1. In the algorithm \texttt{SELECT} the input elements are divided into groups of 5. Will the algorithm work in linear time if they are divided into groups of 7? Argue that \texttt{SELECT} will not work in linear time if groups of size 3 are used.

\textbf{solution:} If the numbers are divided into groups of 7, then the number of elements guaranteed to be less than the median-of-medians is at least $4 \left\lceil \frac{n}{7} \right\rceil / 2 - 1$. Similarly, the number of elements guaranteed to be greater than the median-of-medians is at least $4 \left\lfloor \frac{n}{7} \right\rfloor / 2 - 2$.

In either case, the number of elements on the small side is at least $\frac{2n}{7} - 8$. Assuming the worst case scenario, when we make a recursive call to search on the larger part, we get a subproblem of size at most $n - \frac{2n}{7} + 8 \leq \frac{5n}{7} + 8$. The recursive call to find the median of medians is of size $\left\lceil \frac{n}{7} \right\rceil$. Thus we get the recurrence:

$$T(n) = T \left( \frac{5n}{7} + 8 \right) + T \left( \left\lceil \frac{n}{7} \right\rceil \right) + cn$$

If $n \geq 60$, then we have the sum of the problem sizes on the right side of the recurrence as strictly than the size on the left. Whenever this is the case, and the cost of a problem at an internal node is $c$ times the size of the problem, the total cost is linear (see solutions to problems in the recurrence relations assignment/lecture notes). Thus $T(n) = c'n$, for some constant $c'$.

The corresponding recurrence, if we split the input into groups of size 3 each is

$$T(n) = T \left( \left\lceil \frac{n}{3} \right\rceil \right) + T \left( \frac{2n}{3} + 4 \right) + cn$$
The cost at an internal node is linear in the size of the subproblem as can be inferred from the $cn$ term. The total size of the subproblems at a level exceed the size of the problem at the higher level. This is true between an internal node and its children as well. Thus the total cost at every level is at least $\Omega(n)$. The number of levels is at least $\Omega(\log n)$. Hence the solution to the recurrence, and hence the running time of the select algorithm in the worst case is at least $\Omega(n \log n) = \omega(n)$.

2. Let $X[1 \ldots n]$ and $Y[1 \ldots n]$ be two arrays, each containing $n$ elements already in sorted order. Give an $O(\log n)$-time algorithm to find the median of all $2n$ elements in arrays $X$ and $Y$. (The median of a set of size $n$ is the $\lceil \frac{n}{2} \rceil$ smallest element).

**solution:** Compare the elements $X[\lceil \frac{n}{2} \rceil]$ and $Y[\lfloor \frac{n}{2} \rfloor]$, and vice versa (indices and arrays interchanged). If $n$ is odd, then both cases are identical. Call whichever these elements are $x_0$ and $y_0$. Assume without loss of generality that the outcome is $x_0 < y_0$. In that case we know that the median has to be from among the elements in $X$ to the right of $x_0$ and those in $Y$ to the left of $y_0$. Thus, with a constant number of comparisons, we have reduced the problem to one of half the size. The median we are seeking must also be the median of the reduced set. Since we have found an equal number of elements which are guaranteed to be greater than and less than the median respectively, in the set of elements eliminated. Thus, we can find the median in $O(\log n)$ time.

3. Suppose you have a “black-box” worst-case linear-time median subroutine. Give a simple, linear-time algorithm that solves the selection problem for an arbitrary order-statistic.

**solution** If the required order statistic is the median then it can be solved in one step of the linear time black-box algorithm. If not, then find the median in linear time by the black-box algorithm and then apply the linear time partition algorithm with the median as the pivot. This splits the array roughly in half. If we are looking for a lower order statistic, recurse in the left part and same index using the black-box and partition combination. Else, recurse on the right part, with a modified index (order statistic - median’s rank) again using the combination of black-box and partition around median. Thus in successive recursive calls the problem size keeps halving and at each stage the cost is linear in the problem size. The linear time can be taken to be $(c + c')n$, where the constant $c$ is for the linear
time median finding black-box algorithm and the constant $c'$ is for the linear time partitioning procedure. Thus the total running time is at most $(c + c') \left( n + \frac{n}{2} + \frac{n}{4} + \cdots \right) \leq 2(c + c')n$.

4. Prove a lower bound of $n + \lceil \log n \rceil - 3$ on the number of comparisons required to find the second largest element of a set of distinct elements. An upper bound of this value was shown in the class, in the form of an algorithm using that many comparisons. You might try and use ideas similar to the proof of lower bound for the problem of finding the simultaneous-max-and-min.