Problem 1 ***(25 points)*** For each of the following pairs of functions f and g, state ***(2 points)*** whether (i) $f(n) \in \Theta(g(n))$ or, if $f(n) \notin \Theta(g(n))$, then whether (ii) $f(n) \in O(g(n))$, or (iii) $f(n) \in \Omega(g(n))$, or (iv) none of the above. *Justify* your answers ***(3 points)***.

Exam 1a

(a)	$f(n)$	=	$n^{1/2}$	$g(n)$	=	$n^{3/2}$
(b)	$f(n)$	=	$n\log_2 n$	$g(n)$	=	$10n\log_2(10n)$
(c)	$f(n)$	=	2^n	$g(n)$	=	2^{n+1}
(d)	$f(n)$	=	n^3	$g(n)$	=	$n^{(1+\sin(n))}$
(e)	$f(n)$	=	$4^{\log_2 n}$	$g(n)$	=	n^2

Answers for Problem 1 (Quiz 1a). LR = the limit rule

(a) O. By LR, as: $\lim_{n\to\infty} \frac{n^{1/2}}{n^{3/2}} = \lim_{n\to\infty} \frac{1}{n} = 0$.

(b) Θ. By LR, as: $\lim_{n\to\infty} \frac{n\log_2 n}{10n\log_2(10n)} = \lim_{n\to\infty} \frac{1}{10(1+\frac{\log_2 10}{\log_2 n})} = \frac{1}{10}$.

(c) Θ. By LR, as: $\lim_{n\to\infty} \frac{2^n}{2^{n+1}} = \lim_{n\to\infty} \frac{1}{2} = \frac{1}{2}$.

(d) Ω. By LR, as: $\frac{n^3}{n^{1+\sin(n)}} \geq \frac{n^3}{n^2} = n$ and $\lim_{n\to\infty} n = \infty$.

(e) Θ. By LR, as: $f(n) = 4^{\log_2 n} = 2^{2\log_2 n} = 2^{\log_2 n^2} = n^2 = g(n)$.

Exam 1b

(a)	$f(n)$	=	n^3	$g(n)$	=	$2^{2^{2^{100}}} n^2$
(b)	$f(n)$	=	$n(\log_2 n)^2$	$g(n)$	=	$n\log_2(n^2)$
(c)	$f(n)$	=	3^n	$g(n)$	=	2^{n+1}
(d)	$f(n)$	=	n^2	$g(n)$	=	$n^{(4+\cos(n))}$
(e)	$f(n)$	=	n^2	$g(n)$	=	$3^{\log_2 n}$

Answers for Problem 1 (Quiz 1a). LR = the limit rule

(a) Ω. By LR, as: $\lim_{n\to\infty} \frac{n^3}{C\cdot n^2} = \lim_{n\to\infty} \frac{n}{C} = \infty$, where $C = 2^{2^{2^{100}}}$.

(b) Ω. By LR, as: $\lim_{n\to\infty} \frac{n(\log_2 n)^2}{n\log_2(n^2)} = \lim_{n\to\infty} \frac{\log_2 n}{2} = \infty$.

(c) Ω. By LR, as: $\lim_{n\to\infty} \frac{3^n}{2^{n+1}} = \lim_{n\to\infty} \frac{1}{2}(\frac{3}{2})^n = \infty$.

(d) O. By LR, as: $\frac{n^2}{n^{4+\cos(n)}} \leq \frac{n^2}{n^3} = \frac{1}{n}$ and $\lim_{n\to\infty} \frac{1}{n} = 0$.

(e) Ω. By LR, as: $3^{\log_2 n} = 2^{(\log_2 3)(\log_2 n)} = n^{\log_2 3}$ and $\lim_{n\to\infty} \frac{n^2}{n^{\log_2 3}} = \infty$, since $\log_2 3 < 2$.

