**0641 Test Plan Review**

**Introduction**

This document is intended to provide quality feedback to the author of the 0641 test plan, so that they could possibly use to redevelop their test plan towards a more comprehensive test.

**Analysis**

Upon first inspection the test looks very minimal which could be a good attribute, for the assignment stated to have a test plan that has a minimal testing set defined by testing principles. The first black-box test addresses the possibility of a user entering an incorrect *N* value (i.e. number of integers in the table and must be squared). It tests a correct, two incorrect, and what the reviewer believes to be a null value (in the test plan it is called <empty>). The test missed that a user could input a 0, which would be an empty table which is different from a null value. In this test case, the values in the array of values are not indicative of the full integer set (i.e. [-231, 231-1]). This could make a tester believe they have restrictions to the indicated set of [1,2, … ,5]. The test is also lacking what module we are testing. Is it the isSorted(), the sortable(), the Table class? The assignment says test the TableSorter class, but this should be explicitly detailed to any reviewer of the plan. This first test case does well in presenting the information in a readable and understandable format, which for a tester makes the process of implementing it easy. It also does well in documenting what should be expected, the purpose, and comments of assumptions made.

The second test case is used with valid table values. The purpose of this test is to evaluate the completeness of the handling of correct positive values. It is assumed that the tester will supply valid *N* values for the table. The tests are expected to have the matrices return the correct sorted array for an empty array, and positive integers. It must be noted that the test does not cover valid negative entries and how the sorting methods might handle those possibilities. It does not provide the myriad of edge cases that could occur for more than a 3 x 3 matrix, such as unsorted values on the ends of rows and columns or at the corners of a square matrix. In addition, if someone with a math background who understands sequences were to see [0, 1, 2, …, 5] would not know the range of values the array can take on, for sequences (i.e. using an ellipsis) indicates that the numbers not shown will progress in the same fashion those shown. In other words, the tester will come to believe that testing values 0 to 5 are enough for the tests, and never explore their negative counterparts nor the maximum or minimum values an integer can assume. The test case does show the manner in which to perform the test in an elegant and concise format, which allows a tester to have a very clear plan of action.

The third test case aims at addressing valid/invalid *N* values and table values. Moreover, the purpose is testing how well the methods can handle combinations of inputs, in this case the number of values and the values themselves. The tests are specifically designed to diagnose the type of exception being thrown. To make the tests a little more thorough, the test should place the values within the table at the beginning of the array of values as was done for the center and end of the array of values. Another recommendation would be to test this on a bigger table such as a 200 x 200 table. Again, this test exemplified a simple and concise test that was to the point and clearly-defined.

-End of Document-