

Brian Laccone

CS 362

4/14/2018

Assignment-2

Refactor

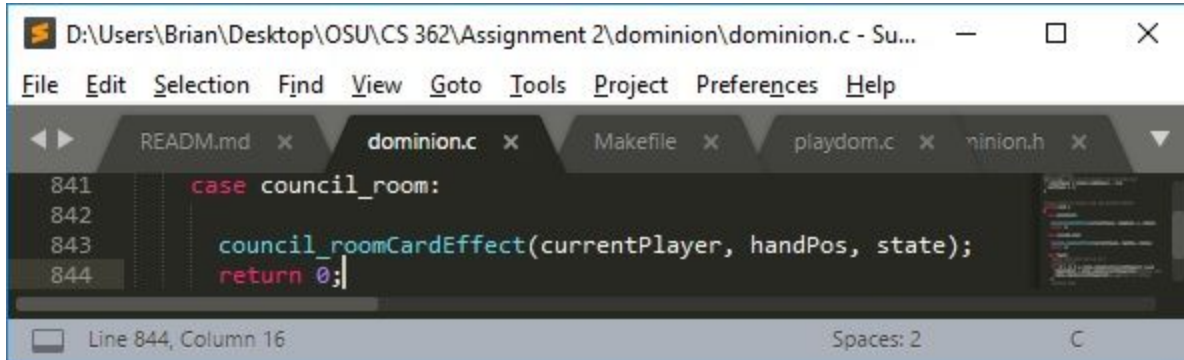
1. Adventurer

```
836     case adventurer:
837
838         adventurerCardEffect(currentPlayer, temphand, z, state);
839         return 0;
```

```
651 void adventurerCardEffect(int currentPlayer, int temphand[], int z, struct gameState *state)
652 {
653     //initialize the variables that are needed for this function
654     int drawntreasure;
655     int cardDrawn;
656
657     while(drawntreasure < 2){
658
659         if (state->deckCount[currentPlayer] < 1){//if the deck is empty we need to shuffle discard and add to deck
660             shuffle(currentPlayer, state);
661         }
662
663         drawCard(currentPlayer, state);
664         cardDrawn = state->hand[currentPlayer][state->handCount[currentPlayer]-1]; //top card of hand is most recently drawn card.
665
666         if (cardDrawn == copper || cardDrawn == silver || cardDrawn == gold)
667             drawntreasure++;
668
669         ///////////////////////////////////////////////////////////////////
670         // BUG #1 - Instead of removing cards that were revealed. All
671         // cards revealed go to the hand and stay there until the current
672         // players turn is over or are used.
673         ///////////////////////////////////////////////////////////////////
674
675         /*
676         else{
677             temphand[z]=cardDrawn;
678             state->handCount[currentPlayer]--; //this should just remove the top card (the most recently drawn one).
679             z++;
680         }
681         */
682     }
683
684     /*
685     while(z-1>=0){
686         state->discard[currentPlayer][state->discardCount[currentPlayer]++]=temphand[z-1]; // discard all cards in play that have been drawn
687         z=z-1;
688     }
689     */
690 }
```

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.

