1. Suppose we want to find the $k$ smallest numbers in a list of $n$ numbers, where $k = \sqrt{n}$. Design an algorithm that solves this task in worst-case time that is linear in $n$. How far can you increase $k$ so that you still have a worst-case linear time algorithm? (20 points)

2. In class we showed using an adversary argument that any algorithm to compute the MAX and the MIN of a set of $n$ distinct numbers simultaneously using pairwise comparisons must, in the worst-case, use at least $\lceil 3n/2 \rceil - 2$ comparisons.

   (a) Use the decision tree argument (finding a lower bound on the number of possible responses, which is a lower bound on the number of leaves, and using this as a bound on the height of the tree) to develop a worst-case lower bound on the number of pairwise comparisons.

   (b) If this bound is different than $\lceil 3n/2 \rceil - 2$, explain how seemingly contradictory bounds can both be correct.

   (15 points)

3. The input is two sets $S_1$ and $S_2$ containing $n$ real numbers in total, and a real number $x$.

   (a) Find a $O(n \log n)$-time algorithm that determines whether there exists an element from $S_1$ and an element from $S_2$ whose sum is exactly $x$.

   (b) Suppose now that the two sets are given in sorted order. Find a $O(n)$-time algorithm solving this problem. (15 points)
4. Show that the second largest element can be found with $n + \lceil \log n \rceil - 2$ comparisons in the worst case. (15 points)

5. In our linear-time selection algorithm, the inputs are divided into groups of 5. What if you used groups of 3 instead? What if used groups of 7 or larger (odd integers)? (15 points)

6. Show that $2n - 1$ comparisons are necessary in the worst case to merge two sorted lists containing $n$ elements each. (20 points)