The “Classroom Problems” will be discussed and solved ex tempore at the tutorials in the same week as this problem set is handed out. These are intended to help you in getting started with the material, and will not yield any course credit. 

The “Homework Problems” you should solve at home, and attend one of the tutorial sessions in the following week, prepared to present your solutions at class if requested. You earn one tutorial credit point (= 0.33 course points, for a maximum total of 6) for each of these problems that you have solved, and marked as solved at the tutorial session.

The “Demonstration Problems” illustrate further interesting aspects of the material. These will be discussed at the tutorials in the same week as the Homework Problems (time permitting) and printed solutions will be provided. You are encouraged to work on these, but no course credit is given for them.

The “Challenge Problems” are extra problems related to the material, which may be exceptionally difficult and/or require tools beyond the course curriculum.

**Classroom problems:**

1. Denote by $T(n)$ the number of elementary operations performed by the naive recursive Fibonacci algorithm $Fib$ discussed at Lecture 1. Show that $T(n) \geq F_n$ for all $n \geq 0$. (Cf. also Challenge Problem 8 below.)

2. Solve the following recurrence equations by the unwinding technique:

   (a) $T(0) = 1$, $T(n) = 2T(n-1) + 1$ for $n = 1, 2, \ldots$
   
   (b) $T(1) = 1$, $T(n) = 2T(n/3) + 1$ for $n = 3^k$, $k = 1, 2, \ldots$

3. [Dasgupta et al., Ex. 0.1] In each of the following situations, indicate the relation (if any) between the orders of growth of functions $f$ and $g$, i.e., whether $f = \Theta(g)$, $f = o(g)$, $g = o(f)$, or the two functions are incomparable. (The notation “$\log n$” denotes by default base-2 logarithms.)

<table>
<thead>
<tr>
<th>$f(n)$</th>
<th>$g(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) $n-100$</td>
<td>$n-200$</td>
</tr>
<tr>
<td>(b) $n^{1/2}$</td>
<td>$n^{2/3}$</td>
</tr>
<tr>
<td>(c) $10n \log n$</td>
<td>$n \log n^2$</td>
</tr>
<tr>
<td>(d) $n^{1.01}$</td>
<td>$n \log^2 n$</td>
</tr>
</tbody>
</table>
Homework problems:

4. Similarly as in Classroom Problem 3, indicate the relations between the orders of growth of the following functions:

\[
\begin{align*}
\text{(a)} & \quad (\log n)^{\log n} & \text{and} & \quad n/\log n \\
\text{(b)} & \quad \sqrt{n} & \text{and} & \quad \log^3 n \\
\text{(c)} & \quad n^{1/2} & \text{and} & \quad 5^{\log n} \\
\text{(d)} & \quad n^{2^n} & \text{and} & \quad 3^n \\
\text{(e)} & \quad 2^n & \text{and} & \quad 2^{n+1} \\
\text{(f)} & \quad n! & \text{and} & \quad 2^n \\
\text{(g)} & \quad (\log n)^{\log n} & \text{and} & \quad 2^{\log^2 n} \\
\text{(h)} & \quad \sum_{i=1}^{n} i^k & \text{and} & \quad n^{k+1}
\end{align*}
\]

5. Show that

\[
\log(n!) = \sum_{i=1}^{n} \log i = \Theta(n \log n).
\]

(Hint: Construct an upper bound for the value of the sum using the \(\log n\) term, and a lower bound using the term \(\log^2 n\).)

6. Analyse the worst-case time complexity of the following algorithm:

```
Algorithm 1: Bubble sort

1 function BUBBLE_SORT (A[1...n])
    Input: Integer array A[1...n].
    Output: Array A, with same items in increasing order.
2 for i ← n downto 2 do
3     for j ← 1 to i - 1 do
5             Exchange elements A[j] and A[j + 1]
6         end
7     end
8 end
```

Demonstration problems:

7. [Dasgupta et al., Ex. 0.3] The Fibonacci numbers \(F_0, F_1, F_2, \ldots\) are defined by the rule

\[
F_0 = 0, \quad F_1 = 1, \quad F_n = F_{n-1} + F_{n-2} \quad (n \geq 2).
\]

In this problem we will confirm that this sequence grows exponentially fast and obtain some bounds on its growth.

(a) Use induction to prove that \(F_n \geq 2^{0.5n}\) for \(n \geq 6\).

(b) Find a constant \(c < 1\) such that \(F_n \leq 2^{cn}\) for all \(n \geq 0\). Show that your answer is correct.

(c) What is the largest \(c\) you can find for which \(F_n = \Omega(2^{cn})\)?

Challenge problem:

8. In the setting of Problem 1, show that there is some constant \(c > 0\) such that \(T(n) \leq cF_n\) for all \(n \geq 1\). (Numerically one can establish that this holds e.g. for \(c = 4\).)