Quiz 19: String manipulation pt 2 SOLUTIONS

CSCI 110 Section 1

Friday, October 14, 2016

1) Write a function that takes a String and returns a String where the word "millennials" is replaced with "snake people". (follow-up: why can't we write a void function that changes a String parameter?) [warmup]

What should the return type be? String
What should the parameters be? one String

```
String replaceMillenials(String s) {
    return s.replace("millenials", "snake people");
}
```

Why can't we write a void function that modifies the String parameter? Because Strings are immutable. That's why many of the methods in the String class return new Strings instead of simply changing the object.

2) We've run your code from Assignment 3 and we're tired of looking at the words Whiz and Zap. Write a function that takes a String and replaces "Whiz" with "Fizz" and "Zap" with "Buzz". So it would turn a String like "12Whiz4Zap" into "12Fizz4Buzz". [30 points]

What should the return type be? String
What should the parameters be? one String

```
String replaceWhizzesAndZaps(String s) {
    String whizReplaced = s.replace("Whiz", "Fizz");
    String bothReplaced = whizReplaced.replace("Zap", "Buzz");
    return bothReplaced;
}
```

Calling this function results in the creation of two new Strings. First, a String is created where all the occurrences of "Whiz" are replaced with "Fizz". Then, we take that String and call .replace() on it, replacing each "Zap" with "Buzz" and creating yet another String.

One common mistake was calling s.replace("Whiz", "Fizz") and putting that in one new variable, then calling s.replace("Zap", "Buzz") and putting that in another new variable. Remember that "s" never gets changed because Strings are immutable, so if you return the result of s.replace("Zap", "Buzz"), only "Zap" has been replaced and not "Whiz".

3) Write a function named equalsIgnoreCase() that takes two Strings and returns true if they're equal, but does a case-insensitive comparison. So if your function got "hello" and "hELLo" as parameters, it should return true. Don't use the built-in equalsIgnoreCase() method. [20 points]

What should the return type be? boolean
What should the parameters be? The two Strings we're comparing

```
boolean equalsIgnoreCase(String a, String b) {
    String lowercaseA = a.toLowerCase();
    String lowercaseB = b.toLowerCase();
    return lowercaseA.equals(lowercaseB);
}
```

First, we create a lowercase version of a. Then we create a lowercase version of b. When we compare them, it's equivalent to doing a case-insensitive comparison.

4) Write a function named contains() that returns true if one string contains another. It should take two parameters: one String that it will search, and a second String that will be a one-character string it searches for. For example, if you pass in the parameters "abcde" and "z" the function should return false. With "abzcde" and "z", the function should return true. Don't use the built-in contains() or indexOf() methods. [50 points]

What should the return type be? boolean
What should the parameters be? two Strings: one that we're searching, one that's the search term

```
boolean contains(String original, String term) {
    for (int i = 0; i < original.length(); i++) {
        if (original.substring(i, i+1).equals(term)) {
            return true;
        }
    }
    return false;
}
```

This uses the substring method on "original", going letter by letter from start to finish with a for loop. Then we use the equals method to check whether each one-letter substring is equal to "term", another one-letter String.

5) Write a function indexOf() that takes two Strings and returns the index of the second String in the first. It should take two parameters: one String that it will search, and a second String that will be a one-character string it searches for. With "abzcde" and "z", the function should return 2.

   This doesn't have to be recursive. Don't use the built-in indexOf() method.
   Hint: use a loop. [extra credit, 50 points]

What should the return type be? int
What should the parameters be? two Strings: one that we're searching, one that's the search term

```
int indexOf(String original, String term) {
    for (int i = 0; i < original.length(); i++) {
        if (original.substring(i, i+1).equals(term)) {
            return i;
        }
    }
    return -1;
}
```

This is very similar to contains(), but we return the index instead of simply true or false.

6) Write a function numToString() that takes a number and returns that number described in words. For example, calling numToString(21) should result in "twenty one". Your function should work up to 100. [extra credit, 50 points]

What should the return type be? int
What should the parameters be? one int (the number we're converting to a String)

```
String digitToString(int n) {
    // don't return anything for zero since this gets called
    // to add a word to "twenty", "thirty" etc
    if (n == 0) {
        return "";
    } else if (n == 1) {
        return "one";
    } else if (n == 2) {
        return "two";
    } else if (n == 3) {
        return "three";
    } else if (n == 4) {
        return "four";
    } else if (n == 5) {
        return "five";
    } else if (n == 6) {
        return "six";
    } else if (n == 7) {
        return "seven";
    } else if (n == 8) {
        return "eight";
    } else if (n == 9) {
        return "nine";
    }
    // this isn't a great way to handle this - would be better
    // to throw an Exception
    return "ERROR";
}

String numToString(int n) {
    if (n == 0) {
        return "zero";
    } else if (n < 10) {
        return digitToString(n);
    } else if (n < 20) {
```

```
        if (n == 10) {
            return "ten";
        } else if (n == 11) {
            return "eleven";
        } else if (n == 12) {
            return "twelve";
        } else if (n == 13) {
            return "thirteen";
        } else if (n == 14) {
            return "fourteen";
        } else if (n == 15) {
            return "fifteen";
        } else if (n == 16) {
            return "sixteen";
        } else if (n == 17) {
            return "seventeen";
        } else if (n == 18) {
            return "eighteen";
        } else if (n == 19) {
            return "nineteen";
        }
    } else if (n < 30) {
        return "twenty " + digitToString(n % 10);
    } else if (n < 40) {
        return "thirty " + digitToString(n % 10);
    } else if (n < 50) {
        return "forty " + digitToString(n % 10);
    } else if (n < 60) {
        return "fifty " + digitToString(n % 10);
    } else if (n < 70) {
        return "sixty " + digitToString(n % 10);
    } else if (n < 80) {
        return "seventy " + digitToString(n % 10);
    } else if (n < 90) {
        return "eighty " + digitToString(n % 10);
    } else if (n < 100) {
        return "ninety " + digitToString(n % 10);
    }
    return "one hundred";
}
```

Not great because it prints extra spaces behind "twenty", "thirty", etc but it gets the job done. See this solution at https://repl.it/DvgA