Your solutions must be typed (preferably typeset in \LaTeX) and submitted as a hard-copy at the beginning of class on the day it's due. The answers that you provide should clearly demonstrate that you understand the assignment and should provide enough information to clearly explain your solution to a peer.

**Problem 1** [10 points] As described in class, insertion sort goes sequentially through the array when making comparisons to find the proper place for the element that is currently being processed. Suppose that the sequential search is replaced by a binary search. Will the change decrease the asymptotic cost of insertion sort? Explain.

**Problem 2** In the programming language of your choice, implement both Insertion Sort and Merge Sort as described in class. Feel free base your code on implementations that you find online, just ensure that both operate on similar objects and properly cite your sources. Add code to both algorithms that tracks the time it takes to process an input. You will also need to create a method that generates arrays of integers of various sizes and orderings. (You do not need to print your code as part of your submission, but must be able to provide source code on request)

(a) [5 points] Briefly describe your development process. As a starting point consider the following questions: What sources did you use for the algorithms? What programming language did you choose and why? Did you run into any difficulties in the implementation? How are you timing the algorithms?

(b) [5 points] Run your algorithms on various sized lists of random integers and graph the running time. Does your graph match the theoretical analysis we did in class, explain why or why not? (Try to get to the point where the slower of the two algorithms finishes in a little more than one second.)

(c) [5 points] Repeat part (a) using already sorted lists of integers. Graph and explain your results.

(d) [5 points] Choose a list size and experimentally determine what percentage of array elements can be out of order before Insertion sort is not longer the faster option. Graph and explain your results. Will this percentage change as the list size grows?