Ashik Md Alim Chowdhury

Assignment 4

Random Testing

Randomtestadventurer.c: This is the random testing file for the Adventurer card. The purpose of this test is to determine if the deck has returned 2 Treasure cards. To perform this test properly, I randomly set the number of tests that the program is going to do. I set the minimum number of tests to 20 and the maximum to 100. After the random number is set, the program runs for that number of times inside a *for* loop and all the outcomes are recorded. This ensures that the game runs multiple times so that all the branches and statements are covered. After every run the number of treasure cards before and after are recorded which helps to determine if the test is running properly or not. There are variables called gold, silver and copper which increment and decrement according to the number of cards in the deck. This variables are then used to make sure that the function is running properly. The coverage improvement was very significant for my case because the random test achieved 100% on both statement and branch coverage compared to only 20% in unittest.

Randomtestcard1.c: This the random testing file for Smithy Card. The purpose of this card is to add 3 cards to the deck. To perform this test I again use random number generator between 20 and 100 to generate the number of times the game is supposed to run. After every run the number of cards before using the card Smithy and after using the card is recorded. This is used to make sure if the code is running properly or not. The variables hand_before and hand_after is used to compare the number of cards in hand before and after the card Smithy is drawn. The coverage improvement for this card is also very significant because I achieved between 98 and 100% of statement coverage and 100% statement coverage compared to 30% statement coverage in the unittest.

Randomtestcard2.c: This is the random testing file for Village Card. The purpose of this card is to add 1 card to the deck and perform two actions. To check if a card is added to the deck I followed the same procedure as the test for Smithy Card. The test is designed to keep track of the hand, discard cards, number of coins and number of actions before and after the card is used. This is how I know if the card is doing what it is supposed to do. I cannot compare this because I did not use this code for my unittest in Assignment 3.

Code Coverage

Randomtestadventurer.c: The code coverage for this card was 100% of statement and branch coverage. The code took less than 1 minute to run and ran about 500 times. The test covered all the codes. This is definitely an improvement from the previous unit tests.

Randomtestcard1.c: The code coverage for Smithy card was 100% branch coverage and 98% of statement coverage (fluctuates between 98 and 100%). The code took less than 1 minute to run and the code ran for about 500 times. The test covered all the codes. This is definitely an improvement from the previous unit tests.

Randomtestcard2.c: The code coverage for Village card was 100% branch coverage but the statement coverage was 98%(fluctuates between 97 and 100%). The code took less than 1 minute to run and the code ran for about 400 times. The test covered all the codes. This is definitely an improvement from the previous unit tests.

Unit vs Random:

Coverage Comparison:

	Adventurer	Smithy	Village
Random Test	Lines executed:100.00% of 65 Branches executed:100.00% of 48 Taken at least once:97.92% of 48 Calls executed:100.00% of 22	Lines executed:98.21% of 56 Branches executed:100.00% of 18 Taken at least once:94.44% of 18 Calls executed:95.00% of 20	Lines executed:97.62% of 42 Branches executed:100.00% of 12 Taken at least once:58.33% of 12 Calls executed:95.83% of 24
Unit Test	Lines executed: 20%	Lines executed: 30%	N/A

As we can see from the table above, Random Test has much better results than unit tests which is as expected. In Unit tests the parameters were given by the users and the expected test results were compared but in random testing I am using random number generators to feed different values to the functions. Through random testing we are producing inputs a user would not think to try. Random testing gives us the ability to explore the unknown.

Random Tests has better fault detection capability because of the huge range of values that are being tested.