Problem 2 ***(20 points)*** The following algorithm takes an array $A[i..j]$ of numbers (where $i \leq j$) and returns a pair of numbers $(\min, \max)$ where min is the smallest value in $A[i..j]$ and max is the largest value in $A[i..j]$. Each comparison in the algorithm is in a box.

```
function minMax(A[i..j])
  if j ≤ i+1 then    // So, n ≤ 2
    if [A[i] ≤ A[j]] then return (A[i],A[j]) else return (A[j],A[i])
  else    // So, n > 2.
    k ← i + ⌊(j−i)/2⌋
    (min1, max1) ← minMax(A[i..k])
    (min2, max2) ← minMax(A[k+1..j])
    if [min1 ≤ min2] then min ← min1 else min ← min2
    if [max1 ≥ max2] then max ← max1 else max ← max2
    return (min, max)
```

(a) ***(5 points)*** Let $n = j - i + 1$, i.e., the number of elements in $A[i..j]$, and $T(n) =$ the total number of comparisons done in running minMax on array of size n, where $n \geq 1$. Which of the following is a recurrence for $T(n)$? ***(2 points)*** *Justify* your answer. ***(3 points)***

(i) $t(1) = t(2) = 1$ and $t(n) = 2t(n/2) + 2$ when $n > 2$.

(ii) $t(1) = t(2) = 3$ and $t(n) = 2t(n/2) + 3$ when $n > 2$.

(iii) $t(1) = t(2) = 1$ and $t(n) = 2t(n/3) + 3$ when $n > 2$.

(b) ***(5 points)*** Use the Master Method Theorem to give a Θ-bound on $t(n)$. Spell out what a, b, and k are.

(c) ***(10 points)*** Prove by induction that $t(n) = \frac{3}{2}n - 2$ when $n \geq 2$ and n is a power of two. (Base case, ***(3 points)***; induction step, ***(7 points)***)

Answers for Problem 2.

(a) (i) since for $n = 1, 2$ there is just one comparison and, for $n > 2$, there are two comparisons plus two recursive calls on half-sized inputs.

(b) $a = b = 2$ and $k = 0$. So, $a = 2 > 1 = 2^0 = b^k$ and, by the Master Theorem, $T(n) = \Theta(n^{\log_2 2}) = \Theta(n^1) = \Theta(n)$.

(c) *Base case:* $n = 2$. Then $T(n) = 1 = \frac{3}{2} \cdot 2 - 2$.

Induction step: $n = 2^{k+1}$. **IH:** Suppose $T(2^k) = \frac{3}{2} \cdot 2^k - 2$. Then

$$T(2^{k+1}) = 2T(\frac{2^{k+1}}{2}) + 2 = 2T(2^k) + 2 \overset{\text{by IH}}{=} 2(\tfrac{3}{2} \cdot 2^k - 2) + 2 = \tfrac{3}{2} \cdot 2^{k+1} - 2.$$

Problem 3 (Exam 1a) ***(20 points)*** Consider a sorted array of numbers $A[1..n]$ where we have $A[1] < 0 < A[n]$. Provide the pseudocode for a ***correct*** $O(\log n)$-time algorithm that, given such an $A[1..n]$, finds the ***least*** $i \in \{1,\ldots,n\}$ such that $0 < A[i]$. ***(10 points)***

Justify the $O(\log n)$ run-time. ***(10 points)***

EXAMPLE: For $A[1..7] = [-8,-3,-1,2,5,6,9]$, $i = 4$.

Answer for Problem 3 (Quiz 1a). Here is a variant of binary search that suffices.

```
return find(A, 1, n)
function find(A, i, j)    // Invariant: A[j] > 0 and A[k] ≤ 0 for each k < i.
    if i = j then return i
    m ← i + ⌊(j − i)/2⌋
    if A[m] > 0 then return find(A, i, m) else return find(A, m + 1, j)
```

Runtime: This is a divide-and-conquer algorithm with $a = 1$, $b = 2$, and $k = 0$. So $a = 1 = 2^0 = b^k$; thus by the Master Theorem, $T(n) = \Theta(n^k \log n) = \Theta(\log n)$.

