1. (15 points) For each pair of functions in the table below, determine whether \( f(n) \in O(g(n)) \), 
\( f(n) \in \Omega(g(n)) \), \( f(n) \in \Theta(g(n)) \), or all of them. It is NOT necessary to justify your answer.

<table>
<thead>
<tr>
<th>( f(n) )</th>
<th>( O )</th>
<th>( \Omega )</th>
<th>( \Theta )</th>
<th>( g(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n^2 + 3n + 4 )</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>( n^3 )</td>
</tr>
<tr>
<td>( n + \log n )</td>
<td>✓</td>
<td></td>
<td></td>
<td>( n \log n )</td>
</tr>
<tr>
<td>( n )</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>( n + \log(n^2) )</td>
</tr>
<tr>
<td>( 4^n )</td>
<td>✓</td>
<td></td>
<td></td>
<td>( 5^n )</td>
</tr>
<tr>
<td>( n^2 )</td>
<td>✓</td>
<td></td>
<td></td>
<td>( \log(n!) )</td>
</tr>
</tbody>
</table>

2. (8 + 6 points)
(a) (8 points) Use the basic definition of \( O \), prove that \( 6n^2 + 2n + 1 \in O(n^2) \)

\[
6n^2 + 2n + 1 \leq 6n^2 + 2n + n \quad \forall n \geq 1 \\
\leq 6n^2 + 3n \quad \forall n \geq 1 \\
\leq 6n^2 + 3n^2 \quad \forall n \geq 1 \\
\leq 9n^2 \quad \forall n \geq 1.
\]

Therefore, by definition, \( 6n^2 + 2n + 1 \in O(n^2) \).

(b) (Extra credit - 6 points) Given three asymptotically positive functions \( f(n) \), \( g(n) \), and \( h(n) \), prove that the following statement is correct: \( f(n) \in O(g(n)) \) and \( g(n) \in O(h(n)) \) imply that \( f(n) \in O(h(n)) \).

\[ (1) \quad f(n) \in O(g(n)) \Rightarrow f(n) \leq c_1 g(n) \quad \text{for some} \quad c_1 > 0 \quad \text{and} \quad \forall n \geq n_0. \]

\[ (2) \quad g(n) \in O(h(n)) \Rightarrow g(n) \leq c_2 h(n) \quad \text{for some} \quad c_2 > 0 \quad \text{and} \quad \forall n \geq n_2. \]

Combining (1) & (2), we have

\[ f(n) \leq c_1 g(n) \leq c_1 c_2 h(n) \quad \text{for} \quad \forall n \geq \max(n_0, n_2). \]

Therefore, if we choose \( c = c_1 c_2 \), we have \( f(n) \leq c h(n) \quad \forall n \geq n_0. \)

By definition, \( f(n) \in O(h(n)). \)
3a. (9 points) Analysis of iterative algorithm. You are given the following code snippet:

```java
int sum = 0;
for (int i = 1; i <= n; i = i + 1) {
    sum = sum + i;
}
print (sum);
```

a. How many times will line 3 be executed, as a function of $n$?

   (a) $\Theta(\log n)$
   (b) $\Theta(n^2)$
   (c) $\Theta(n)$
   (d) $\Theta(n \log n)$

b. What is the output of the program if $n = 16$?

   (a) 526
   (b) 136
   (c) 1985
   (d) 5860

   \[ \sum_{i=1}^{16} i \]

   (c) $\Theta(n)$
   (d) $\Theta(n \log n)$

   \[ \frac{(1+n) \cdot n}{2} \]
3b. (Extra credit - 9 points) Analysis of iterative algorithm. You are given the following code snippet:

```cpp
int sum = 0;
for (int i = 1; i <= n; i = i * 2) {
    sum = sum + i;
}
print (sum);
```

a. How many times will line 3 be executed, as a function of $n$?

(a) $\Theta(\log n)$
(b) $\Theta(n^2)$
(c) $\Theta(n)$
(d) $\Theta(n \log n)$

b. What is the output of the program if $n = 16$?

(a) 31
(b) 165
(c) 1085
(d) 5865

c. In general, what is the output of the program if $n = 2^k$, where $k$ is some positive integer?

\[
\sum_{i=0}^{k} 2^i = 2^{k+1} - 1 = 2 \cdot 2^k - 1 = 2n - 1
\]

d. Prove by induction that your answer above in (c) is correct.

