EL9343 Homework 1

Due: Sept. 14th 5:00 p.m.

1. Prove the following properties of asymptotic notation:

   (a) $n = \omega(\sqrt{n})$
   Following the ω definition:

   $$\forall c > 0, \exists n_0 = c^2 + 1 > 0, \text{ such that } n > c\sqrt{n}, \forall n \geq n_0$$
   $$\therefore n = \omega(\sqrt{n})$$

   (b) If $f(n) = \Omega(g(n))$, and $h(n) = \Theta(g(n))$, then $f(n) = \Omega(h(n))$
   For the relationship between $h(n)$ and $g(n)$, $h(n) = \Theta(g(n)) \implies h(n) = O(g(n))$, and the big-Omega side is not needed in the proof:

   $$f(n) = \Omega(g(n)) \implies \exists c_1 > 0, n_1 > 0, \text{ such that } f(n) \geq c_1 g(n), \forall n \geq n_1$$
   $$h(n) = \Theta(g(n)) \implies \exists c_2 > 0, n_2 > 0, \text{ such that } h(n) \leq c_2 g(n), \forall n \geq n_2$$
   $$\therefore g(n) \geq \frac{1}{c_2} h(n), \forall n \geq n_2$$
   $$\therefore \exists c_3 = \frac{c_1}{c_2} > 0, n_3 = \max(n_1, n_2) > 0, \text{ such that } \forall n \geq n_3, f(n) \geq c_1 g(n) \geq c_1 \frac{1}{c_2} h(n) = c_3 h(n)$$
   $$\therefore f(n) = \Omega(h(n))$$

   (c) $f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$ (*Transpose Symmetry* property)
   For the if and only if type of questions, we must prove from both sides:

   $$\text{Part1: } f(n) = O(g(n)) \implies g(n) = \Omega(f(n))$$
   $$f(n) = O(g(n)) \Leftrightarrow \exists c_1 > 0, n_1 > 0, \text{ such that } f(n) \leq c_1 g(n), \forall n \geq n_1$$
   $$\therefore c_1' = \frac{1}{c_1}, \forall n \geq n_1, g(n) \geq c_1' f(n) \Leftrightarrow g(n) = \Omega(f(n))$$
   $$\text{Part2: } f(n) = O(g(n)) \impliedby g(n) = \Omega(f(n))$$
   $$g(n) = \Omega(f(n)) \Leftrightarrow \exists c_2 > 0, n_2 > 0, \text{ such that } g(n) \geq c_2 f(n), \forall n \geq n_2$$
   $$\therefore c_2' = \frac{1}{c_2}, \forall n \geq n_2, f(n) \leq c_2' g(n) \Leftrightarrow f(n) = O(g(n))$$

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	yes	yes	no	no	no
b	n^k	c^n	yes	yes	no	no	no
c	$\sqrt{n}$	$n^{\sin n}$	no	no	no	no	no
d	2^n	$2^{n/2}$	no	no	yes	yes	no
e	$n^{\lg c}$	$c^{\lg n}$	yes	no	yes	no	yes
f	$\lg(n!)$	$\lg(n^n)$	yes	no	yes	no	yes

A few notes:

$$\lg(n^{\lg c}) = \lg c \times \lg n = \lg(c^{\lg n}) \implies n^{\lg c} = c^{\lg n}$$
$$\text{By Stirling's approximation, } n! \approx n^n e^{-n} \sqrt{2\pi n}, \textit{and}$$
$$\lg(n!) \approx \lg(n^n e^{-n} \sqrt{2\pi n}) = n \lg n - n \lg e + \lg \sqrt{2\pi n} = \Theta(n \lg n) = \lg(n^n)$$

3. You have 5 algorithms, A1 took $O(n)$ steps, A2 took $\Theta(n \log n)$ steps, and A3 took $\Omega(n^2)$ 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.

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

(a) $4(5^{3\log_5 n}) + 12n + 9527$

$$4(5^{3\log_5 n}) + 12n + 9527 = \Theta(5^{3\log_5 n} + n) = \Theta(n^3 + n) = \Theta(n^3)$$

(b) $\sqrt[5]{3n!}$

$$\text{By Stirling's approximation, } n! \approx n^n e^{-n}\sqrt{2\pi n}$$
$$\sqrt[5]{3n!} \approx \sqrt[5]{3}(2\pi n)^{1/10}(\frac{n}{e})^{n/5} = \Omega(n^n)$$

(c) $\frac{5^{\log_{16} n}{}^2}{6} + 4n + 17$

$$\frac{5^{\log_{16} n}{}^2}{6} + 4n + 17 = \Theta(n^{\log_{16} 25} + n) = \Theta(n^{\log_{16} 25}) \approx \Theta(n^{1.20})$$

(d) $3n\log_3 n + (\log_2 n)^3$

$$3n\log_3 n + (\log_2 n)^3 = \Theta(n\log n)$$

(e) $\log_4 \log_2 n + 61$

$$\log_4 \log_2 n + 61 = \Theta(\log\log n)$$

(f) $2^{5\log_4 n}$

$$2^{5\log_4 n} = \Theta(n^{2.5})$$

(g) $(\log_2 n)^2 + \log_3 \log_3 n$

$$(\log_2 n)^2 + \log_3 \log_3 n = \Theta((\log_2 n)^2)$$

4. We want to check if there is an 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?
The time complexity is $\Theta(n)$. There is only a for-loop in the algorithm, and the time for operations inside the loop is constant.

(b) 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. As in the algorithm. **Re-counting the appearance of element v is a must.** If you check with **if** $c > \frac{n}{2}$ after counting, you will get full points this time, but in fact you are also using inequality checks. One way to avoid is shown.

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: Need to check whether the element in v is the wanted one
13: c = 0
14: if n%2 == 1 then
15:    t = (n+1)/2        # n is odd
16: else
17:    t = n/2 + 1
18: end if
19: for i = 1, 2, ..., n do
20:    if L[i] == v then
21:       c = c + 1
22:    end if
23:    if c == t then
24:       return True, v
25:    end if
26: end for
27: return False
```