Matthew Damiani 7/2/17

CS1632 Mr. Laboon

**Deliverable 3 (property based testing laboonify() method)**

I tried to approach this project in a methodical manner. Firstly, I knew this project involved testing a function that I wrote based off of Mr. Laboon’s specifications. This led me to start this project by first creating the laboonify file and method based off the requirements provided by Mr. Laboon. I needed to do this first because I would be unable to actually test the laboonify() method if I did not have a working/functional form of it. After that, in the same file in the main function, I included a couple of sample pre-determined arrays (the same example arrays given in the assignment description) and I printed out the sample arrays themselves as well as the array the was returned from laboonify() after passing each of the sample arrays into the laboonify() method. I did this just to show that my laboonify() method worked. As mentioned in the project description, I had to generate 100 random arrays so I wrote a helper method in the test file so that I could call that method to have 100 random arrays returned whenever I needed it for testing. I then proceeded to think of 3 properties I would expect from the array returned by laboonify(). After I decided on those 3 properties, I wrote unit tests for each of them and made sure they all passed.

The first property I tested was the size of the “laboonified” array. The laboonify() method squared each value of the array passed in, and then added the sum of the squared values to the end of the array. Therefore, I could always expect the size of the laboonified array to be 1 greater than the array passed in. The second property I decided on was testing each of the actual value in the laboonified array to make sure they were correct. For all values in the laboonified array except for the last, I would expect it to be the squared value of the value at the corresponding index in the original array. For the last value in the laboonified array, I expected it to be the sum of all the other values in the laboonified array. The third property I chose was that for all the squared values, they would be of equal or greater value than the value that was squared from the corresponding index in the original array. I was able to expect this property only because we weren’t dealing with decimal values below 1, otherwise there could’ve been a squared value that was less than its original value. But due to the range of numbers that each array value could be, I was able to expect that squaring a value would result in an equal or larger value. To check this, I checked every value in the laboonified array except for the last (since the last value didn’t involve squaring a value from the original array) and made sure that the squared value was greater than or equal to the value being squared from the original array.

The two biggest challenges for this project were coming up with the properties and dealing with 100 arrays of random sizes. I thought of the first 2 properties rather quickly but it took a little longer to realize that the squared value should always be equal or greater to the original value due to the range of possible values the assignment specified. Secondly, it took me a little while to come up with the get100RandomArrays() method so I could get the 100 random arrays whenever they were needed for testing. The hard part was coming up with how to create an array of arrays (2D array) ahead of time without knowing the size of each array ahead of time since they were randomly generated. The solution was trivial but it involved me researching Java syntax and discovering that I could initialize a 2D array of ints like: int[][] array = new int[100][] without needing to specify the second size argument. Again, it did not have to do much with testing, but rather simple Java syntax, however it was still an issue that took up a lot of my time and involved me independently researching Java syntax.

My files for this project can be found at the url below:

https://github.com/mdamiani610/CS1632\_Deliverable4.git

Screenshot of the results of the executed unit tests:

