**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: sorting (50 points).** Consider the following array of integers:

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

You have to sort it in ascending order with each of the following algorithms, illustrating intermediate results as indicated below:

1. Heapsort: show the following in the same form as in fig. 6.4 (but show each max-heap both as an array and as an almost complete binary tree):
   - (10 points) Show the max-heap produced by `Build-Max-Heap` (p. 157).
   - (13 points) Show the max-heap just after each call to `Max-Heapify` during the `HeapSort` loop (p. 160).

2. (12 points) Quicksort (non-randomized version where the pivot is always the last element): draw a recursion tree corresponding to each recursive call to `Quicksort`, where in each node you show the state of the array just after the call to `Partition`.

3. (8 points) Radix-sort: show the array after each of the 3 sorting steps.

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

**Exercise 2: 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 3: 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.)