Jason Harkness

Project 1 Test Plan/Approach

10/10/2017

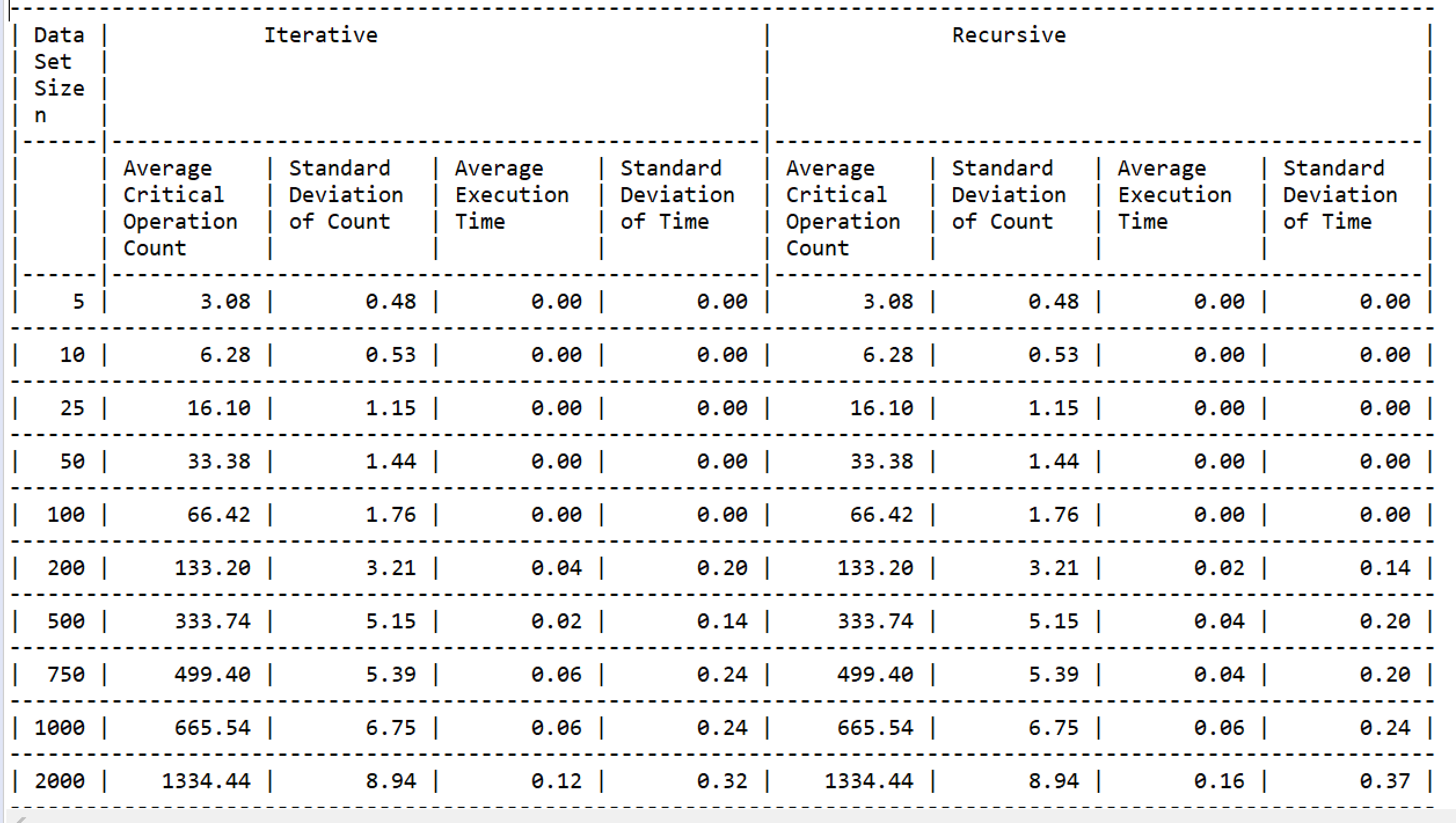
I chose to use Quicksort as my sorting algorithm to implement and benchmark for project 1. Both iterative and recursive versions of the algorithm were used for benchmarking. My project conformed to the design given in the project rubric including a class to implement the main() method, a class for implementing benchmarking, a class for accomplishing the logic for iterative and recursive quicksort, an interface, and a custom unsorted exception class. I set the counter for each time the pivot point is changed in the quicksort algorithm for benchmarking purposes. The BenchmarkSorts class was created to implement the quicksort algorithms and to calculate the average and standard deviation of time and the chosen critical count operation. Also, in the BenchmarkSorts class, a displayReports() method was used to print out the required data to the user. If the input data was not sorted correctly a unsortedexception notified the user.

