Problem 1 For every integer $n \geq 1$, prove by induction that $1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$.

Solution 1 Basis: If $n = 1$, then $1^2 = \frac{1 \times 2 \times 3}{6}$.

Induction Hypothesis: Assume the statement holds for $n-1$. That is $1^2 + 2^2 + \cdots + (n-1)^2 = \frac{(n-1)n(2n-1)}{6}$.

Inductive Step:

\[
1^2 + 2^2 + \cdots + n^2 = \frac{(1^2 + 2^2 + \cdots + (n-1)^2) + n^2}{6} \quad (\text{based on I.H.})
\]

\[
= \frac{(n-1)n(2n-1) + 6n^2}{6}
\]

\[
= \frac{n(n-1)(2n-1) + 6n}{6}
\]

\[
= \frac{n}{6}((n-1)(2n-1) + 6n)
\]

\[
= \frac{n}{6}(n^2 + 3n + 1)
\]

\[
= \frac{n}{6}((n+1)(2n+1))
\]
Problem 2 For every integer \( n \geq 1 \), prove by induction that \( 1 \times 1! + 2 \times 2! + \cdots + n \times n! = (n + 1)! - 1 \).

Recall that \( n! = 1 \times 2 \times 3 \times \ldots \times n \).

Solution 2 Basis: If \( n = 1 \), then \( 1 \times 1! = 2! - 1 \).

Induction Hypothesis: Assume the statement holds for \( n - 1 \). That is \( 1 \times 1! + 2 \times 2! + \cdots + (n - 1) \times (n - 1)! = n! - 1 \).

Inductive Step:

\[
1 \times 1! + 2 \times 2! + \cdots + n \times n! = (1 \times 1! + 2 \times 2! + \cdots + (n - 1) \times (n - 1)! + n \times n! \\
(\text{based on I.H.}) = n! - 1 + n \times n! = (n + 1) \times n! - 1
\]
**Problem 3** We have \( n \times k \) cups. Each of these cups has one of the \( k \) different colors. Assume \( k \) boxes of size \( n \) are available for packing these cups. Prove we can pack these cups in a way that each box has cups of at most two different colors. (Hint: Use induction on \( k \).)

**Solution 3** Note that there might be any number of cups of any color.

**Basis:** If \( k = 1 \), then we have \( n \) cups of 1 color and only one box of size \( n \). Clearly, we can put all the cups in the box and the box does not contain more than one color.

**Induction Hypothesis:** Assume the statement holds for \( k - 1 \). That is if we have \( n \times (k - 1) \) cups (for any positive integer \( n \), of \( k - 1 \) colors and we have \( k - 1 \) boxes of size \( n \) then we can put all the cups in the boxes so that no box has more than 2 colors.

**Inductive Step:** Now suppose we have \( n \times k \) cups of \( k \) different colors and \( k \) boxes of size \( n \). There exists one color for which we have less than or equal to \( n \) cups (Otherwise all the colors have more than \( n \) cups in which case there would be more than \( n \times k \) cups in total). Say this color is yellow.

Also there exists another color for which we have more than or equal to \( n \) cups. (Otherwise all the colors have less than \( n \) cups in which case there would be less than \( n \times k \) cups in total). Say this color is blue.

Now take one of the boxes. Put all yellow cups in that box. Fill the rest of the box with blue cups. Since the number of blue cups are more than or equal to \( n \) this is possible. Also since the number of yellow cups is less than or equal to \( n \), no yellow cups would be left. Now one of the boxes is filled. We are left with \( n \times k - n = n \times (k - 1) \) cups and \( k - 1 \) boxes. Furthermore, no yellow cup is left. So the cups are of \( k - 1 \) different colors now. By induction hypothesis, the boxes can be filled so that no box has more than two different colors, and we filled the \( k \)-th box with only yellow and blue cups. So the proof is complete.
**Problem 4** Suppose we have a $2^n$ by $2^n$ chessboard with one cell removed. Prove that the board can be covered with L-trominoes.

For example if $n = 2$ and one of the center cells is removed, the tiling is as shown in figure 1.

**Solution 4** Basis: If $n = 1$ the table is $2 \times 2$. No matter which cell is removed the rest is a L-triomino.

Induction Hypothesis: Suppose any $2^{n-1} \times 2^{n-1}$ table with a removed cell can be covered by L-triominos.

