You will submit your solution to this assignment to the Curator System (as `HW03`). 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
   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,

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

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

$$= 8N - 2$$

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).

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
   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,

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

$$= \sum_{k=1}^{N-1}\left(\sum_{i=1}^{k-1} 4 + 9\right) + 5$$

$$= \sum_{k=1}^{N-1}(4k+5) + 5$$

$$= 4\frac{(N-1)N}{2} + 5(N-1) + 5$$

$$= 2N^2 + 3N$$

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

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

$$= 2N^2 + 4N$$

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

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

a(n) is Theta(n^3) by Theorem 13 and Theorem 5.

b) $b(n) = 3n \log n + 5n$

b(n) is Theta(n log n) by Theorem 13 and Theorem 5.

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

This is not covered by Theorem 5, so you needed to make a guess and apply Theorem 8:

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

$$= \lim_{n\to\infty}\left(\frac{6\log(n)}{n\log n} + 3\right) = \lim_{n\to\infty}\left(\frac{6}{n} + 3\right) = 0 + 3 = 3$$

So, c(n) is Theta(n^2 log n).

d) $d(n) = n^2 + 2^n + 3^n$

d(n) is Theta(3^n) by Theorems 13 and 5 again.

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

This is also not covered by Theorem 5, but Theorem 8 settles the issue if you make the right guess:

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

So, e(n) is Theta(1).

4. [15 points] Suppose that executing an algorithm on input of size N requires executing T(N) = 8N + log N instructions. How long would it take to execute this algorithm on hardware capable of carrying out 2^{28} instructions per second if N = 2^{40}? (Give your answer in hours, minutes and seconds, to the nearest second.)

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

$$T(2^{40}) = 8 \cdot 2^{40} + \log 2^{40} = 8 \cdot 2^{40} + 40$$

The number of seconds required is

$$\frac{T(2^{40})}{2^{28}} = \frac{8 \cdot 2^{40} + 40}{2^{28}} \approx 8 \cdot 2^{12} = 32768$$

That works out to be about 9 hours, 6 minutes, 8 seconds.

---

5. [25 points] Design an efficient 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:

$$\text{if } x \text{ is a real number then } \lfloor x \rfloor + 1 = \lfloor x + 1 \rfloor$$

proof:

If x is a real number, then there is an integer $k \le x$ **and a real number** $0 \le \alpha < 1$ **such that** $x = k + \alpha$. **Therefore**

$$k \le x < k + 1 <= k + 1 + \alpha = x + 1 < k + 2$$

Now, k, k+1 and k+2 are consecutive integers, so it's clear that $k = \lfloor x \rfloor$ **and** $k + 1 = \lfloor x + 1 \rfloor$, **and therefore**

$$\lfloor x \rfloor + 1 = k + 1 = \lfloor x + 1 \rfloor$$