## Tutorial 3

1. Determine the order of growth of the following sums. Use the *O(g(n))* notation with the simplest function *g(n)* possible.

   (i) $\sum_{i=o}^{n-1}(i^2+1)^2$

   (ii) $\sum_{i=2}^{n-1}\lg i^2$

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

2. For each of the following algorithm, give an asymptotic notation for the number of times which the statement $x=x+1$ is executed.

```
(i)   for $i$ = 1 to $n$
         for $j$ = 1 to $i$
            for $k$ = 1 to $j$
               $x=x+1$

(ii)  for $i$ = 1 to 2$n$
         for $j$ = 1 to $n$
            $x=x+1$

(iii)   $j = n$
       while ($j \geq 1$) {
         for $i$ = 1 to $j$
            $x=x+1$
         $j = j/3$
       }
```

2. Prove that $\lg(n^k + c) = \Theta(\lg n)$ for every fixed $k > 0$ and $c > 0$.

4. Determine the complexity of the following recursive function. (You may assume that $n=2^k$).

   $T(n) = 2\,T(n/2)+cn$ if $n > 1$
   $T(n) = 1$ if $n = 1$.

5. Consider the following recursive algorithm.

```
Algorithm $Q(n)$
Input:  positive integer $n$
   if  $n = 1$
         return 1
   else
         return $Q(n-1) + 2*n-1$
```

   Set up a recurrence relation for the number of multiplications made by the algorithm and solve it.