Asymptotic Analysis (12 points)

1. Show that

   (a) [1 point] $5n^2 = O\left(n^2\right)$

   Solution: $\forall n \geq n_0 : 5n^2 \leq c \cdot n^2 \Leftrightarrow c = 5$ and $n_0 = 1$

   (b) [1 point] $n = O\left(n^2\right)$

   Solution: $\forall n \geq n_0 : n \leq c \cdot n^2 \Leftrightarrow c = 1$ and $n_0 = 1$

   (c) [1 point] $5n^2 = \Omega\left(n^2\right)$

   Solution: $\forall n \geq n_0 : 5n^2 \geq c \cdot n^2 \Leftrightarrow c = 5$ and $n_0 = 1$

   (d) [1 point] $n \neq \Theta\left(5n^2\right)$

   Solution: Suppose $n = \Theta\left(5n^2\right) \Leftrightarrow c_1 \cdot 5n^2 \leq n \leq c_2 \cdot 5n^2$ for all $n \geq n_0$. But then $\forall n \geq n_0 : c_2 \geq \frac{5n^2}{n} = 5n$ which is impossible.

   (e) [1 point] $a^n = O\left(b^n\right), b > a > 1$

   Solution: $\forall n \geq n_0 : a^n \leq c \cdot b^n \Leftrightarrow c = 1$ and $n_0 = 1$. Because as $b > a > 1$, $\forall n \geq 1 : b^n \geq a^n$.

2. For each of the following statements, decide whether it is **always true**, **never true**, or **sometimes true** for asymptotically nonnegative functions f and g . If it is **always true** or **never true**, explain why. If it is **sometimes true**, give one example for which it is true, and one for which it is false.

   (a) [1 point] $f(n) = O\left(f(n)^2\right)$

   Solution: Sometimes true: For $f(n) = n$ it is true, while for $f(n) = 1/n$ it is not true. (The statement is always true for $f(n) = \Omega(1)$, and hence for most functions with which we will be working in this course, and in particular all time and space complexity functions).

   (b) [1 point] $f(n) + g(n) = \Theta(\max(f(n), g(n)))$

   Solution: Always true: $\max(f(n), g(n)) \leq f(n) + g(n) \leq 2\max(f(n), g(n))$

   (c) [1 point] $f(n) = \Omega(g(n))$ and $f(n) = o(g(n))$ (note the little-o notation: for any real constant $c > 0$, there exists an integer constant $n_0 \geq 1$ such that $0 \leq f(n) < c \cdot g(n)$)

   Solution: Never true: If $f(n) = \Omega(g(n))$ then there exists positive constant c_Ω and n_Ω such that for all $n > n_\Omega, cg(n) \leq f(n)$. But if $f(n) = o(g(n))$, then for any positive constant c , there exists $n_o(c)$ such that for all $n > n_o(c), f(n) < cg(n)$. If $f(n) = \Omega(g(n))$ and $f(n) = o(g(n))$, we would have that for $n > \max(n_\Omega, n_o(c_\Omega))$ it should be that $f(n) < c_\Omega g(n) \leq f(n)$ which cannot be.

3. [1 point] What is the smallest value of n such that an algorithm whose running time is $100n^2$ runs faster than an algorithm whose running time is 2^n on the same machine?

Solution: We want that $100n^2 < 2^n$. Note that if $n = 14$, this becomes $100(14)^2 = 19600 > 2^14 = 16384$. For $n = 15$ it is $100(15)^2 = 22500 < 2^{15} = 32768$. So, the answer is $n = 15$.

4. [1 point] Express the function $n^3/1000 - 100n^2 - 100n + 3$ in terms of Θ-notation.

Solution: $n^3/1000 - 100n^2 - 100n + 3 \in \Theta\left(n^3\right)$

5. [2 points] Is $2^{n+1} = O(2^n)$? Is $2^{2n} = O(2^n)$?

Solution: $2^{n+1} \geq 2 \cdot 2^n$ for all $n \geq 0$, so $2^{n+1} = O(2^n)$. However, 2^{2n} is not $O(2^n)$. If it was, there would exist n_0 and c such that $n \geq n_0$ implies $2^n \cdot 2^n = 2^{2n} \leq c2^n$, so $2^n \leq c$ for $n \geq n_0$ which is clearly impossible since c is a constant.