Problem 3 (Exam 1b) ***(20 points)*** Consider a sorted array $A[1..n]$ in which each $A[i]$ is either 0 or 1, e.g.: $A[1..7] = [0,0,0,0,1,1,1]$. Provide the pseudocode for a ***correct*** $O(\log n)$-time algorithm that, given such an $A[1..n]$, finds how many 1's the array contains. ***(10 points)***

Justify the $O(\log n)$ run-time. ***(10 points)***

EXAMPLE: For $A[1..7] = [0,0,0,1,1,1,1]$, there are four 1's.

Answer for Problem 3 (Quiz 1b).. Here is a variant of binary search that suffices.

```
if A[n] = 0 then return 0 else return find(A, 1, n)
function find(A, i, j)    // Invariant: A[j] = 1 and A[k] = 0 for each k < i.
    if i = j then return n − (i − 1)
    m ← i + ⌊(j − i)/2⌋
    if A[m] = 1 then return find(A, i, m) else return find(A, m + 1, j)
```

Runtime: This is a divide-and-conquer algorithm with $a = 1$, $b = 2$, and $k = 0$. So $a = 1 = 2^0 = b^k$; thus by the Master Theorem, $T(n) = \Theta(n^k \log n) = \Theta(\log n)$.

Problem 4 ***(20 points)*** For a directed graph $G = (V,E)$ and $v \in V$:

$\text{outDegree}(v) =_{\text{def}}$ the number of edges of the form $(v,u) \in E$.

$\mathit{inDegree}(v) =_{\text{def}}$ the number of edges of the form $(u,v) \in E$.

EXAMPLE: See the example graph in the Math Facts section.

(a) (10 points) Suppose you are given an *adjacency list* representation of G. Give an $O(|V|+|E|)$ time algorithm for that constructs a table that gives the out-degree of each vertex. ***(5 points)*** *Hint:* This is quite easy.
Justify the running time. ***(5 points)***

(b) (10 points) Suppose you are given an *adjacency list* representation of G. Provide the pseudocode for an $O(|V|+|E|)$-time algorithm for that constructs a table that gives the in-degree of each vertex. ***(5 points)*** *Hint:* This is a little more work.
Justify the running time. ***(5 points)***

Answer for 4(a+b). We compute both in- and out-degrees at once. *Strategy:* Scan the edges and tally the number of edges entering and leaving each vertex.

```
1. for each u ∈ V do
2.     outDeg[u] ← 0;    inDeg[u] ← 0.
3. for each u ∈ V do
4.     for each v ∈ AdjList[u] do    // Consider edge (u, v)
5.         outDeg[u] ← 1 + outDeg[u];    inDeg[v] ← 1 + inDeg[v]
6. return (outDeg, inDeg)
```

Runtime: We execute line 2 once for each $u \in V$ and we execute line 5 once for each $e \in E$. Hence we have a $O(|V|+|E|)$ runtime.

Problem 5 ***(15 points)*** BACKGROUND. In a directed graph $G = (V,G)$, a *super-sink* is an $s \in V$ with $\text{outDegree}(s) = 0$ and $\text{inDegree}(s) = |V| - 1$.

EXAMPLE: In the Math-Facts example graph, vertex x is a super-sink.

YOUR TASK. Let $G = (V,E)$ be a directed graph, with $V = \{0,\ldots,n-1\}$. Suppose we represent G through an *adjacency matrix*. Provide the pseudocode for a $\Theta(n)$-time algorithm what, given G, finds the super-sink vertex of G, if it exists, and returns -1, if G fails to have a super-sink vertex. ***(8 points)***

Justify the running time. ***(7 points)***

Hints: If $(u,v) \in E$, what does that tell us about u and/or v? If $(u,v) \notin E$, what does that tell us about u and/or v?

Answer for Problem 5. Note:

- if $(i,j) \in E$, then i cannot be a super-sink.
- if $(i,j) \notin E$, then j cannot be a super-sink.

So:

```
i ← 1      // i is our possible super-sink
for j ← 2,...,n do
    if AdjMat[i,j]
       then i ← j     // i isn't a super-sink, try j.
       else {}        // j isn't a super-sink.
// i is the only possible super-sink, so check it
for j ← 1,...,n do
    if j ≠ i and (AdjMat[i,j] or (not AdjMat[j,i])) return −1
return i
```

Runtime: The first loop has $(n-1)$-many iterations and the second loop has n-many iterations. The bodies of both loops are constant time. Hence the runtime is $\Theta(n)$.

