CS361 Algorithm Lab 1

What to do

1. Read the test data from the data file provided into an array of integers. The file contains 10,000,000 integers, one line per integer. If you read an empty line, then skip it. The sum of these integers is 49,999,995,000,000. Make sure you are using this information to verify the accuracy of your input routine.
2. Write a recursive method auxMergeSort that takes three parameters: the array, the startIndex, and the endIndex and sorts the elements between the startIndex and endIndex using Merge Sort. Consider coding the Merge part in a separate method that does not need to be recursive.
3. Write a recursive method auxQuickSort that takes three parameters: the array, the startIndex, and the endIndex and sorts the elements between the startIndex and endIndex using Quick Sort with the pivot to be the average of the values at startIndex, endIndex, and the middle element between startIndex and endIndex using m = (startIndex + endIndex)/2. Consider coding the splitting part in a separate method that does not need to be recursive.
4. Write a ***recursive*** method flgIsSorted to check if a given array (provided as a parameter) is sorted in increasing order. The method returns true if and only if the array is sorted in increasing order. Hint, when the array has only one element, it is sorted. If the first half is sorted, the second half is sorted, and the first element of the second half is not smaller than the last element in the first half, the array is sorted. Your initial method can only take one parameter – the array. That method can call another auxiliary method that takes other parameters.
5. Look into System.nanoTime() (or an equivalent); to time the number of nanoseconds needed to perform the above four methods. Now change your code so that, rather than perform all these steps on the 10 million integers, starts with 1,000 and increases at 10x until it reads more than 10 million numbers. Make sure to check whether the array is sorted using your flgIsSorted method. Run your code 3 times, record the execution time in milliseconds for each run on each size, enter the milliseconds reading into an Excel spreadsheet, calculate the average execution time in milliseconds for each run on each size and display your results in both a table and as a line chart. I am expecting to see a chart with two lines. Clearly indicate which line is which algorithm. Also on your output show the result of using your flgIsSorted to check whether the array is actually sorted. Show the screen dump indicating your array is actually sorted and the time it takes for each run.

Code:

You are expected to update (or push) your code on your GitHub repository five times during the two weeks (Week 1 Monday to Week 2 Sunday) you are working on your lab. Each update should show reasonable progress and be accompanied by a brief (but clear) explanation of the changes you have made to your code. These updates are graded, so make sure you are keeping up and not waiting until the last minute. The deadlines for these five updates are:

1. Week 1 Wednesday
2. Week 1 Saturday
3. Week 2 Tuesday
4. Week 2 Friday
5. Week 2 Sunday – This is your final update, so your code should be complete

What to turn in to Moodle:

You will turn in a ONE PDF file lab report. This lab report must have the following components:

- Code segments each task listed above. These code segments must include comments. Make sure to thoroughly comment your code.
- Screen dumps from your output for each part. It is ok if each screen dump doesn't include 10,000,000 integers, but you should thoroughly convince me that your code is working.
- Explaining thoroughly how you accomplished each task above, including citations. Did you use pseudocode from the book? Did you get support from a website (cite your source)? Did you work with a peer? Also include any stumbling blocks along the way.
- Your Excel charts and graphs from part 5, along with a written analysis of your results.