CP 312, Fall 2017
Assignment 2 (5% of the final grade)
(due Monday, October 15, at 23:30)

There are four questions in this assignment. Note that all logarithms in this assignment are base 2, i.e. $\log f(n) \equiv \log_2 f(n)$.

1. [10 marks] In the COMMON SUM problem, we are given two arrays A and B of length n containing non-negative (not necessarily distinct) integers, and we must determine whether there are indices $i_1, i_2, j_1, j_2 \in \{1, 2, \ldots, n\}$ for which

$$A[i_1] + A[i_2] = B[j_1] + B[j_2].$$

   Design an algorithm that solves the COMMON SUM problem and has time complexity $O(n^2 \log n)$ in the setting where operations on individual integers take constant time.

   Your solution must include a description of the algorithm in words, the pseudocode for the algorithm, a justification of its correctness, and an analysis of its time complexity in big-Θ notation.

2. [4 marks] Analyze the following pseudocode and give a tight (Θ) bound on the running time as a function of n. You can assume that all individual instructions are elementary. Show your work.

```
m := 1; s := 1;
while m <= n do
  for j = 1 to 2⌈log m⌉ do
     s := s + 1
  od
  m := 3 * m
od.
```

3. [8 marks] (a) [6 marks] Solve the following recurrences by the recursion-tree method (you may assume that n is a power of 2):

$$T(n) = \begin{cases} 4, & n = 1, \\ 5T\left(\frac{n}{2}\right) + 2n^2, & n > 1. \end{cases}$$

(b) [2 marks] Solve part (a) using Master theorem.

4. [8 marks] Consider the following algorithm prototype in pseudocode:

```
int Fiction( A:: array, n:: integer) {
 if (n>1) {
  B <- A[1],...,A[n/3];        // copy 1st "third" of A to B
  C <- A[n/3+1],...,A[2*n/3];  // copy 2nd "third" of A to C
  D <- A[2*n/3+1],...,A[n];    // copy 3rd "third" of A to D
  C <- Perturb(C);
  cond2 <- Fiction(C, n/3);
  if (cond2) {
    cond1 <- Fiction(B, n/3);
    cond3 <- Fiction(D, n/3);
    cond2 <- (cond1 + cond3)/2;
  }
  return cond2;
 }
 else
  return 1;
}
```

(a) [5 marks] What is the worst case complexity of this algorithm assuming that the algorithm `Perturb` when applied to an array of length n has running time complexity $\Theta(n \log n)$? To justify your answer provide and solve a divide-and-conquer recurrence for this algorithm. You may assume that n is a power of 3.

(b) [3 marks] What is the "best case" complexity of this algorithm under the same assumptions as in (a)?