1. Let \( f(n) = 2n^2 + 15n + 20 \).
   (a) Find a constant \( c \) and a natural number \( n_0 \) such that
   \[
   f(n) \leq c \cdot n^2 \text{ for all } n \geq n_0.
   \]
   Prove that your choice of \( c \) and \( n_0 \) work. From this you can conclude that \( f = O(n^2) \).
   (b) Find a constant \( c \) and a natural number \( n_0 \) such that
   \[
   f(n) \geq c \cdot n \text{ for all } n \geq n_0.
   \]
   Prove that your choice of \( c \) and \( n_0 \) work. From this you can conclude that \( f = \Omega(n) \).

2. Let \( f(n) = 7n \log_2 n \). Prove that \( f = O(n^2) \) and \( f = \Omega(n) \).

3. Take the following list of functions (from natural numbers to natural numbers) and arrange them in ascending order of growth. Thus, if a function \( g \) immediately follows \( f \) in the list, then \( f = O(g) \).
   (a) \( 2^n \)
   (b) \( n^2 \log_2 n \)
   (c) \( n^{4/3} \)
   (d) \( 2^{\sqrt{\log n}} \)
   (e) \( n \cdot (\log_2 n)^3 \)
   (f) \( 100n^2 \)
   (g) \( n^3 / \log_3 n \)

4. Analyze the running times of the following code fragments. For each code fragment, express your answer as a function of \( n \), using the \( \Theta \) notation. Please show your work in order to receive partial credit.

(a) \[
j \leftarrow n
\]
   \[
\text{for } i \leftarrow 1 \text{ to } n \text{ do}
\]
   \[
j \leftarrow 1
\]
   \[
\text{while } j \leq i \text{ do}
\]
   \[
\text{print("hello")}
j \leftarrow 2 \times j
\]

(b) \[
\text{for } i \leftarrow 1 \text{ to } n \text{ do}
\]
   \[
\text{for } j \leftarrow i \text{ to } n \text{ do}
\]
   \[
\text{for } k \leftarrow 1 \text{ to } n/3 \text{ do}
\]
   \[
\text{print("hello")}
\]

Here are some mathematical identities that will help you solve this problem:
• **Arithmetic series:**

\[
a + (a + d) + (a + 2d) + \cdots + (a + (n-1)d) = \frac{n(2a + (n-1)d)}{2}.
\]

If we set \(a = 1\) and \(d = 1\), we get the particularly useful special case

\[
1 + 2 + 3 + \cdots + n = \frac{n(n + 1)}{2}.
\]

• **Stirling’s Approximation:**

\[
\ln(n!) = n \ln n - n + O(\ln n).
\]

5. You are given a list of \(n\) integers. Design an algorithm that returns the number of pairs of duplicates in this list. For example, if the list is \((13, 3, 8, 3, 13, 7, 13, 9, 13)\), then the four 13’s in list yield 6 pairs of duplicates and the two 3’s in the list yield one pair. The other elements in the list are all unique. Thus your algorithm should return 7 as the total number of pairs of duplicates. Your algorithm should run asymptotically faster than the simple \(\Theta(n^2)\) time algorithm that examines all pairs of elements in the list.

6. Problem 6 from Chapter 2 in the Kleinberg-Tardos textbook.