CS_362_400_F2018
Joseph Sullivan, sullijos
Assignment 5

Bug-Reports

Title: Adventurer does not go to played card pile after being played.
Classification: Serious bug
Can it be reproduced: Yes. It happens every time the card is played.
Summary: When running fixed unit tests and random unit tests, a gameState struct is created and the cardEffect() function is called for the adventurer card. Every time, the tests show that no adventurer card is placed in the playedCards array as is expected. Failed tests in the unit tests pointed this bug out. The bug was obvious after reviewing the dominion.c code.
A standalone test case:

```
#include "dominion.h"
#include "dominion_helpers.h"
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "rngs.h"
int main(){
    //Declare the following variables.
    int numPlayers = 2;
    int seed = 33343455;
    int kingdomCards[10] = {adventurer, embargo, village, minion, mine,
cutpurse, sea_hag, tribute, smithy, council_room};
    struct gameState pre, post;
    int player, handPos, choice1, choice2, choice3, bonus, numPlayedCards;
    player = handPos = choice1 = choice2 = choice3 = bonus =
numPlayedCards = 0;
    //Initialize the game state and create a copy of it.
    initializeGame(numPlayers, kingdomCards, seed, &pre);
    pre.whoseTurn = player;
    pre.hand[player][handPos] = adventurer;
    memcpy(&post, &pre, sizeof(struct gameState));
    // Play the adventurer card.
    cardEffect(adventurer, choice1, choice2, choice3, &post, handPos,
&bonus);
    // Adventurer is not in playedCards array nor has the playedCardCount
increased by one as expected.
    printf("Cards in played pile before playing adventurer: %d, expected:
%d\n", pre.playedCardCount, numPlayedCards);
    numPlayedCards++;
    printf("Cards in played pile after playing adventurer: %d, expected:
%d\n", post.playedCardCount, numPlayedCards);
    return 0;
}
```

Expected output: After playing the adventurer card, the playedCardCount should be 1 and the adventurer card should be in position 0 of the playedCards array.
Actual output: The playedCardCount is still 0 and no cards are in the playedCards array.

CS_362_400_F2018
Joseph Sullivan, sullijos
Assignment 5

Cause: dominion.c, line 28: while (temp_counter – 1 > 0)
This while loop condition prevents the card in temp_hand[0] from being transferred to the discard pile.
Fix: replace dominion.c, line 28 with: while(temp_counter – 1 >= 0)

Title: Non-coin card drawn by adventurer card does not go to discard pile, gets left in temporary hand.
Classification: Serious bug
Can it be reproduced: Yes. It happens any time a non-coin card is drawn before two coins are drawn when the adventurer card is played.
Summary: Non-coin cards that are drawn when playing the adventurer card are placed in a temporary hand. These cards are supposed to be transferred to the discard pile once two coins have been drawn. The card in the 0 position of the temporary hand array does not get placed in the discard pile. Failed tests in the unit tests pointed this bug out. The bug was obvious after reviewing the dominion.c code.
A standalone test case:

```
#include "dominion.h"
#include "dominion_helpers.h"
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "rngs.h"
int main(){
    //Declare the following variables.
    int numPlayers = 2;
    int seed = 33343455;
    int kingdomCards[10] = {adventurer, embargo, village, minion, mine,
cutpurse, sea_hag, tribute, smithy, council_room};
    struct gameState pre, post;
    int i, player, handPos, choice1, choice2, choice3, bonus;
    player = handPos = choice1 = choice2 = choice3 = bonus = 0;
    //Initialize the game state and create a copy of it.
    initializeGame(numPlayers, kingdomCards, seed, &pre);
    pre.whoseTurn = player;
    pre.hand[player][handPos] = adventurer;
    memcpy(&post, &pre, sizeof(struct gameState));
    // Play the adventurer card.
    cardEffect(adventurer, choice1, choice2, choice3, &post, handPos,
&bonus);
    // Cards in deck from top to bottom
    printf("Cards in deck from top to bottom (4,5,6 are coins, all others
should be discarded):\n");
    for (i = pre.deckCount[player] - 1; i >= 0; i--){
        printf("%d\n", pre.deck[player][i]);
```

```
    }
    printf("Cards in discard before playing adventurer: %d\n",
pre.discardCount[player]);
    printf("Cards in discard after playing adventurer: %d\n",
post.discardCount[player]);
    return 0;
}
```

Expected output: After playing the adventurer card in the test case above, the top card of the deck, which is a 1(estate), will be drawn. This card is added to the temporary hand and should be discarded after two treasure cards are drawn. The discard pile should have 1 card in it at the end of execution.
Actual output: The discard pile has zero cards in it.
Cause: No call to discardCard() is made as with the other cards in dominion.c
Fix: In dominion.c, insert "discardCard(handPos, currentPlayer, state, 0) between lines 32 and 33. Add a int handPos parameter to the adventurer_card() function on line 8. Add a handPos argument to the call to adventurer_card() on line 866.

