Assignment-2
Kyle Yoneshige
04/20/2019

Refactor

Adventurer:
The first card I refactored was adventurer. This card's effect is that you reveal cards from you deck until you have 2 treasure cards. Those treasure cards will be added to your hand and the other cards that are revealed are discarded. I put this card into its own function called adventurerEffect. This function takes one parameter called struct gameState *state. Struct gameState holds the game information such as number of players, whose turn it is, phase of turn, coin count, supply count, etc. The main variables needed to implement adventurerEffect is currentPlayer, temphand, drawntreasure, and cardDrawn. The function works by using a while loop to execute until drawntreasure count is greater than 2. These two drawn treasures will be added to the hand and the cards to be discarded will be added to the temp hand. Then at the end of the function the temp hand will be discarded.

The bug I introduced is that I changed the if statement to check for treasures to if((cardDrawn == copper && cardDrawn == silver) || cardDrawn == gold). By changing the treasure check if statement to the above, the function will not count copper or silver's drawn as drawn treasure as the drawn card cannot be both silver and copper at the same time. The function will then have to execute until 2 golds are found but may leave silvers and coppers discarded at the end of the function.

Smithy:
The second card I refactored was smithy. Smithy is a card in which the user draws three cards. I put smithy into its own function called smithyEffect. This function takes two parameters: struct gameState *state and int handPos. Int handPos is an integer which holds the position of a card. The main variables used are int i and currentPlayer. The function works by using a for loop from int i = 0 to int I < 3 to drawCard three times. Then the smithy card is then discarded.

The bug I introduced is that I changed the for loop for drawCard to run from i = 1 to i < 3. This will make the user only draw two cards while still executing the code fine.

Remodel:
The third card I refactored was remodel. Remodel's effect is that the user will discard a card form their hand and will gain a card that costs up to 2 more than the discarded card. I refactored this card into a function called remodelEffect that takes the following parameters: int choice1, int choice2, struct gameState *state, and int handPos. The main variables used in this function is int i, int j, int currentPlayer. The way this function works is that it will store the discarded cost in variable j. Then it will compare the cost of the card to be discarded with the chosen card to be added to verify that the card chosen has a cost of at most 2 more than the discarded card. If the card is within parameters the card will be then added to the hand and the

card to be discarded will be discarded. Then the trashed card and remodel will be discarded form hand.

I chose to not introduce any bugs to this function.

Great Hall:
Great hall is the fourth card I refactored. This card effect will cause the user to draw a card and will allow the user to play +1 action cards that turn. This card is also a victory card. I refactored this card effect to a function called greatHallEffect which takes the following parameters: struct gameState *state and int handPos. This function works by first drawing a card for the user. Then will increase the numActions by +1. Then the great hall card is discarded.

The bug I introduced to this function is that the card will now increase the action count by 2. Thus, the card will allow the user to use +2 action cards that turn.

Sea Hag:
The last card I refactored was the sea hag. This card effect makes each other player (not including the user of Sea Hag) discard the top card of their deck then gain a curse onto their deck. I refactored this card into a function called seaHagEffect which takes the following parameters: struct gameState *state and int handPos. This function works by using a for loop that goes through each player (not including the user of the card). Each player will discard the card on top of the deck. Then deck count is decreased. Then discard count is increased. Then curse is added to the top of the players deck.

The bug I introduced to this function to change the for loop to only use the effect on the user of Sea Hag by changing the for loop if statement to if(i == currentPlayer). Now the player who uses the card will discard a card and will gain a curse.