Maricel Paz
CS-362 W2019
Assignment-4

Random Testing
Write up the development of your random testers, including improvements in coverage that you gained from random testing.(15 points)

Smithy

The development of the Smithy tested was rather straightforward. The Smithy card increases the number of cards in hand by 2, taking the cards from the deck. It then “discards” the Smithy card as it’s already been played. However, the card is not added to the discard pile until the end of the turn. This action is not handled by the Smithy function but rather by the overall game.

Due to this simplicity, I was able to focus my random tests on the randomization of the deck, hand, and discard cards. I opted to randomize the number of cards in each of these piles. I then made a copy of the game state, running the Smithy function on one copy and manually updating the deck and hand counts for the other copy.

From the beginning it was straightforward to achieve 100% coverage as I had in the unit testing. However, I was able to run through many more test cases with the random generator.

In order to create a better tester for this card, the bug introduced in assignment 2 was fixed.

Village

The development of the Village tested was rather straightforward. The Village card increases the number of cards in hand by 1, taking the cards from the deck. It also increases the number of actions that turn by 2. It then “discards” the Village card as it’s already been played. However, the card is not added to the discard pile until the end of the turn. This action is not handled by the Village function but rather by the overall game.

Due to this simplicity, I was able to focus my random tests on the randomization of the deck, hand, actions, and discard cards. I opted to randomize the number of cards in each of these piles. I then made a copy of the game state, running the Village function on one copy and manually updating the deck, action, and hand counts for the other copy.

From the beginning it was straightforward to achieve 100% coverage as I had in the unit testing. However, I was able to run through many more test cases with the random generator.

In order to create a better tester for this card, the bug introduced in assignment 2 was fixed.

Adventurer

The development of the Adventurer tested was much more complex. The Adventurer card draws cards from the deck until it finds 2 treasure cards, which it moves to the hand. If the deck cards run out before two treasures are found, the discard pile is shuffled and moved back to the deck. The search continues until 2 treasure cards are found or the deck once again runs out.

There were a few observations made about the original code which required alteration:

- The code was designed with an infinite loop. It searches for treasure cards until 2 are found. However, there is no escape clause in the case that the deck and discard piles don't contain 2 treasure cards between them.
    - This was altered by adding a conditional checking for the end of the deck and discard piles.
- The first conditional branch is meant to incorporate the discard pile into the deck in the event that the deck is empty. However, it never moves the discard cards to the deck and instead just shuffles an empty deck.
    - This was commented out completely as the drawCard() function takes care of the shuffling of the discard pile and incorporation to the deck.
- In assignment 2 I introduced a bug that initialized the number of cards draw to 2 instead of 0.
    - In order to improve testing of the card, I fixed this bug and initialized the counter to 0.

Code Coverage

discuss how much of adventurer and the other cards' code you managed to cover. Was there code you failed to cover? Why?For at least one card, make your tester achieve 100% statement and branch coverage, and document this and list how long the test must run to achieve this level of coverage. It shouldn't take more than five minutes to achieve the coverage goal (on a reasonable machine, e.g. flip).(15 points)

Smithy

Function 'smithyCard'
Lines executed:100.00% of 5
Branches executed:100.00% of 2
Taken at least once:100.00% of 2
No calls

function smithyCard called 400 returned 100% blocks executed 100%

```
    -:  648:int smithyCard(int currentPlayer, struct gameState *state, int handPos)
    -:  649:{
```

```
        -:  650:        int i;
        -:  651:
        -:  652:        //+3 Cards [BUG: Should be i < 3]
     3200:  653:        for (i = 0; i < 3; i++)
branch  0 taken 75%
branch  1 taken 25%
        -:  654:        {
     1200:  655:                drawCard(currentPlayer, state);
     1200:  656:        }
        -:  657:
        -:  658:        //discard card from hand
      400:  659:        discardCard(handPos, currentPlayer, state, 0);
      400:  660:        return 0;
        -:  661:}
        -:  662:
```

For Smithy I managed to achieve 100% coverage. All lines were reached and all branches were taken at least once. This took very few tests to achieve. 100 iterations of the random tests were able to produce 100% coverage easily.

Village

```
Function 'villageCard'
Lines executed:100.00% of 4
No branches
No calls

function villageCard called 400 returned 100% blocks executed 100%
        -:  700:int villageCard(int currentPlayer, struct gameState *state, int handPos)
        -:  701:{
        -:  702:        //+1 Card
      400:  703:        drawCard(currentPlayer, state);
        -:  704:
        -:  705:        //+2 Actions [BUG: Should increase actions by 2 instead of 1.]
      400:  706:        state->numActions = state->numActions + 2;
        -:  707:
        -:  708:        //discard played card from hand
      400:  709:        discardCard(handPos, currentPlayer, state, 0);
      400:  710:        return 0;
        -:  711:}
        -:  712:
```

For Village I managed to achieve 100% coverage. All lines were reached and there were no branches to take. This took very few tests to achieve. 100 iterations of the random tests were able to produce 100% coverage easily.

Adventurer

The adventurer code was much more complex with a number of bugs. While I was able to improve my coverage from the unit test (which was only around 60 - 70%) it was still only in the 80%s. It was improved but it likely won't be able to reach 100% unless the bugs introduced are fixed.

