**Total possible marks: 100.** Homeworks must be solved individually. This set covers chapters 6–12 of the textbook *Introduction to Algorithms*, 3rd. ed., by Cormen et al.

**Exercise 1: min-priority queues (35 points).** Using the same notation as in the textbook, write pseudocode to implement a *min-priority queue* using a min-heap, ensuring that all operations run in $\mathcal{O}(\lg n)$. Specifically:

1. (3 points) State the min-heap property.

Then, write pseudocode for the following functions (where $A$ is the min-heap):

2. (5 points) MIN-HEAPIFY$(A, i)$, which assumes that the binary trees rooted at LEFT$(i)$ and RIGHT$(i)$ are min-heaps, but that $A[i]$ may be larger than its children. MIN-HEAPIFY lets the value of $A[i]$ float down so that the subtree rooted at $i$ obeys the min-heap property. Note: write an *iterative* version (the textbook's is recursive).
3. (2 points) HEAP-MINIMUM$(A)$, which returns the element of $A$ with the smallest key.
4. (5 points) HEAP-EXTRACT-MIN$(A)$, which removes and returns the element of $A$ with the smallest key.
5. (5 points) MIN-HEAP-INSERT$(A, key)$, which inserts an element with the given key in $A$.
6. (5 points) HEAP-DECREASE-KEY$(A, i, key)$, which sets the value of the element in node $i$ to *key* (assumed to be smaller than the current key value).
7. (5 points) HEAP-INCREASE-KEY$(A, i, key)$, which sets the value of the element in node $i$ to *key* (assumed to be greater than the current key value).
8. (5 points) HEAP-DELETE$(A, i)$, which deletes the element in node $i$ from $A$.

*Hint*: modify accordingly the corresponding pseudocode for max-heaps from the textbook. You may also want to write max-heap implementations of HEAP-DELETE and HEAP-DECREASE-KEY, which the textbook does not provide.

**Exercise 2: sorting (15 points).**

1. (8 points) Consider the following array of integers:

$$A = [345, 435, 876, 644, 137, 786, 758, 983, 521, 645, 231].$$

   Sort it in ascending order using radix-sort, showing the array after each of the 3 sorting steps.

2. (7 points) What are the worst- and average-case running times for heapsort, quicksort and radix sort for an array with $n$ digits?

**Exercise 3: hash tables (28 points).** Consider inserting the keys $10, 22, 31, 4, 15, 28, 17, 88, 59$ (in that order) into a hash table of length $m = 11$. Show the final table in these two cases:

1. (14 points) Chaining using as hash function $h(k) = k \mod m$.
2. (14 points) Open addressing using linear probing and the same hash function.

**Exercise 4: binary search trees (22 points).**

1. (12 points) Starting from an empty binary search tree, draw the final tree resulting from the insertion of the following keys: $12, 34, 1, 45, 33, 27, 8, 30, 66, 41$ (in that order).
2. (10 points) Starting from the tree obtained in the former question, draw the tree resulting from the removal of the following keys: $34, 33$ (in that order).

**Bonus exercise: (20 points).** (Exercise 8.4-4 in the book.) We are given $n$ points in the unit circle, $p_i = (x_i, y_i)$, such that $0 < x_i^2 + y_i^2 \leq 1$ for $i = 1, 2, \ldots, n$. Suppose that the points are uniformly distributed; that is, the probability of finding a point in any region of the circle is proportional to the area of that region. Design an algorithm with an average-case running time of $\Theta(n)$ to sort the $n$ points by their distances $d_i = \sqrt{x_i^2 + y_i^2}$ from the origin.

*Hint*: Design the bucket sizes in BUCKET-SORT to reflect the uniform distribution of the points in the unit circle.