1. (5 points): Why is inserting an element into an array an $O(n)$ operation?

If the array does not have room for an additional item, what do you do? Arrays in Java cannot change size.

So, allocate a new, bigger array. Copy the $n$ items from the old to the new, and discard the old. Copying $n$ items is $O(n)$.

Aside: since Java initializes arrays to zero/null/false, creating an array of size $n$ is itself an $O(n)$ operation.
2. (10 points): Write a method that receives an array of int and returns a count of the number of integers in the array that are divisible by three.

```java
int count(int[] a) {
    int count = 0;
    for (int value : a) {
        if (value % 3 == 0) {
            ++count;
        }
    }
    return count;
}
```

Note: overloading the name `count` here is okay. Java distinguishes between them based on whether `count` is followed by an open paren or not.
3. (25 points): For each of the following code fragments, indicate the number of times that `doSomething()` is called. Using big-Oh notation is fine.

```c
// (a)
for (int i = 0; i < n; ++i)
    doSomething(i);

// (b)
int b = 1;
while (b <= n) {
    doSomething(b);
    b = b * 2;
} // while

// (c)
int c = n;
while (c > 0) {
    doSomething(c);
    c = c / 3;
} // while

// (d)
int d = n;
while (d > 0) {
    doSomething(d);
    d = d * 0 - 1;
} // while

// (e)
for (int e = 0; e < n; ++e)
    for (int e2 = e; e2 <= n; ++e2)
        doSomething(e + e2);
```

- (a) \(O(n)\)
- (b) \(O(\log n)\)
- (c) \(O(\log n)\)
- (d) \(O(1)\)
- (e) \(O(n^2)\)
4. (10 points): Assume that $a > 0$, $b > 0$, $f(x) = ax$, and that $g(x) = bx^2$. Under what conditions will $f(x) > g(x)$ be true? Express your answer in terms of $a$, $b$, and $x$.

Suppose $f(x) > g(x)$

$$ax > bx^2$$

$$a > bx$$

(assuming $x \neq 0$)

$$\frac{a}{b} > x$$

answer