Name: Haris Sohail
Date: 05/11/2018
Assignment 4
CS362

Random Testing

Randomtestcard1 - (Smithy):

The random tester aims to ensure the total cards drawn are equivalent to 3. The two main aspects to go about this are to check the total amount of cards in the hand before and after. Then to also check the total amount of cards in the deck before and after. Beyond that the tester ensures the hand and deck cards for the other player do not change.

First we randomize the inputs for the base kingdom cards. Every test runs with a random assortment of Kingdom cards and random seed value. Next we randomize the total players as well as the hand count for player 1 who will undergo the testing. By randomizing the players we are spreading or coverage very wide to see where the game will break. By randomizing hand counts we can see if the increments are changed at all depending on card number.

The inputs for the kingdom cards are limited to actual existing cards within the game, seg faults occuring when using outside inputs therefore I kept the pool limited to valid entries only. As for total players the pool is ranged from 0 to 4. 0 is kept to test for game breaking ability yet it demonstrates the game continues regardless, thus a bug was discovered. Hand counts were randomized by adding a random value to the start hand, this ensured full validation of the cards being added.

Randomtestcard2 - (Village):

The random tester aims to ensure the player draws 1 card and gains 2 actions. This required implementing a random start action value along with a random card and deck amount. This type of random testing would check not only that the hand gains one card but that the card is also removed from the players deck.

In a similar manner to Smithy we choose to randomize several elements of the game state before initializing. This includes kingdom cards, total players, hand counts, total actions and deck counts. This function operates in a similar manner to Smithy thus the randomized inputs were generally kept the same.

The results indicate a bug where the total actions added are only 1 and not 2. By randomizing the start actions we saw this bug happen in every variation of the test cementing its existence. Otherwise we find 1 card is gained and no other state changes occur to player 1 or 2. However, this second random test further iterates the point of randomtestcard1 where the total players being 0 leads no bugs and allows the game to play normally.

Randomtestadventurer - (Adventurer):

Adventurer is a more difficult card to test and even further so with random inputs being placed. The main issue arising is the bug that was implemented for assignment 2. It prevents any input, random or manual from reaching certain parts of the code. Regardless we are able to uncover several bugs here.

The random tester randomizes kingdom cards, hand counts, discard counts and deck counts. We are trying to test if the player is properly able to draw cards and add them to their hand. Again by randomizing the starting hand count we can see how the card behaves within a different state. This stands for all other counts as well.

We find the player hand count is always 1 more than expected for all random tests. This is actually uncovered to be the adventurer card not having a discard function and remaining in the hand even after being played. The random test does prove the effect is working where the player draws from their deck till they have two treasure cards. The deck counts continuously match up. The other concern is the discard pile never gains any cards. We expected at least 1 from the adventurer card being thrown out. However this is the result of the bug where the player will draw the first two cards regardless of them being treasure or not.

Code Coverage

Randomtestcard1 - (Smithy):

The random tester for Smithy obtained full 100% coverage. It covers all statements, branches and lines. This is achieved easily, within the first few seconds of running.

Randomtestcard2 - (Village):

The random tester for Village obtained full 100% coverage. It covers all statements, branches and lines. This is achieved easily, within the first few seconds of running.

Randomtestadventurer - (Adventurer):

The random tester obtains 76.92% total coverage. It hits only 50% of call executions. It has 100% branch coverage however. This is due to the bug where the player automatically draws two treasure cards, thus the function never has to trigger the else statement. No amount of random input/state changing can get around this bug to get more coverage. The bug must be fixed first.

Unit Vs. Random

Randomtestcard1 - (Smithy):

In comparison we find a lot of similarities, however there are a few key differences. While both types covered the card 100%, only the random tester uncovered the number of players bug where the game continues even with 0 players present. Beyond that the random tester was better able to prove the presence of bugs with a far more variety of inputs. By having such varying game states we are better able to see the presence of the bugs through each test.

Randomtestcard2 - (Village):

Overall there was little difference between the unit and random tests for this card. Both obtained 100% code coverage while unearthing the same bugs relating to actions. The random test did also uncover the number of players bug that the one for Smithy did. This further illustrates the point that random testing is better than unit testing is in uncovering hidden bugs that normally would not be activated with unit tests.

Randomtestadventurer - (Adventurer):

If the bug inside adventurer is fixed, I am leaning towards saying the random tester can get more coverage. By being able to randomize the deck it can ensure the player has to possible go through the entire deck to get the two treasure cards. However, the random test and unit test both achieved the same code coverage due to bug limitations so we cannot say for sure as of right now.

Beyond that the Adventurer card benefits greatly from random tests in the sense that we can better control the outcome of the function. The unit tests are incapable of reaching that point since they cannot change game inputs/states.