1. (20 Points) Prove that if \( h_1(n) = \Theta(f(n)) \) and \( h_2(n) = \Theta(g(n)) \), then \( h_1(n)h_2(n) = \Theta(f(n)g(n)) \).

**Proof:**
We have:
\[
\exists \text{ positive } a_1, b_1, n_1 \text{ such that } \forall n \geq n_1: 0 \leq a_1 f(n) \leq h_1(n) \leq b_1 f(n)
\]
\[
\exists \text{ positive } a_2, b_2, n_2 \text{ such that } \forall n \geq n_2: 0 \leq a_2 g(n) \leq h_2(n) \leq b_2 g(n)
\]

Define \( a = a_1 a_2, b = b_1 b_2 \) and \( n_0 = \max(n_1, n_2) \). Then \( a, b \) and \( n_0 \) are positive. If \( n \geq n_0 \), then both of the above inequalities are true. Upon multiplying these inequalities, we get
\[
\exists \text{ positive } a, b, n_0 \text{ such that } \forall n \geq n_0: 0 \leq a f(n)g(n) \leq h_1(n)h_2(n) \leq b f(n)g(n)
\]
showing that \( h_1(n)h_2(n) = \Theta(f(n)g(n)) \).  

2. (20 Points) Use Stirling's formula to prove that \( \frac{(3n)!}{(n!)^3} = \Theta\left(\frac{27^n}{n}\right) \).

**Proof:**
\[
\frac{(3n)!}{(n!)^3} = \frac{\sqrt{2\pi \cdot 3n} \cdot \left(\frac{3n}{e}\right)^{3n} \cdot \left(1 + \Theta\left(\frac{1}{3n}\right)\right)^3}{\left(\sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n \cdot \left(1 + \Theta\left(\frac{1}{n}\right)\right)^3\right)^3}
\]

\[
= \frac{\sqrt{3} \cdot 1}{2\pi} \cdot \frac{3^{3n} \cdot n^{3n} \cdot e^{-3n}}{n^{3n} \cdot e^{-3n}} \cdot \frac{1 + \Theta\left(\frac{1}{3n}\right)}{\left(1 + \Theta\left(\frac{1}{n}\right)\right)^3}
\]

\[
= \frac{\sqrt{3}}{2\pi} \cdot \frac{27^n \cdot \left(1 + \Theta\left(\frac{1}{3n}\right)\right)}{n \cdot \left(1 + \Theta\left(\frac{1}{n}\right)\right)^3}
\]

Therefore
\[
\frac{(3n)!}{(n!)^3} = \frac{\sqrt{3}}{2\pi} \cdot \frac{1 + \Theta\left(\frac{1}{3n}\right)}{\left(1 + \Theta\left(\frac{1}{n}\right)\right)^3} \to \frac{\sqrt{3}}{2\pi} \text{ as } n \to \infty
\]

Since \( 0 < \sqrt{3}/2\pi < \infty \), it follows that \( \frac{(3n)!}{(n!)^3} = \Theta\left(\frac{27^n}{n}\right) \).
3. (20 Points) The $n^{th}$ harmonic number is defined to be the sum $H_n = \sum_{k=1}^{n} \left( \frac{1}{k} \right)$. Use induction to prove that for all $n \geq 1$:

$$\sum_{k=1}^{n} H_k = (n + 1)H_n - n$$

(Hint: Use the fact that $H_n$ satisfies the recurrence relation $H_n = H_{n-1} + \frac{1}{n}$.)

**Proof:**

I. If $n = 1$, then $H_1 = 1$ and $\sum_{k=1}^{1} H_k = 1 = 2 - 1 = (1 + 1) \cdot 1 - 1 = (1 + 1)H_1 - 1$, so the base case is satisfied.

