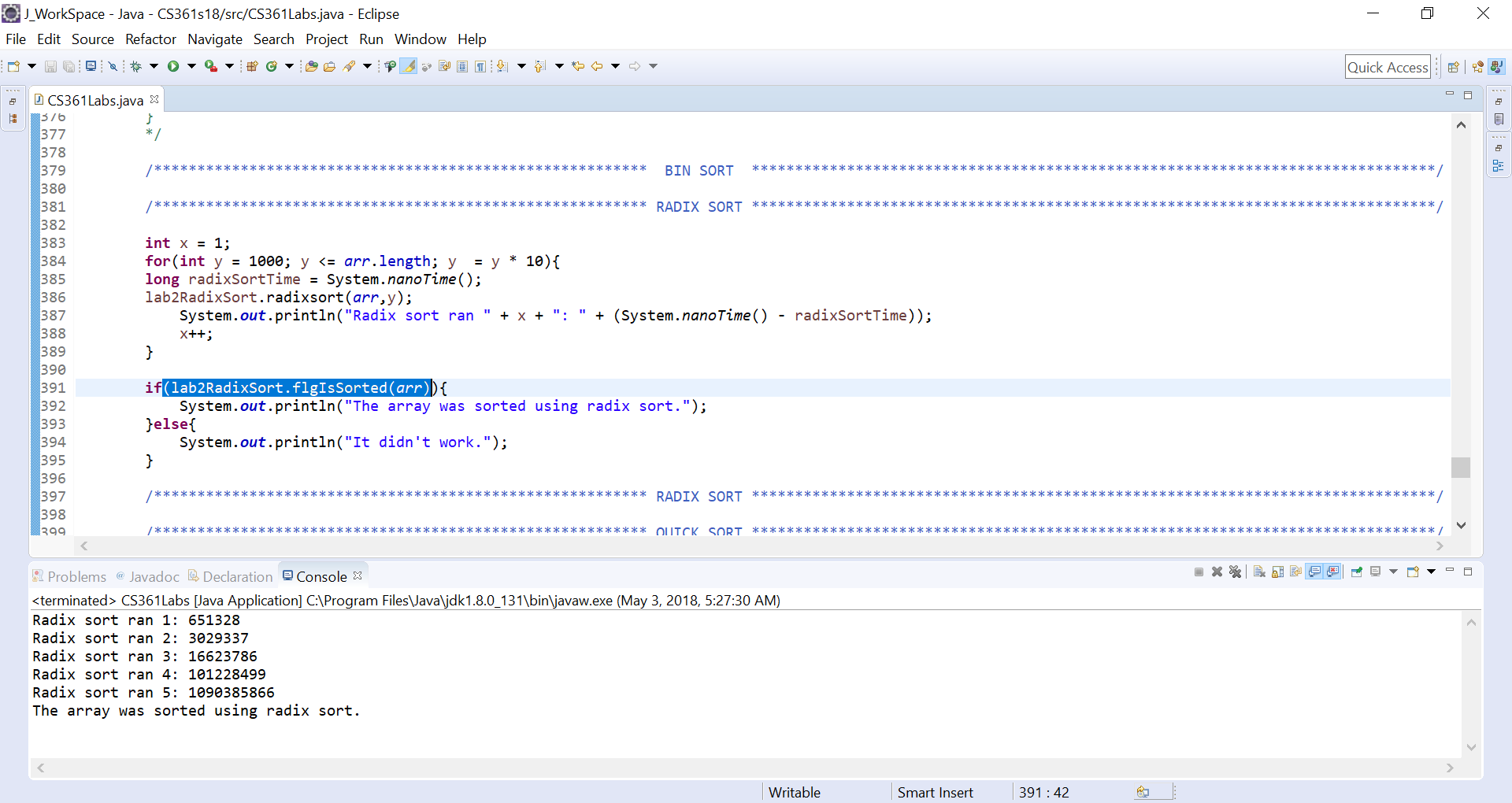
