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:
\[
 n! = \sqrt{2\pi n} \left( \frac{n}{e} \right)^n \left(1 + \Theta(1/n)\right),
\]
to prove that \( \lg(n!) = \Theta(n\lg n) \).

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

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

\textbf{ProcessArray}(A, p, q) \hspace{1cm} (\text{Preconditions: } p \geq 1 \text{ and } q \leq \text{length}[A])

1. Do something which takes constant time.
2. If \( p < q \)
3. \[ r \leftarrow \left\lfloor \frac{p + q - 1}{2} \right\rfloor \]
6. \text{ProcessArray}(A, p, r)
7. \text{ProcessArray}(A, r+1, q)

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

\begin{verbatim}
WasteTime(n)
1. for i ← 1 to n^3
2. waste a constant amount of time
3. for i ← 1 to 7
4. WasteTime(\lceil n/2 \rceil)
5. fiddle around for constant time
\end{verbatim}

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

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

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

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

7. 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.

\begin{verbatim}
HeapIncreaseKey( A, i, k ) (Precondition: 1 \leq i \leq HeapSize[A])
1. A[i] \leftarrow \max(A[i], k)
2. .............
3. .............
\end{verbatim}

8. 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 \( h \).

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

\[
T(n) = \begin{cases} 
5 & n = 1 \\
10 & n = 2 \\
3T(\lceil n/3 \rceil) + n & n > 2 
\end{cases}
\]

Show that there exists a \( c > 0 \) such that \( T(n) \leq cn \lg n \) for all \( n \geq 3 \). Prove this using induction on \( n \) (not the Master Theorem.)

\textbf{More Problems on Heaps:}

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