Test Plan: TableSorter

The purpose of this test is to validate the TableSorter class. The TableSorter is to sort an NxN table. The class will also have a method that will verify if a table is sorted after an NxN table is the input and the output is a sorted table. You are to implement test cases and execute them to verify if this class fulfills the requirements. The TableSorter class is written in Java. A Table class is used to hold the tables, the following is a description of the Table class:

Class Table holds an NxN table of integers. The Table interface has 4 methods:

static Table getTable (String filename) returns an instance of table initialized

with values from the file filename

public int getTableValue(row, column) returns the integer at (row, column)

public void setTableValue(int row, int column, int value) sets the value of the

table at row, column to value

getSize() returns N

Requirements for TableSorter:

Create a Java class TableSorter with two instance methods:

public boolean isSorted(Table t) returns true if every row and every column of

the table t is sorted in ascending order

public static void sortable(Table t) sorts a Table so that isSorted() is true.

Test Cases:

**Verify:** that isSorted(Table t) returns true if every row and every column of the table t is sorted in ascending order.  
**Using:** multiple inputs of NxN Tables containing numbers in ascending order. Test small Table deimentsion and test big Table dimensions. Example, small table 5x5, big tables 1000x1000 etc.

Example (use different Table sizes):

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 4 | …n |
| 2 | 3 | 5 | …n |
| 3 | 5 | 7 | …n |
| …n | …n | …n | …n |

|  |  |  |
| --- | --- | --- |
| 1 | 2 | …n |
| 2 | 3 | …n |
| …n | …n | …n |

**Verify:** that isSorted(Table t) returns true if every row and every column of the table t is sorted in ascending order.  
**Using:** an NxN Table with the same value all through the Table.

Example (use different Table sizes):

k = any arbitrary value that will populate table

|  |  |  |  |
| --- | --- | --- | --- |
| k | k | k | …n |
| k | k | k | …n |
| k | k | k | …n |
| …n | …n | …n | …n |

**Verify:** that isSorted(Table t) returns false if every row and every column of the table t is not sorted in ascending order.  
**Using:** an NxN Table with random values through the Table.

Example (use different Table sizes):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 13 | 4 | 9 | 77 | …n |
| 3 | 11 | 37 | 45 | …n |
| 6 | 1 | 0 | 11 | …n |
| 8 | 9 | 13 | 2 | …n |
| …n | …n | …n | …n | …n |

**Verify**: that sortable(Table t) sorts a Table so that isSorted() is true.

**Using**: multiple inputs of NxN Tables with sorted numbers in ascending order.

Example (use different Table sizes):

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 4 | …n |
| 2 | 3 | 5 | …n |
| 3 | 5 | 7 | …n |
| …n | …n | …n | …n |

**Verify**: that sortable(Table t) sorts a Table so that isSorted() is true.

**Using**: multiple inputs of NxN Tables with sorted numbers in ascending order except one value that is out of order.

Example (use different Table sizes):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 77 | 4 | 6 | …n |
| 2 | 3 | 5 | 8 | …n |
| 3 | 5 | 7 | 11 | …n |
| 5 | 9 | 12 | 20 | …n |
| …n | …n | …n | …n | …n |

**Verify**: that sortable(Table t) sorts a Table so that isSorted() is true.

**Using**: multiple inputs of NxN Tables with random numbers.

Example (use different Table sizes):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 13 | 4 | 9 | 77 | …n |
| 3 | 11 | 37 | 45 | …n |
| 6 | 1 | 0 | 11 | …n |
| 8 | 9 | 13 | 2 | …n |
| …n | …n | …n | …n | …n |

**Verify**: that sortable(Table t) sorts a Table so that isSorted() is true.

**Using**: multiple inputs of NxN Tables with numbers in descending order.

Example (use different Table sizes):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 50 | 35 | 17 | 11 | …n |
| 47 | 33 | 15 | 10 | …n |
| 39 | 29 | 11 | 7 | …n |
| 33 | 21 | 7 | 2 | …n |
| …n | …n | …n | …n | …n |

**Verify**: that sortable(Table t) sorts a Table that is empty and that program will not halt.

**Using**: multiple inputs of NxN Tables with no values.

Example (use different Table sizes):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Null | Null | Null | Null | …n |
| Null | Null | Null | Null | …n |
| Null | Null | Null | Null | …n |
| Null | Null | Null | Null | …n |
| …n | …n | …n | …n | …n |