1 Finish the Runtimes

Below we see the standard nested for loop, but with missing pieces!

```
for (int i = 1; i < ______; i = ______) {
    for (int j = 1; j < ______; j = ______) {
        System.out.println("Circle is the best TA");
    }
}
```

For each part below, some of the blanks will be filled in, and a desired runtime will be given. Fill in the remaining blanks to achieve the desired runtime! There may be more than one correct answer.

**Hint:** You may find `Math.pow` helpful.

(a) Desired runtime: Θ(N^2)
```
for (int i = 1; i < N; i = i + 1) {
    for (int j = 1; j < ______; j = ______) {
        System.out.println("This is one is low key hard");
    }
}
```

(b) Desired runtime: Θ(log(N))
```
for (int i = 1; i < N; i = i * 2) {
    for (int j = 1; j < ______; j = ______) {
        System.out.println("This is one is mid key hard");
    }
}
```

(c) Desired runtime: Θ(2^N)
```
for (int i = 1; i < N; i = i + 1) {
    for (int j = 1; j < ______; j = ______) {
        System.out.println("This is one is high key hard");
    }
}
```

(d) Desired runtime: Θ(N^3)
```
for (int i = 1; i < ______; i = i * 2) {
    for (int j = 1; j < N * N; j = ______) {
        System.out.println("yikes");
    }
}
```
2 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 \( N \). Feel free to draw out the recursion tree if it helps.

```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 \( f \) returns a random number between 1 and \( x \), inclusive. For the following function calls, find the tightest \( \Omega \) and big O bounds. Feel free to draw out the recursion tree if it helps.

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

\[ g(N, 2): \Omega( ), O( ) \]

\[ g(N, N): \Omega( ), O( ) \]
3 Is This a BST?

In this setup, assume a BST (Binary Search Tree) has a key (the value of the tree root represented as an int) and pointers to two other child BSTs, left and right.

(a) The following code should check if a given binary tree is a BST. However, for some trees, it returns the wrong answer. Give an example of a binary tree for which brokenIsBST fails.

```java
public static boolean brokenIsBST(BST tree) {
    if (tree == null) {
        return true;
    } else if (tree.left != null && tree.left.key > tree.key) {
        return false;
    } else if (tree.right != null && tree.right.key < tree.key) {
        return false;
    } else {
        return brokenIsBST(tree.left) && brokenIsBST(tree.right);
    }
}
```

(b) Now, write isBST that fixes the error encountered in part (a).

Hint: You will find Integer.MIN_VALUE and Integer.MAX_VALUE helpful.

Hint 2: You want to somehow store information about the keys from previous layers, not just the direct parent and children. How do you use the parameters given to do this?

```java
public static boolean isBST(BST T) {
    return isBSTHelper(____________________________________________);
}

public static boolean isBSTHelper(BST T, int min, int max) {
    if (________________________________________) {
        ____________________________________________
    } else if (________________________________________________) {
        ____________________________________________
    } else {
        ____________________________________________
    }
}
```