### Step One: Simplifying Expressions

1. **O(n + 10)** simplifies to **O(n)**.
2. **O(100 \* n)** simplifies to **O(n)**.
3. **O(25)** simplifies to **O(1)** (constant time complexity).
4. **O(n^2 + n^3)** simplifies to **O(n^3)** (as n3 is the dominant term).
5. **O(n + n + n + n)** simplifies to **O(n)**.
6. **O(1000 \* log(n) + n)** simplifies to **O(n)**.
7. **O(1000 \* n \* log(n) + n)** simplifies to **O(n \log n)**.
8. **O(2^n + n^2)** simplifies to **O(2^n)** (as exponential growth dominates polynomial growth).
9. **O(5 + 3 + 1)** simplifies to **O(1)**.
10. **O(n + n^(1/2) + n^2 + n \log(n)^10)** simplifies to **O(n^2)** (as n2 is the dominant term).

### Step Two: Calculating Time Complexity

* **function logUpTo(n)**
  + Loops through 1 to n and prints each number.
  + **Time Complexity: O(n)**
* **function logAtLeast10(n)**
  + Loops at least 10 times, up to n times if n is greater than 10.
  + **Time Complexity: O(max(n, 10))**, which simplifies to **O(n)** since it's bounded by the larger of n or 10.
* **function logAtMost10(n)**
  + Loops at most 10 times, less if n is smaller than 10.
  + **Time Complexity: O(min(n, 10))**, which simplifies to **O(1)** since it's a constant bound when n > 10.
* **function onlyElementsAtEvenIndex(array)**
  + Iterates over each element of the array but only processes even indices.
  + **Time Complexity: O(n)**, where n is the length of the array.
* **function subtotals(array)**
  + Nested loops where the inner loop's iterations grow with the outer loop's index.
  + **Time Complexity: O(n^2)**, where n is the length of the array.
* **function vowelCount(str)**
  + Loops through each character of the string and checks for vowels.
  + **Time Complexity: O(n)**, where n is the length of the string.

### Part 3 - Short Answer

1. True or false: n2+n is O(n^2).
   * **True**
2. True or false: n2×n is O(n^3).
   * **True**
3. True or false: n2+n is O(n).
   * **False**
4. What’s the time complexity of the .indexOf array method?
   * **O(n)**
5. What’s the time complexity of the .includes array method?
   * **O(n)**
6. What’s the time complexity of the .forEach array method?
   * **O(n)**
7. What’s the time complexity of the .sort array method?
   * **O(n \log n)** for most sorting algorithms.
8. What’s the time complexity of the .unshift array method?
   * **O(n)** (as elements need to be shifted).
9. What’s the time complexity of the .push array method?
   * **O(1)** on average.
10. What’s the time complexity of the .splice array method?
    * **O(n)** (as elements may need to be shifted).
11. What’s the time complexity of the .pop array method?
    * **O(1)**
12. What’s the time complexity of the Object.keys() function?
    * **O(n)**, where n is the number of keys.

**BONUS**

* What’s the space complexity of the Object.keys() function?
  + **O(n)**, where n is the number of keys (as a new array is created containing all keys).