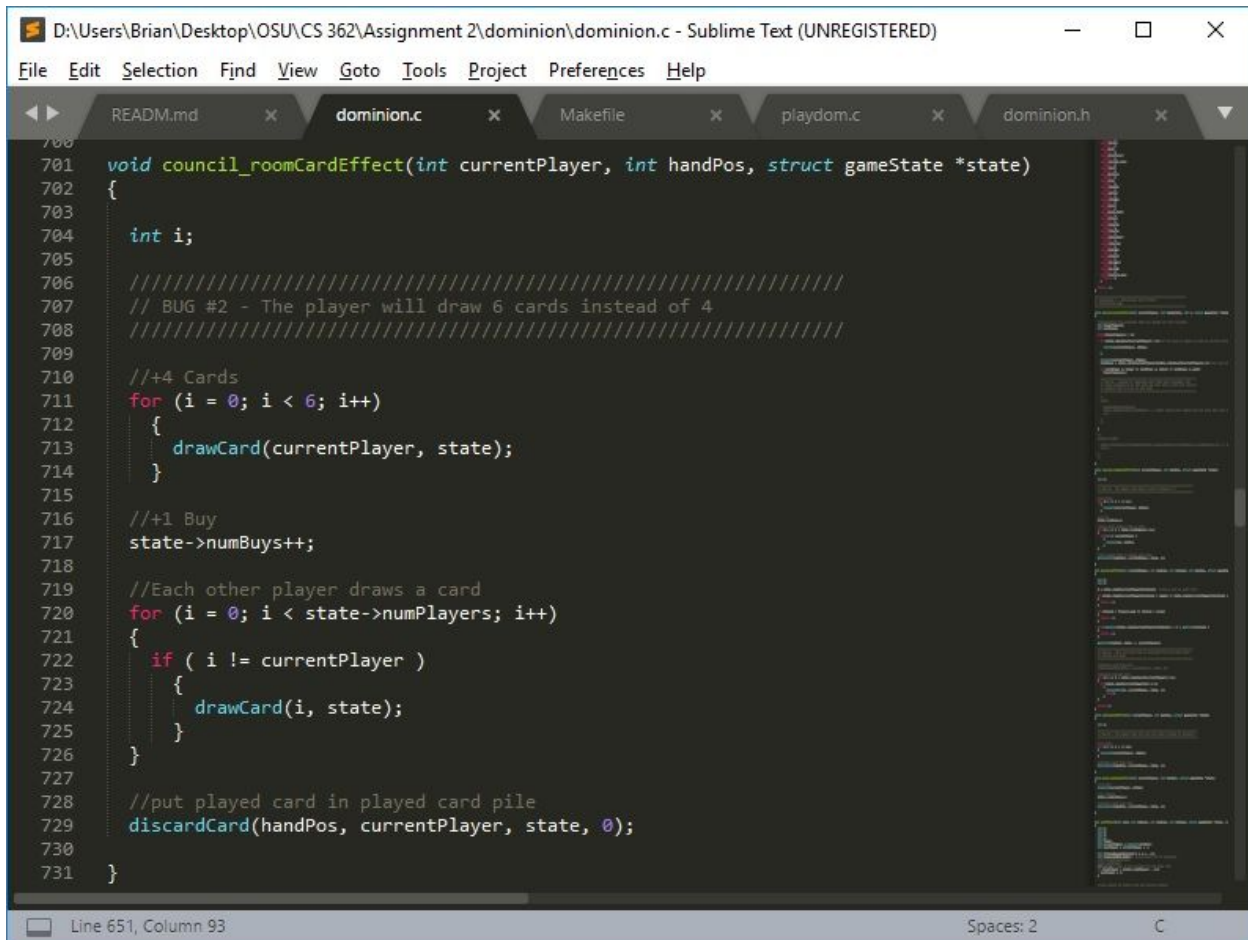
2. Council Room



A screenshot of a code editor window titled "D:\Users\Brian\Desktop\OSU\CS 362\Assignment 2\dominion\dominion.c - Su...". The editor shows the "council_room" case in a switch statement. The code is as follows:

```
841     case council_room:
842
843         council_roomCardEffect(currentPlayer, handPos, state);
844         return 0;
```

The status bar at the bottom indicates "Line 844, Column 16" and "Spaces: 2".



