1 Read Me

Describe what each of the following methods does. You may assume that values contains at least one element.

```java
private static boolean method1 (int[] values) {
    int k = 0;
    while (k < values.length - 1) {
        if (values[k] > values[k+1]) {
            return false;
        }
        k = k + 1;
    }
    return true;
}

private static void method2 (int[] values) {
    int k = 0;
    while (k < values.length / 2) {
        int temp = values[k];
        values[k] = values[values.length - 1 - k];
        values[values.length - 1 - k] = temp;
        k = k + 1;
    }
}
```
2 CopyCat

For the following class, write a non-static method called `cloneCat` that allows the current `Cat` to clone itself. (Hint: This means incrementing the `clones` field and returning a clone of the current `Cat` object using the provided constructor.)

```java
class Cat {
    public static int clones = 5;
    String name;

    public Cat() {
        name = "Catherine";
    }

    public Cat(Cat c) {
        name = c.name;
    }

    public Cat cloneCat() {
    }
}
```

Could you call `cloneCat` from an instance object? How about from a class?

What would happen if we added the `static` keyword to `cloneCat` without modifying the body of the method? If we changed the method body as well, how could we call `cloneCat` from the class? Would we be able to call `cloneCat` from an instance object?
3 Flatten

Write a method flatten that takes in a 2-D int array `x` and returns a 1-D int array that contains all of the arrays in `x` concatenated together. For example, `flatten([[1, 3, 7], {}, {9}])` should return `{1, 3, 7, 9}.

```java
public static int[] flatten(int[][] x) {
    int newSize = ____________________________________;
    for (_____________________________________________) {
        _______ += __________________________________;
    }
    int[] toReturn = _________________________________;
    int toReturnIndex = ______________________________;
    for (_____________________________________________) {
        for (___________________________________________) {
            _______ = __________________________________;
            _______ += _________________________________;
        }
    }
    return _________________________________________
}
```