CS412 MEAN

Problem set 1: functions (three problems)

Part 1

For this assignment, write JavaScript code to satisfy each of the following problems. Each solution should be in a separate file which is named to reflect the problem number. For example, PS1/PS1.P1.js would be the file holding your solution to Problem Set 1, Problem 1.

For these problems, you'll need to both write the function (and any helper functions you require), and also a line or two that executes the function with sample input, printing the result on the console. You'll also need to dig around to find ways to solve them; for example, when working with strings, take a look at the docs for the String library to get ideas of what can be done. The 'official' docs are at developer.mozilla.org. Most solutions require chaining a few functions.

Note: For each problem, write your solution using 'fat arrow' (=>) notation.

Example 1
Write a function that returns the largest integer of an array of integers

```
const biggie = items => Math.max(...items);
console.log(`Biggest int is: ${biggie([4,8,1,4,3,9,2])}`)
```

Problem 1 (PS1.P1.js)
Write a function that takes a string as its input and returns a new string that contains all of the letters in the original string, but in reverse alphabetical order. Ignore punctuation and numbers. Duplicates are fine, so 'exioi' -> 'xoiie'. Test your function using the string 'supercalifragilisticexpialidocious'.

Problem 2 (PS1.P2.js)
Write a function that takes as its input the following formatted strings:

'4+2'
'5*7'
'6-1'
'9/2'
'2^8' (where ^ is exponentiation)

This function should
        Determine the operator (+, *, -, ^, or /) embedded in the string
        Return a function to implement the input operator that returns the result

For example, if the input string is '8%3', return `(left, right) => left % right`

Execute the returned function to evaluate and print each expression. For example,

```
const expression = '8%3';
```

```
let operator = evaluate(expression);
console.log(`${expression} = ${operator(expression)}`)
```

You can assume that the format of the input string is fixed (i.e. a digit, an operator, and a digit, always the same length). Write a helper function to parse the expression that uses destructuring to assign values to *left* and *right*.

Problem 3 (PS1.P3.js)
Write a function that accepts two input parameters: a string, and a decorator function. The function should execute the passed decorator function on the passed string and return the result.

Next, write two expressions that call this function. For the first, pass the string 'supercalifragilisticexpialidocious' and a lambda (unnamed) function that returns an array containing fragments of the input string broken on the character 'c'. For the input string 'supercalifragilisticexpialidocious', you should get

['super', 'califragilisti', 'cexpialido', 'cious']

This is actually a little tougher than it sounds…a hint would be to take a look at how bit/character stuffing is done in networking.

For the second expression, pass the string 'supercalifragilisticexpialidocious' and a lambda function that replaces all of the 'a' characters with 'A' characters. Return an object that contains the original string, the modified string, the total number of As in the modified string, and the overall length of the modified string:

```
{
        originalString:      xxx,
        modifiedString:      xxx,
        numberReplaced:      xxx,
        length:              xxx,
}
```

Print the data from the returned object on the console (console.table would be good for this).

Submitting your work

To submit, upload your solution files (PS1.P1.js, PS1.P2.js, PS1.P3.js) to Gradescope. Note that there are no extensions offered to the deadline.