1. Asymptotics is fun!

(a) Using the function \( g \) defined below, what is the runtime of the following function calls? Write each answer in terms of \( N \).

```c
void g(int N, int x) {
    if (N == 0) {
        return;
    }
    for (int i = 1; i <= x; i++) {
        g(N - 1, i);
    }
}
```

\( g(N, 1) : \Theta( ) \)
\( g(N, 2) : \Theta( ) \)

(b) Suppose we change line 6 to \( g(N - 1, x) \) and change the stopping condition in the for loop to \( i \leq f(x) \) where \( f(x) \) returns a random number between 1 and \( x \), inclusive. For the following function calls, find the tightest \( \Omega \) and big \( O \) bounds.

```c
void g(int N, int x) {
    if (N == 0) {
        return;
    }
    for (int i = 1; i <= f(x); i++) {
        g(N - 1, x);
    }
}
```

\( g(N, 2) : \Omega( ) , O( ) \)
\( g(N, N) : \Omega( ) , O( ) \)
2 Flip Flop

For each part, give the best and worst case runtime in $\Theta(.)$ notation as a function of $N$. Your answer should be simple with no unnecessary leading constants or summations.

```java
public static void flip(int N) {
    if (N <= 100) {
        return;
    }
    for (int i = 1; i < N; i++) {
        // Assume g(i, N) will be equal to i for at least one i
        if (g(i, N) == i) {
            flop(i, N);
            return;
        }
    }
}
```

Given the method `flip` defined above, we will determine the best and worst case runtime when `flop` is defined as:

```java
public static void flop(int a, int b) {
    flop(b - a);
}
```

Best Case: $\Theta()$, Worst Case: $\Theta()$

```java
public static void flop(int a, int b) {
    int low = Math.min(a, b - a);
    flip(low);
    flip(low);
}
```

Best Case: $\Theta()$, Worst Case: $\Theta()$

```java
public static void flop(int a, int b) {
    flip(a);
    flip(b - a);
}
```

Best Case: $\Theta()$, Worst Case: $\Theta()$
3 Prime Factors

What is the best and worst case runtime of the function below?

```java
int prime_factors(int N) {
    int factor = 2;
    int count = 0;
    while (factor * factor <= N) {
        while (N % factor == 0) {
            System.out.println(factor);
            count += 1;
            N = N / factor;
        }
        factor += 1;
    }
    return count;
}
```

Best Case: Θ( ), Worst Case: Θ( )