COMP 4030/6030 - Spring 2017
Exam 1 Name: _____

Note: Do not write on the backs; only front pages are digitized and graded.

1. (*20 points*) Answer with True or False.

- $3n^2 + n^3 \in \Omega(n^3)$ T
- $3n^2 + n^3 \in O(n^4)$ T

2. (*10 points*) Use the definition of O to show that $5n^2 + 10n \in O(n^2)$.

$5n^2 + 10n \le 5n^2 + 10n^2 = 15n^2$ for all $n > 1$

$(c = 15)$

3. (*20 points*) Use the definition of Θ to show that $3n^3 + 5n - 5 \in \Theta(n^3)$.

$3n^3 + 5n - 5 \le 3n^3 + 5n^3 + 0 = 8n^3$ for $n > 1$

$3n^3 + 5n - 5 \in O(n^3)$

$3n^3 + 5n - 5 > n^3$, $n > 1$. So, $3n^3 + 5n - 5 \in \Omega(n^3)$

Therefore, $3n^3 + 5n - 5 \in \Theta(n^3)$

4. (*10 points*) Use Θ to specify the running time of the following procedure in terms of n.

```
sum = 0
for i in range(n):
    for j in range(5):
        sum = sum + i + j
```

5 steps

running time $= c \cdot 5 \cdot n \in \Theta(n)$

5. (*10 points*) Use Θ to specify the running time of the following procedure in terms of n.

```
sum = 0
for i in range(n):
    j = n
    while j > 1:
        sum = sum + i*j
        j = j / 2
```

step	j
0	n
1	$n/2$
2	$n/2^2$
k	$n/2^k$

$j = \frac{n}{2^k} = 1 \Rightarrow n = 2^k \Rightarrow k = \log_2 n$

r.t. is $c \cdot n \cdot \log_2 n \in \Theta(n \log n)$

6. (*10 points*) Is the running time of the procedure in Question 4 is in $\Omega(n)$? ~~Explain briefly.~~

$5 \cdot c \cdot n \in \Omega(n)$. Yes $5 \cdot c \cdot n \ge c \cdot n \Rightarrow$ r.t $\in \Omega(n)$

7. (*10 points*) What is the space complexity of the procedure in Question 4? $\Theta(1)$

8. (*10 points*) Explain what the following function does. The input is a list of numbers.

```
def foo(L):
    if len(L) == 0:
        return 1
    return L[0] * foo(L[1:])
```

the product of all numbers in L

(if L is empty, return just 1).

9. (*10 points*) Use mathematical induction to explain that the following function correctly adds up all numbers that are divisible by 5 in the input list.

```
def bar(L):
    if len(L) == 0:
        return 0
    if L[0] % 5 == 0:
        return L[0] + bar(L[1:])
    return bar(L[1:])
```

① When input size is smallest (len(L) = 0), then bar works correctly by returning 0.

② Let's say k = len(L). Then, k-1 = len(L[1:]).
If bar works correctly for input size less than k, then bar(L[1:]) correctly adds all numbers in L[1:] that are divisible by 5. This means:

A) If L[0] is divisible by 5, then bar is correct by adding L[0] to the sum of all numbers divisible by 5 starting from L[1].

B) If L[0] is not divisible by 5, then bar is also correct by simply returning the sum of #'s divisible by 5 starting from L[1]

10. (*10 points*) The input of the following function is a binary tree T. Assume that such a tree T has 3 attributes: T.left, T.right (both of which are also binary trees), and T.color (which is a string of either "red", "green", or "blue"). If T is empty, the function *is_empty*(T) returns True (if not, it returns False). Complete defining the following Python function so that it correctly counts the number of red nodes in the input binary tree T.

```
def count_red_nodes(T):
    # fill in your codes below to count the number of red nodes in the binary tree T
    if isempty(T): return 0
    count = 0
    if T.color == "red":
        count = 1

    rnLST = count_red_nodes(T.left)
    rnRST = count_red_nodes(T.right)
    return count + rnLST + rnRST
```