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	$\overset{\leq}{O}$	$\overset{<}{o}$	$\overset{\geq}{\Omega}$	$\overset{>}{\omega}$	$\overset{=}{\Theta}$
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

2、

$\lg(\lg^* n)$	$2^{\lg^* n}$	$(\sqrt{2})^{\lg n}$	n^2	$n!$	$(\lg n)!$
$(\frac{3}{2})^n$	n^3	$\lg^2 n$	$\lg(n!)$	2^{2^n}	$n^{1/\lg n}$
$\ln \ln n$	$\lg^* n$	$n \cdot 2^n$	$n^{\lg \lg n}$	$\ln n$	1
$2^{\lg n}$	$(\lg n)^{\lg n}$	e^n	$4^{\lg n}$	$(n+1)!$	$\sqrt{\lg n}$
$\lg^*(\lg n)$	$2^{\sqrt{2 \lg n}}$	n	2^n	$n \lg n$	$2^{2^{n+1}}$

$1 = n^{1/\lg n} < \lg(\lg^* n) < \lg^* n < \lg^*(\lg n) < 2^{\lg^* n} < \ln(\ln n)$

$< \sqrt{\lg n} < \lg n < \lg^2 n < 2^{\sqrt{2\lg n}} < (\sqrt{2})^{\lg n} < n = 2^{\lg n} < n\lg n = \lg(n!)$

$< n^2 = 4^{\lg n} < n^3 < n^{\lg(\lg n)} = (\lg n)^{\lg n} < (\lg n)! < (\frac{3}{2})^n < 2^n < e^n < n2^n$

$< n! < (n+1)! < 2^{2^n} < 2^{2^{n+1}}$

3.

Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove or disprove each of the following conjectures.

a. $f(n) = O(g(n))$ implies $g(n) = O(f(n))$.

b. $f(n) + g(n) = \Theta(\min(f(n), g(n)))$.

c. $f(n) = O(g(n))$ implies $\lg(f(n)) = O(\lg(g(n)))$, where $\lg(g(n)) \geq 1$ and $f(n) \geq 1$ for all sufficiently large n.

d. $f(n) = O(g(n))$ implies $2^{f(n)} = O\left(2^{g(n)}\right)$.

a. false $\quad n = O(n^3) \quad O(n^3) \neq n$

b. false $\quad n + n^3 \neq \Theta(n) \quad (n>1)$

c. true $\quad 0 \leq f(n) \leq c_1 g(n)$

easy to prove that : $0 \leq \lg(f(n)) \leq \lg(c_1) + \lg(g(n))$

$c_2 \lg(g(n))$, $c_2 = 1 + \lg(c_1)$

for $f(n) \geq 1$ and $\lg(g(n)) \geq 1$

d. false $\quad 2n = O(n) \quad 2^{2n} \neq O(2^n)$

4.

1-4 Consider sorting n numbers stored in array A by first finding the largest element of A and exchanging it with the element in A[n]. Then find the second largest element of A, and exchange it with A[n-1]. Continue in this manner for all n elements of A. Write pseudocode for this algorithm, and answer the following questions: What loop invariant does this algorithm maintain? Give the best-case and worst-case running times of selection sort in asymptotic notation.

(1) Pseudocode:

```
SELECTION-SORT(A)
 for j = n to 1
   max_p = 1
   for i = 1 to j
     if A[i] > A[max_p]
        max_p = i
   swap ( A[max_p], A[j] )
```

(2) loop invariant

A[j...n] 按从小到大的顺序依次排列

(3) best-case:

$O(n^2)$

(4) worst-case:

$O(n^2)$