CS-546 Lab 1

An Intro to Node

For this lab, you will be creating several functions and run them in a simple script!

You will submit a single file named `lab1.js`. In this lab, you will write the 5 functions below, and run them with test input.

sumOfSquares(num1, num2, num3)

For your first function, you will calculate the sum of the squares of 3 numbers and return that result. That means `sumOfSquares(5, 3, 10)` would return `134`.
To test this function, you will log the result of 5 calls to `sumOfSquares(x, y, z)` with different inputs, like so:

```
console.log(sumOfSquares(5, 3, 10));
```

sayHelloTo(firstName, lastName, title);

For the second function, you will make a simple function that uses `console.log` to print hello to someone!

The interesting thing about this function is that you don't have to have *all* the inputs to run.

Your function should print a string in the following format:

```
sayHelloTo(); // throws
sayHelloTo("Phil"); // logs: Hello, Phil!
sayHelloTo("Phil", "Barresi"); //logs: Hello, Phil Barresi. I hope you are having a good day!
sayHelloTo("Phil", "Barresi", "Mr."); // logs: Hello, Mr. Phil Barresi! Have a good evening!
```

This function does not return any content.

cupsOfCoffee(howManyCups)

For the third function, you will create and return a simple song called `99 Cups of Coffee on the Desk`.

This function should not log the song, but rather concatenate and return a string with the contents of the song.

The lyrics of this song grow longer depending on how many cups of coffee there are on the desk.

If you run `cupsOfCoffee(5)` it would return:

```
5 cups of coffee on the desk! 5 cups of coffee!
Pick one up, drink the cup, 4 cups of coffee on the desk!
```

```
4 cups of coffee on the desk! 4 cups of coffee!
Pick one up, drink the cup, 3 cups of coffee on the desk!

3 cups of coffee on the desk! 3 cups of coffee!
Pick one up, drink the cup, 2 cups of coffee on the desk!

2 cups of coffee on the desk! 2 cups of coffee!
Pick one up, drink the cup, 1 cup of coffee on the desk!

1 cup of coffee on the desk! 1 cup of coffee!
Pick it up, drink the cup, no more coffee left on the desk!
```

This would not print any content, unless you used `console.log` on the result of the function.

Take note for the subtle grammar changes!

occurrencesOfSubstring(fullString, substring)

For the fourth function, you will calculate how many times a substring occurs in a given string.

For example, calling `occurrencesOfSubstring("hello world", "o");` should return `2`, because the letter *o* appears two times in the string.
However, you must also factor in a case where there are overlaps! When you call `occurrencesOfSubstring("Helllllllo, class!", "ll");` should return `6`.
This would not print any content, unless you used `console.log` on the result of the function.

randomizeSentences(paragraph)

For your final function, you will take in a paragraph and randomize the sentences in it.

```
var paragraph = "Hello, world! I am a paragraph. You can tell that I am a paragraph because there are multiple sent
ences that are split up by punctuation marks. Grammar can be funny, so I will only put in paragraphs with periods,
exclamation marks, and question marks -- no quotations.";

console.log(randomizeSentences(paragraph));
```

Would print something *like*:

```
You can tell that I am a paragraph because there are multiple sentences that are split up by punctuation marks. I a
m a paragraph. Grammar can be funny, so I will only put in paragraphs with periods, exclamation marks, and questi
on marks -- no quotations.  Hello, world!
```

This one is tricky! You'll have to work with string manipulation, and probably an array or two as well.

This function should not log the new paragraph, but rather concatenate and return a string with the contents of the new paragraph.

Error Checking

1. Expect and account for bad input, and handle it accordingly! You can `throw "A string describing an error"` when given bad input. You can read about throwing on the MDN (Links to an external site.)Links to an external site.
1. You should throw if data is not of an expected type (Links to an external site.)Links to an external site.: ie, expecting a number and receiving an integer.
2. You should throw if your data is an out of bounds situation; ie: receiving a negative side length for certain values, or data that does not make sense given the requirements of the function.

Requirements

1. You will have to write each function
2. You must check that all arguments are valid and of the proper type