ANSWER KEY

Java Strings Review: Multiple Choice

1. Consider the declaration...

```
String line = "Some more silly stuff on strings!";
// assume the words are separated by a single space
```

What string will `str` refer to after execution of the following code segment?

```
int x = line.indexOf("m");
String str = line.substring(10, 15) + line.substring(25, 25 + x);
```

(A) "sillyst"

B. "silly str"

C. "sillystr"

D. "sillystrin"

E. "silly st"

2. A program has a `String` variable `fullName` that stores a first name, followed by a space, followed by a last name. There are no spaces in either the first or last names. Consider the following code segment that extracts the last name from a `fullName` variable, and stores it in a variable identified as `lastName`, with no surrounding blanks:

```
int k = fullName.indexOf(" ") // find index of the blank space

String lastName = /* expression */
```

Which is a correct replacement for /* *expression* */ ?

~~I.~~ `fullName.substring(k);`
✓ II. `fullName.substring(k + 1);`
✓ III. `fullName.substring(k + 1, fullName.length());`

A. I only

B. II only

C. I and III only

D. III only

(E.) II and III only

Give yourself some examples:

"Kelly Kennedy" (k = 5)

I. substring(5) → grabs onto the blank space - No!

II. substring(6) → starts at K and gives "Kennedy" - Yes!

III. substring(6, 13) → gives "Kennedy" so Yes!

3. This question refers to the `getString` method shown below:

```
public static String getString(String s1, String s2) {
 int index = s1.indexOf(s2);
 return s1.substring(index, index + s2.length());
}
```

← this affects the data you want to test. Try situations where s2 can be found in s1

Which is true about `getString`? It may return a string that ...

✓ I. ... is equal to `s2`.

∅ II. ... has no characters in common with `s2`.

✓ III. ... is equal to `s1`.

} produce tests for each of these conditions!

A. I and III only

B. I, II, and III

C. II and III only

D. None is true

(E.) I and II only

✓ I. getString("Kelly", "Ke")
index = 0, s2.length = 2
s1.substring(0, 2) → "Ke"
equal to s2!

NO! II. getString("Kelly", "hi")
index = -1
s1.substring(-1, -1+2)
★ index always equals -1 if s2 isn't in s1, and the method won't take that so it has to return something with characters in common w/ s1, and therefore s2

✓ III. getString("mario", "mario")
index = 0, s2.length() = 5
s1.substring(0, 5) → "mario"
a copy of s1!