Note: These are just notes and not necessarily full solutions for each exercise. Please report any mistakes you may find.

**Exercise 1**
The following list shows the functions in “ascending order”: If $f$ is shown to the left of $g$, then $f = O(g)$. If they are separated by a semicolon, then $f = o(g)$; and if they are separated by a comma, then $f = \Theta(g)$.

\[13; \log(n^{1337}), \log n; \log^2 n; \sqrt{n}; 2^{3+\log n}, 3n; \log n^n; e^{\log n}; 2^{4\log n}; n^6 - 5n^2; 2^{\log^2 n}; 2^n; 4^n.\]

Proofs are left to the reader. The function $-n^6 + 5n^2$ is strictly negative for $n \geq 2$, hence definitions of the big O notation do not apply.

**Exercise 2**
Let $f, g : \mathbb{N} \to \mathbb{R}^+$. To show: $f = O(g)$ if and only if $\limsup_{n \to \infty} \frac{f(n)}{g(n)} < \infty$.

Let $f = O(g)$. By definition, there is a $c > 0$ and an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ we have $f(n) \leq cg(n)$. This means that $\frac{f(n)}{g(n)} \leq c$ for all $n \geq n_0$, hence $\sup_{n \geq n_0} \frac{f(n)}{g(n)} \leq c$. As the limsup can only decrease from this value, we get $\limsup_{n \to \infty} \frac{f(n)}{g(n)} \leq c < \infty$.

Now suppose $\limsup_{n \to \infty} \frac{f(n)}{g(n)} = c < \infty$. Clearly $c \geq 0$, since $f$ and $g$ are strictly positive functions. By definition, this means that for all $\epsilon > 0$ there exists an $n_0 \in \mathbb{N}$ such that for all $n_1 \geq n_0$ one has $|\sup \left\{ \frac{f(n)}{g(n)} \mid n \geq n_1 \right\} - c| \leq \epsilon$. In particular, $\sup \left\{ \frac{f(n)}{g(n)} \mid n \geq n_0 \right\} \leq c + \epsilon$. Let us choose $\epsilon = 1$. Then it follows that for all $n \geq n_0$ we have $\frac{f(n)}{g(n)} \leq c + 1$, i.e. $f(n) \leq (c + 1)g(n)$ and thus $f = O(g)$.

**Exercise 3**

(a) Let $f, g : \mathbb{N} \to \mathbb{R}^+$ with $f \sim g$. This implies that $\forall \epsilon > 0$, there exists $N \in \mathbb{N}$ such that $|\frac{f(n)}{g(n)} - 1| \leq \epsilon$ for all $n \geq N$. Choosing $\epsilon = 1$, we deduce that $f(n) \leq 2g(n)$ for each $n \geq N$. Since $\sim$ is clearly symmetric, $g(n) \leq 2f(n)$ for $n \geq N'$, where $N'$ is an appropriate natural number. Hence $f = \Theta(g)$. The converse is in general false, take e.g. $f(n) = n$ and $g(n) = 2n$. 


Exercise 4

\( n \log n = \log n^n \geq \log(n!) \) for all \( n \in \mathbb{N} \), so \( \log(n!) = O(n \log n) \).

Conversely, we have

\[
\log(n!) = \log(n \cdot (n-1) \cdot \ldots \cdot 1) = \sum_{i=2}^{n} \log i = \sum_{i=2}^{n} \int_{i-1}^{i} \log x \, dx = \int_{1}^{n} \log x \, dx
\]

Hence \( \log(n!) \in \Omega(\log n) \), which implies that all three relations are true.

Exercise 5

Let us call \( s_1(n) = T_1(n) \cdot 10^{-10} \) and \( s_2(n) = T_2(n) \cdot 10^{-6} \) the number of seconds to process an input of size \( n \) in the first and second scenarios, respectively. One easily sees that \( s_1 \) starts out smaller but grows faster than \( s_2 \). So where is the point where \( s_1 \) and \( s_2 \) reach parity?

\[
s_1(n) = s_2(n) \implies 5 \cdot 10^{-10} \cdot n^2 = 1000 \cdot 10^{-6} \cdot n \log n \implies \frac{n}{\log n} = 2 \cdot 10^6
\]

There is no elementary closed form solution to such an equation, so we have to do it numerically using e.g. Python. We obtain that the first scenario is faster until a problem size of about \( 5.1 \cdot 10^7 \), or 51 millions, at which point both settings take a little more than 15 days to run.