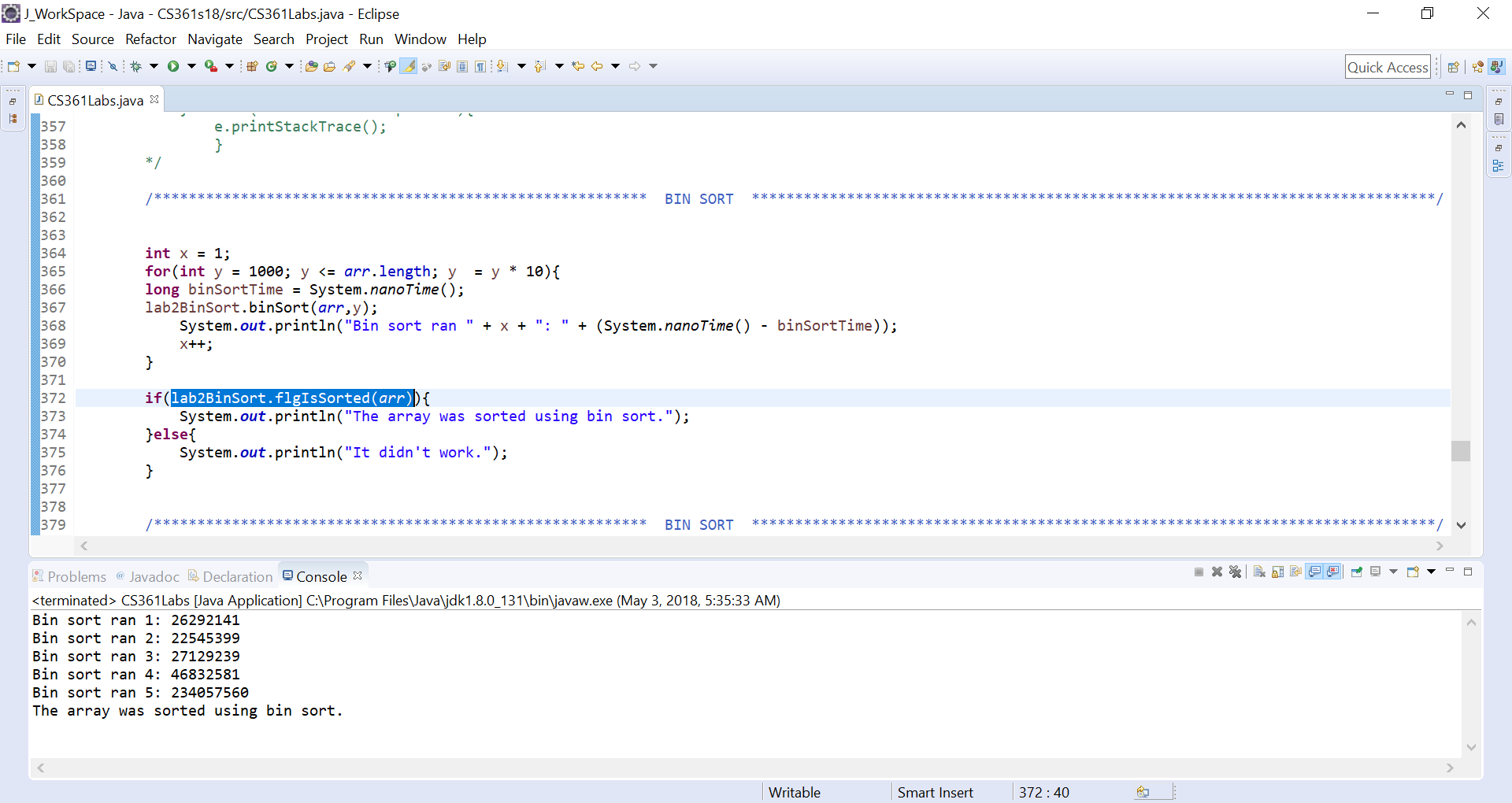
# CS361 Algorithm Lab 2

