Slightly Harder (Spring 2017, MT2)

Here is a video walkthrough of the solutions.

Give the runtime of the following functions in Θ or O notation as requested. Your answer should be as simple as possible with no unnecessary leading constants or lower order terms. For `f5`, your bound should be as tight as possible (so don't just put $O(N^{NM!})$ or similar for the second answer).

```java
public static void f4(int N) {
    if (N == 0) {return;}
    f4(N / 2);
    f4(N / 2);
    f4(N / 2);
    f4(N / 2);
    g(N); // runs in Θ(N^2) time
}
```

Runtime: $\Theta($ $)$

Solution: Runtime: $\Theta(N^2 \log N)$

Explanation: We will try a sample input, N = 4.

```
                                f4(4)

        f4(2)               f4(2)               f4(2)               f4(2)

   f4(1)    f4(1)     f4(1)    f4(1)      f4(1)    f4(1)     f4(1)    f4(1)

f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0)
```

For the first layer, the time needed is dominated by `g(N)`, which runs in $\Theta(N^2)$ time. Therefore, the time taken at this level is 4^2.

At the second level, each call is 2^2. But there are four calls to `f4(2)`, so the total time is $(4)(2)^2 = 4^2$.

In general, at the i-th level, the total time is $(4^i)(N/2^i)^2$, which is equal to exactly N^2.

Therefore:

```
                         f4(N)                                    N^2

      f4(N/2)            f4(N/2)            f4(N/2)            f4(N/2)        (4*N^2)/4

   f4(1)    f4(1)     f4(1)    f4(1)     f4(1)    f4(1)     f4(1)    f4(1)     (8*N^2)/8

f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0)  (16*N^2)/16
```

Each layer takes total time N^2, and the number of layers is $log_2 N$ (one layer when $N = 2$, three layers when $N = 8$, etc.). The total time is $\sum_{i=0}^{\log N} N^2 = \Theta(N^2 \log N)$.

```
public static void f5(int N, int M) {
    if (N < 10) {return;}
    for (int i = 0; i <= N % 10; i++) {
        f5(N / 10, M / 10);
        System.out.println(M);
    }
}
```

Runtime: $O(\quad)$

Solution:
Runtime: $O(N)$

Explanation:
Again, we can think of this as a tree. Each call to `f5` does $N\%10$ work. We can consider this to be constant, since even as N gets massive, $N\%10$ will always be between 0 and 9, inclusive. So let's assume the worst case, which means we assume $N\%10 = 9$ all the time. This means we make 10 calls to `f5` each time. Now, how many levels does our tree have before it ends? We are dividing `N` by 10 each call and we end when we hit `N < 10`. So, there are $log(N)$ levels to our tree. We can now sum up the work done at each level:

$$1 + 10 + 10^2 + ... + 10^{\log N} = \sum_{i=0}^{N} 10^i = \frac{1 - 10^{\log N}}{1 - 10}$$

Remember that asymptotics ignores constant coefficients and lower order terms! So we can get rid of all those to get:

$$10^{\log N} = O(N)$$