

Due by the start of class, Oct. 6.

1. The merge algorithm presented in class was not *in place*. Assume you are given an array  $A[1], \dots, A[n]$  where  $A[1], \dots, A[k]$  is one sorted list and  $A[k+1], \dots, A[n]$  is another sorted list. Write an *in place* algorithm that merges  $A[1], \dots, A[k]$  and  $A[k+1], \dots, A[n]$ . Explain how your algorithm works and derive its worst case running time. Show an example where the running time is worst case.
2. An array is said to be *k-sorted* if no item is more than  $k$  positions from its position in the sorted array. For example, the array  $[3, 2, 1, 5, 4]$  is 2-sorted.
  - (a) Give an array of length 8 that is 4-sorted.
  - (b) What is the running time of bubblesort on a  $k$ -sorted array?
  - (c) What is the running time of mergesort on a  $k$ -sorted array?
  - (d) Give an algorithm that is faster than bubblesort and mergesort for  $k$ -sorted arrays. It goes without saying that you need to show that your algorithm is correct and prove its run-time.
3. Extra Credit: Give the best lower-bound you can for comparison sorting of  $k$ -sorted arrays.
4. Extra Credit: Suppose  $T(1) = 1$  and  $T(n) = T(n/5) + T(7n/10) + n$ . Give a  $\Theta$  bound for this recurrence and prove the correctness of your answer.