Fall 2025 CIS 390/550 Homework #1
(Due September 23 – before class) - 50 points

Queston 1. (*10 points*) **Euclid's Algorithm** [*hint: Lecture 1, slide 8*]

a. (*5 points*) Find gcd (67890,12345) by applying Euclid's algorithm.

b. (*5 points*) Estimate how many times faster it will be to find gcd (67890,12345) by Euclid's algorithm compared with the algorithm based on checking consecutive integers from $\min\{m, n\}$ down to $\gcd(m, n)$.

Queston 2. (*10 points*) **Consider the algorithm for the sorting problem that sorts an array by counting, for each of its elements, the number of smaller elements and the information to put the elements in its appropriate position in the sorted array.**

ALGORITHM *ComparisonCountingSort*$(A[0..n-1])$

```
//Sorts an array by comparison counting
//Input: Array A[0..n − 1] of orderable values
//Output: Array S[0..n − 1] of A's elements sorted
//    in nondecreasing order
for i ← 0 to n − 1 do
    Count[i] ← 0
for i ← 0 to n − 2 do
    for j ← i + 1 to n − 1 do
        if A[i] < A[j]
            Count[j] ← Count[j] + 1
        else Count[i] ← Count[i] + 1
for i ← 0 to n − 1 do
    S[Count[i]] ← A[i]
return S
```

a. (*6 points*) Apply this algorithm to sorting the list 48, 19, 24, 58, 21, 2.

b. (*2 points*) Is this algorithm stable? [*hint: An algorithm is stable if it doesn't change the position of equal elements in the input when producing the output.*]

c. (*2 points*) Is it in-place? [*hint: An algorithm is in-place if it doesn't require much extra memory and mainly operates on the original data, without creating a significant additional data structure.*]

Queston 3. (*10 points*) Rank the following functions by order of growth, from slowest-growing to fastest-growing. That is, find an arrangement f1, f2, ...f13 of the following functions satisfying f1 ∈ O(f2), f2 ∈ O(f3), etc. Hint: Try applying the rules of Theorem in slide. [*hint: Lecture 2, slide 19*]

$10n \log n$; 5^{100} ; $\log(\log n)$; $\log^2 n$; $2^{\log n}$; $6\sqrt{n}$; $n^{0.001}$; $n^3 \log n$
$5 \log n$; $7n$; n^3 ; n^2 ; $n!$

Queston 4. (*20 points*) For the following algorithms, derive the big O running times.

- **a.** (*5 points*) A proposed function representing the number of primitive operations for the algorithm in terms of the input size, addressing each line and/or loop of the algorithm. You do not need to be precise counting constant numbers of primitive operations (e.g., figuring out exactly how many primitive operations a line does). However, you should be precise about how many times a loop runs.
- **b.** (*3 points*) For your proposed function representing the number of primitive operations f(n), provide the upper bound for the function f(n) such that $f(n) = O(g(n))$. If $f(n) \neq O(g(n))$, explain why.
- **c.** (*2 points*) Clear communication: a mix of mathematical notation and explanations in words.

Algorithm 1

```
for i = 1 to n*n do
    if i is even then
        for j = 1 to n do
            x = x + 1
```

Algorithm 2

```
for i = 1 to n*n do
    if n|i then
        for j = 1 to n do
            x = x + 1
```