9.3-6
The $k$th quantiles of an $n$-element set are the $k - 1$ order statistics that divide the sorted set into $k$ equal-sized sets (to within 1). Give an $O(n \lg k)$-time algorithm to list the $k$th quantiles of a set.

Solution:
Unsorted array : $A[]$
distinct keys: an integer $k$
An empty array $Q$ of length $k - 1$
We want to find the $k$th quantiles of $A$.

QUANTILES($A, k, Q$)
1. if $k == 1$ then return
2. else
3. $n = $ length of $A[]$
4. $i = \lfloor k/2 \rfloor$
5. $x = \text{SELECT}(A, \lfloor i \cdot n/k \rfloor)$
6. $\text{PARTITION}(A, x)$
7. Add to list $Q$: QUANTILES($A[1]$ to $A[\lfloor i \cdot n/k \rfloor], \lfloor k/2 \rfloor, Q)$
8. Add to list $Q$: QUANTILES($A[\lfloor i \cdot n/k \rfloor + 1]$ to $A[n], \lfloor k/2 \rfloor, Q)$
9. return $x$

Consider a recursion tree for this algorithm. At the top level we need to find $k - 1$ order statistics, and it costs $O(n)$ to find one. The root has two children, one contains at most $\lfloor (k - 1)/2 \rfloor$ order statistics, and the other $\lfloor (k - 2)/2 \rfloor$ order statistics. The sum of the costs for these two nodes is $O(n)$.

At depth $i$ we find $2^i$ order statistics. The sum of the costs of all nodes at depth $i$ is $O(n)$, for $0 \leq i \leq \log_2(k - 1)$, because the total number of elements at any depth is $n$. The depth of the tree is $d = \log_2(k - 1)$. Hence, the worstcase running time of QAUNTILES is $\theta(n \lg k)$. 
9.3-7
Describe an $O(n)$ algorithm that, given a set $S$ of $n$ distinct numbers and a positive integer $k \leq n$, determines the $k$ numbers in $S$ that are closest to the median of $S$.

Solution:
Assume for simplicity that $n$ is odd and $k$ is even. If the set $S$ was in sorted order, the median is in position $n/2$ and the $k$ numbers in $S$ that closest to the median are in positions $(n-k)/2$ through $(n+k)/2$. We first use linear time selection to find the $(n-k)/2$, $n/2$, and $(n+k)/2$th elements and then pass through the set $S$ to find the numbers less than $(n+k)/2$th element, greater than the $(n-k)/2$th elements, and not equal to the $n/2$th elements. The algorithm takes $O(n)$ time as we use linear time selection exactly three times and traverse the $n$ numbers in $S$ once.