EL9343 Homework 1

Due: Jan. 30th 5:00 p.m.

1. Prove the following properties of asymptotic notation:
   (a) $n = \omega(\sqrt{n})$
   (b) If $f(n) = \Omega(g(n))$, and $h(n) = \Theta(g(n))$, then $f(n) = \Omega(h(n))$
   (c) $f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$ (*Transpose Symmetry* property)
2. Indicate, for each pair of expressions (A, B) in the table below, whether A is O, o, Ω, ω, or Θ of B. Assume that $k \geq 1, \epsilon > 0$, and $c > 1$ are constants. Your answer should be **in the form of the table** with "yes" or "no" written in each box.

	A	B	O	o	Ω	ω	Θ
a	$\lg^k n$	n^ϵ					
b	n^k	c^n					
c	$\sqrt{n}$	$n^{\sin n}$					
d	2^n	$2^{n/2}$					
e	$n^{\lg c}$	$c^{\lg n}$					
f	$\lg(n!)$	$\lg(n^n)$					

3. You have 5 algorithms, A1 took $O(n)$ steps, A2 took $\Theta(n \log n)$ steps, and A3 took Ωn^2 steps, A4 took $o(n^3)$ steps, A5 took $\omega(n^{3/2})$ steps. You had been given the exact running time of each algorithm, but unfortunately you lost the record. In your messy desk you found the following formulas:
   (a) $4(5^{3\log_5 n}) + 12n + 9527$
   (b) $\sqrt[5]{3n!}$
   (c) $\frac{1}{6}(5^{\log_{16} n})^2 + 4n + 17$
   (d) $3n \log_3 n + (\log_2 n)^3$
   (e) $\log_4 \log_2 n + 61$
   (f) $2^{5\log_4 n}$
   (g) $(\log_2 n)^2 + \log_3 \log_3 n$

   For each algorithm check all the possible formulas that could be associated with it in the following table. Your submitted answer should be **in the form of the table**.

	a	b	c	d	e	f	g
A1							
A2							
A3							
A4							
A5							

4. We want to check if there is an element (called *majority element*) occurs more than $\frac{n}{2}$ times in an array containing n elements, assuming only equality checks are allowed.
   (a) Algo. 1 is part of the required algorithm. What is the time complexity now?
   (b) The algorithm shown is incomplete. Please show an example input where the output of the algorithm is not the majority element.
   (c) Make the algorithm complete by adding a few more lines to substitute the underlined text. Your modification should **NOT** change the time complexity. Be sure to return things as indicated.

Algorithm 1 Find majority element in an array

Input: $L[1, ..., n]$ as input list containing n real numbers
Output: True or False. If true, also returning the majority element

```
1: c = 0, v = L[1]
2: for i = 1, 2, ..., n do
3:     if c == 0 then
4:         v = L[i]
5:     end if
6:     if v == L[i] then
7:         c = c + 1
8:     else
9:         c = c − 1
10:    end if
11: end for
12: Future steps
```