Test case

Table Sorter function  
<<author name redacted>>  
Software Testing and Integration

Black Box test Case:



Black Box testing should be based solely on the Software Requirements and Specifications, no need to be able to see what’s inside the methods, loops, conditionals of the source code. Thus the requirements and specifications of the Sorter Table method will be defined as follows.

Requirements and Specifications attained from Programming Assignment 1 by Dr. Roach:

1. The method TableSorter shall get an object of type Table.
2. The method TableSorter shall return the Table object with the array sorted row-wise, and column-wise in ascending order.
3. Any Table object from considere as input for the TableSorter method shall contain an NxN (squared 2D array) of integers.
4. Values from the 2D array can be obtained by using the method getTableValue(row,column) which the return the value at the given row and column indices:
   1. row values are row indices of the 2D array
   2. column values are column indices of the 2D array
5. The method setValue(row,column,value) sets an integer value in the row and column given as inputs.
6. The 2D array must be square. Meaning row size must equal column size.
7. The method isSorted shall get a Table object and return true if the 2D array of this object is sorted row-wise and column-wise in ascending order.

Test Case:

1. The method ‘isSorted’ must be validated first to ensure the method does return correct outputs.
   1. Create a txt file that contains a 3x3 array with the following values:  
      
   2. Instantiate a Table object and store the array in the object.
   3. Test the isSorted method by passing the Table object as input and observe the output by printing it. Expected output must be ‘true’.
   4. Perform the same procedure from step ‘a’ to ‘c’ with several more examples in which we expect ‘true’ as output.
      1. Use negative numbers
      2. Use a combination of negative and positive numbers
   5. Perform then the same procedure with cases in which expected outputs are ‘false’.
   6. Create an Excel Table, have several inputs, a column which represents your expected output, and another column representing the actual output.
   7. Document and mark all mismatches. Run at least 10 different inputs. Example of table:

|  |  |  |  |
| --- | --- | --- | --- |
| inputs | Expected output | Actual output | Match |
| 1 | TRUE | TRUE | Y |
| 2 | TRUE | TRUE | Y |
| 3 | TRUE | FALSE | N |
| 4 | TRUE | TRUE | Y |
| 5 | TRUE | TRUE | Y |
| 6 | FALSE | FALSE | Y |
| 7 | FALSE | FALSE | Y |
| 8 | FALSE | TRUE | N |
| 9 | FALSE | FALSE | Y |
| 10 | FALSE | FALSE | Y |

1. If there is no mismatches between all desire outputs vs actual outputs, proceed with the following step. Otherwise, the test has been set to fail.
2. Once isSorted method is validated, we can use it for validation of the SorterTable method.
   1. Create a Print() method which prints the 2D array to observe the output after sorting is implemented.
   2. The Test case will be divided into two different sub test cases. First test case will be performed by passing ‘unsorted arrays’ and second test case will be performed by passing ‘sorted arrays’ as inputs. Document every test case based on the given input.
   3. Sorted Arrays test case:
      1. Create various 2D sorted arrays (can be the same inputs for testing the isSorted method).
      2. Implement the SorterTable method and print the output. The expected output must be the same. Meaning that we must not see change between the input and the output.
      3. Document every mismatch. If there is at least one mismatch, the test is set to fail.
   4. Unsorted Arrays test case:
      1. Create various 2D unsorted arrays.
      2. Implement the SorterTable method and print the output.
      3. Use the isSorted method to test its correlation.
      4. Document the inputs, expected outputs, actual outputs from isSorted table, and record its match or mismatch.
      5. Based on that, we can determine the test to pass or fail.