Computer Science 3600 (Winter 2027):
Problem-set #1

1. For the algorithm below, derive a **worst-case** time complexity function $T(n)$.

```
i = 1
sum = 57
finished = false
while ((i <= n) and (not finished))
    for j = 1 to i do
        if (COND(i))
            sum = sum + (i/j)
            for k = 1 to log(n) do
                sum = sum + k
        else
            sum = sum - (j/i)
    if COND(sum) then
        finished = true
        i = 57
    else
        i = i + 1
sum = sum / i + 63
```

Note that method `COND()` runs in 4 timesteps.

2. For the algorithm below, derive an **asymptotic worst-case**, *i.e.*, Big-Oh, complexity function $O(f(n))$. Briefly explain the reasoning behind your derivation.

```
sum = 42
for i = 1 to n * log(n) do
    j = 1
    finished = false
    for k = 1 to  n do
        if COND(sum)
            sum = sum / (k * i) + j
        while ((j <= n) and (not finished)) do
            finished = true
```

Note that method `COND()` runs in $(n + 13)$ timesteps.

3. For the algorithm below, derive a parameterized asymptotic worst-case time complexity function.

```
sum = 0
tsum = -15
for i = 1 to n do
    x = P1(n)
    sum = sum - x + 5
    for j = 1 to n * n do
        y = x / (P2(n) + P1(n))
    if (P3(n))
        if (P4(n))
            tsum = tsum + tsum
     else
        for j = 1 to log(n) do
    if (P4(n))
                y = y * i - j
    tsum = tsum / y
sum = sum - tsum * tsum
```

4. Prove or disprove the following:

a) $f(n) = (n-2)(n-6)$ is not $\Theta(n^2)$.

b) $f(n) = n^d + 10n^2$, where d is some integer constant greater than or equal to 2, is $O(n^d)$.

c) $f(n) = 10^{127}2^n$ is $\Omega(3^n)$.

5. Determine the longest common subsequence (LCS) of the strings GAAGCCTA and TATCGA using the algorithms given on pages 394 and 395 of the textbook. Show the filled-in dynamic programming matrix, all created matrix-cell backpointers (as arrows between matrix vells rather than in a separate matrix), the backpointer path that gives an optimal LCS, and the LCS associated with that path.