Title: Smithy card gets trashed instead of being placed in the played card pile after being played.
Classification: Serious bug
Can it be reproduced: Yes. It happens every time the smithy card is played.
Summary: The smithy card does not end up in the played card pile after being played. The call to discardCard() sets the trashFlag set to 1 which trashes smithy instead. Failed tests in the unit tests pointed this bug out. The bug was obvious after reviewing the dominion.c code.
A standalone test case:

```
#include "dominion.h"
#include "dominion_helpers.h"
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "rngs.h"
int main(){
    //Declare the following variables.
    int numPlayers = 2;
    int seed = 33343455;
    int kingdomCards[10] = {adventurer, embargo, village, minion, mine,
cutpurse, sea_hag, tribute, smithy, council_room};
    struct gameState pre, post;
    int player, handPos, choice1, choice2, choice3, bonus, numPlayedCards;
    player = handPos = choice1 = choice2 = choice3 = bonus =
numPlayedCards = 0;
    //Initialize the game state and create a copy of it.
```

```
    initializeGame(numPlayers, kingdomCards, seed, &pre);
    pre.whoseTurn = player;
    pre.hand[player][handPos] = smithy;
    memcpy(&post, &pre, sizeof(struct gameState));
    // Play the adventurer card.
    cardEffect(smithy, choice1, choice2, choice3, &post, handPos, &bonus);
    // Smithy is not in playedCards array nor has the playedCardCount
increased by one as expected.
    printf("Cards in played pile before playing smithy: %d, expected:
%d\n", pre.playedCardCount, numPlayedCards);
    numPlayedCards++;
    printf("Cards in played pile after playing smithy: %d, expected:
%d\n", post.playedCardCount, numPlayedCards);
    return 0;
}
```

Expected output: After playing the smithy card, three cards are drawn, then the smithy card is placed played card pile. The played card pile should have 1 card in it.
Actual output: The discard pile has zero cards in it.
Cause: dominion.c, line 43: discardCard(handPos, currentPlayer, state, 1);
The last argument in this function call sets the trashFlag in discardCard, causing the smithy card to be trashed rather than be placed in the played card pile.
Fix: Change the function call to: discardCard(handPos, currentPlayer, state, 0);

Title: Playing the village card only adds 1 action instead of 2.
Classification: Serious bug
Can it be reproduced: Yes. It happens every time the village card is played.
Summary: The village card only gains the player who played it 1 action instead of the 2 that they should gain. Failed tests in the unit tests pointed this bug out. The bug was obvious after reviewing the dominion.c code.
A standalone test case:

```
#include "dominion.h"
#include "dominion_helpers.h"
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "rngs.h"
int main(){
    //Declare the following variables.
    int numPlayers = 2;
    int seed = 33343455;
    int kingdomCards[10] = {adventurer, embargo, village, minion, mine,
cutpurse, sea_hag, tribute, smithy, council_room};
    struct gameState pre, post;
```

```
    int actions = 1;
    int player, handPos, choice1, choice2, choice3, bonus;
    player = handPos = choice1 = choice2 = choice3 = bonus = 0;
    //Initialize the game state and create a copy of it.
    initializeGame(numPlayers, kingdomCards, seed, &pre);
    pre.whoseTurn = player;
    pre.hand[player][handPos] = village;
    memcpy(&post, &pre, sizeof(struct gameState));
    // Play the adventurer card.
    cardEffect(village, choice1, choice2, choice3, &post, handPos,
&bonus);
    // Smithy is not in playedCards array nor has the playedCardCount
increased by one as expected.
    printf("Actions before playing village: %d, expected: %d\n",
pre.numActions, actions);
    actions += 2;
    printf("Actions after playing village: %d, expected: %d\n",
post.numActions, actions);
    return 0;
}
```

Expected output: After playing the village card the player should gain 2 actions, leaving them with a total of 3 actions.
Actual output: The player only gains 1 action, giving them a total of 2.
Cause: dominion.c, line 52: state->numActions = state->numActions + 1;
The wrong number is added to numActions.
Fix: Change line 52 to: state->numActions = state->numActions + 2;

Test Report

None of my tests needed to be altered in order to work properly with my teammate's code. Thus, I would consider my tests highly maintainable. 100% branch and statement coverage was achieved for the adventurer, smithy, village, and council room cards. The all lines were covered at a minimum of 20002 for the smithy and village cards. The adventurer card had all lines covered tens of thousands of times with the exception of line 15 which was only covered 8533 times. The great hall card only had its lines covered 2 times each as I had not developed random tests for this card. I am confident that all bugs for these cards were identified during my tests. I would say that this code is highly unreliable due to the fact that the code for multiple cards have serious bugs that occur every time that the card is played. These bugs cause cards to function incorrectly and for the game rules a stated on cards to not be followed accurately. While my testing has revealed several bugs, many more unit tests for the other functions and cards would need to be performed in order to fully assess the quality of this code.

CS_362_400_F2018
Joseph Sullivan, sullijos
Assignment 5

Debugging

I identified bugs in my teammate's code by reviewing what failures my tests revealed. All bugs reported were easy to find after reviewing the associated card functions within the dominion.c file. I still went through the exercise of using gdb to locate where bugs were occurring. Once I identified a failed test in my unit test, I would create a break point at the function of the card in question. Once the break was reached, I would step through function with the 'next' command. This revealed which lines were being executed and which were not. While the number of actions that were added with the village card, the trashing of the smithy card, and the omission of a call to discarCard() in the adventurer function were obvious without the use of gdb, it was worthwhile to step through the adventurer card in order to confirm the bug that left the non-coin card stuck in the temporary hand. Stepping through the function allowed me to confirm that a card was being added to the temporary hand as lines 23-25 were executed once, but was never being transferred to the discard pile as lines 30 and 31 were never executed. The identification of bugs and fixes for them are mentioned in the bug report above. As per the assignment 5 description, I am also submitting my random and unit test results as my coverage data that is discussed above is pulled from these files.