1. (30 pts.) Indicate whether true or false.
   ANS. Solution to (a-c) uses the fact that $\ln n \in o(n^a)$ for all $a > 0$.
   (a) F $(\ln n)^2 \in \Theta(n)$. [Take $a \le 1/2$ in the fact and square.]
   (b) T $n \ln n \in o(n^{1.2})$. [Take $a \le 0.2$ in the fact and multiply by n.]
   (c) F $n^{1.2} \in o(n \ln n)$.
   (d) T If $f(n) \in \Theta(g(n))$, then $g(n) \in \Theta(f(n))$.
   (e) T If $f_1(n) \in O(g(n))$ and $f_2(n) \in O(g(n))$, then $(f_1(n) + f_2(n)) \in O(g(n))$.
   (f) T Let $W_M(n)$ and $W_Q(n)$ be the worst case times for mergesort and quicksort, respectively. True or false: $W_M(n) \in o(W_Q(n))$. [$W_M(n) \in \Theta(n \ln n)$ and $W_Q(n) \in \Theta(n^2)$]

2. (25 pts.) Consider the following eight complexity categories (remember $\lg = \log_2$):

   $\Theta(2^{\ln n})$ $\Theta(2^{\lg n})$ $\Theta(n \lg(\lg n))$ $\Theta(n \lg n)$ $\Theta(n(1+\lg n))$ $\Theta(n!)$ $\Theta(2^n)$ $\Theta(n)$

   (a) Which are equal? (There may be more than one pair.) Give a reason for any equalities.
   ANS. $\Theta(2^{\lg n}) = \Theta(n)$ since $2^{\lg n} = n$ and $\Theta(n(1 + \lg n)) = \Theta(n \lg n)$ since

$$\lim_{n\to\infty} \frac{n(1+\lg n)}{n \lg n} = 1.$$

   (b) Arrange the distinct categories in order from slowest growing to fastest growing. In other words, if $\Theta(f(n))$ is to the left of $\Theta(g(n))$, then $f(n) \in o(g(n))$.
   ANS. We use $2^{\ln n} = 2^{(\lg n)\ln 2} = n^{\ln 2}$. Also, $\ln 2 < 1$ since $e > 2$.

   $\Theta(2^{\ln n})$ $\Theta(n)$ $\Theta(n \lg(\lg n))$ $\Theta(n \lg n)$ $\Theta(2^n)$ $\Theta(n!)$

3. (20 pts.) It is known that $T(1) = 0$ and that $T(n+1) = 7T(n) + 12$ for $n > 0$. Prove that $T(n) = 2(7^{n-1} - 1)$.
   PROOF. For $n = 1$, the formula gives $T(1) = 2(7^{1-1} - 1) = 0$, proving the base case.
   Suppose the result is true for n, Then, using the recursion and then the formula for n, we have

$$\begin{aligned} T(n+1) &= 7T(n) + 12 \\ &= 7\Big(2(7^{n-1} - 1)\Big) + 12. \end{aligned}$$

   After some algebra, this becomes $T(n+1) = 2(7^n - 1) = 2(7^{(n+1)-1} - 1)$, which proves the formula for $n + 1$. This completes the induction step.

4. (25 pts.) In the following algorithm, $\cdots$ stands for some simple calculations that take constant time.

```
procedure(n)
    for k from 1 to n do
        ···  /* produces a number j */
        if k divides j, then mergesort an n-long list
        ···
    end for loop
    ···
end
```

Note: Think of j as a random integer, so the probability that "k `divides` j" is $1/k$.

(a) Suppose the sorting were free (which it is not). What is the complexity class for the average running time of this algorithm. **You MUST give a reason for your answer**. (The class should be of the form $\Theta(f(n))$ where $f(n)$ is a simple function.)

ANS. If the sorting were free, the loop is executed n times and so the running time is $\Theta(n)$.

(b) Suppose that the basic operation is a comparison in mergesort. What is the complexity class for the average running time of this algorithm. (You may give your answer in the form $\Theta(\sum f(k))$ where $f(k)$ is a simple function and the sum runs from 1 to n.) **You MUST give a reason for your answer**.

ANS. The probability that the mergesort is executed is $1/k$ for each value of k. Since the average number of comparisons for mergesort is $\Theta(n \log n)$, the average running time is

$$\sum_{k=1}^{n} (1/k)\Theta(n \log n) = \Theta\left(\sum_{k=1}^{n} (1/k) n \log n\right) = \Theta\left(n \log n \sum_{k=1}^{n} 1/k\right).$$

Any of these forms is okay. Also, if you happen to know that $\sum_{k=1}^{n} 1/k = \Theta(\log n)$, you could also have the answer $\Theta(n(\log n)^2)$. In all cases, the base of the logarithm does not change the complexity class, so it can be any base.

(c) Use (a) and (b) to find the complexity class for the average running time of this algorithm. **You MUST give a reason for your answer**.

ANS. The running time is the time to do the work without counting mergesort plus the time to do mergesort. Hence the answer is the sum of the answers to (a) and (b). You get credit for stating this, even if your answers to (a) and (b) are incorrect.

By comparing the complexity class for (a) and (b), you can see that the answer to (b) grows faster, so the complexity class for (c) is the same as that for (b).