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 $N$.

```cpp
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 $f$ returns a random number between 1 and $x$, inclusive. For the following function calls, find the tightest $\Omega$ and big O bounds.

```cpp
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

Suppose we have the `flip` function as defined below. Assume the method `unknown` returns a random `integer` between 1 and $N$, exclusive, and runs in constant time. For each definition of the `flop` method below, give the best and worst case runtime of `flip` in $\Theta(\cdot)$ notation as a function of $N$.

```java
public static void flip(int N) {
    if (N <= 100) {
        return;
    }
    int stop = unknown(N);
    for (int i = 1; i < N; i++) {
        if (i == stop) {
            flop(i, N);
            return;
        }
    }
}
```

(a) `public static void flop(int i, int N) {
    flip(N - i);
}

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

(b) `public static void flop(int i, int N) {
    int minimum = Math.min(i, N - i);
    flip(minimum);
    flip(minimum);
}

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

(c) `public static void flop(int i, int N) {
    flip(i);
    flip(N - i);
}

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

Determine the best and worst case runtime of \texttt{prime_factors} in \( \Theta(.) \) notation as a function of \( N \).

```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: \( \Theta( \quad ) \), Worst Case: \( \Theta( \quad ) \)