Problem 1. Let $S$ be a set of 9 integers $\{75, 23, 12, 87, 90, 44, 8, 32, 89\}$, stored in an array of length 9. Let us use quicksort to sort $S$. Recall that the algorithm randomly picks a pivot element, and then, recursively sorts two sets $S_1$ and $S_2$, respectively. Suppose that the pivot is 89. What are the contents of $S_1$ and $S_2$, respectively? The ordering of the elements in $S_1$ and $S_2$ does not matter.

Problem 2. Let $S_1$ be a set of $n$ integers, and $S_2$ another set of $n$ integers. Each of $S_1$ and $S_2$ is stored in an array of length $n$. The arrays are not necessarily sorted. Design an algorithm to determine whether $S_1 \cap S_2$ is empty. Your algorithm must terminate in $O(n \log n)$ time.

Problem 3 (Sorting a Multi-Set). Let $A$ be an array of $n$ integers. Note that some of the integers may be identical. Design an algorithm to arrange these integers in non-descending order. For example, if $A$ stores the sequence of integers $(35, 12, 28, 12, 35, 7, 63, 35)$, you should output an array $(7, 12, 12, 28, 35, 35, 35, 63)$.

Problem 4* (Inversions). Consider a set $S$ of $n$ integers that are stored in an array $A$ (not necessarily sorted). Let $e$ and $e'$ be two integers in $S$ such that $e$ is positioned before $e'$ in $A$. We call the pair $(e, e')$ an inversion in $S$ if $e > e'$. Design an algorithm to count the number of inversions in $S$. Your algorithm must terminate in $O(n \log n)$ time.

For example, if the array stores the sequence $(10, 15, 7, 12)$, then your algorithm should return 3, because there are 3 inversions: $(10, 7), (15, 7)$, and $(15, 12)$.

Problem 5* (Maxima). In two-dimensional space, a point $(x, y)$ dominates another point $(x', y')$ if $x > x'$ and $y > y'$. Let $S$ be a set of $n$ points in two-dimensional space, such that no two points share the same x- or y-coordinate. A point $p \in S$ is a maximal point of $S$ if no point in $S$ dominates $p$. For example, suppose that $S = \{(1, 1), (5, 2), (3, 5)\}$; then $S$ has two maximal points: $(5, 2)$ and $(3, 5)$.

Suppose that $S$ is given in an array of length $n$, where the $i$-th $(1 \leq i \leq n)$ element stores the x- and y-coordinates of the $i$-th point in $S$ (i.e., each element of the array occupies 2 memory cells). For example, $S = \{(1, 1), (5, 2), (3, 5)\}$ is given as the sequence of integers: $(1, 1, 5, 2, 3, 5)$. Design an algorithm to find all the maximal points of $S$ in $O(n \log n)$ time.