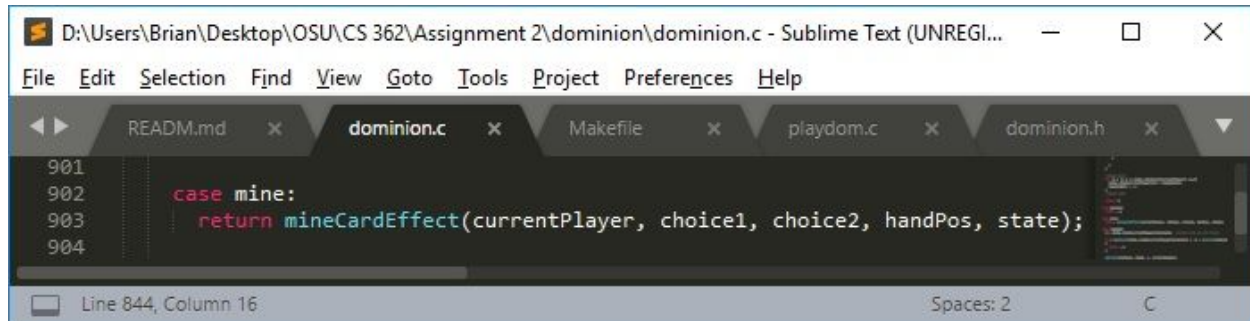
A screenshot of a code editor window titled "D:\Users\Brian\Desktop\OSU\CS 362\Assignment 2\dominion\dominion.c - Sublime Text (UNREGISTERED)". The editor shows the implementation of the "council_roomCardEffect" function. The code is as follows:

```
700
701 void council_roomCardEffect(int currentPlayer, int handPos, struct gameState *state)
702 {
703     int i;
704
705     //////////////////////////////////////
706     // BUG #2 - The player will draw 6 cards instead of 4
707     //////////////////////////////////////
708
709     //4 Cards
710     for (i = 0; i < 6; i++)
711     {
712         drawCard(currentPlayer, state);
713     }
714
715     //+1 Buy
716     state->numBuys++;
717
718     //Each other player draws a card
719     for (i = 0; i < state->numPlayers; i++)
720     {
721         if ( i != currentPlayer )
722         {
723             drawCard(i, state);
724         }
725     }
726
727     //put played card in played card pile
728     discardCard(handPos, currentPlayer, state, 0);
729
730 }
731
```

The status bar at the bottom indicates "Line 651, Column 93" and "Spaces: 2".

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.

