Name: ______________________________

Computer Science 245: Data Structres & Algorithms
Midterm 1 Problems Sheet
Solutions

1. Give the $\Theta()$ running time of the following code fragments, in terms of n. Show your work! (Be careful, some of these are tricky!)

   (a)
   ```
   for (i=0; i < n; i++)                    Executed n times
   {
      for (j = n; j > 1;  j--)                Executed n times
         sum++;                                  O(1)
      for (j = n; j > 1; j = j - 3)           Executed n/3 times
         sum++                                   O(1)
   }
   ```
   Total: $O(n^2)$

   (b)
   ```
   for (i=1; i < n; i = i + 2)                      Executed n/2 times
      for (j = n; j > n / 2; j = j - 2)             Executed n/4 times
         for (k = 1; k < n / 2; k = k * 2)          Executed lg(n/2) times
            sum++;                                  O(1)
   ```
   Total: $O(n^2 \lg n)$

   (c)
   ```
   for (i=1; i < n; i++)                        Executed n times
   {
      for (j = 1; j < i; j++)                      Executed n times
         sum++;                                       O(1)
      for (j = 1; j < n; j++)                      Executed n times
         sum++;                                       O(1)
      for (j = 1; j  < n; j = j * 2)               Executed lg n times
         sum++;                                       O(1)
      for (j = 0; j < n; j = j + 2)                Executed n/2 times
         sum++                                        O(1)
   }
   ```
   Total: $O(n^2)$

2. Consider the following function:

```
int recursive(int n)
{
  if (n <= 1)
      return 1;
  else
      return recursive(n - 1) + recursive(n - 1) + recursive(n - 1);
}
```

(a) What does this function calculate?
This function calculates $f(n) = 3^{n-1}$ for $n > 1$ and $f(n) = 1$ for $n <= 0$

(b) Give a recurrence relation ($T(n) = \ldots$) for this function (be sure to include both base and recursive cases!)

$$\begin{aligned} T(0) &= c_1 \\ T(1) &= c_1 \\ T(n) &= c_2 + 3T(n-1) \end{aligned}$$

(c) Solve the recurrence relation to get the $\Theta()$ running time of the function, in terms of n. Show your work, using either repeated substitution, the master method, or a recursion tree.

$$\begin{aligned} T(n) &= 3T(n-1) + c_2 \\ &= 3[3T(n-2) + c_2] + c_2 \\ &= 9T(n-2) + 3c_2 + c_2 \\ &= 9[3T(n-3)c_2] + 3c_2 + c_2 \\ &= 27T(n-3)9c_2 + 3c_2 + c_2 \\ &= 27[3T(n-4) + c_2] + 9c_2 + 3c_2 + c_2 \\ &= 81T(n-4) + 27c_2 + 9c_2 + 3c_2 + c_2 \\ &\ldots \\ &= \sum_{i=0}^{k} 3^i c_2 + T(n-k) \end{aligned}$$

Set k=n, giving us

$$\left(\sum_{i=0}^{n-1} 3^i c_2 \right) + c_1 \in O(3^n)$$

```
int recursive2(int n)
{
   if (n <= 1)
      return n;
   sum = 0;
   for (int i = 0; i < n; i++)
      sum++
   return recursive2(n/3) + recursive2(n/3) + recursve2(n/3)  + sum;
}
```

(a) Give a recurrence relation ($T(n) = \ldots$) for this function (be sure to include both base and recursive cases!)

$$\begin{aligned} T(0) &= c_1 \\ T(1) &= c_1 \\ T(n) &= c_2 n + 3T(n/3) \end{aligned}$$

(b) Solve the recurrence relation to get the $\Theta()$ running time of the function, in terms of n. Show your work, using either repeated substitution, the master method, or a recursion tree.

By the master method: $T(n) = aT(n/b) + f(n)$, where

$$\begin{aligned} a &= 3 \\ b &= 3 \\ f(n) &= n \end{aligned}$$

$n^{log_b a} = n^{log_3 3} = n$. $f(n) = n \in \Theta(n^{log_b a})$ so by the second case of the master method, $T(n) \in \Theta(n \lg n)$