Inductive Step: Consider a $2^n \times 2^n$ table with one removed cell. Divide this table into 4 tables of size $2^{n-1} \times 2^{n-1}$ each. The removed cell is inside one of these 4 tables. That subtable can be be covered by L-triominos by induction hypothesis. Put one L-triomin in the middle of the original table such that it covers one cell from each of the remaining 3 subtables. Now these three subtables are also $2^{n-1} \times 2^{n-1}$ tables with one removed cell, and each can be covered by L-triominos by induction hypothesis. Therefore, the whole table can be covered by L-triominos. An illustration of this for $n = 3$ is shown in figure 2.
Problem 5  Consider the following code:

```
s ← 0
c ← 0
for i ← 1 to n do
    for j ← 2i to 2n do
        s ← s + j
        c ← c + 1
    end for
end for
```

(a) What is $s$ after running this code?

Solution  

\[
s = \sum_{i=1}^{n} \sum_{j=2i}^{2n} j = \sum_{i=1}^{n} \left( \sum_{j=1}^{2n} j - \sum_{j=1}^{2i-1} j \right)
\]

\[
= \sum_{i=1}^{n} \frac{2n(2n+1)}{2} - \frac{(2i-1)(2i)}{2} = \sum_{i=1}^{n} n(2n + 1) - i(2i - 1)
\]

\[
= \sum_{i=1}^{n} n(2n + 1) - 2 \sum_{i=1}^{n} i^2 + \sum_{i=1}^{n} i
\]

\[
= n^2(2n + 1) - \frac{n(n+1)(2n+1)}{3} + \frac{n(n+1)}{2} = n^2(2n + 1) - \frac{n(n+1)(4n-1)}{6}
\]

(b) What is $c$ after running this code?

Solution  

\[
c = \sum_{i=1}^{n} \sum_{j=2i}^{2n} 1 = \sum_{i=1}^{n} 2n - 2i + 1 = \sum_{i=1}^{n} 2n + 1 - 2 \sum_{i=1}^{n} i = n(2n + 1) - n(n + 1) = n^2.
\]

(c) What is the running time of the algorithm?

Solution  For every operation that is done by the algorithm, $c$ is incremented by one (note that $c$ is $n^2$ at the end). Therefore, the running time of the algorithm is $O(n^2)$. 
Problem 6 The following code gets $n$ and $c$ as input. What is the running time of the following code?

\begin{verbatim}
  i ← c
  while i < n do
    i ← i × c.
  end while
\end{verbatim}

Solution Suppose the loop runs in $k$ rounds. At the end of the algorithm we have $i = c^k$. So $k$ is the minimum integer such that $c^k \geq n$. This means $k$ should be the minimum integer such that $k \geq \log_c n$. Therefore, $k = O(\log(n))$. 
Problem 7 The following code represents a version of the well-known sorting algorithm "Bubble Sort":

```
for i ← 1 to n do
    for j ← 1 to n - i do
            Swap(A[j], A[j + 1])
        end if
    end for
end for
```

(a) Prove that at the end of the first iteration of the outer for loop, which is indexed by $i$, the largest element of the array would be at its correct position which is the end of the array.

Solution Suppose the largest element is located in $A[k]$ initially. In the first iteration ($i = 1$) $j$ moves from 1 to $n - 1$. As soon as $j = k$, $A[k]$ and $A[k+1]$ would be swapped and the largest element would be swapped until the end of this iteration which is $j = n - 1$. At this point the largest element would be in the last cell $A[n]$.

(b) Prove by induction the correctness of the algorithm.

Solution Induction Hypothesis:
Bubblesort algorithm presented above successfully sorts an array of size $n - 1$. The following code is Bubblesort for an array of size $n - 1$.

```
for i ← 1 to n - 1 do
    for j ← 1 to n - 1 - i do
            Swap(A[j], A[j + 1])
        end if
    end for
end for
```

Induction Step: After the first iteration the largest element is moved to the end of the array and is never touched again.

The remaining part of the algorithm after this iteration is:

```
for i ← 2 to n do
    for j ← 1 to n - i do
            Swap(A[j], A[j + 1])
        end if
    end for
end for
```

This routine is exactly equivalent to Bubblesort for $n - 1$ elements and by induction hypothesis can sort the first $n - 1$ elements. The last element is in the order so the whole array would be sorted.

(c) What is the running time of Bubble Sort?

Solution The running time of Bubblesort can be written in the following form: (note that $c$ is a constant)

$$
\sum_{i=1}^{n} \sum_{j=1}^{n-i} c = \sum_{i=1}^{n} c \times (n - i) = c \times \frac{n(n-1)}{2} = O(n^2)
$$
Problem 8 Order the following pairs of functions in terms of order of magnitude. In each case, briefly explain whether \( f(n) = O(g(n)), \ f(n) = \Omega(g(n)), \) and/or \( f(n) = \Theta(g(n)) \).

