1. (16 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>X</td>
<td></td>
<td></td>
<td>( n^3 )</td>
</tr>
<tr>
<td>( 100 + \log n )</td>
<td>X</td>
<td></td>
<td></td>
<td>( 10n )</td>
</tr>
<tr>
<td>( n + \log n )</td>
<td>X</td>
<td></td>
<td>( n \log n )</td>
<td></td>
</tr>
<tr>
<td>( n + n \log n )</td>
<td>X</td>
<td></td>
<td>( n^2 )</td>
<td></td>
</tr>
<tr>
<td>( n )</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>( n + \log(n^2) )</td>
</tr>
<tr>
<td>( n )</td>
<td></td>
<td>X</td>
<td></td>
<td>( 2^{\log_4 n} )</td>
</tr>
<tr>
<td>( 4^n )</td>
<td>X</td>
<td></td>
<td></td>
<td>( 5^n )</td>
</tr>
<tr>
<td>( n^2 )</td>
<td></td>
<td>X</td>
<td></td>
<td>( \log(n!) )</td>
</tr>
</tbody>
</table>

2. (10 points)
(a) Use the **basic definition** of \( O \), prove that \( 2n^2 + 5n + 1 \in O(n^2) \).

Proof sketch: \( 2n^2 + 5n + 1 \leq 2n^2 + n^2 + n^2 = 8n^2 \) for \( n \geq 1 \).

(b) 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 \( f(n) \in O(h(n)) \).

Proof sketch: \( f(n) \in O(g(n)) \) implies that \( f(n) < c_1 g(n) \) for \( n \geq n_01 \), and \( g(n) \in O(h(n)) \) implies that \( g(n) < c_2 h(n) \) for \( n \geq n_02 \). Therefore \( f(n) \leq c_1 g(n) \leq c_1 c_2 h(n) \) for \( n > \max(n_01, n_02) \).

3. (9 points) Choose the correct order of growth for the following sums.

a. \( \sum_{i=0}^{n} (2^i + 2i) \)
   \( \Theta(2^n) \)

b. \( \sum_{i=0}^{n} (i + \log i) \)
   \( \Theta(n) \)

c. \( \sum_{i=0}^{\log_2 n} (2^i)n \)
   \( \Theta(n^2) \)

4. (9 points) You are given the following iterative algorithm:
Enigma(A[1..n])

// Input: An array A[1..n] of n real numbers
x = A[1];
y = A[1];
for (i = 2; i <= n; i++)
    if (A[i] < x)
        x = A[i];
    if (A[i] > y)
        y = A[i];
end
return y - x;
end

a. Which line is the algorithm’s basic operation (i.e., the most executed line)?
   Line 5

b. What is the time complexity of the algorithm, as a function of n?
   $\Theta(n)$

c. What does this algorithm compute?
   The difference between the largest value and the smallest value in the array.
5. (29 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.

a. \( T(n) = 9T(n/3) + n^{3/2} \);  
   \( \Theta(n^2) \)

b. \( T(n) = 4T(n/2) + n^3 \);  
   \( \Theta(n^3) \)

c. \( T(n) = 3T(n/2) + n \log n \);  
   \( \Theta(n^{\log_2 3}) \)

d. \( T(n) = T(n/2) + n \log n \);  
   \( \Theta(n \log n) \)

e. \( T(n) = 2T(n/4) + \sqrt{n} + \log n \);  
   \( \Theta(\sqrt{n} \log n) \)

f. \( T(n) = T(n/4) + n \log n \);  
   \( \Theta(n \log n) \)

g. (Extra credit - 5 points) \( T(n) = 2T(n-2) + n^2 \); Show details if necessary.  
   \( \Theta(2^n) \)
6. (10 points) Assume that $T(1) \in \Theta(1)$. Solve the following recurrence function using the recursion tree method to get an asymptotically tight bound.

$$T(n) = 2T(n/2) + n^2$$

See lecture notes.

7. (10 points) Assume that $T(1) \in \Theta(1)$ and $T(n) = T(n/2) + 2T(n/3) + n^2$. Prove that $T(n) \in O(n^2)$ using the substitution method.

See lecture notes.

8. (15 points) Analysis of recursive algorithm. Consider the pseudocode of the following two algorithms. The input $A$ is an array of size $n$. In Alg1, $A$ is divided into two subarrays, and the algorithm is recursively applied to only one subarray. In Alg2, $A$ is divided into five subarrays, and the algorithm is recursively applied to two or three of them, depending on values in $A$. Assume that parameter passing takes constant time.

```
Alg1 (A[1..n])
  if (n <= 1) return;
  mid = floor(n/2);
  x = rand(); // 0 < x < 1
  if (x <= 0.5)
    Alg1 (A[1..mid]);
  else
    Alg1 (A[mid+1..n]);
end

Alg2 (A[1..n])
  if (n <= 1) return;
  s = floor(n/5);
  x = rand(); // 0 < x < 1
  if (x <= 0.5)
    Alg2 (A[1..s]);
    Alg2 (A[s+1..2s]);
  else
    Alg2 (A[2s+1..3s]);
    Alg2 (A[3s+1..4s]);
    Alg2 (A[4s+1..n]);
end
```

a. What is the worst-case running time of Alg1? What about best-case?
   Both are $\Theta(\log n)$.

b. What is the worst-case running time of Alg2? What about best-case?
   Worst case running time is $\Theta(n^{\log_5 3})$ and best case is $\Theta(n^{\log_5 2})$.

c. We have assumed that it parameter passing takes constant time. Now let’s say it actually takes $f(n) = \Theta(n)$ time to pass an array of size $n$. Re-do (a) and (b).
   The running time is now always $\Theta(n)$ for both Alg1 and Alg2 in worst and best case.
9. (10 points) Using substitution method, prove that the solution you got for the recurrence in 5(g) is tight.
   Details later.
Scratch paper
Scratch paper