Algorithm Analysis Exercises

1. Tracing and Counting

Consider the following code:

```
public static int someFunc1(int[] numbers) {
    int sum = 0;
    for (int num : numbers) {
        sum += num;

        for (int i = 0; i < 10; i++) {
            sum += i;
        }
    }
    return sum;
}
```

a. What would be an appropriate basic operation? _____

b. **Check with your instructor first**, then underline all occurrences of said operation in the code.

c. Develop a function *T(n)* that exactly maps the input size, *n*, to the number of basic operations:

d. Circle the closest time complexity: 1 $\log_2 n$ n n^2 2^n

For this function, count **all assignments to total** for your basic operation. Don't forget the initial assignment!

```
public int magicSums(int n) {
    int total = 0;

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            total += j;
        }
    }

    total += 1;
    return total;
}
```

e. Develop a function *T(n)* that exactly maps the input size, *n*, to the number of basic operations:

f. Circle the closest time complexity in terms of Big-O: 1 $\log_2 n$ n n^2 2^n

2. Big-O Intuition

Determine whether the following claims are True or False:

(True / False) $n^2 \in O(n^2)$

(True / False) $n^2 \in O(n^5)$

(True / False) $n^2 \in O(n \log n)$

(True / False) $5n + 100 \in O(n)$

(True / False) $\log n \in O(n)$

(True / False) $5 \in O(n)$

(True / False) $n \in O(n \log n)$

(True / False) $n \in O(1)$

3. Big-O Definitions

For a non-negative function $\boldsymbol{T}(n)$,

$$\mathbf{T}(n) \in O(f(n))$$

iff there exist positive constants c and n_0 such that
$\mathbf{T}(n) \leq c * f(n)$ for all $n > n_0$

a. Use the definition of Big-O to demonstrate that: $10n^2 + 1 \in O(n^2)$

(You will need to find constants c and n_0 that satisfy the definition)

b. Use the definition of Big-O to demonstrate that: $n^3 + 2n \in O(n^3)$