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 \( n \).

1. For this problem, assume that the static method \( \text{constant} \) 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. ```java
public static void barsRearranged(int n) {
    for (int i = 1; i <= n; i *= 2) {
        for (int j = 0; j < i; j += 1) {
            System.out.println("mooove");
        }
    }
}
```
3 A Bit with some Bits

Complete the following method. When given a list of integers, `bitVote` 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 `ints` are 32 bits long!

For example, if `bitList` was \([1,3]\), then in binary this would be \([01,11]\) (with 30 more zeros in front of each number), and the result would be \(01 \implies 1\), since the right-most digit was 1 for more than half the numbers, but the second-from-the-right digit was not 1 for more than half the numbers.

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

            for (int k : bitList) {
                // For each integer

            }

    }

}
```