(a) \( f(n) = n^3 + 3n + \log(n)^5, \quad g(n) = 100n^2 + 3n^3 + 10\sqrt{n} \)

Solution \( f(n) = \Theta(n^3), \ g(n) = \Theta(n^3) \Rightarrow f(n) = \Theta(g(n)) \)

(b) \( f(n) = \sqrt{n} \log(n), \quad g(n) = 2\sqrt{n} \)

Solution For any \( n > 10, \ f(n) = \sqrt{n} \log(n) < \sqrt{n} \times \sqrt{n} = n = 2^{\log(n)} < 2\sqrt{n} = g(n) \). Therefore, \( f(n) = O(g(n)) \).

Note that \( f(n) < 2^{\log(n)} \) and \( \lim_{n \to \infty} \frac{2^{\log(n)}}{2\sqrt{n}} = 0 \). Hence, \( f(n) = o(g(n)) \).

This means \( f(n) = O(g(n)) \).

(c) \( f(n) = \log(\log(n^2)), \quad g(n) = \log(\log(3n + 10)) \)

Solution \( f(n) = \log(\log(n^2)) = \log(2\log(n)) = \log(2) + \log(\log(n)) = \Theta(\log(\log(n))) \)

\( g(n) = \log(\log(3n + 10)) = \Theta(\log(\log(3n))) = \Theta(\log(\log(n))) \).

\( \Rightarrow f(n) = \Theta(g(n)) \).

(d) \( f(n) = 3^n, \quad g(n) = n^22^n \)

Solution Note that you cannot basically do what you do with normal polynomials to find order of the polynomials, to exponents in functions. For example, \( 2^n \) and \( 2^{3n} \) are not of the same order though \( n \) and \( 3n \) are of the same order. Even simpler than that \( n^3 \) and \( n^2 \) are not of the same order though \( 1 \) and \( 2 \) are both constants.

\( \lim_{n \to \infty} \frac{g(n)}{f(n)} = \lim_{n \to \infty} \frac{n^22^n}{3^n} = \lim_{n \to \infty} \frac{n^2}{3^n} = 0. \)

\( \Rightarrow g(n) = o(f(n)) \) which means \( f(n) = \Omega(g(n)) \).

(e) \( f(n) = n^{\frac{1}{2}} \quad g(n) = n^{\frac{1}{2}} \log(n) \)

Solution \( f(n) = n^{\frac{1}{2}} n^{\frac{1}{2}} \log(n) \)

\( \lim_{n \to \infty} \frac{g(n)}{f(n)} = \lim_{n \to \infty} \frac{\log(n)}{n^{\frac{1}{2}}} = 0 \)

So \( g(n) = o(f(n)) \) which means \( f(n) = \Omega(g(n)) \).
Problem 9 Prove the following:

(a) \( n! = o(n^n) \)

Solution Let \( f(n) = n^n \) and \( g(n) = n! = 1 \times (2 \times 3 \times \cdots \times n) < 1 \times (n^{n-1}) = n^{n-1} \)

\[
\lim_{n \to \infty} \frac{g(n)}{f(n)} \leq \lim_{n \to \infty} \frac{n^{n-1}}{n^n} = \lim_{n \to \infty} \frac{1}{n} = 0
\]

\( \Rightarrow g(n) = o(f(n)) \)

(b) \( \log(n!) = \Theta(\log(n^n)) \).

Solution Obviously, \( \log(n!) \leq \log(n^n) \) for any \( n > 1 \). So \( \log(n!) = O(\log(n^n)) \).

Now, we want to show that \( \log(n!) \) is also \( \Omega(\log(n^n)) \). First of all note that \( \log(n^2) = \log(n) - 1 \geq \log(n^2) \) for all \( n \geq 4 \). Now we prove \( \log(n!) \geq \frac{n \log(n)}{4} \) for all \( n \geq 4 \).

\[
\log(n!) = \log(1) + \log(2) + \cdots + \log(n) = \sum_{i=1}^{n} \log(i)
\]

\[
\geq \sum_{i=\frac{n}{2}+1}^{n} \log(i) \geq \sum_{i=\frac{n}{2}+1}^{n} \log\left(\frac{n}{2}\right)
\]

\[
= \frac{n}{2} \times \log\left(\frac{n}{2}\right) = \frac{n}{2} \times (\log(n) - \log(2))
\]

\[
= \frac{n}{2} \times (\log(n) - 1) \geq \frac{n}{2} \times \frac{\log(n)}{2} \) (for all \( n \geq 4 \))
\]

Hence, by setting \( n_0 = 4 \) and \( c = \frac{1}{4} \), we have proved for all \( n \geq n_0 \) we have \( \log(n!) \geq c \times n \log(n) \).