Problem 1. Selection sort finds the largest element and puts it at the end of the array in each iteration. Consider a version of Selection sort that finds the two largest elements and puts both of them at the end of the array (in order) in each iteration.

(a) Write the pseudo-code for this version of selection sort. Make sure that its works when the size of the array is odd.

(b) Calculate the exact worst-case number of comparisons for an even n. Show your work.

Problem 2. One way to find the median of a list is to sort the list and then take the middle element. Assume you use Bubble Sort to sort a list with 7 elements (i.e. $n = 7$). Exactly how many comparisons do you use (in the worst case)?

Problem 3. You can actually find the median by running a sorting algorithm and stopping early, as soon as you know the median. Assume you use Bubble Sort to find the median of 7 elements (i.e. $n = 7$), but stop as soon as you know the median. Exactly how many comparisons do you use (in the worst case)?

Problem 4. Suppose an array of n values is divided into k subarrays. Each of those subarrays is sorted using insertion sort. What would be the asymptotic runtime of sorting the entire array in the worst case? For simplification, you may assume sorting each of these subarrays, also sorts the entire array. Show your work.