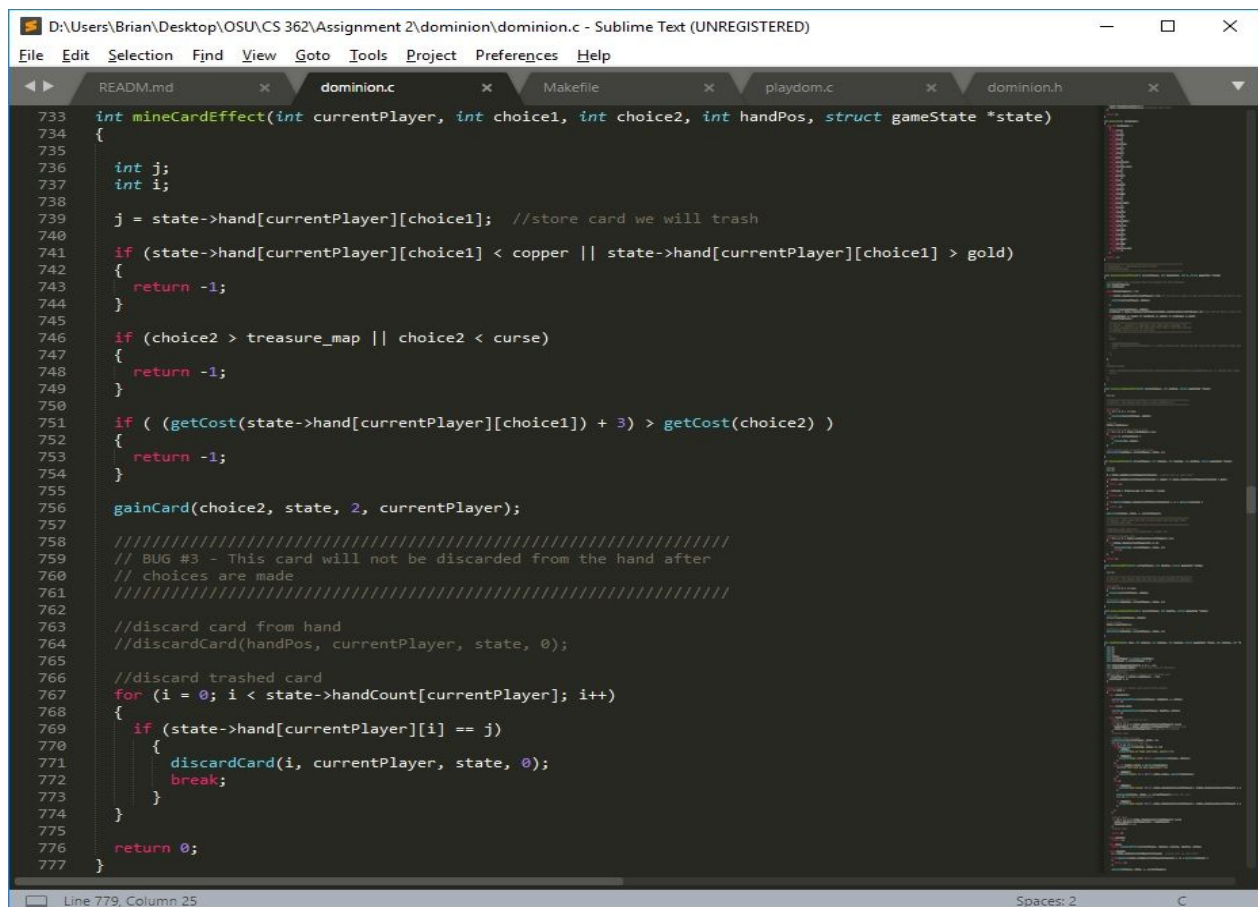
3. Mine



A screenshot of the Sublime Text editor window titled "D:\Users\Brian\Desktop\OSU\CS 362\Assignment 2\dominion\dominion.c - Sublime Text (UNREGI...". The editor shows the "case mine:" block in dominion.c, with lines 901 to 904. The code is as follows:

```
901
902     case mine:
903         return mineCardEffect(currentPlayer, choice1, choice2, handPos, state);
904
```

The status bar at the bottom indicates "Line 844, Column 16" and "Spaces: 2".



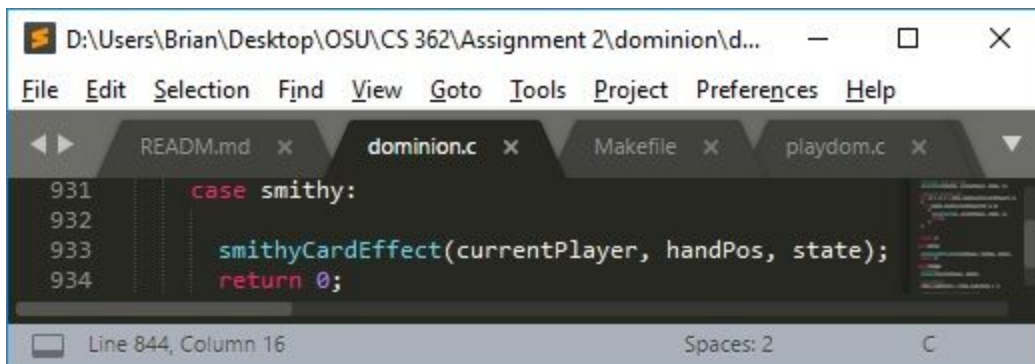
A screenshot of the Sublime Text editor window titled "D:\Users\Brian\Desktop\OSU\CS 362\Assignment 2\dominion\dominion.c - Sublime Text (UNREGISTERED)". The editor shows the mineCardEffect function in dominion.c, with lines 733 to 777. The code is as follows:

```
733 int mineCardEffect(int currentPlayer, int choice1, int choice2, int handPos, struct gameState *state)
734 {
735
736     int j;
737     int i;
738
739     j = state->hand[currentPlayer][choice1]; //store card we will trash
740
741     if (state->hand[currentPlayer][choice1] < copper || state->hand[currentPlayer][choice1] > gold)
742     {
743         return -1;
744     }
745
746     if (choice2 > treasure_map || choice2 < curse)
747     {
748         return -1;
749     }
750
751     if ( (getCost(state->hand[currentPlayer][choice1]) + 3) > getCost(choice2) )
752     {
753         return -1;
754     }
755
756     gainCard(choice2, state, 2, currentPlayer);
757
758     //////////////////////////////////////
759     // BUG #3 - This card will not be discarded from the hand after
760     // choices are made
761     //////////////////////////////////////
762
763     //discard card from hand
764     //discardCard(handPos, currentPlayer, state, 0);
765
766     //discard trashed card
767     for (i = 0; i < state->handCount[currentPlayer]; i++)
768     {
769         if (state->hand[currentPlayer][i] == j)
770         {
771             discardCard(i, currentPlayer, state, 0);
772             break;
773         }
774     }
775
776     return 0;
777 }
```

The status bar at the bottom indicates "Line 779, Column 25" and "Spaces: 2".

