COMP 152 **Object-Oriented Data Structures and Algorithms**

Lab 4

Assigned: September 09 *Due: September 11*

1. Analyze each of the following three Python functions. In each case you may assume that `values` is a list of integers. The length of values should be used as the input size (n). For each function provide both the exact growth function and the appropriate big-O class.

   NOTE: Integer addition is the *only* primitive operation that you should count when determining the exact operation counts and growth rates for this question.

```
def someFunc1(values):
    sum = 0
    for i in range(len(values)):
        sum += i
    for val in values:
        sum += 4 * val
        sum += 8 * val
    return sum

def someFunc2(values):
    sum = 0
    for i in values:
        for j in values:
            sum += 2 * j
            for k in range(5):
                sum += k * k
    return sum

def someFunc3(values):
    sum = 0
    for i in values:
        j = len(values)
        while j > 1:
            sum += 1
            sum += i
            j = j/2
    return sum
```

2. For the following pairs of functions, indicate whether the ?? could be replaced with O, Ω or Θ. More than one may be correct: indicate all that apply.

	$f(n)$	$g(n)$	$f(n) \in$??$(g(n))$
a)	$2n$	n^2	
b)	n	$\log_2 n$	
c)	$.001n^3$	$100n^2 + 1000n$	
d)	n^2	2^n	
e)	$5n$	$8n + 1$	
f)	$n!$	$n^4 + 10n^2 \log_2 n$	
g)	n	$\log_2(2^n)$	

3. List each of the functions from the table above from slowest to fastest growing. Indicate which functions are in the same complexity class (same big-Θ).

Complexity Class	Functions from slowest to fastest growing

NOTE: You may need fewer rows than provided.

4. Consider the following two algorithms: Algorithm A requires $3n + 4$ steps to complete on an input of size n. Algorithm B requires n^2 steps. For what values of n should we prefer algorithm A? For what values of n should we prefer algorithm B? Justify your answer.

5. Using the definition of big-Θ, demonstrate that $n^3 + 2n^2 + 3n \in \Theta(n^3)$.

6. Consider the following loop template containing integer constants a and s:

```
i = a
while i < n:
    # do some work, O(1)
    i += s
```

(a) What constraints must be placed on the values of a, s, and n in order for this loop to execute at least one iteration and terminate properly?

(b) Assuming the above constraints are met, exactly how many iterations will it perform before it terminates (in terms of a, s, and n)?

(c) What is the Big O complexity of this loop and what effect, if any, do the values of a and s have on its Big O complexity?

7. Consider the following loop template containing integer constants a and s:

```
i = n
while i > a:
    # do some work, O(1)
    i /= s
```

(a) What constraints must be placed on the values of a, s, and n in order for this loop to execute at least one iteration and terminate properly?

(b) Assuming the above constraints are met, exactly how many iterations will it perform before it terminates?

(c) What is the Big O complexity of this loop and what effect, if any, do the values of a and s have on its Big O complexity?