Unit vs Random

compare your coverage to that of your unit tests that you created in assignment-3 and discuss how the tests differ in ability to detect faults. Which tests had higher coverage –unit or random? Which tests had better fault detection capability? Be detailed and thorough.(15 points)

Smithy - Unit Test

Function 'smithyCard'
Lines executed:100.00% of 5
Branches executed:100.00% of 2
Taken at least once:100.00% of 2
No calls

```
function smithyCard called 1 returned 100% blocks executed 100%
        -:  648:int smithyCard(int currentPlayer, struct gameState *state, int handPos)
        -:  649:{
        -:  650:        int i;
        -:  651:
        -:  652:        //+3 Cards [BUG: Should be i < 3]
       10:  653:        for (i = 0; i < 4; i++)
branch  0 taken 80%
branch  1 taken 20%
        -:  654:        {
        4:  655:                drawCard(currentPlayer, state);
        4:  656:        }
        -:  657:
        -:  658:        //discard card from hand
        1:  659:        discardCard(handPos, currentPlayer, state, 0);
        1:  660:        return 0;
        -:  661:}
        -:  662:
```

Smithy - Random Test

```
Function 'smithyCard'
Lines executed:100.00% of 5
Branches executed:100.00% of 2
Taken at least once:100.00% of 2
No calls

function smithyCard called 400 returned 100% blocks executed 100%
        -:  648:int smithyCard(int currentPlayer, struct gameState *state, int handPos)
        -:  649:{
        -:  650:        int i;
        -:  651:
        -:  652:        //+3 Cards [BUG: Should be i < 3]
     3200:  653:        for (i = 0; i < 3; i++)
branch  0 taken 75%
branch  1 taken 25%
        -:  654:        {
     1200:  655:                drawCard(currentPlayer, state);
     1200:  656:        }
        -:  657:
        -:  658:        //discard card from hand
      400:  659:        discardCard(handPos, currentPlayer, state, 0);
      400:  660:        return 0;
        -:  661:}
        -:  662:
```

Smithy - Comparison

Both the Smithy unit test and random test were able to achieve 100% coverage. In both instances, all lines and branches were reached. That being said, the random test generator was able to go through many more instances of the game than the unit test.

Village - Unit Test

```
Function 'villageCard'
Lines executed:100.00% of 4
No branches
No calls

function villageCard called 1 returned 100% blocks executed 100%
```

```
-:  690:int villageCard(int currentPlayer, struct gameState *state, int handPos)
-:  691:{
-:  692:        //+1 Card
1:  693:        drawCard(currentPlayer, state);
-:  694:
-:  695:        //+2 Actions [BUG: Should increase actions by 2 instead of 1.]
1:  696:        state->numActions = state->numActions + 1;
-:  697:
-:  698:        //discard played card from hand
1:  699:        discardCard(handPos, currentPlayer, state, 0);
1:  700:        return 0;
-:  701:}
-:  702:
```

Village - Random Test

Function 'villageCard'
Lines executed:100.00% of 4
No branches
No calls

function villageCard called 400 returned 100% blocks executed 100%

```
-:  700:int villageCard(int currentPlayer, struct gameState *state, int handPos)
-:  701:{
-:  702:        //+1 Card
400:  703:        drawCard(currentPlayer, state);
-:  704:
-:  705:        //+2 Actions [BUG: Should increase actions by 2 instead of 1.]
400:  706:        state->numActions = state->numActions + 2;
-:  707:
-:  708:        //discard played card from hand
400:  709:        discardCard(handPos, currentPlayer, state, 0);
400:  710:        return 0;
-:  711:}
-:  712:
```

Village - Comparison

Both the Village unit test and random test were able to achieve 100% coverage. In both instances, all lines were reached and there were no branches. That being said, the random test generator was able to go through many more instances of the game than the unit test.

Adventurer - Unit Test

Function 'adventurerCard'
Lines executed:68.75% of 16
Branches executed:66.67% of 12
Taken at least once:50.00% of 12
No calls

```
function adventurerCard called 1 returned 100% blocks executed 73%
        -:  663:int adventurerCard(int drawntreasure, struct gameState *state, int currentPlayer, int
temphand[])
        -:  664:{
        -:  665:        // [BUG: Should be z=0]
        1:  666:        int z=1;
        -:  667:        int cardDrawn;
        -:  668:
        6:  669:        while(drawntreasure<2){
branch  0 taken 67%
branch  1 taken 33%
        2:  670:                if (state->deckCount[currentPlayer] <1){//if the deck is empty we need to
shuffle discard and add to deck
branch  0 taken 0%
branch  1 taken 100%
    #####:  671:                    shuffle(currentPlayer, state);
    #####:  672:                }
        2:  673:                drawCard(currentPlayer, state);
        2:  674:                cardDrawn =
state->hand[currentPlayer][state->handCount[currentPlayer]-1];//top card of hand is most
recently drawn card.
        2:  675:                if (cardDrawn == copper || cardDrawn == silver || cardDrawn == gold)
branch  0 taken 0%
branch  1 taken 100%
branch  2 never executed
branch  3 never executed
branch  4 never executed
branch  5 never executed
        2:  676:                    drawntreasure++;
        -:  677:                else{
    #####:  678:                    temphand[z]=cardDrawn;
    #####:  679:                    state->handCount[currentPlayer]--; //this should just remove the
top card (the most recently drawn one).
    #####:  680:                    z++;
        -:  681:                }
```

```
        -:  682:        }
        4:  683:        while(z-1>=0){
branch  0 taken 50%
branch  1 taken 50%
        1:  684:
state->discard[currentPlayer][state->discardCount[currentPlayer]++]=temphand[z-1]; // discard
all cards in play that have been drawn
        1:  685:                z=z-1;
        -:  686:        }
        1:  687:        return 0;
        -:  688:}
        -:  689:
```