Problem 1: You are given an array $A$ of (decimal) integers, where different integers may have different number of digits, but the total number of digits over all the integers in the array is $n$. Give an algorithm to sort $A$ in $O(n)$ worst-case time.

Your answer should include a high-level description of the algorithm, a good explanation of its correctness and a running time analysis to show that its (worst-case) running time is $O(n)$.

Problem 2: You are given a collection $S$ of $m$ keys $k_1, k_2, \ldots, k_m$, where each key is an integer in the range $[1 \ldots n]$ for some integer $n$. Note that the collection may include duplicates; that is, there may be multiple occurrences of the same key. We want to handle the following query operations.

(a) MEMBER($i$): Here, $i$ is an integer in the range $[1 \ldots n]$. This operation should return TRUE if the value $i$ is in the current collection of keys and FALSE otherwise.

(b) LESS($i$): Here also, $i$ is an integer in the range $[1 \ldots n]$. This operation should return the number of keys in the collection which are strictly less than $i$.

(c) RANGE($i, j$): Here, $i$ and $j$ are both integers in the range $[1 \ldots n]$ and further $i \leq j$. This operation should return the number of keys in the collection that are in the range $[i \ldots j]$.

You are required to specify a data structure that supports each of the above operations in $O(1)$ worst-case time. You may use $O(n)$ additional space and $O(m + n)$ preprocessing time.

Your answer must include the following.

(i) A clear description of the data structure,

(ii) pseudocode for the preprocessing steps along with an explanation of why the preprocessing time is $O(m + n)$ and

(iii) the pseudocode for each of the three operations above along with a clear explanation of why the worst-case time to answer each query is $O(1)$.

Problem 3: Let $S = \{s_1, s_2, \ldots, s_n\}$ and $T = \{t_1, t_2, \ldots, t_r\}$ be two sets such that each element of $S \cup T$ is an integer in the range 1 to $m$ for some positive integer $m$. Give an algorithm to determine whether or not $S \subseteq T$ in $O(n + r)$ time. Your answer should include a high-level description of the algorithm, a proof of its correctness and a proof that its running time is $O(n + r)$.

Note: If you need to use an array of size $m$ to solve this problem, bear in mind that you cannot initialize the whole array. (This is because initializing all the elements takes $\Theta(m)$ time, and $m$ may be much larger than $n + r$.)