Test Cases

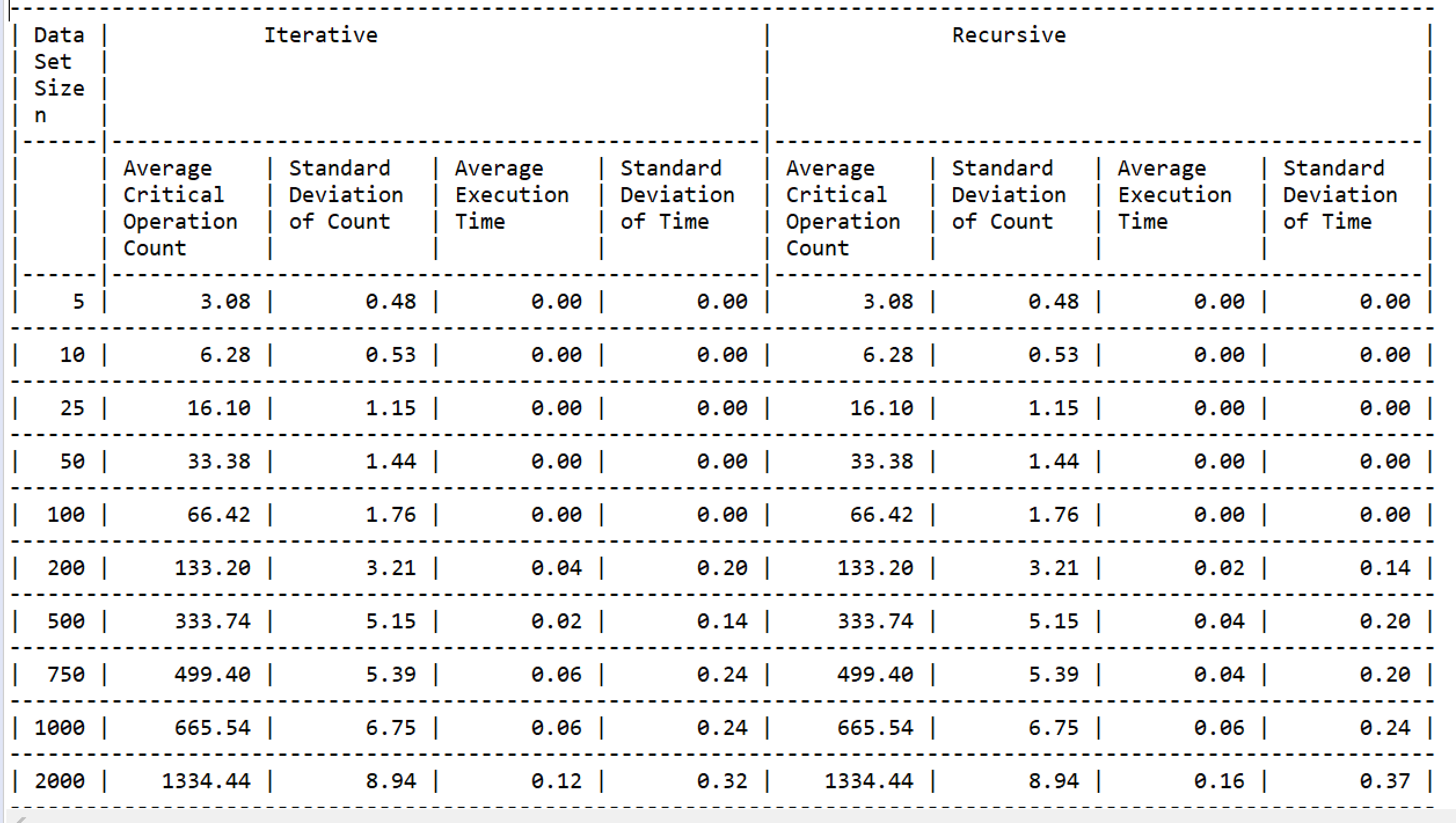
1.) Input: Randomized data set with size 5 in this case

[12504, 9008, 14019, 750, 1442]

Expected output: [750, 1442, 9008, 12504, 14019]



