Problem 1-1 (Asymptotic Comparisons)  10 Points

For each of the following pairs of functions \( f(n) \) and \( g(n) \), state whether \( f \) is \( O(g) \); whether \( f \) is \( o(g) \); whether \( f \) is \( \Theta(g) \); whether \( f \) is \( \Omega(g) \); and whether \( f \) is \( \omega(g) \). (More than one of these can be true for a single pair!)

(a) \( f(n) = 3n^9 + \log(n) + 38; \quad g(n) = \frac{4n^{30} + 5n^2 + 4}{111} - 52n. \)
(b) \( f(n) = \log(n^2 + 3n); \quad g(n) = \log(n^4 - 1). \)
(c) \( f(n) = \log(2^{n^2} + n^2); \quad g(n) = \log(n^{372}). \)
(d) \( f(n) = n^{37} \cdot 2^n; \quad g(n) = n^2 \cdot 5^n. \)
(e) \( f(n) = (n^n)^3; \quad g(n) = n(n^3). \)

Problem 1-2 (Counting Inversions)  13 points

Let \( A[1, \ldots, n] \) be an array of \( n \) distinct numbers. If \( i < j \) and \( A[i] > A[j] \), then the pair \((i, j)\) is called an inversion of \( A \).

(a) (2 points) List all inversions of the array \( \langle 8, 5, 2, 7, 9 \rangle \).
(b) (3 points) Which arrays with distinct elements from the set \( \{1, 2, \ldots, n\} \) have the smallest and the largest number of inversions and why? State the expressions exactly in terms of \( n \).
(c) (5 points) What is the relationship between the running time of INSERTION-SORT and the number of inversions \( I \) in the input array? Justify your answer.
(d) (3 points) [Extra credit] Let \( A[1, \ldots, n] \) be a random permutation of \( \{1, 2, \ldots, n\} \). What is the expected number of inversions of \( A \). What can you conclude about the average case running time of INSERTION-SORT (where the average is over all arrays \( A \) of size \( n \))?  

**Hint:** Recall the linearity of expectation, i.e., for any real \( a, b, c \) and any random variables \( X, Y, \)

\[
E(aX + bY + c) = aE(X) + bE(Y) + c.
\]
Problem 1-3 (The Same or Not the Same?) 10 points

The following two functions both take as arguments two \( n \)-element arrays \( A \) and \( B \):

**Magic-1**\((A, B, n)\)

\[
\text{for } i = 1 \text{ to } n \\
\quad \text{for } j = 1 \text{ to } n \\
\quad \quad \text{if } A[i] \geq B[j] \text{ return FALSE} \\
\text{return TRUE}
\]

**Magic-2**\((A, B, n)\)

\[
\text{temp} := A[1] \\
\text{for } i = 2 \text{ to } n \\
\quad \quad \text{if } A[i] > \text{temp then } \text{temp} := A[i] \\
\quad \text{for } j = 1 \text{ to } n \\
\quad \quad \text{if } \text{temp} \geq B[j] \text{ return FALSE} \\
\text{return TRUE}
\]

(a) (2 points) Both of these procedures return TRUE if and only if the same condition holds on the arrays \( A \) and \( B \) holds. Describe this condition (in words).

(b) (5 points) Analyze the worst-case running time for both algorithms using the \( \Theta \)-notation.

(c) (3 points) Does the situation change if we consider the best-case running time for both algorithms?

Problem 1-4 (Selection Sort) 12 points

Consider sorting \( n \) numbers stored in array \( A \) by first finding the largest element of \( A \) and exchanging it with the element in \( A[n] \). Then find the second largest element of \( A \) and exchange it with \( A[n-1] \). Continue in this manner for the first \( n-1 \) elements of \( A \).

(a) (5 points) Write (non-recursive) pseudocode for this algorithm, which is known as selection sort. What loop invariant does this algorithm maintain? Why does it need to run for only the first \( n-1 \) elements, rather than for all \( n \) elements? Give the best-case and worst-case running times of selection sort in \( \Theta \)-notation.

(b) (2 points) Compare the running time of selection sort to the one of insertion sort.

(c) (5 points) Devise a recursive variant of your algorithm in (a) by following the divide-and-conquer paradigm. Find a recurrence relation describing the running time of your algorithm and solve it.