Software 1301 was tested using test plan 0641. The test plan did not explicitly say how to enter N and values to test the software. We assumed the test plan was written not to test the assignment by using values in a file, but assuming by creating a table directly with Table(int N, int[] values) since N and values were being tested.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test: ***Black Box #1*** | | | Current Status: ***In progress*** | | |
| Test title: Invalid N input. | | | | | |
| Testing approach: This test consists of four (4) separate steps with different input variations, to help determine how the program responds to invalid ***and*** valid inputs. Test will cover invalid N inputs, such as: (1) Perfect square input, (2) None perfect square input, (3) Empty input and (4) Negative input.  This test is categorized as a ***Black Box (Boundary Value Analysis)*** testing. | | | | | |
| STEP  1.1  1.2  1.3  1.4 | OPERATOR ACTION  ***N =*** 9  ***Values =*** [1, 2,…, 5]  ***N =*** 5  ***Values =*** [1, 2,…, 5]  ***N =*** <*empty*>  ***Values =*** [1, 2,…, 5]  ***N =*** -9  ***Values =*** [1, 2,…, 5] | PURPOSE  To verify that the program will catch and throw an exception if the N input is invalid. | | EXEPCTED RESULTS  *Valid*: The test will go through, and the program will proceed accordingly. Output will be a Table.  *Invalid*: An exception will be thrown stating the N input is invalid. Program will stop. There will be no output. | COMMENTS  Step(s) 1.1 area *Valid.*  Step(s) 1.2, 1.3 and 1.4 are *Invalid.*  N inputs are assuming the Values [] input is correct. |
| Testing Notes: 1.1 gives an error, the comment that 1.1 should be valid is incorrect.  1.2 prints “ERROR!: Input must be square.”  1.3 will not compile. Test cannot be conducted to make an integer value empty and compile.  1.4 throws exception and prints “ERROR!: Input must be square.” | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test: ***Black Box #2*** | | | Current Status: ***In progress*** | | |
| Test title: Valid Values [] input. | | | | | |
| Testing approach: This test consists of three (3) separate steps with different variations, to help determine how the program responds to invalid ***and*** valid inputs. Test will cover valid Values [] inputs, such as: (1) Sorted array, (2) None sorted array and (3) Empty array.  This test is categorized as a ***Black Box (Boundary Value Analysis)*** testing. | | | | | |
| STEP  1.1  1.2  1.3 | OPERATOR ACTION  ***N =*** 9  ***Values =*** [0, 1, 2,…, 5]  ***N =*** 9  ***Values =*** [1, 4, 3, 2, 5, 0]  ***N =*** 9  ***Values =*** [*empty*] | PURPOSE  To verify that the program will be able to handle multiple types of the standard Values [] input. | | EXEPCTED RESULTS  Step(s) 1.1 and 1.2 should both output a sorted 3x3 table containing the input in ascending order.  Step(s) 1.3 should output an empty 3x3 table, since no integers are present in the Values []. | COMMENTS  Values [] inputs are assuming the N input is valid. |
| Testing Notes: 1.1 error -n input in test is incorrect  1.2 error -n input in test is incorrect  1.3 error –test incorrect -there is no specification in the assignment that it will output an empty 3 X 3 | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test: ***Black Box #3*** | | | Current Status: ***In progress*** | | |
| Test title: Valid/Invalid N & Values [] inputs. | | | | | |
| Testing approach: This test consists of four (4) separate steps with different input variations to help determine how the program responds to invalid ***and*** valid inputs. Test will cover (1) Invalid N / Invalid Values [], (2) Invalid N / Valid Values [], (3) Valid N / Invalid Values [] and (4) Valid N / Valid Values [].  This test is categorized as a ***Black Box*** testing. | | | | | |
| STEP  1.1  1.2  1.3  1.4 | OPERATOR ACTION  ***N =*** -2  ***Values =*** [1, 3, a, 6, b, @]  ***N =*** 3  ***Values =*** [1, 4, 3, 6, 7, 8]  ***N =*** 9  ***Values =*** [1, 3, a, 6, b, @]  ***N =*** 9  ***Values =*** [1, 4, 3, 6, 7, 8] | PURPOSE  To verify that the program will be able to handle and catch multiple types of the standard N and Values [] inputs. | | EXEPCTED RESULTS  Step 1.1: The program will stop and throw an exception on the invalid N and/or invalid Values [] input.  Step 1.2: The program will stop and throw an exception on ***only*** the invalid N input.  Step 1.3: The program will stop and throw an exception on ***only*** the invalid Values [] input.  Step 1.4: The program will run and output an ascending ordered table, containing the values from the Values []. | COMMENTS  Step(s) 1.2 and 1.4 are the basic requirements the program should handle.  Step(s) 1.1 and 1.3 are extra requirements the program is ***recommended*** to handle. To fully test all inputs and variations. |
| Testing Notes:1.1 program will not compile with values  1.2 Prints “ERROR!: Input must be square.”  1.3 program will not compile with values  1.4 error | | | | | |

There may be some confusion in how the test plan is meant to be implemented, or by the author of the test plan when it was written. Most all of the tests had a different actual output vs the expected output, but the test cases do not make sense to us.

We continued to test the software using values read from a file like the program is meant to be ran. These include the following quick test inputs followed by the result after the “->” symbol. The parenthesis mention what they are testing.

1 / 2 3 -> error (ascii near 0-9)

1 : 2 3 -> error (ascii near 0-9)

1 2.1 3 4 -> error (double value)

1 “” 2 3 -> error (null value)

-2147483648 1 2 2147483647 -> true (integer BVA and already sorted)

-2147483649 1 2 3 -> error (integer BVA)

1 2 3 2147483648 -> error (integer BVA)

1 -> true (1 integer matrix)

-3 -4 1 1 2 3 4 5 6 -> false (non sorted row)

-1 2 3 4 -2 5 6 7 8 -> false (non sorted col)

1 then sort-> true (how handles sorting 1 val)

-3 -4 1 1 2 3 4 5 6 -> false (completely unsorted) then sort -> true

When running the tests we added above, the program behaves as expected. Due to the confusion of the test plan and the results of our tests we included, we conclude that the program behaves as expected.