1. Using the Master Theorem:

Let $f$ be an increasing function that satisfies

$$f(n) = a \cdot f(n/b) + c \cdot n^d$$

whenever $n = b^k$, where $k$ is a positive integer, $a \geq 1$, $b$ is
an integer $> 1$, and $c$ and $d$ are real numbers with $c$ positive
and $d$ nonnegative. Then

$$f(n) = \begin{cases}
O(n^d) & \text{if } a < b^d \\
O(n^d \log n) & \text{if } a = b^d \\
O(n^{\log_b a}) & \text{if } a > b^d
\end{cases}$$

What are the big-O bounds recurrence relations? (Simplify logs and exponents.)

a) $f(n) = 4 \cdot f(n/2) + n$

b) $f(n) = 2 \cdot f(n/4) + n$

c) $f(n) = 4 \cdot f(n/4) + n^2$

d) $f(n) = 2 \cdot f(n/2) + n$

e) $f(n) = 2 \cdot f(n/2) + 1$

2. Which of the above describes the complexity of

a) Binary Search

b) Merge Sort
3. Given the following method:

```java
public int recMax (int[] A){
    return recMax(A,0,A.length-1);
}

private int recMax(int[] A, int lo, int hi){
    if(lo==hi) return A[lo];
    else{
        int mid = (lo+hi)/2;
        int m1 = recMax(A,lo,mid);
        int m2 = recMax(A,mid+1,hi);
        return Math.max(m1, m2);
    }
}
```

a) Derive a recurrence $rM(n)$ relation for recMax(A, lo, hi), where $n = hi-lo+1$.

$rM(n) = 1$ for $n = 1$

$rM(n) = \text{ }$ for $n > 1$

b) Use the Master Theorem to solve the recurrence and obtain the big O complexity of recMax.

$rM(n) = O( )$

4. Find a solution to the following recurrence relation, using repeated substitution:

$f(1) = 2000$
$f(n) = 1.1 \cdot f(n-1)$ for $n>1$