II. Let $n > 1$ be chosen arbitrarily, and assume $\sum_{k=1}^{n-1} H_k = ((n - 1) + 1)H_{n-1} - (n - 1)$. We must show that

$$\sum_{k=1}^{n} H_k = (n + 1)H_n - n$$

We have

$$\sum_{k=1}^{n} H_k = \sum_{k=1}^{n-1} H_k + H_n$$

$$= ((n - 1) + 1)H_{n-1} - (n - 1) + H_n$$

by the induction hypothesis

$$= nH_{n-1} - n + 1 + H_n$$

$$= nH_n - nH_n + nH_{n-1} - n + 1 + H_n$$

$$= (n + 1)H_n - n + 1 - n(H_n - H_{n-1})$$

$$= (n + 1)H_n - n + 1 - n \left( \frac{1}{n} \right)$$

by the recurrence for $H_n$

$$= (n + 1)H_n - n,$$

as required. It follows that $\sum_{k=1}^{n} H_k = (n + 1)H_n - n$ for all $n \geq 1$. ■
4. (20 Points) Define $T(n)$ by the recurrence

$$T(n) = \begin{cases} 
0 & \text{if } n = 1 \\
4T([n/2]) + 2n^2 & \text{if } n \geq 2 
\end{cases}$$

a. (15 Points) Determine $c > 0$ such that $T(n) \leq cn^2 \lg (n)$ for all $n \geq 1$, hence $T(n) = O(n^2 \log(n))$.

**Solution:**
Let $c = 2$. We show by induction that $\forall n \geq 1$: $T(n) \leq 2n^2 \lg (n)$, from which $T(n) = O(n^2 \log(n))$ follows.

I. For $n = 1$ we have $T(1) = 0 \leq 0 = 2 \cdot 1^2 \lg (1)$, establishing the base case.

II. Let $n > 1$ be chosen arbitrarily, and assume $T(k) \leq 2k^2 \lg (k)$ for $k$ in the range $1 \leq k < n$. We must show that $T(n) \leq 2n^2 \lg (n)$. Then

$$T(n) = 4T([n/2]) + 2n^2 \quad \text{by the definition of } T(n)$$

$$\leq 4 \cdot 2 \left[\frac{n}{2}\right]^2 \lg \left[\frac{n}{2}\right] + 2n^2 \quad \text{by the induction hypothesis with } k = [n/2]$$

$$\leq 8 \left(\frac{n}{2}\right)^2 \lg \left(\frac{n}{2}\right) + 2n^2 \quad \text{since } [x] \leq x \text{ for any } x \in \mathbb{R}$$

$$= 2n^2 (\lg(n) - 1) + 2n^2$$

$$= 2n^2 \lg(n) - 2n^2 + 2n^2$$

$$= 2n^2 \lg(n)$$

The result follows for all $n \geq 1$ by the 2nd PMI. ■

b. (5 Points) Use the Master Theorem to find a tight asymptotic bound on $T(n)$.

**Solution:**
Simplifying as appropriate for the Master Theorem gives $T(n) = 4T(n/2) + n^2$. We compare $n^2$ to $n^\log_2(4) = n^2$. Case 2 yields $T(n) = \Theta(n^2 \log(n))$. ■
5. (20 Points) The following recursive algorithm determines whether an array is sorted. Variables $B_1, B_2$ and $B_3$ are Boolean, and $\land$ represents the Logical And operator.

```python
Sorted(A, p, r) precondition: $r \geq p$
1. if $r = p$
2. return TRUE
3. else
4. $q = \lfloor (p + r) / 2 \rfloor$
5. $B_1 = \text{Sorted}(A, p, q)$
6. $B_2 = \text{Sorted}(A, q + 1, r)$
7. $B_3 = (A[q] \leq A[q + 1])$
8. return $(B_1 \land B_2 \land B_3)$
```

a. (10 Points) Use induction on $m = \text{length}(A[p \cdots r])$ to prove the correctness of the above algorithm, i.e. prove that Sorted($A, p, r$) returns TRUE if and only if $A[p \cdots r]$ is sorted in increasing order.

