1 Basic Algorithmic Analysis

For each of the following function pairs $f$ and $g$, list out the $\Theta, \Omega, O$ relationships between $f$ and $g$, if any such relationship exists. For example, $f(x) \in O(g(x))$.

1. $f(x) = x^2, g(x) = x^2 + x$
2. $f(x) = 5000000x^3, g(x) = x^5$
3. $f(x) = \log(x), g(x) = 5x$
4. $f(x) = e^x, g(x) = x^5$
5. $f(x) = \log(5^x), g(x) = x$

2 Practice with Runtime

For each of the following functions, find the Big-Theta expression for the runtime of the function in terms of the input variable $n$.

You may find the following relations helpful:

$$1 + 2 + 3 + 4 + \cdots + N = \Theta(N^2)$$
$$1 + 2 + 4 + 8 + \cdots + N = \Theta(N)$$

1. For this problem, you may assume that the static method constant runs in $\Theta(1)$ time.

```java
public static void bars(int n) {
    for (int i = 0; i < n; i += 1) {
        for (int j = 0; j < i; j += 1) {
            System.out.println(i + j);
        }
    }
    for (int k = 0; k < n; k += 1) {
        constant(k);
    }
}
```
2. Determine the runtime for \texttt{barsRearranged}.

```
public static void cowsGo(int n) {
    for (int i = 0; i < 100; i += 1) {
        for (int j = 0; j < i; j += 1) {
            for (int k = 0; k < j; k += 1) {
                System.out.println("moove");
            }
        }
    }
}
```

```
public static void barsRearranged(int n) {
    for (int i = 1; i <= n; i *= 2) {
        for (int j = 0; j < i; j += 1) {
            cowsGo(j);
        }
    }
}
```

3. A Bit on Bits

Recall the following bit operations and shifts:

1. Mask (x & y): yields 1 only if both bits are 1.
   
   \[
   \begin{array}{c}
   & \text{01110} \\
   & \text{10110} \\
   \hline
   & \text{00110}
   \end{array}
   \]

2. Set (x | y): yields 1 if at least one of the bits is 1.
   
   \[
   \begin{array}{c}
   & \text{01110} \\
   & \text{10110} \\
   \hline
   & \text{11110}
   \end{array}
   \]

3. Flip (x ^ y): yields 1 only if the bits are different.
   
   \[
   \begin{array}{c}
   & \text{01110} \\
   & \text{10110} \\
   \hline
   & \text{11000}
   \end{array}
   \]

4. Flip all (~ x): turns all 1’s to 0 and all 0’s to 1.
   
   \[
   \begin{array}{c}
   & \text{01110} \\
   \hline
   & \text{10001}
   \end{array}
   \]

5. Left shift (x << left_shift): shifts the bits to the left by \texttt{left_shift} places, filling in the right with zeros.
   
   \[
   \begin{array}{c}
   & \text{10110111} \\
   & \text{3} \\
   \hline
   & \text{10111000}
   \end{array}
   \]

6. Arithmetic right shift (x >> right_shift): shifts the bits to the right by \texttt{right_shift} places, filling in the left bits with the current existing leftmost bit.
   
   \[
   \begin{array}{c}
   & \text{10110111} \\
   & \text{3} \\
   \hline
   & \text{11110110}
   \end{array}
   \]
   
   \[
   \begin{array}{c}
   & \text{00110111} \\
   & \text{3} \\
   \hline
   & \text{00000110}
   \end{array}
   \]

7. Logical right shift (x >>> right_shift): shifts the bits to the right by \texttt{right_shift} places, filling in the left with zeros.
   
   \[
   \begin{array}{c}
   & \text{10110111} \\
   & \text{3} \\
   \hline
   & \text{00010110}
   \end{array}
   \]
Implement the following two methods. For both problems, \( i=0 \) represents the least significant bit, \( i=1 \) represents the bit to the left of that, and so on.

1. Implement `isBitOn` so that it returns a boolean indicating if the \( i \)th bit of \( num \) has a value of 1. For example, `isBitOn(2, 0)` should return `false` (the 0th bit is 0), but `isBitOn(2, 1)` should return `true` (the 1st bit is 1).

   ```java
   /** Returns whether the \( i \)th bit of num is a 1 or not. */
   public static boolean isBitOn(int num, int i) {
       int mask = 1 __________________________________________;
       return __________________________________________________;
   }
   ```

2. Implement `turnBitOn` so that it returns the input number but with its \( i \)th significant bit set to a value of 1. For example, if \( num \) is 1 (1 in binary is 01), then calling `turnBitOn(1, 1)` should return the binary number 11 (aka 3).

   ```java
   /** Returns the input number but with its \( i \)th bit changed to a 1. */
   public static int turnBitOn(int num, int i) {
       int mask = 1 __________________________________________;
       return __________________________________________________;
   }
   ```