AP Computer Science Name______________________________
Sort Lab

In this lab you will write a program that uses recursive and iterative sorting algorithms, and analyze their performance on several types of lists. First read chapter 13 sections 13.4 – 13.9 in JAVA Methods A & AB.

1. Copy the following three files to your project src folder: `SortsLab.java, Sorts.java, ListSetup.java`. Compile and run your program on an array of 20 randomly generated integers first with quicksort only. To get a rough idea of the algorithm's running time, modify the program to declare a variable `qcount` that keeps track of how many times during program execution an array element is compared to the pivot. Use proper labeling to display the array before and after sorting along with `qcount`, its running time measure. What did you get for `qcount?` ______. Complete the missing code for the selection, insertion, and merge sort methods, and test them on the random-ordered array to verify that the lists are properly sorted. Calculate the `scount, icount, and mcount` the number of comparisons between array elements in the selection, insertion, and merge sort algorithms, and record the data for the four sorts on a list of 20 numbers.

Data	Quick Sort	Selection Sort	Insertion Sort	Merge Sort
random order				

2. Write the portions of missing code to run each sorting method on ascending-ordered, and descending-ordered arrays. Compare the running times of the four sorts on arrays of 1000 elements where the data is randomly or already ordered. Be sure to comment out the print calls when you change the size of the arrays. Complete the table below.

Data	Quick Sort	Selection Sort	Insertion Sort	Merge Sort
random order				
ascending order				
descending order				

On one graph plot the number of comparisons for each sort on random data. On a second and third graph, repeat for ascending and descending data.

3. Comment out all sorting calls except the quick sort portions of the program. The quicksort algorithm you've been using selects the first element as the pivot element. Explore how the choice of quicksort's pivot element affects the running time. Modify the method to pivot about the middle element, then a randomly chosen location before proceeding with the splitting. Record the running time measure, `qcount`, for each pivot strategy on a 1000 element arrays.

Data	Split first	Split middle	Split random
random order			
ascending order			
descending order			

Which pivot value gives the best running time performance over all?_____________

4. Counting the number of comparisons disregards movement of elements. For example, a swap requires three moves. The merge sort uses a temporary array and moves elements to and from it. Incorporate another counter into your program to keep track of the number of moves as well as comparisons. Run some data and discuss the results. Does this change your ideas of the efficiency of each sort?

5. Read Chapter 18 in JAVA Methods A & AB to learn about the concept of Big-Oh and proper notation. Finally after learning the Big-Oh times for each sort, discuss which sort is best suited for each type of array.