**Black Box Test Plan – TableSorter: Review**

As the tester for Code 9914, a test plan from Project 8618 was provided. It was instructed to implement the test cases that it contained. After performing the simple test cases that were provided, it was determined that Code 9914 is not an efficient TableSorter. The Test Plan was conducted with a Tester class that contained the different Tables for the tests.

For the fist test, it was instructed to test a matrix with a dimension of 1x1 with any numeric value. The test confirmed that the methods to sort and determine if matrix is sorted, is suitable to sort a 1x1 matrix.

The second test was to input a 4x4 matrix with a 1 all throughout the matrix. After testing the matrix, it was determined that the code written for the sortable method was not able to sort the matrix or the isSorted method is not capable of detecting a sorted Table.

In the third test case, the to input a 4x4 Table of values 4, 3, 2, 1 in each row but the tests returned false for the Table when it checked to verify if the Table was sorted.

The fourth test case, a null Table was the input and like the Test Plan mentioned, the program did crash. For that issue, an exception handler can be introduced in the TableSorter implementation.

For the last test case, the liberty to input an NxN Table, meaning that the dimension was up for decision. The values for the Table are to be arbitrary. It was concluded that the TableSorter implementation was not able to sort an arbitrary NxN Table.

It cannot be determined if the issue is with the sortable method is causing issues or if it is the isSorted method that cannot determine if Table is sorted.

My recommendations for the Test Plan are that the test cases provided are good. The Code 9914 itself was not able to sort Tables and all test cases but the first failed. More variations of Table dimension sizes could be of great help since it is not specified if only small Tables are to be sorted. The test cases were good enough to determine if Code 9914 was efficient but what if an implementation can handle those test cases. Does it mean that it will be reliable with bigger Tables?

**Results**

|  |  |  |
| --- | --- | --- |
| Tests: | Expected results: | Actual results: |
| T1  { 1 } | Before sorting: true  After sorting: true | Before sorting: true  After sorting: true |
| T2  {{1, 1, 1, 1},  {1, 1, 1, 1},  {1, 1, 1, 1},  {1, 1, 1, 1}} | Before sorting: true  After sorting: true | Before sorting: false  After sorting: false |
| T3  {{4, 3, 2, 1},  {4, 3, 2, 1},  {4, 3, 2, 1},  {4, 3, 2, 1}} | Before sorting: false  After sorting: true | Before sorting: false  After sorting: false |
| T4  null | Before sorting: error | Before: error |
| T5  {{x, x, x, …n},  {x, x, x, …n},  {x, x, x, …n},  {n, n, n, …n}}  x=any arbitrary value  n=any finite value | Before sorting: false  After sorting: true | Before sorting: false  After sorting: false |