1  Asymptotic growth rates (3 pts)

Rank these functions in order of growth:

\[
\frac{n}{2}, 4^n, n! \cdot \log n, 2^n, (n+1)! \cdot \left(\frac{100}{99}\right)^n, n \cdot \log n, n^3, 2^n, n \cdot 2^n
\]

In other words, you should order the functions in a sequence \(g_1, g_2, \ldots, g_{11}\) such that \(g_1 = O(g_2), g_2 = O(g_3), \ldots, g_{10} = O(g_{11})\). You should also justify each relation by either referring to well known relations (where “well known” means there is a specific place in the course book that you can refer to) or give a proof. Chapter 3 in the book provides excellent reference for this exercise.

**Hint:** Make sure the function is in its simplest form. If not, use the rewriting rules in the book to make it simpler.

2  Tradeoffs (3 pts)

The `Merge-Sort` procedure described in the lectures calls `Merge` once for each recursive call. Each time memory space of the same size as input is allocated (the \(L\) and \(R\) arrays). `Insertion-Sort`, on the other hand, allocates no extra memory, but is an algorithm with quadratic running time. For each case below, justify in at most three sentences which of the two sorting algorithms (`Merge-Sort` and `Insertion-Sort`) is the asymptotically better one.

1. Memory allocation is done in constant time no matter how much memory is allocated.
2. Memory allocation is done in linear time: allocating something of size \(n\) takes \(O(n)\).
3. Memory allocation is done in quadratic time: allocating something of size \(n\) takes \(O(n^2)\).

3  Foo, Bar and Baz (4 pts)

Below the procedures `Foo`, `Bar` and `Baz` are given in pseudocode. For `Foo` and `Bar`, answer the following four questions:

1. What does the procedure do?
2. What is the best case running time?
3. What is the average case running time?
4. What is the worst case running time?

Each answer should be justified in one sentence. Answers without justifications give 0 points.
Foo\(A\)
1 \(n \leftarrow \text{size}(A)\)
2 \(\text{for } i \leftarrow 1 \text{ to } \left\lfloor \frac{n}{2} \right\rfloor \text{ do if not } A[i] = A[n - i + 1] \text{ then return False}\)
3 \text{return True}\)

Bar\(A, z\)
1 \text{return Baz}(A, 1, \text{size}(A), z)\)

Baz\(A, x, y, z\)
1 \text{if } x > y \text{ or } (x = y \text{ and } A[x] \neq z) \text{ then return } -1\)
2 \(i \leftarrow x + \left\lfloor \frac{y - x}{2} \right\rfloor\)
3 \text{if } A[i] = z \text{ then return } i\)
4 \text{if } A[i] > z \text{ then Baz}(A, x, i - 1, z)\)
5 \text{else Baz}(A, i + 1, y, z)\)

\textbf{Hint:} Use pen and paper to run the algorithm for a number of inputs, or even better implement the procedure in some simple programming language.