Problem 1. (1 point) Illustrate the operation of heapsort on the array:
\( A = (19, 2, 11, 14, 7, 17, 4, 3, 5, 15) \)
By showing the values in array \( A \) after initial heapification and after each call to \texttt{max-heapify}.

Problem 2. (1 point) Illustrate the operation of counting sort on the array:
\( A = (4, 6, 3, 5, 0, 5, 1, 3, 5, 5) \)
By showing the values in array \( C \) after each loop, and the final array \( B \).

Problem 3. (1 point) Illustrate the operation of radix sort on the array:
\( A = (392, 517, 364, 931, 726, 912, 299, 250, 600, 185) \)
By showing the values in array \( A \) after each intermediate sort.

Problem 4. (1 point) Illustrate the operation of bucket sort on the array:
\( A = (0.88, 0.23, 0.25, 0.74, 0.18, 0.02, 0.69, 0.56, 0.57, 0.49) \)
By showing the final array \( B \) of sorted buckets.

Problem 5. (3 points) Consider a \( d \)-ary heap – a generalization of a binary heap, in which all internal nodes (with at most one exception) have \( d \) children. Design a method to store a \( d \)-ary heap in an array and derive the expressions for:
(a) Parent of \( i \)-th node in a \( d \)-ary heap.
(b) \( j \)-th child of \( i \)-th node in a \( d \)-ary heap.
(c) Maximum number of nodes of height \( h \) in any \( n \)-element \( d \)-ary heap.
(d) Maximum height of any \( n \)-element \( d \)-ary heap.

Problem 6. (4 points) Consider a min-priority queue representing a set of integers and supporting the following operations: \texttt{insert(k)} to insert element with value \( k \), \texttt{get-min()} to return the minimum element, and \texttt{extract-min()} to remove and return the minimum element.

Give pseudocode and worst-case running time for each operation, assuming the priority queue is implemented with the following data structures:
(a) Unordered array.
(b) Ordered array.
(c) Unordered linked list.
(d) Ordered linked list.
(e) Min-heap.