**Review Activity 7**

## Algorithmic Analysis

1) When a recursive function creates a very large call tree that is not collapsed over time, and that eventually crashes the program due to the lack of memory, this situation is referred to as:
   a. Super stack
   b. Stack magnification
   c. State explosion
   d. Heap exploitation
   e. None of the above

→ Stack Overflow

2) Suppose that the runtime efficiency of an algorithm is defined as the function T(n), which is given below. Determine the algorithm's order of growth in terms of the Big-O notation by simplifying the expression. You do not have to provide a formal proof using the Big-O definition. You may find the following formula useful: $\sum_{i=1}^{n-1} i = \frac{(n-1)n}{2}$

$$T(n) = \sum_{i=1}^{n-1}\sum_{j=1}^{n-1}(i+j) =$$

$$T(n) = \sum_{i=1}^{n-1}\sum_{j=1}^{n-1}(i+j)$$
$$= \sum_{i=1}^{n-1}\left[i\sum_{j=1}^{n-1} + \sum_{j=1}^{n-1} j\right]$$
$$= \sum_{i=1}^{n-1}\left[i(n-1) + \frac{(n-1)n}{2}\right]$$
$$= (n-1)\left[\sum_{i=1}^{n-1} i + \frac{n}{2}\sum_{i=1}^{n-1}\right]$$
$$= (n-1)\left[\frac{(n-1)n}{2} + \frac{n(n-1)}{2}\right]$$
$$= (n-1)^2 n$$
$$= n^3 - 2n^2 + n$$
$$= O(n^3)$$

3) If f(n) = O(g(n)) and g(n) = O(h(n)) then it must hold that f(n) = O(h(n)).
**True | False**

$f(n) = O(g(n))$ → equal or slower than $g(n)$
$g(n) = O(h(n))$ → equal or slower than $h(n)$

Complexity
$h(n)$ ↓ $g(n)$ ↓ $f(n)$

TRUE

4) Solve the given recurrence relationship for the runtime efficiency function T(n) by unrolling the recurrence (use back substitution), and give T(n)'s order of growth in terms of the Big-O notation as a function of n.

```
T(1) = 4
T(n) = 2T(n-1), for n > 1
```

$T(n) = 2T(n-1)$

$= 2(2T(n-1))$

$= 2(2(2T(n-1)))$

$= 2^i T(n-i)$

let $i = c$, when $n - i = 1$, $\therefore c = n-1$

$= 2^{n-1} T(1)$

$= 2^{n-1} \cdot 4$

$= O(2^n)$

5) Suppose that the runtime efficiency of an algorithm is defined as T(n), which is given below. Determine the algorithm's order of growth in terms of Big-O by simplifying the expression. You do not have to provide a formal proof using the Big-O definition, but you need to simplify the given expression and show simplification steps. You may find the following formula useful: $\sum_{i=0}^{n-1} i = \frac{(n-1)n}{2}$

$$T(n) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} ij + \sum_{j=0}^{n-1} 4 =$$

$= i \sum_{i=0}^{n-1} \frac{(n-1)n}{2} + 4n$

$= \frac{(n-1)^2 n^2}{4} + 4n$

$= (n^2 - 2n + 1)n^2 \cdot \frac{1}{4} + 4n$

$= \frac{1}{4}n^4 - \frac{1}{2}n^3 + \frac{1}{4}n^2 + 4n$

$= O(n^4)$