# What to do

1. Implement the Radix sort algorithm and use it to sort roughly 10,000,000 numbers. I am providing a new data file (on Moodle).
2. Implement the Bin sort algorithm and use it to sort roughly 10,000,000 numbers.
3. Make sure the results are sorted for 1 and 2. Show the screen dump indicate the sorting algorithms are actually sorting correctly.

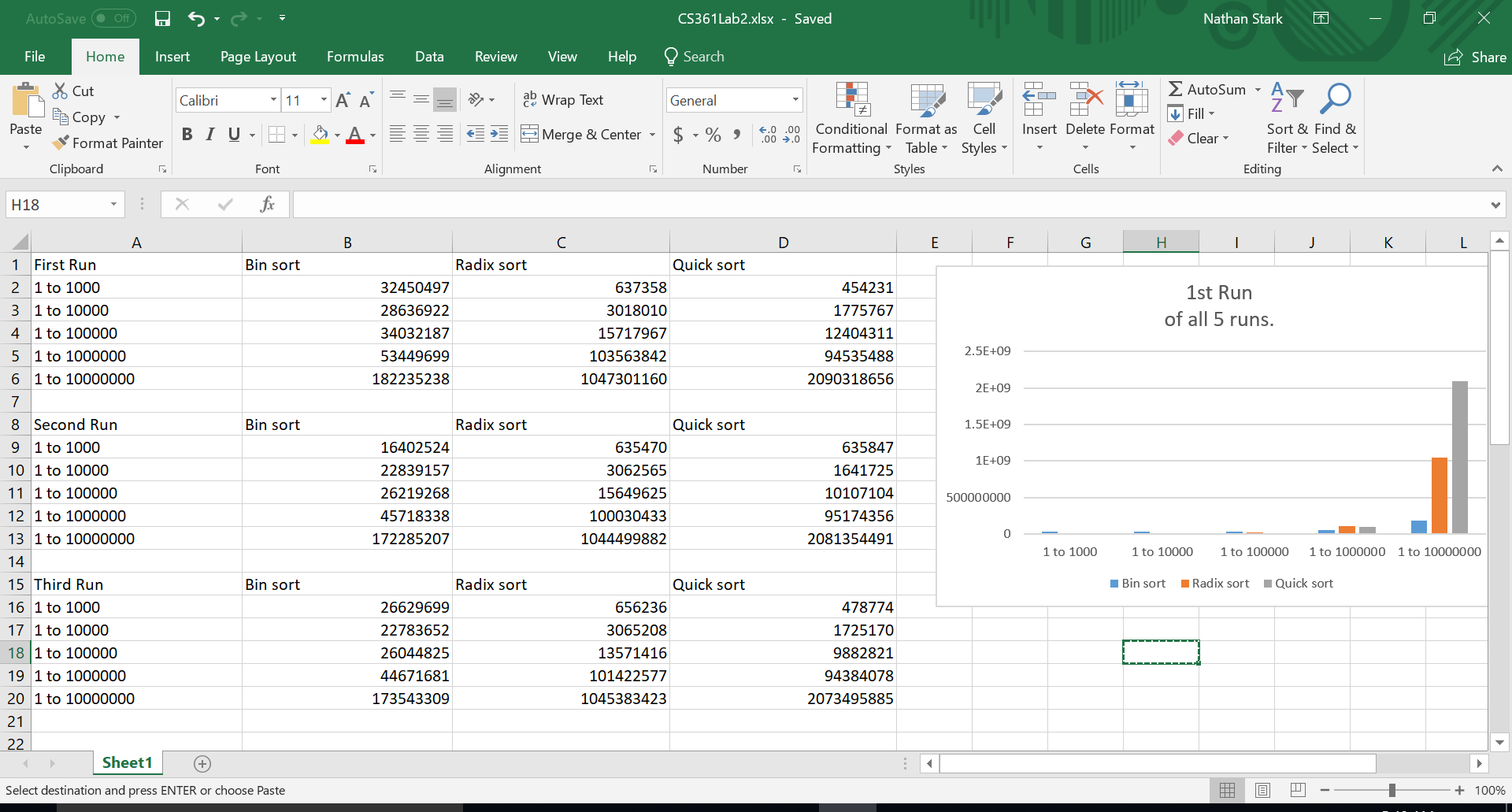


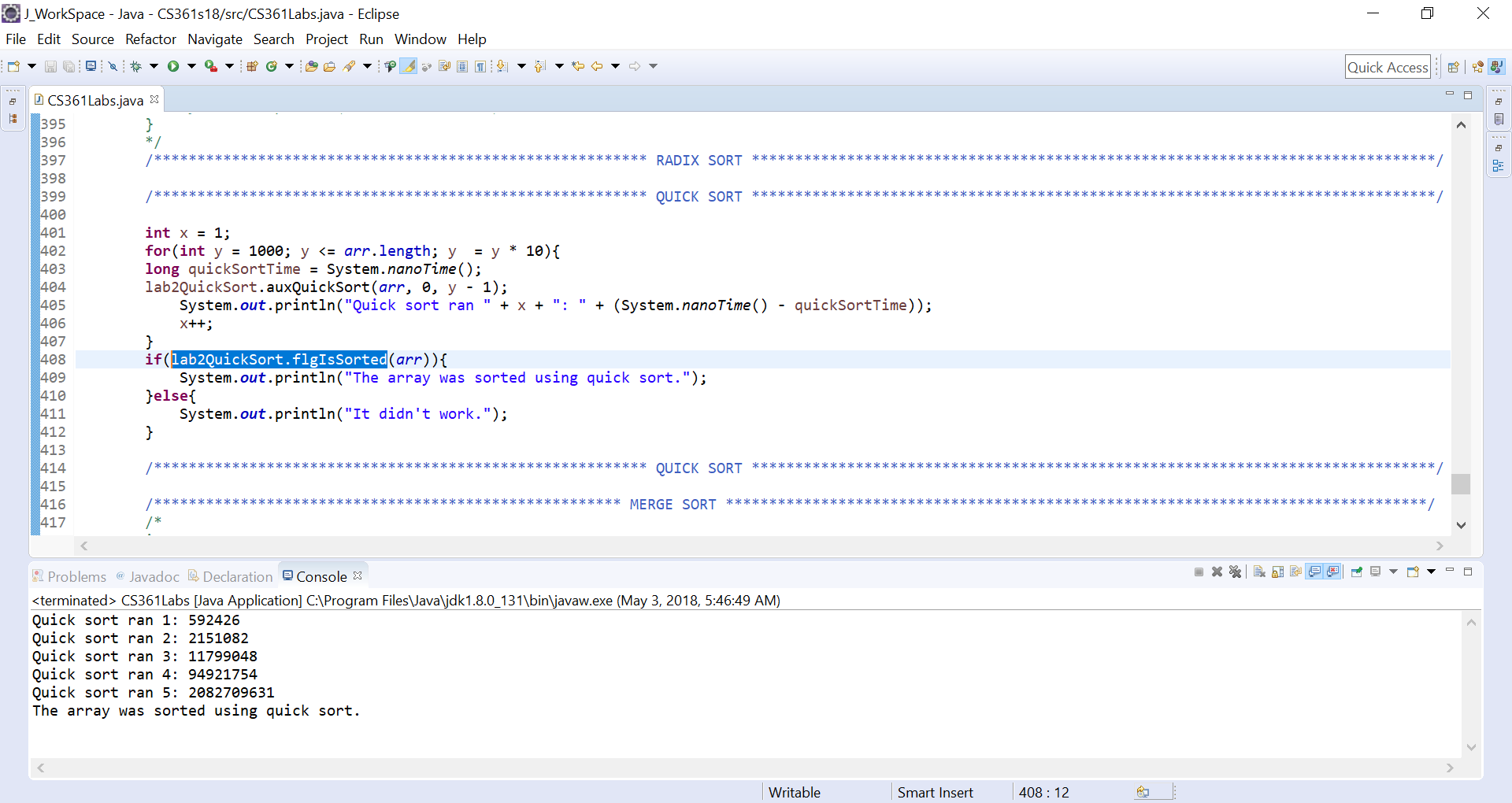
**As we can see the flgIsSorted() method is use in an if statement so that the statement “The array was sorted using radix sort.” Will print to the console only if the array is sorted. Otherwise the statement “It didn’t work.” Prints to the console. As we can see the proper statement is printed.**



**As we can see the flgIsSorted() method is use in an if statement so that the statement “The array was sorted using bin sort.” Will print to the console only if the array is sorted. Otherwise the statement “It didn’t work.” Prints to the console. As we can see the proper statement is printed.**

