CS 1632 – DELIVERABLE 4 – PROPERTY-BASED TESTING

Dan Kaufman

I started by writing the laboonify() method. An array called *result* is created, with a size 1 larger than the input array. A variable named *total* is initialized to zero. This will hold the sum of the squares. A FOR loop iterates through each integer in the input array. For each integer, the integer is squared, then stored in the same index in the *result* array, and the squared number is added to *total*. After the FOR loop, *total* is stored at the last index of *result*. *result* is returned.

Based on intuition, and my understanding of the code, this method should almost always work. The limitation would be integers that when squared, are larger than MAX\_INT. Since we are assuming all input integers are between 1 and 100 inclusive, this should never be a problem. The only other problem also relates to MAX\_INT, and occurs if the size of the input array is the maximum size. The function will not be able to make an array one larger than the input.

I chose three tests that I felt tested every aspect of the *laboonify* function. One test ensures that there are no additional spaces in the returned array. The output array should always be one larger than the input array, with one space for each squared term, and one space at the end for the sum of squares. There should be no additional spaces.

Another test verifies that each number was properly squared. Each number from the input array is squared and compared to the corresponding number in the output array. The third test checks the sum of squares by squaring each number as it generated and added to the input array, and adding the squared number to a total. This total is compared with the number at the last index of the output array; they should be the same.

Other tests I considered included ensuring that each output number is larger than the input, and that all output numbers are positive. I decided that these tests were already covered by the first test that checks that each number is properly squared. A properly squared number will always be positive and larger than its root. Any other tests I could think of tested outside the given assumption that the numbers and array sizes would always be between one and one hundred.

The only sort-of problem I encountered was that initially I put the laboonify method in its own class, before realizing I could just make it a part of the test class. I found no test failures.

