Tutorial 3

Question 1:

Find the computational complexity of the following piece of code using Big-oh notation:

```
for (int i = 1; i < n; i *= 2) {
 for (int j = n; j > 0; j /=2) {
  for (int k = j; k < n; k += 2) {
   sum += (i + j * k);
  } } }
```

Question 2:

Write a recursive function GCD(n,m) that returns the greatest common divisor of two integer n and m according to the following definition (recurrence relation):

```
GCD(n,m) = {
 m, if m <= n and n mod m = 0 {
  GCD(m,n), if n < m {
   GCD(m, n mod m), otherwise
```

Example:
Enter the first number: 54
Enter the second number: 24
The GCD of 54 and 24 is 6

Question 3:

Use the master method to give tight asymptotic bounds for the following recurrences (if the master method cannot be applied give your argument):

(a) $T(n) = 4T(n/2) + n$.

(b) $T(n) = 4T(n/2) + n^3$.

Question 4:

The following is the running time of a recursion merge sort algorithm:

$$T(n) = 2T(n/2) + O(n)$$

Using the substitution method, proof that the time complexity of this algorithm is O(n lg n). Verify your answer with the tree method and the master method.