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. The log function here denotes the natural logarithm.

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. $f(x) = e^x$, $g(x) = x^5$ (hint: $5 > e$)
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 $n$.

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

   ```java
   public static void thisIsANestedLoop(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. ```java
   public static void thisIsMoreConfusing(int n) {
       for (int i = 1; i <= n; i *= 2) {
           for (int j = 0; j < i; j += 1) {
               System.out.println("moo");
           }
       }
   }
   ```
3  A Bit with some Bits

Complete the following method such that it does what it is intended to do: given a list of integers, it returns an integer such that the i-th bit of the return value is 1 if and only if more than half of the integers in the list have 1 in the i-th bit. Keep in mind that Java int s are 32 bits long!

Note: the solution to this question isn’t very complicated, but it’s not short! Try breaking it down into components, and ask your neighbors for help!

```java
public static int bitVote(int[] bitList) {

    for (int i = 0; i < 32; i++) {  // For each bit index
        int count = 0;
        for (int k : bitList) {   // For each integer
            if ((k >> i) & 1) count++;
        }
        int threshold = (bitList.length / 2) + 1;
        if (count >= threshold) return i;
    }

    return -1;  // Should never reach here.
}
```