Christopher Eardensohn

Om Kanwar

December 20, 2017

Assignment 7 Reflection

Testing approach for radix sort was fairly straightforward, after creating the pair of functions, one to go through each digit and one to sort by value in buckets, we used a loop through the array of integers to check if any values were greater than the one before it to make sure it was in ascending value order. The code contains an error message of the index where a function is out of order. We tested the option of printing all integers but it is unnecessary to keep printing all one million integers each run time after seeing the results once.

Testing approach for the animate quick sort included a series of implement one task, test, then add. This series started with creating the integers and displaying them and the index positions to the screen, followed by adjustments of positioning and creating borders for each integer value so it clearly shows each number separated from another. Next implementation was the reset button which was tested by making sure the display cleared and randomized a new set of integers with the indexes reset. Last implementation was the step button which was tested by making sure the pivot was swapped with low and high index conditions met, then making sure each index moved as it should by following the logic of quick sort with each click of step. After several reset and step throughs and checking our if blocks for accuracy, we were able to successfully partition a random array of integers for one execution of quick sort.

The two parts for the assignment went very well. Radix sort was smooth with no issues in implementation or testing. However, some problems did come with animate quick sort. The largest problem was taking the logic from the quick sort functions provided by the book and breaking it down in to if-else statements so that each click of step would do one action. As we broke it down further into smaller cases for what each condition should do it became clear how to step through the sort function. Other issues for the animation were how to present nodes in center for the integers. At first, we used a stacked HBox but could not control the position of the index text as easy as we wanted. Instead, we chose a Grid Pane to hold our integers and index text and it instantly became much easier to track positioning and swapping of text and integers by using the indexes of rows and columns on the grid. Overall, we learned there's multiple ways that javafx could be handled and trying to force one style on a program is not always the best solution.