You will submit your solution to this assignment to the Curator System (as `HW02`). Your solution must be either a plain text file (e.g., NotePad) or a typed MS Word document; submissions in other formats will not be graded.

Except as noted, credit will only be given if you show relevant work.

---

1. [15 points] Using the rules given in the course notes, perform an exact count complexity analysis, for the worst case, of the body of the following function.

```
public double eval(double[] c, double x) {

   double polyx = c[0];                    // 2
   for (int k = 1; k < c.length; k++) {    // 1 before, 2 per pass, 1 exit
      double xToK = x;                     // 1
      for (int i = 1; i < k; i++) {        // 1 before, 2 per pass, 1 exit
         xToK = x * xToK;                  // 2
      }
      polyx = polyx + c[k] * xToK;         // 4
   }
   return polyx;                           // 1
}
```

State both a complexity function T(N) and the Θ-complexity of T(N).

From the line-by-line analysis above,

$$
\begin{aligned}
T(N) &= 2 + 1 + \sum_{k=1}^{N-1}\left(2 + 1 + 1 + \sum_{i=1}^{k-1}(2+2) + 1 + 4\right) + 1 + 1 \\
&= \sum_{k=1}^{N-1}\left(\sum_{i=1}^{k-1} 4 + 9\right) + 5 \\
&= \sum_{k=1}^{N-1}(4k+5) + 5 \\
&= 4\frac{(N-1)N}{2} + 5(N-1) + 5 \\
&= 2N^2 + 3N
\end{aligned}
$$

If you counted the "dot" operation to access `length`, as 1, then you would get a slightly different result:

$$
\begin{aligned}
T(N) &= 2 + 1 + \sum_{k=1}^{N-1}\left(2 + 1 + 1 + 1 + \sum_{i=1}^{k-1}(2+2) + 1 + 4\right) + 2 + 1 \\
&= 2N^2 + 4N
\end{aligned}
$$

And either way, it's clear from the theorems that T(N) is Theta(N^2).

2. [15 points] Using the rules given in the course notes, perform an exact count complexity analysis, for the worst case, of the body of the following function.

```
public double eval(double[] c, double x) {

   double polyx = c[0];                      // 2
   double xToK  = x;                         // 1
   for (int k = 1; k < c.length; k++) {      // 1 before, 2 per pass, 1 exit
      polyx = polyx + c[k] * xToK;           // 4
      xToK = x * xToK;                       // 2
   }
   return polyx;                             // 1
}
```

State both a complexity function T(N) and the Θ-complexity of T(N).

From the line-by-line analysis above,

$$T(N) = 2+1+1+\sum_{k=1}^{N-1}(2+4+2)+1+1$$

$$= \sum_{k=1}^{N-1} 8+6$$

$$= 8N-2$$

Again, if you counted the "dot" operator, you'd get a slightly different answer:

$$T(N) = 2+1+1+\sum_{k=1}^{N-1}(3+4+2)+2+1$$

$$= \sum_{k=1}^{N-1} 9+7$$

$$= 9N-2$$

And either way, it's clear from the theorems that T(N) is Theta(N).

3. [20 points] For each part, determine the simplest possible function g(n) such that the given function is $\Theta(g)$. No justification is necessary, but you might have to do some analysis using the theorems from the notes.

a) $a(n) = 14n^2 + 3n\log n$

$a(n)$ is $\theta(n^2)$ by Theorem 13

b) $b(n) = 3n^2 \log n$

$b(n)$ is $\theta(n^2 \log n)$ by Theorem 8 and

$$\lim_{n\to\infty} \frac{3n^2 \log n}{n^2 \log n} = \lim_{n\to\infty} 3 = 3$$

c) $c(n) = 3n\log^2 n + 3n^2 \log n$

$c(n)$ is $\theta(n^2 \log n)$ by Theorem 8 and

$$\lim_{n\to\infty} \frac{3n\log^2 n + 3n^2 \log n}{n^2 \log n} = \lim_{n\to\infty}\left(\frac{3\log n}{n} + 3\right)$$

$$= 3 + \lim_{n\to\infty} \frac{3/n\ln 2}{1} = 3 + 0 = 3$$

d) $d(n) = 10n^2 + 2^n$

$d(n)$ is $\theta(2^n)$ by Theorem 8 and

$$\lim_{n\to\infty} \frac{10n^2 + 2^n}{2^n} = \lim_{n\to\infty}\left(\frac{10n^2}{2^n} + 1\right)$$

$$= 1 + \lim_{n\to\infty} \frac{20n}{2^n \ln 2} = 1 + \lim_{n\to\infty} \frac{20}{2^n \ln^2 2} = 1$$

e) $e(n) = \dfrac{n^2 + 2n + 3}{n}$

$e(n)$ is $\theta(n)$ by Theorem 8 and

$$\lim_{n\to\infty} \frac{(n^2 + 2n + 3)/n}{n} = \lim_{n\to\infty} \frac{n^2 + 2n + 3}{n^2}$$

$$= \lim_{n\to\infty}\left(1 + \frac{2}{n} + \frac{3}{n^2}\right) = 1$$

4. [15 points] Suppose that executing an algorithm on input of size N requires executing T(N) = N log N + 16N instructions. How long would it take to execute this algorithm on hardware capable of carrying out 2^{22} instructions per second if N = 2^{30}? (Give your answer in hours, minutes and seconds, to the nearest second.)

The number of instructions that the algorithm would execute is given by

$$T(2^{30}) = 2^{30}\log 2^{30} + 16\cdot 2^{30} = 30\cdot 2^{30} + 16\cdot 2^{30} = 46\cdot 2^{30}$$

The number of seconds required is

$$\frac{T(2^{30})}{2^{22}} = \frac{46\cdot 2^{30}}{2^{22}} = 46\cdot 2^{8} = 11776$$

That works out to be 3 hours, 16 minutes, 16 seconds.

5. [25 points] Design an <u>efficient</u> algorithm for solving the following problem:

> Given an array `A` holding `N` elements, such that `A[0] < A[1] < A[2] < . . . < A[N-1]`, determine whether there is an index `k` such that `0 <= k <= N-1` and `A[k] = k`.

Write your algorithm as a Java function and state its Θ-complexity.

This can be solved by simply changing the binary search algorithm in the notes. The key insights are:

- **if A[k] < k then there cannot be a solution for i < k**
- **if A[k] > k then there cannot be a solution for I > k**

The changes are minimal, and left to you. The complexity is that of binary search, Theta(log N).

6. [10 points] Prove the following:

$$\text{if } x \text{ is a real number then } \lceil x \rceil + 1 = \lceil x + 1 \rceil$$

proof:

If *x* is a real number, then there is an integer k such that *k < x <= k + 1*, and by definition

$$\lceil x \rceil = k$$

But then, *k + 1 < x + 1 <= k + 2*, so

$$\lceil x + 1 \rceil = k + 2 = k + 1 + 1 = \lceil x \rceil + 1$$