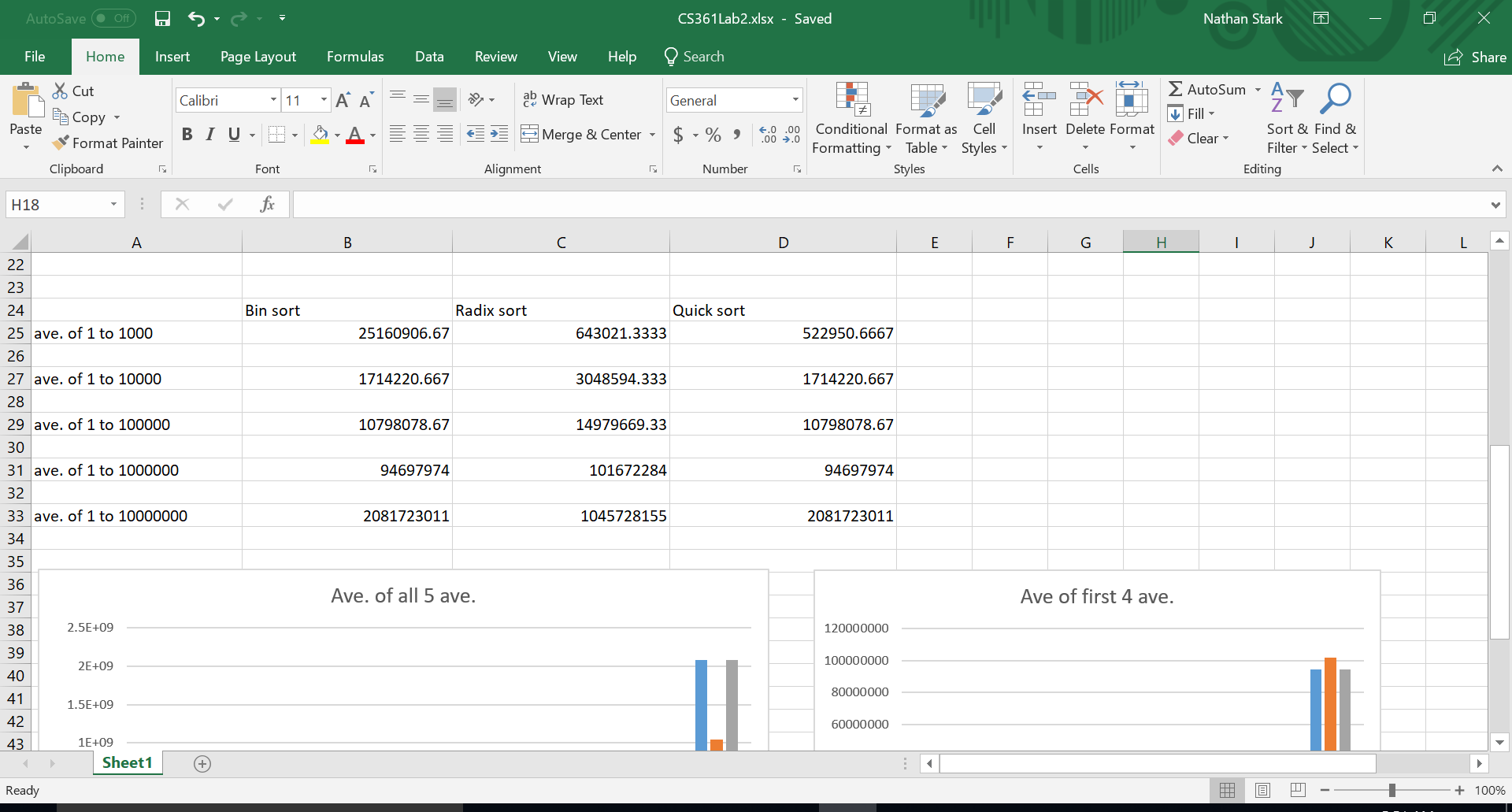
1. Show the execution time comparison with your either quick sort or merge sort. Also make sure the result of your quick sort or merge sort is sorted.





**As we can see the flgIsSorted() method is use in an if statement so that the statement “The array was sorted using quick sort.” Will print to the console only if the array is sorted. Otherwise the statement “It didn’t work.” Prints to the console. As we can see the proper statement is printed.**

1. Run your code for 1~3 three 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 and provide your results in a table and/or as a line chart.



1. Use your Lab 1 read method to from my data file. Then write **recursive** algorithm to list the largest 10 elements of the data you read, and listing them in decreasing order as the output. Again, starts with 1,000 and increases at 10x until it needs to read more than 10 million numbers. Output the execution time of your approach.
2. Test your result by calling one of your sorting algorithm to sort the data first and display largest numbers in decreasing order as the output. Output the execution time of your approach.
3. Run your code for part 6 and 7 three 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.
4. Write a half to one-page report to explain your execution time observation and discuss the problem-solving approach you applied for step 6. Is it DP, greedy algorithm, or divide-and-conquer?

# What to turn in:

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

* Code segments for 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.
* All Excel charts and graphs, along with a written analysis of your results.