Recursion test follow-up

The following is based on the 3 most-missed questions on the test with fewer than 75% of correct responses. Please work through them an discuss your answers.

Tricky string recursion

Describe what the following method does.

```
public boolean check (String s)
{
    return s.length () >= 2 && (s.charAt(0) != s.charAt(1) || check(s.substring(1)));
}
```

To check your answer, trace each of the following

- `check("aaab")`
- `check("aaaa")`
- `check("abcd")`

Recursion depth

As described in class, how deep is the recursion tree for MergeSort on a list of size 64? Assume the base case is a list of length 1. (Another way to ask this is, if you count the original call to MergeSort as "1", how many nested recursive calls are made before hitting the base case?)

To further practice this concept, find the depth of trees for the following

```
public int f(int x) {
    if (x <= 1) return x;
    return f(x-1) + f(x-2);
}
```

Depth of tree on each of

```
* `f(3)`
* `f(10)`
* `f(n)`
```

```
public int f(int x) {
    if (x <= 1) return x;
    return 1 + f(x/2);
}
```

Depth of tree on each of

* `f(16)`
* `f(100)`
* `f(n)`

Tricky recursion tracing

Consider the following method

```
public void doSomething(int value)
{
    System.out.println(value);

    if(0 < value && value < 10)
    {
        doSomething(value - 1);
        doSomething(value + 1);
    }
}
```

What will be printed as a result of the call `doSomething(4)`?

Extension What is the shortest proof that this code loops forever?