**Proof:**

**Base Case:** $k = 0$, i.e., $n = 1$, output = $\sqrt{n} = 1$.

**Step:** Assume output is $2(2^{k-1}) - 1$ for $n = 2^{k-1}$, then output for $n = 2^k$ is:

\[
(2 \cdot 2^{k-1} - 1) + 2^k = 2 \cdot 2^k - 1 = 2n - 1.
\]
4. (25 + 5 points) Assume that \( T(1) \in \Theta(1) \). Solve the following recurrence functions using the **master method**. Mark the correct answer. You do NOT need to show all details, but showing so could get you some partial credits.

a. \( T(n) = 4T(n/2) + n^{3/2} \);
   \[ n^2 \text{ vs } n^{3/2} \]
   \[ \text{Case 1} \]
   (a) \( \Theta(n^2) \)
   (b) \( \Theta(n^{3/2}) \)
   (c) \( \Theta(n^4) \)
   (d) Master Method cannot be applied.
   (e) None of the above.

b. \( T(n) = 5T(n/2) + n^3 \);
   \[ n^{\log_2 5} \text{ vs } n^3 \]
   \[ \text{Case 3} \]
   (a) \( \Theta(n^2) \)
   (b) \( \Theta(n^{2.5}) \)
   (c) \( \Theta(n^3) \)
   (d) Master Method cannot be applied.
   (e) None of the above.

c. \( T(n) = 4T(n/2) + n \log n \);
   \[ n^2 \text{ vs } n \log n \]
   \[ \text{Case 1} \]
   (a) \( \Theta(n \log n) \)
   (b) \( \Theta(n^2) \)
   (c) \( \Theta(n \log^2 n) \)
   (d) Master Method cannot be applied.
   (e) None of the above.

d. \( T(n) = 2T(n/2) + n^2 \log n \);
   \[ n \text{ vs } n^2 \log n \]
   \[ \text{Case 3} \]
   (a) \( \Theta(n) \)
   (b) \( \Theta(n^2 \log n) \)
   (c) \( \Theta(n \log n) \)
(d) Master Method cannot be applied.
(e) None of the above.

\( T(n) = 4T(n/4) + n; \)

\( n \text{ vs } n \)

\( \text{Case 2} \)

(a) \( \Theta(n \log n) \)
(b) \( \Theta(n) \)
(c) \( \Theta(n^2) \)
(d) Master Method cannot be applied.
(e) None of the above.

f. (Extra credit - 5 points) \( T(n) = 2T(n-2) + n \log n; \) Show details if necessary.

(a) \( \Theta(n^2) \)
(b) \( \Theta(n^2 \log n) \)
(c) \( \Theta(\sqrt{2^n}) \)
(d) \( \Theta(2^n) \)
(e) \( \Theta(2^n \log n) \)

\[
\begin{align*}
T(n) &= 2T(n-2) + n \log n \\
S(m) &= 2S(m/4) + \log m \cdot \log m \\
S(m) &= \Theta(m^{1/2}) \\
T(n) &= (\sqrt{2})^n
\end{align*}
\]
5. (15 points) Assume that \( T(1) \in \Theta(1) \). Solve the following recurrence function using the recursion tree method.

\[
T(n) = 4T(n/2) + n
\]

\[
\begin{align*}
\frac{n}{2^2} & \cdot 4 = n \cdot 2 \\
\frac{n}{2^3} \cdot 4^2 = n \cdot 2^2 \\
\frac{n}{2^3} \cdot 4^3 = n \cdot 2^3 \\
\sum_{i=0}^{h} n \cdot 2^i = \Theta(n \cdot 2^{\log_2 n}) = \Theta(n^2)
\end{align*}
\]

6. (10 points) Assume that \( T(1) \in \Theta(1) \) and \( T(n) = T(n/3) + 2T(n/4) + n \). Prove that \( T(n) \in O(n) \) using the substitution method.

To prove \( T(n) \in O(n) \), we need to show \( T(n) \leq cn \) for some \( c > 0 \) and all \( n \geq n_0 \).

Assume this is true for \( T(n/3) \) and \( T(n/4) \).

We then have \( T(n/3) \leq c \cdot \frac{n}{3} \) and \( T(n/4) \leq c \cdot \frac{n}{4} \).

