Exercise 1

Due: 20/03/2019

Question 1

Find and prove the limit at ∞ using the definition for each of the following functions:

1. $\log(n)$
2. k^{-n} for some $k \in \mathbb{N} \setminus \{0\}$
3. $1 - \frac{1}{n}$

Question 2

Prove or give a counter example:

1. For every function $f : \mathbb{N} \to \mathbb{R}$, if $\lim_{n\to\infty} f(n) = \infty$then $\lim_{n\to\infty} \frac{1}{f(n)} = 0$
2. For every function $f : \mathbb{N} \to \mathbb{R}$, if $\lim_{n\to\infty} f(n) = 0$ then $\lim_{n\to\infty} \frac{1}{f(n)} = \infty$

Question 3

Prove the following using the definition of the asymptotic notations:

1. $\log(n^k) = O(n)$ for any $k \in \mathbb{N}$
2. $c_1 n + c_2 \cdot n^2 = \theta(n^2)$ for any $c_1, c_2 > 0$
3. $f(n) = \Omega(\log(f(n)))$ for any$f : \mathbb{N} \to \mathbb{R}_+$
4. $\sum_{i=1}^{n} i = \Theta(n^2)$
5. $\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$
6. $\sum_{i=1}^{\lfloor \log(n) \rfloor} \log\left(\frac{n}{2^i}\right) = \Theta(\log^2(n))$ (For the simplicity of the proof you may prove for n's such that $n = 2^k$ for some $k \in \mathbb{N}$)

Question 4

Prove the following for $f, g : \mathbb{N} \to \mathbb{R}_+$:

1. Transitivity:

$$f(n) = O(g(n)) \wedge g(n) = O(h(n)) \implies f(n) = O(h(n))$$

2. Symmetry:

$$f(n) = \Theta(g(n)) \iff g(n) = \Theta(f(n))$$

3. Transpose Symmetry:

$$f(n) = O(g(n)) \iff g(n) = \Omega(f(n))$$

Question 5

Prove the correctness of the effecient (correct) algorithm for finding a 1D local peak you saw in the lecture. You can find its pseudocode in the recitation summary (the routine named *Find_1D_Local_Peak* in the recitation summary in the moodle).

Question 6

1. In the recitation we saw the Greedy Ascent algorithm that finds a local 2D peak in a 2D array of n rows and m columns (see the algorithm "Find_2D_Local_Peak_Naive" in the recitation summary of week 1 in the course's moodle). In the recitation we saw an upper asymptotic bound for its worst-case running time of $O(nm)$. **Show a 2D array with 5 rows and 4 columns for which the run of this algorithm enters the loop at least 12 times.** (You are not required to prove this is in fact the case. Just draw/sketch the array). Note that the example you are to build may be generalised to show that there are inputs for which the run of the algorithm enters the loop at least $nm/2$ times. This implies that the worst-case running time of this algorithm is also $\Omega(nm)$ and altogether - that it's $\Theta(nm)$.

2. **Write an algorithm** that, given a 1D array of length n finds a cell with a global maximum (i.e. an index i_0 for which $\forall 0 \leq j \leq n : arr[j] \leq arr[i_0]$) and which has an asymptotic efficiency with the tight bound $\Theta(n)$. **Prove** both its correctness as an algorithm and the correctness of the tight bound.