1. Give an $O(n \lg k)$-time algorithm to merge $k$ sorted lists into one sorted list, where $n$ is the total number of elements in all the input lists.

2. Given a set of $n$ numbers, we wish to find the $i$ largest in sorted order using a comparison-based algorithm. Find the algorithm that implements each of the following methods with the best asymptotic worst-case running time, and analyze the running times of the algorithms on terms of $n$ and $i$.
   
   (a) Sort the numbers, and list the $i$ largest.
   (b) Build a max-priority queue from the numbers, and call EXTRACT-MAX $i$ times.
   (c) Use a SELECT algorithm to find the $i$th largest number, partition around that number, and sort the $i$ largest numbers.


   We can obviously find a local minimum in $O(n)$ time by scanning through $A$. Describe an $O(\log n)$ algorithm for finding a local minimum.

4. 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 (in value) to the median of $S$.

5. Let $A$ be a list of $n$ (not necessarily distinct) integers. Describe an $O(n)$-algorithm to test whether any item occurs more than $\lceil n/2 \rceil$ times in $A$. Your algorithm should use $O(1)$ additional space.

6. Give an $O(n \lg k)$ algorithm to find the $k-1$ elements in a set that partition the set into (approx.) $k$ equal-sized sets $A_1, A_2, \ldots, A_k$ such that all elements in $A_i$ are smaller than all elements in $A_{i+1}$. Assume $k$ is a power of 2.

7. Show how to sort $n$ integers in the range $1$ to $n^2$ in $O(n)$ time.