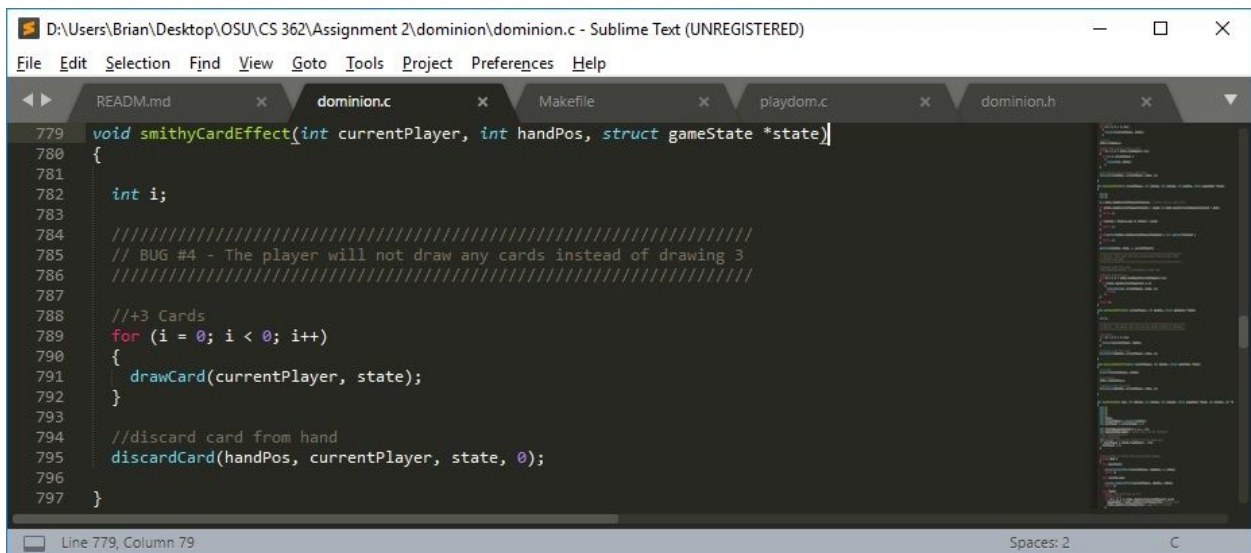
The Mine card case was refactored and moved to a new function called mineCardEffect. Unlike the first two card, mineCardEffect needs a return type of int instead of void because the function needs to return -1 if the choices were invalid. The code is almost the exact same as when it was in case mine 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



```
931     case smithy:
932
933         smithyCardEffect(currentPlayer, handPos, state);
934     return 0;
```

