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!)

\[(a) \quad f(n) = 3n^9 + \log(n) + 38; \quad g(n) = \frac{4n^{20} + 5n^3 + 4}{11} - 52n.\]

**Solution:** INSERT YOUR SOLUTION HERE

\[(b) \quad f(n) = \log(n^2 + 3n); \quad g(n) = \log(n^4 - 1).\]

**Solution:** INSERT YOUR SOLUTION HERE

\[(c) \quad f(n) = \log(2n^2 + n^2); \quad g(n) = \log(n^{372}).\]

**Solution:** INSERT YOUR SOLUTION HERE

\[(d) \quad f(n) = n^{37} \cdot 2^n; \quad g(n) = n^2 \cdot 5^n.\]

**Solution:** INSERT YOUR SOLUTION HERE

\[(e) \quad f(n) = (n^n)^3; \quad g(n) = n^{(n^3)}.\]

**Solution:** INSERT YOUR SOLUTION HERE
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 \( \langle 8, 5, 2, 7, 9 \rangle \).

**Solution:** INSERT YOUR SOLUTION HERE

(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 \( n \).

**Solution:** INSERT YOUR SOLUTION HERE

(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.

**Solution:** INSERT YOUR SOLUTION HERE

(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 \( X, Y \),

\[
E(aX + bY + c) = aE(X) + bE(Y) + c.
\]

**Solution:** INSERT YOUR SOLUTION HERE
The following two functions both take as arguments two $n$-element arrays $A$ and $B$:

**MAGIC-1**(A, B, n)

for $i = 1$ to $n$  
  for $j = 1$ to $n$  
    if $A[i] \geq B[j]$ return FALSE  
  return TRUE

**MAGIC-2**(A, B, n)

temp := A[1]  
for $i = 2$ to $n$  
  if $A[i] > temp$ then temp := A[i]  
for $j = 1$ to $n$  
  if temp $\geq B[j]$ return FALSE  
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).

**Solution:** INSERT YOUR SOLUTION HERE

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

**Solution:** INSERT YOUR SOLUTION HERE

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

**Solution:** INSERT YOUR SOLUTION HERE
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.

Solution: INSERT YOUR SOLUTION HERE

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

Solution: INSERT YOUR SOLUTION HERE

(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.

Solution: INSERT YOUR SOLUTION HERE