Then

\[
T(n) = T(n/3) + 2T(n/4) + n \\
\leq c \cdot \frac{n}{3} + 2 \cdot c \cdot \frac{n}{4} + n \\
\leq \frac{5}{6} cn + n \\
\leq cn + (n - \frac{f}{6} cn) \\
\leq cn \text{ if } n \leq \frac{f}{6} cn, \text{ i.e., } c > 6, \text{ and } n > 0.
\]

Therefore, by definition, \( T(n) \in O(n) \).
7. (18 points) Analysis of recursive algorithms.

Consider the pseudocode of the following two algorithms. In both algorithms, the input $A$ is an array of size $n$, which is then split into 3 subarrays of equal sizes to be used in subsequent recursive calls. Note that it takes constant time to split and pass an array.

\[
\begin{array}{ll}
\text{AlgX} (A[1..n]) & \quad \mathcal{O}(1) \\
\text{if } (n < 3) \text{ return sum}(A[1..n]); & \quad \mathcal{O}(1) \\
p = \text{floor}(n/3); & \quad \mathcal{O}(1) \\
/ * r \text{ is a random number}\ & \quad \mathcal{O}(1) \\
\text{between 0 and 1 } */ & \quad \mathcal{O}(1) \\
r = \text{rand}(); & \quad \mathcal{O}(1) \\
\text{if } (r < 0.5) & \quad \mathcal{O}(1) \\
\quad \text{return AlgX} (A[1..p]); & \quad \mathcal{O}(1) \\
\text{else} & \quad \mathcal{O}(1) \\
\quad b = \text{AlgX} (A[(p+1)..2p]) & \quad \mathcal{O}(1) \\
\quad c = \text{AlgX} (A[(2p+1)..n]) & \quad \mathcal{O}(1) \\
\quad \text{return } b + c; & \quad \mathcal{O}(1) \\
\end{array}
\]

\[
\begin{array}{ll}
\text{AlgY} (A[1..n]) & \quad \mathcal{O}(1) \\
\text{if } (n < 3) \text{ return sum}(A[1..n]); & \quad \mathcal{O}(1) \\
p = \text{floor}(n/3); & \quad \mathcal{O}(1) \\
r = \text{rand}(); & \quad \mathcal{O}(1) \\
a = \text{AlgY} (A[1..p]); T(n/3) & \quad \mathcal{O}(1) \\
b = \text{AlgY} (A[(p+1)..2p]); T(n/3) & \quad \mathcal{O}(1) \\
c = \text{AlgY} (A[(2p+1)..n]); T(n/3) & \quad \mathcal{O}(1) \\
\text{if } (r < 0.5) & \quad \mathcal{O}(1) \\
\quad \text{return } a; & \quad \mathcal{O}(1) \\
\text{else} & \quad \mathcal{O}(1) \\
\quad \text{return } b + c; & \quad \mathcal{O}(1) \\
\end{array}
\]

a. What is the worst-case running time of AlgX and AlgY, respectively? Write down the recurrences and solve them. Which algorithm is asymptotically more efficient?

\[
\begin{align*}
\text{AlgX:} & \quad T(n) = 2T(n/3) + \Theta(1) \\
& \quad \text{By M.M., } T(n) \in \Theta(n \log_3^2) \\
\text{AlgY:} & \quad T(n) = 3T(n/3) + \Theta(1) \\
& \quad \text{By M.M., } T(n) \in \Theta(n \log_3 2) \\
\text{AlgX is more efficient because } n \log_3^2 \ll n.
\end{align*}
\]
b. What is the best-case running time of AlgX and AlgY, respectively? Write down the recurrences and solve them.

**Alg X:**
\[ T(n) = T(n/3) + O(1) \]
\[ T(n) \in O(\log n) \]

**Alg Y:**
\[ T(n) = 3T(n/3) + O(1) \]
\[ T(n) \in \Theta(n) \]

c. What is the average-case running time of AlgX and AlgY, respectively? Write down the recurrences and solve them.

**Alg X:**
\[ \overline{T}(n) = 0.5(\overline{T}(n/3) + O(1)) + 0.5(2\overline{T}(n/3) + O(1)) \]
\[ = 1.5 \overline{T}(n/3) + O(1) \]
\[ \overline{T}(n) \in \Theta(n^{\log_{1.5} 3}) \]

**Alg Y:**
\[ \overline{T}(n) = 0.5(3\overline{T}(n/3) + O(1)) + 0.5(3\overline{T}(n/3) + O(1)) \]
\[ = 3 \overline{T}(n/3) + O(1) \]
\[ \overline{T}(n) \in \Theta(n) \]