**Proof:**

I. Let $m = 1$. Then length($A[p \cdots r]$) = $r - p + 1 = 1 \Rightarrow r = p$, and TRUE is returned on line 2 of the algorithm. Indeed, an array of length 1 is always sorted, so the algorithm returns a correct value. The base case is therefore established.

II. Let $m > 1$ and assume Sorted() returns a correct value on all sub-arrays of length less than $m$. We must show that Sorted() returns a correct value when run on any sub-array of length $m$. Since $m > 1$, we have $m = r - p + 1 > 1 \Rightarrow r > p$, so line 2 is skipped and lines 4-8 are executed.

Also

\[
p < r \Rightarrow p + r < 2r \Rightarrow \lfloor (p + r) / 2 \rfloor < r \Rightarrow q < r \\
\Rightarrow q - p + 1 < r - p + 1 \\
\Rightarrow \text{length}(A[p \cdots q]) < m
\]

and

\[
p < r \Rightarrow 2p < p + r \Rightarrow p < \frac{p + r}{2} \Rightarrow p \leq \lfloor (p + r) / 2 \rfloor \\
\Rightarrow p < \lfloor (p + r) / 2 \rfloor + 1 \Rightarrow p < q + 1 \\
\Rightarrow r - q < r - p + 1 \\
\Rightarrow \text{length}(A[q + 1 \cdots r]) < m
\]

The induction hypothesis guarantees that lines (5) and (6) return correct values for sub-arrays $A[p \cdots q]$ and $A[q + 1 \cdots r]$. Observe $A[p \cdots r]$ is sorted in increasing order if and only if: $A[p \cdots q]$ is sorted, $A[q + 1 \cdots r]$ is sorted and $A[q] \leq A[q + 1]$. Thus $A[p \cdots r]$ is sorted if and only if the value of the Boolean expression $B_1 \land B_2 \land B_3$ returned on line (8) is TRUE. Therefore, Sorted($A, p, r$) returns TRUE if and only if $A[p \cdots r]$ is sorted in increasing order, as required.

b. (10 Points) Let $T(n)$ denote the number of array comparisons performed by Sorted() on an array of length $n$. Write a recurrence relation for $T(n)$. Determine a tight asymptotic bound for $T(n)$.

**Solution:**

If $p = 1$, $r = n$, and $q = \lfloor (n + 1) / 2 \rfloor$ then length($A[1 \cdots q]$) = $q = \lfloor (n + 1) / 2 \rfloor = [n/2]$, and length($A[q + 1 \cdots n]$) = $n - q = n - [n/2] = [n/2]$. Therefore $T(n)$ must satisfy the recurrence
\[ T(n) = \begin{cases} 
0 & n = 1 \\
T([n/2]) + T([n/2]) + 1 & n \geq 2 
\end{cases} \]

First, simplify the recurrence to \( T(n) = 2T(n/2) + 1 \). We compare \( 1 = n^0 \) to \( n^{\log_2(2)} = n^1 \). Let \( \epsilon = 1 - 0 = 1 \). Then \( \epsilon > 0 \) and \( 1 = O(n^0) = O(n^{\log_2(2) - \epsilon}) \), and by case (1) we have \( T(n) = \Theta(n) \).

**Alternative Solution:**

One can show directly that \( T(n) = n - 1 \) is an exact solution to this recurrence. First note that when \( n = 1 \), \( T(1) = 0 \). If \( n \geq 1 \) then

\[
RHS = T([n/2]) + T([n/2]) + 1 \\
= ([n/2] - 1) + ([n/2] - 1) + 1 \\
= ([n/2] + [n/2]) - 1 \\
= n - 1 \\
= T(n) \\
= LHS
\]

so \( T(n) = n - 1 \) solves the recurrence, and \( T(n) = \Theta(n) \).