CS 1632 - DELIVERABLE 4: PROPERTY-BASED TESTING

Michael Bowen

Summary

Writing the code for this deliverable proved to be simple enough, as the project description was very explicit about what input and output were to be expected. Writing the laboonify method was a simple process of creating a new array with a length one greater than the argument array, then filling it with the squares of the original array. While the squares are calculated (via a simple helper method), a variable keeps track of the sum of those squares and tacks that value on to the new array as the last element. The new array is then returned to the caller.

The real challenge of the project lay in switching mindsets from value-based testing to property-based testing. I struggled a bit on deciding which properties could or should be tested. I wanted the properties to be well defined but, at the same time, not so explicit as to just recreate the laboonify function. This is why I decided to not specifically test whether or not the last element in the array actually equaled the sum of the previous elements, though maybe I should have.

I eventually decided on testing four different properties. The first asserts that the result array contains no zeroes or negative numbers. This is evident by the fact that we are dealing with squares of number from 1 to 100. The second property asserts that the result array should always be one element larger than the original array. The third property asserts that the last number in the laboonify array cannot be less than any other number in either array. Finally, the fourth property tested ensures that no number returned is larger than the upper limits that are mathematically defined by the input and the act of squaring those numbers.

Writing the tests for these properties was fairly easy. I created a function that would generate an array with length 1 to 100 and populated them with random numbers from 1 to 100. I also used a constant to define the number of arrays to be tested by each test. From there, writing each test was simple as the logic involved is not very complicated. I did run into one unexpected failure, however. I had originally wrote the test for property three as the sum of the squares needing to always be explicitly greater than any value before it, forgetting that if a one element array consisting of only the number one were passed to laboonify, the resultant sum value would be equal to the previous element. This was corrected in the sumMagnitudeTest function by changing the comparison operators to “less than” instead of “less than or equal to”.

Code for the JUnit tests: github.com/mjb236/CS1632/tree/master/Deliverable%204

Screenshot

