1. Use the Master Theorem to find the asymptotic solution for the following recurrence: $T(n) = T(\frac{7n}{10}) + n$.

**Solution:** We have $a = 1$, $b = 10/7$, 

$$f(n)/n^{\log_{10} 7} = n,$$

we get Case 3, and thus $T(n) = \Theta(n)$. Note that the regularity condition is satisfied as $7n/10 \leq cn$ for $c = 7/10$ and $n \geq 1$. (20 points)

2. In this midterm there are 5 problems and each problem is worth 20 points. Suppose that you either get full credit for a problem or 0 points (this will not be the case). Denote the probability that you answer question $i$ correctly by $p_i$, $i = 1, \ldots, 5$. Assume that $p_1 = 0.9$, $p_2 = 0.8$, $p_3 = 0.7$, $p_4 = 0.6$, $p_5 = 0.5$. Use indicator random variables to find your expected score on the final.

**Solution:** Let 

$$X_i = \begin{cases} 
1 & \text{the } i\text{th problem is solved correctly} \\
0 & \text{otherwise} 
\end{cases}$$

Then by the linearity of expectation

$$20E(X) = 20E\left(\sum_{i=1}^{5} X_i\right) = 20 \sum_{i=1}^{5} E(X_i) =$$

$$= 20(0.9 + 0.8 + 0.7 + 0.6 + 0.5) = 70.$$  (20 points)

3. Given a sequence of $n$ distinct numbers $a_1, a_2, \ldots, a_n$, an inversion is a pair $(i, j)$, where $i < j$ but $a_i > a_j$. Design a divide-and-conquer algorithm that runs in $O(n \log n)$ time to determine the number of inversions in the sequence.
Solution: We can easily modify Merge-sort to do this. In addition to sorting we determine the number of inversions. Recursively we sort the left half $A$ and the right half $B$ and find the number of inversions in $A$ and in $B$. Then the only task left is to determine the number of crossing inversions $a_i \in A$, $a_j \in B$ but $a_i > a_j$ in $O(n)$ time. We can do this as we are merging. Say $a$ is the current smallest element in $A$ and $b$ is the current smallest element of $B$. We compare $a$ and $b$. If $a$ is the smaller one, then there is no new inversion. However, if $b$ is the smaller one, then $b$ forms an inversion with all the remaining elements in $A$, so we add this number to the number of inversions. This is in time $O(n)$, so we get the same recursion as for Merge-sort. (Note that there could be $\binom{n}{2}$ inversions; we find their number in just $O(n \log n)$ time.) (20 points)

4. Given a sequence of $n$ distinct numbers we want to find the $\lfloor \log n \rfloor$ largest numbers in sorted order. Give an algorithm for this task for which the worst-case running time is $O(n)$. Show that this is optimal (apart from the constant coefficient in the big-O notation), i.e. give a linear lower bound for the worst-case running time of any algorithm solving this task.

Solution: Let $i = \lfloor \log n \rfloor$. Use the worst-case linear time selection algorithm to find the $i$-th largest number in $O(n)$ time. Partition around that number in $O(n)$ time. Sort the $i$ largest numbers in $O(i \log i)$ worst-case time (with Merge-sort or Heap-sort). Thus the total running time is $O(n) + O(\log n \log \log n) = O(n)$. Finding the maximum alone must take at least $n - 1$ comparisons. (20 points)

5. Show that in a set of $n$ distinct elements the 13-th largest element can be found with at most $n + 12 \lceil \log n \rceil$ comparisons in the worst case.

Solution: We use the tournament method again. We set up a tournament on the $n$ elements (using $n - 1$ comparisons). We remove the largest element and we replace it with $-\infty$ in the tournament. We find the winner again using at most $\lceil \log n \rceil$ comparisons (only one path has to be recomputed). We remove this element again and replace it with $-\infty$, etc. we repeat this 12 times. The total number of comparisons is at most $n + 12 \lceil \log n \rceil$. (20 points)