
```

Case 2

- Set choice2 == 2
- Set more than 1 duplicates exist in current hand

- Set choice1 that points to a card with an enum that is less than current handCount (i.e iterator i will match the enum once), the bug will be triggered and will be detected in the original code.

For both test cases

- creating a second iterator to tally count for assert function (see below)
- Create an array to store the ith position

Functions

==========

Function to be tested is mainly the card effect function:

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

Specifically the case ambassador:

```
case ambassador:
	j = 0;		//used to check if player has enough cards to discard

	if (choice2 > 2 || choice2 < 0)
	{
	    return -1;
	}

	if (choice1 == handPos)
	{
	    return -1;
	}

	for (i = 0; i < state->handCount[currentPlayer]; i++)
	{
	    if (i != handPos && i == state->hand[currentPlayer][choice1] && i != choice1)
	    {
	        j++;
	    }
	}
	if (j < choice2)
	{
	    return -1;
	}

	if (DEBUG)
```

```
            printf("Player %d reveals card number: %d\n", currentPlayer,
state->hand[currentPlayer][choice1]);

        //increase supply count for choosen card by amount being discarded
        state->supplyCount[state->hand[currentPlayer][choice1]] += choice2;

        //each other player gains a copy of revealed card
        for (i = 0; i < state->numPlayers; i++)
        {
            if (i != currentPlayer)
            {
                gainCard(state->hand[currentPlayer][choice1], state, 0, i);
            }
        }

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

        //trash copies of cards returned to supply
        for (j = 0; j < choice2; j++)
        {
            for (i = 0; i < state->handCount[currentPlayer]; i++)
            {
                if (state->hand[currentPlayer][i] == state->hand[currentPlayer][choice1])
                {
                    discardCard(i, currentPlayer, state, 1);
                    break;
                }
            }
        }

        return 0;
```

- There is a risk that should we call the cardEffect function, the chances that it would trigger another case statement exist. To reduce the risk of that occurring, I have refactored the code to call the playAmbassador function directly. This ensures all variables can be set manually to trigger the bug occurrence.
- The case statement also calls on two difference functions namely the gainCard function on line 1123 and discardCard function on 1137. To ensure functions are performing correctly, the tally of cards should be recorded and printed for observation purposes

```
gainCard(state->hand[currentPlayer][choice1], state, 0, i);
```

```
discardCard(i, currentPlayer, state, 1);
```

Assertions

==========

To ensure the bug is caught, we need to make sure that the if statement is triggered when the correct conditions are met. I.e. when the card iterated at the ith position is not the ambassador card played, the card is of the same card at position[choice1] and it is not equal to choice1.

```
for (i = 0; i < state->handCount[currentPlayer]; i++)
	{
		if (i != handPos && i == state->hand[currentPlayer][choice1] && i != choice1)
		{
			j++;
		}
	}
```

Thus we need to assert the following within the if function above:

```
 state->hand[currentPlayer][i]== state->hand[currentPlayer][choice1]
```

- This would assert that the correct card is detected and j is incremented appropriately

```
 newIterator == j
```

- Using the second iterator to tally the count, compare this to j and assert they equal

```
 iArray == duplicateArray //duplicate array contains the position of the first two duplicates in the current player's hand
```

- Using the array created, append array with ith position when the assert function is triggered, compare this to the position of the duplicates places in the current player's deck and ensure they are equal

Dominion Bug 11 Test Plan

==========

Description

==========

In the case statement for minion starting around line 955, as the rule says, it will increase action by 1, and then player can make choice either to gain 2 money or draw cards. The code around line 960 calls the function discardCard() before the player makes the choice. And the second issue is the 'if, else if' for choice 1 and choice 2 statement is not right, and should be 'if, else'

Functions

==========
NA - not a valid bug

Assertions
==========
NA - not a valid bug

Team Work: (10 points)

Describe in detail the contribution of each of the team members using the format below.

	Bug Report	**Test Plan**
Bug 1 Contribution	Report completed by Alex	Report completed by Alex
Bug 2 Contribution	Report completed by Alex	Report completed by Alex
Bug 3 Contribution	Report completed by Alex	Report completed by Alex
Bug 4 Contribution	Report completed by Alex	Report completed by Alex
Bug 5 Contribution	Report completed by Eric	Test plan completed by Eric
Bug 6 Contribution	Report completed by Eric	Test plan completed by Eric
Bug 7 Contribution	Report completed by Eric	Test plan completed by Eric
Bug 8 Contribution	Report completed by Eric	Test plan completed by Eric
Bug 9 Contribution	Report completed by Serena	Test plan completed by Serena
Bug 10 Contribution	Report completed by Serena	Test plan completed by Serena
Bug 11 Contribution	Report completed by Serena	NA - not a valid bug. Serena