1. Design an algorithm called Extrema$(A, p, r)$ on the divide and conquer paradigm which finds and returns the maximum and minimum values in $A[p \cdots r]$. Your algorithm should perform exactly $\lceil 3n/2 \rceil - 2$ comparisons, on an input array of length $n$.

   a. Prove the correctness of your algorithm.

   b. Write a recurrence for the number of comparisons performed on arrays of length $n$, and solve it exactly.

2. Design a variation of MergeSort which, instead of recurring until the subarray has length 0 or 1, recurs at most a constant $k$ times, then calls InsertionSort on the resulting $2^k$ subarrays of length (approximately) $n/2^k$. Call this algorithm Depth-$k$-MergeSort.

   For example let $q = \left\lfloor \frac{1+n}{2} \right\rfloor$, $u = \left\lfloor \frac{1+q}{2} \right\rfloor$, and $v = \left\lfloor \frac{(q+1)+n}{2} \right\rfloor$. We can picture the operation of Depth-2-MergeSort by the following recursion tree.

   The nodes at depth $i$ represent subarrays of length $n/2^i$. Depth-2-MergeSort calls InsertionSort on the 4 subarrays at Depth 2. Depth-1-MergeSort recurs down to depth 1 and calls InsertionSort on the 2 subarrays at Depth 1. Depth-0-MergeSort simply calls InsertionSort on the full array.

   a. Write pseudo-code for this algorithm. (Hint: Notice that each call to Depth-$k$-MergeSort must know its own level in the recursion tree in order to know whether it should recur again, or call InsertionSort.)

   b. Determine the asymptotic run time of Depth-$k$-MergeSort as a function of both $k$ and $n$. In order to simplify the analysis, you may assume that $n$ is always an exact power of 2. (Hint: First review the discussion of InsertionSort in the text, and note that it runs in time $\Theta(n^2)$.)
3. Recall the RandSelect($A, p, r, i$) algorithm which returns the $i$th order statistic of the subarray $A[p \cdots r]$. The recurrence

$$t(n) = (n-1) + \left( \frac{n-1}{n^2} \right) \sum_{q=1}^{n-1} t(q)$$

was derived in class for the average number of comparisons performed by RandSelect on subarrays of length $n = r - p + 1$. Use this recurrence to prove that $t(n) = \Theta(n)$. (Hint: first show directly that $t(n) \geq \Omega(n)$. Then prove by induction that $\forall n \geq 1: t(n) \leq 2n$, showing $t(n) = O(n)$.)