Eric Thomas
9/4/2018
CSC 345
Project 1

Intro:

For my debug strategy I focused primarily on visual guides. This includes a label stating what is being done and a printout of the values being modified in some way. I found this to be most useful to me to trace what was happening throughout the sorting process. I didn't want too much information being printed on the console where it would be overwhelming to attempt to understand what was actually happening. I believed that sticking to a minimalistic approach would be best for me personally. I also included a begin and end statement to each algorithm to clearly see when it starts and finishes. This became very useful in merge sort and quick sort when multiple algorithms were being used.

Bubble Sort:

For my bubble sorting algorithm, I chose to display three possible debug statements depending on what was happening. Those three options are, no-swap, pre-swap, and post-swap. I felt like these three cases covered what I needed for debugging purposes. In each printout I printed the entire array with the two values being compared highlighted by asterisks and each value enclosed in brackets. The array would be printed on each iteration of the loop, so I could see the comparisons happening as I go down the list. The asterisks also gave a nice visual indicator for the overall process of the sort as it was being completed.

Insertion Sort:

For my insertion sorting algorithm, I chose to display three possible debug statements again. This time they were, current state, after insert, and comparing. The comparing debug statement is identical to the bubble sort one previously mentioned. The current state prints the state of the array between processes. It is the entire array separated by a decimal point representing the two logical arrays contained within the data structure. This allows me to see the array being sorted one value at a time to make sure it is being moved logically. The last option is the after-insert statement. This is the entire array printed again with the value just inserted contained within brackets. It also includes the decimal point that serves as the logical separator for the two parts of the array. This makes it easy to verify that the new value was inserted into the correct location on the left side of the decimal point.

Merge Sort:

For my merge sort algorithm, I once again chose three options for this debug strategy. They are midpoint, split, and sort half. Technically, this sorting algorithm also includes the debug statements from the selection sort algorithm which I will describe after quick sort. The midpoint statement is simply a printout of the midpoint value the merge will be splitting at. This is mainly to identify where the split took place exactly. The split statement is the main debug statement in this because it shows the values split into two halves by curly braces. Since this algorithm is recursive it does this repeatedly starting with the left half each time. So, following the statements you can see each half being broken up into two halves based on the midpoint. Once it gets at or below the baseLen requirement it prints out the half that will be sorted by selection sort, this is preceded by a label saying sort half. After this, the debug statements switch over to those in the selection sort until that individual half has been sorted. Once complete, it returns to the merge sort to do the other half.

Eric Thomas
9/4/2018
CSC 345
Project 1

Quick Sort:

For my quick sort algorithm, I stuck with my rule of three statements as I felt that it was telling me enough information about what was happening. They are, pivot, after partition, and sort partition. The pivot simply prints the entire array with the pivot value that was chosen encased in brackets to make it easier to see in relation to the rest of the array. After partition also prints the entire array but this time it separates the array into two parts, each separated by curly braces. These two parts are decided based on the pivot value. Once the parts have been partitioned enough ("baseLen"), the sort partition statement prints out just the part of the array that will be sorted. Again, selection sort takes over from that point to print out what is happening. Selection sort ends with a statement that shows the array that was passed to it fully sorted. Together, this lets me see that the partition I sent to selection sort was sorted successfully and it continues on to the other partitions in the original array. I will go into selection sort more in the next part.

Selection Sort:

The time has come to talk about the last algorithm, selection sort. For this one I broke tradition and only needed two debug statements to describe what was happening during the sorting process. Those two are, current and smallest. Current prints the entire array with the index that will be switched in brackets. This lets me see which value should be changed and I can also see the progression of the sorting process. The smallest statement prints out the entire array as well but highlights the smallest value in respect to the index about to be switched, this value is encased in curly braces. Each time it finds a value smaller it prints another statement and highlights just that one value that is smallest.