Distribution of Scores

```
Range   Scores
17-20:  20
21-24:
25-28:
29-32:
33-36:  33 33 35
37-40:  38 38 39 39
41-44:  44
45-48:  45 45 47 47
49-52:  51
53-56:  53 54 56 56
57-60:  60
61-64:  61 62 62 63 63 63 64 64
65-68:  65 66 66 66 67 68 68
69-72:  69 69 71 71 72 72
73-76:  73 74 74 75 75 76
77-80:  77 77 78 79 80 80 80
81-84:  81 81 82 84 84
85-88:  85 85 85 87 87 87
89-92:  89 89 90 92 92
93-96:  94 94
97-100: 97

Average: 67.89      Median: 71.0
```

Some Reference Math Facts

a. $a^m \cdot a^n = a^{m+n}$.

b. $a^{m \cdot n} = (a^m)^n = (a^n)^m$.

c. $a^n \cdot b^n = (a \cdot b)^n$.

d. $\log_a a^n = n$.

e. $a^{\log_a n} = n$.

f. $c \cdot \log_2 a = \log_2 a^c$.

g. $\log_2(a \cdot b) = (\log_2 a) + (\log_2 b)$.

h. $(\log_a x)/(\log_a b) = \log_b x$.

i. $a^{\log_b c} = c^{\log_b a}$.

j. $\sum_{k=1}^{n} k = \frac{1}{2} \cdot n \cdot (n+1)$.

k. $\sum_{k=0}^{n} a^k = \frac{a^{n+1}-1}{a-1}$.

l. $\sum_{k=1}^{\infty} 2^{-k} = 1$.

m. $\sum_{k=1}^{n} k^2 = \frac{n \cdot (n+1) \cdot (2n+1)}{6}$.

n. $\sum_{k=1}^{\infty} k \cdot 2^{-k} = 2$.

o. For $a > 1$ and $b, c > 0$:

$$\lim_{n \to \infty} \frac{(\log_a n)^b}{n^c} = 0.$$

p. For $a > 1$ and $b, c > 0$:

$$\lim_{n \to \infty} \frac{n^b}{a^{c \cdot n}} = 0.$$

q. If $\lim_{n \to \infty} \frac{f(n)}{g(n)} = c$, then:

	$c = 0$	$0 < c < \infty$	$c = \infty$
$f(n) \in O(g(n))$	True	True	False
$f(n) \in \Theta(g(n))$	False	True	False
$f(n) \in \Omega(g(n))$	False	True	True

r. *The Master Recurrence Theorem:* Suppose for all $n > n_0$, we have $T(n) = a \cdot T(n/b) + c \cdot n^k$ for $a, b \geq 1$ and $k \geq 0$, then:

(i) if $a < b^k$, then $T(n) \in \Theta(n^k)$;
(ii) if $a = b^k$, then $T(n) \in \Theta(n^k \log n)$; and
(iii) if $a > b^k$, then $T(n) \in \Theta(n^{\log_b a})$.

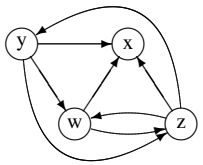
Above, we can take n/b can be taken as either $\lfloor n/b \rfloor \pm c$ or $\lceil n/b \rceil \pm c$ for constant c.

Floor and ceiling. For each $x \in \mathbb{R}$:

$\lfloor x \rfloor =_{\text{def}} \max\{n \in \mathbb{Z} : n \leq x\}$. (E.g., $\lfloor 2.5 \rfloor = 2$.)

$\lceil x \rceil =_{\text{def}} \min\{n \in \mathbb{Z} : n \geq x\}$. (E.g., $\lceil 2.5 \rceil = 3$.)

Example graph.



Vertex	w	x	y	z
outDegree	2	0	3	3
inDegree	2	3	1	2