Actual Ouptut: [750, 1442, 9008, 12504, 14019]



2.) For the second test case I placed a bug in the code as shown below to show the error checking through the exception class working:

**int** partition (**int** arr[], **int** l, **int** h)

{

**int** x = arr[h];

**int** i = (l - 1);

**for** (**int** j = l; j <= h- 1; j++)

{

**if** (arr[j] <= x)

{

i++;

// swap arr[i] and arr[j]

swap(arr,i,j);

}

}

// swap arr[i+1] and arr[h]

swap(arr,i+1,h);

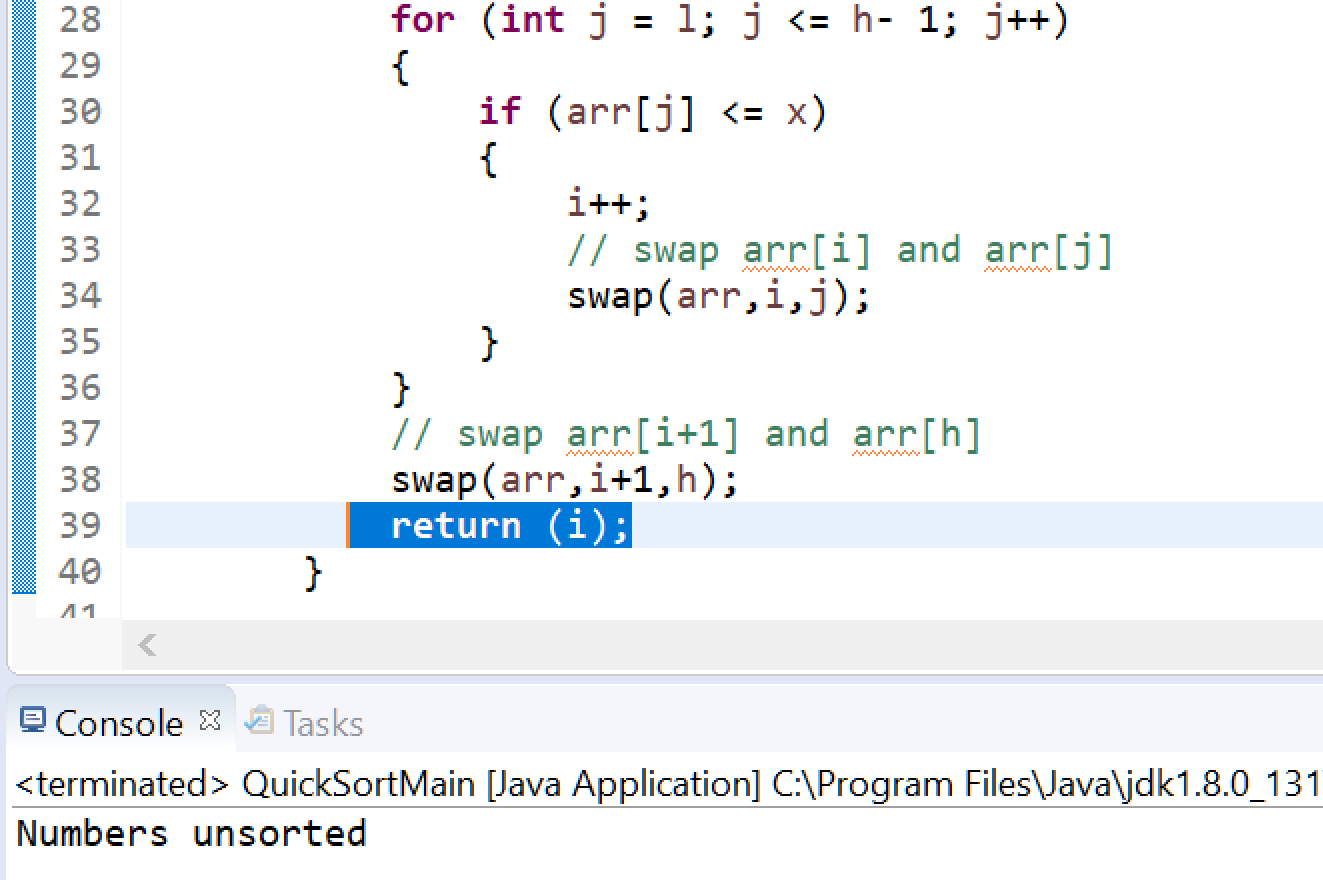
**return** (i); - Removed the + 1 at this line to return wrong integer