Line 844, Column 16 Spaces: 2 C



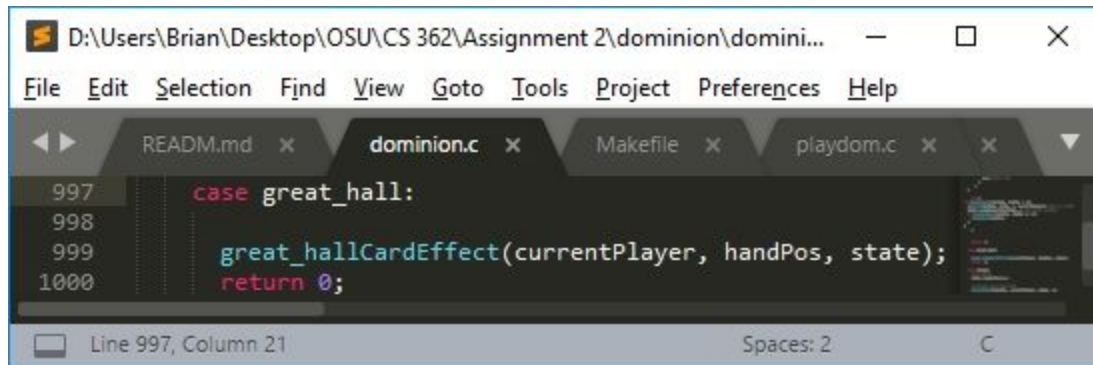
```
779 void smithyCardEffect(int currentPlayer, int handPos, struct gameState *state)
780 {
781     int i;
782
783     ///////////////////////////////////////////////////
784     // BUG #4 - The player will not draw any cards instead of drawing 3
785     ///////////////////////////////////////////////////
786
787     // +3 Cards
788     for (i = 0; i < 0; i++)
789     {
790         drawCard(currentPlayer, state);
791     }
792
793     //discard card from hand
794     discardCard(handPos, currentPlayer, state, 0);
795
796 }
797
```

Line 779, Column 79 Spaces: 2 C

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.

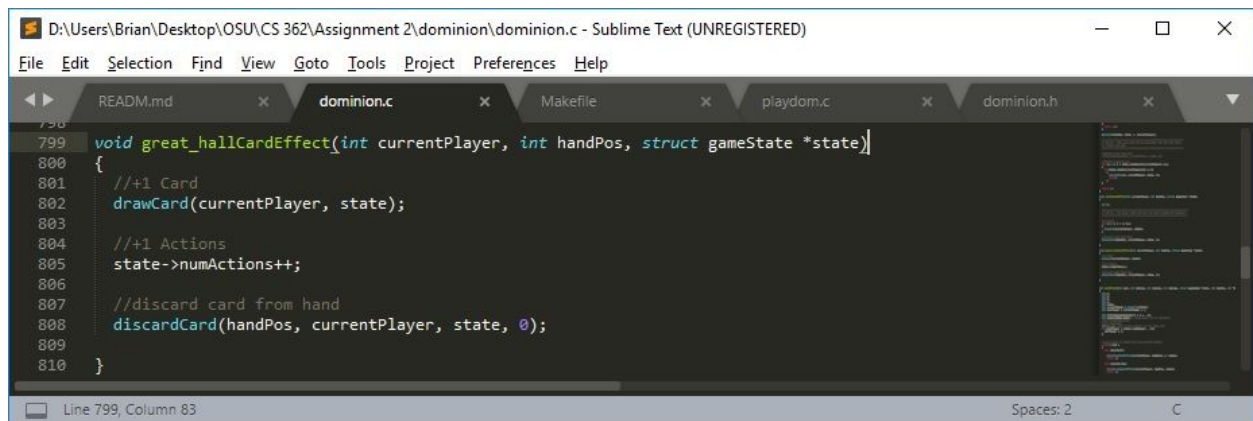
5. Great Hall



This screenshot shows a code editor window with the file path `D:\Users\Brian\Desktop\OSU\CS 362\Assignment 2\dominion\domini...`. The editor displays the `dominion.c` file with tabs for `README.md`, `dominion.c`, `Makefile`, and `playdom.c`. The code at lines 997-1000 shows the refactored `case great_hall` block:

```
997     case great_hall:
998
999         great_hallCardEffect(currentPlayer, handPos, state);
1000     return 0;
```

The status bar at the bottom indicates "Line 997, Column 21" and "Spaces: 2".



This screenshot shows the same code editor window, now displaying the implementation of the `great_hallCardEffect` function in `dominion.c`. The function signature is `void great_hallCardEffect(int currentPlayer, int handPos, struct gameState *state)`. The code at lines 799-810 shows the function body:

```
799 void great_hallCardEffect(int currentPlayer, int handPos, struct gameState *state)
800 {
801     //+1 Card
802     drawCard(currentPlayer, state);
803
804     //+1 Actions
805     state->numActions++;
806
807     //discard card from hand
808     discardCard(handPos, currentPlayer, state, 0);
809 }
810
```

The status bar at the bottom indicates "Line 799, Column 83" and "Spaces: 2".

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.