1. Let \( f(n) \) and \( g(n) \) be asymptotically non-negative functions which are defined on the positive integers.
   a. State the definition of \( f(n) = O(g(n)) \).
   b. State the definition of \( f(n) = \omega(g(n)) \).

2. State whether the following assertions are true or false. If any statements are false, give a related statement which is true.
   a. True or False: \( f(n) = O(g(n)) \) implies \( f(n) = o(g(n)) \).
   b. True or False: \( f(n) = O(g(n)) \) if and only if \( g(n) = \Omega(f(n)) \).
   c. True or False: \( f(n) = \Theta(g(n)) \) if and only if \( \lim_{n \to \infty} (f(n)/g(n)) = L \), where \( 0 < L < \infty \).

3. Use Stirling’s formula: \( n! = \sqrt{2\pi n} \left( \frac{n}{e} \right)^n \cdot \left(1 + \Theta(1/n) \right) \), to prove that \( \lg(n!) = \Theta(n \lg n) \).

4. Use Stirling’s formula to prove that \( \left( \frac{2n}{n} \right)^n = \Theta \left( \frac{4^n}{\sqrt{n}} \right) \).

5. Consider the following sketch of an algorithm called \textbf{ProcessArray} which performs some unspecified operation on an array \( A[1 \cdots n] \).

\begin{verbatim}
 ProcessArray(A, p, q)    (Preconditions: \( p \geq 1 \) and \( q \leq \text{length}[A] \))
1. Do something which takes constant time.
2. If \( p < q \)
3. \( r \left\lfloor \frac{p+q-1}{2} \right\rfloor \)
4. ProcessArray(A, p, r)
5. ProcessArray(A, r+1, q)
\end{verbatim}

Write a recurrence which gives the running time \( T(n) \) of this algorithm, when called on the full array: \textbf{ProcessArray}(A, 1, n). Give a tight asymptotic solution to this recurrence.
6. Consider the following algorithm

\[
\text{WasteTime}(n)
\begin{align*}
1. & \quad \text{for } i \leftarrow 1 \text{ to } n^3 \\
2. & \quad \text{waste a constant amount of time} \\
3. & \quad \text{for } i \leftarrow 1 \text{ to } 7 \\
4. & \quad \text{WasteTime}\left(\left\lceil n/2 \right\rceil\right) \\
5. & \quad \text{fiddle around for constant time}
\end{align*}
\]

Write a recurrence which gives the running time \(T(n)\) of this algorithm. Give a tight asymptotic solution to this recurrence.

7. Use the Master Theorem to find asymptotic approximations to the solutions of the following recurrences.

a. \(T(n) = 2T(n/4) + \sqrt{n}\)

b. \(T(n) = 7T(n/3) + n^2\)

8. Complete the following algorithm called \textbf{HeapIncreaseKey}( A, i, k ) which sets \(A[i] \leftarrow \max(A[i], k)\), then updates the heap structure accordingly.

\[
\text{HeapIncreaseKey}( A, i, k ) \quad \text{(Pre: } 1 \leq i \leq \text{HeapSize}[A] \text{)}
\begin{align*}
1. & \quad \text{if } k \geq A[i] \\
2. & \quad \text{…………} \\
3. & \quad \text{…………}
\end{align*}
\]

9. Prove that an n element heap has exactly \(\left\lfloor n/2^h \right\rfloor - \left\lfloor n/2^{h+1} \right\rfloor\) nodes at height \(h\).

10. Let \(T(n)\) be the solution to the recurrence

\[
T(n) = \begin{cases} 
5 & n = 1 \\
10 & n = 2 \\
3T(\left\lfloor n/3 \right\rfloor) + n & n > 2 
\end{cases}
\]

Show that there exists a \(c > 0\) such that \(T(n) \leq cn \ln n\) for all \(n \geq 3\). Prove this using induction on \(n\) (not the Master Theorem.)
Some Problems on Heaps and Priority Queues:
p.129: 6.1-1, 2, 3, 4, 5, 6, 7
p.132: 6.2-1, 2, 3, 5, 6
p.135: 6.3-1, 2, 3

Some Problems on Graphs and Breadth First Search:
p. 530: 22.1-1,2
p. 538: 22.2-1, 2, 4