}

Input: Randomized data set

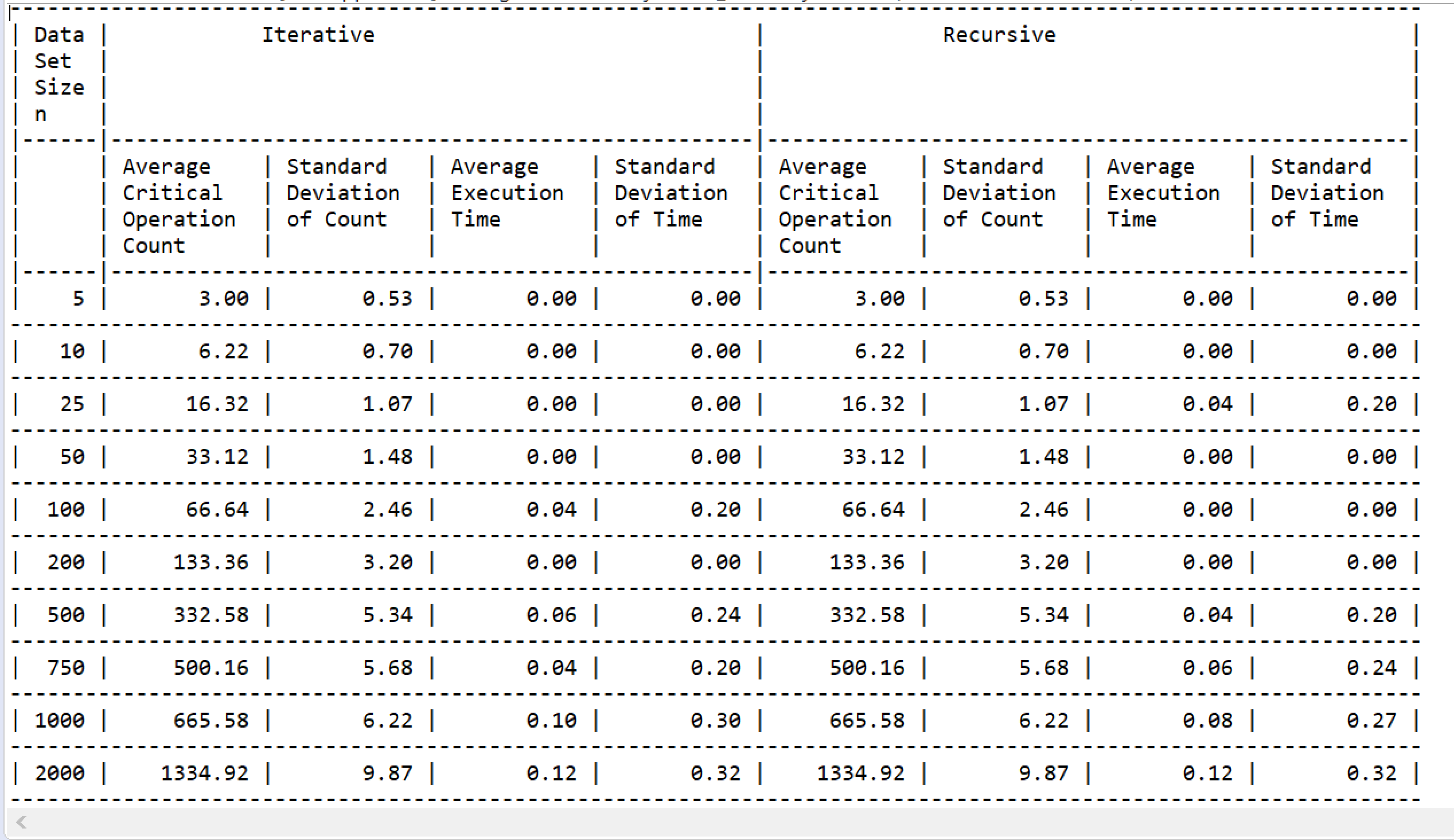
Expected Output: “Numbers Unsorted”

Actual Output:

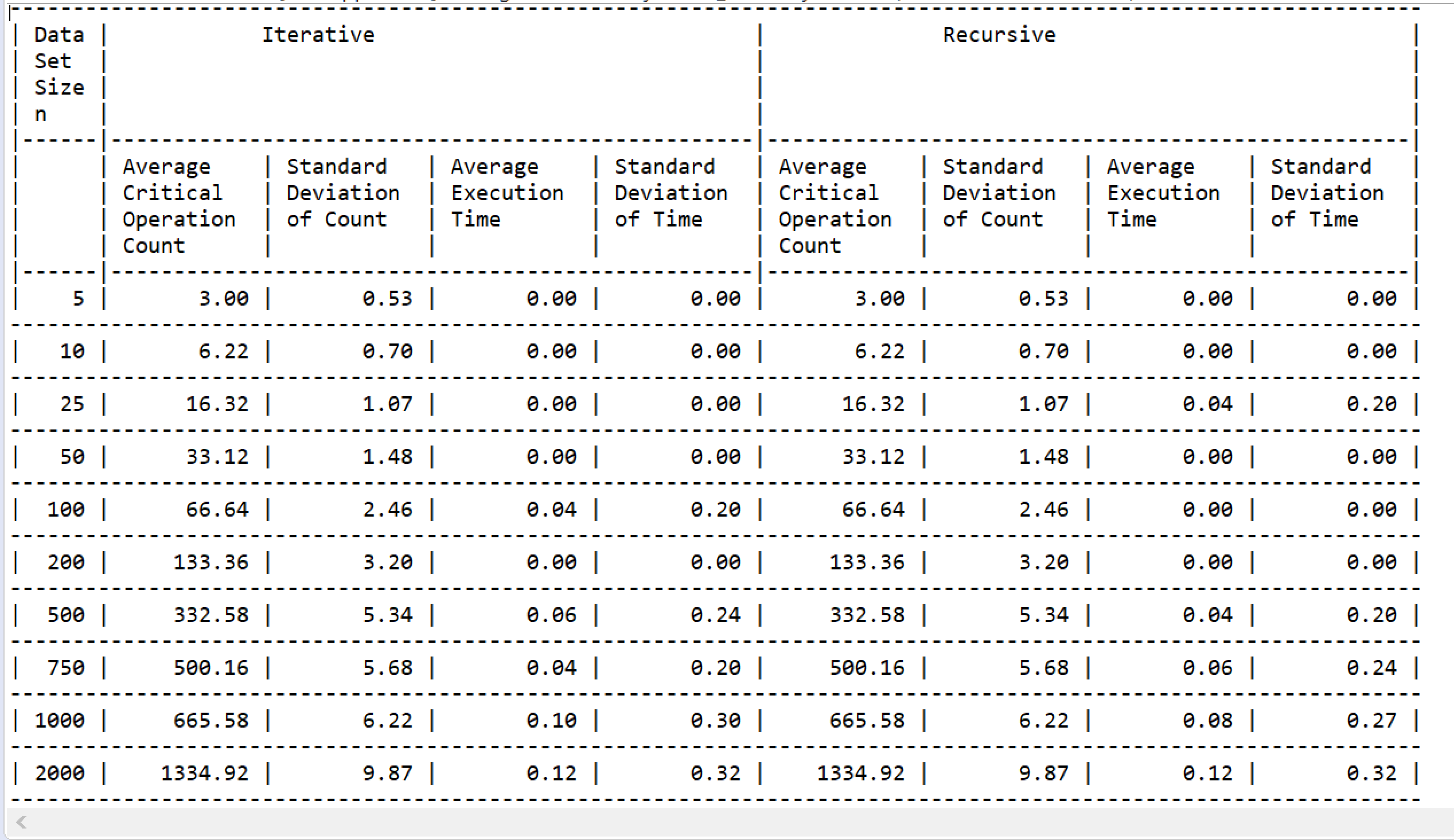


3.) Input: 10 different sizes of data sets { 5, 10, 25, 50, 100, 200, 500, 750, 1000, 2000 }

Expected output: Display showing proper benchmarking data for all data sets



Actual Output below:



Lessons Learned:

Until taking this class I had never used quicksort and reading through the weekly material and learning that in many cases it is the fastest of the sorting algorithms made me interested to choose it for this project. It took me awhile to understand the concept of focusing on a pivot and rearranging the numbers around the pivot until they are in order. Another challenging concept was implementing standard deviation. Looking on line for how to calculate it and implement it from a coding standpoint took me awhile.

Other lessons learned:

* Utilizing interfaces
* Custom exceptions

References:

<http://www.geeksforgeeks.org/quick-sort/>

<http://www.geeksforgeeks.org/iterative-quick-sort/>

<http://www.chem.tamu.edu/class/fyp/keeney/stddev.pdf>