**Functions**

**DEADLINE:** 31/12/2019

**FOLDER STRUCTURE**

|  |  |
| --- | --- |
| FL12\_HW8/\*     └─ task/      └─ FL12\_HW8.docx  └─ homework/\*     └─ js/\*  └─ isBigger.js\*  └─ isSmaller.js\*  └─ getMin.js\*  └─ pipe.js\*  └─ countNumbers.js\*  └─ makeNumber.js\*  └─ isLeapYear.js\*  └─ .eslintrc.js\* | \* ­­­- required |

**TASK**

**Task #1**

Write a function - *isBigger*

It should accept two arguments and returns **true** if first one has **greater** value than second one or false otherwise.

**Tip**: no need for if/else clause nor ternary operator  
**For example**:

isBigger(5, -1); // => true

**Task #2**

Write a function - *isSmaller*

It should accept two arguments and returns **true** if first one has **lesser** value than second one or false otherwise.

**Tip**: consider reusing *isBigger* function  
**For example**:

isSmaller(5 -1); //=> false

**Task #3**

Write a function - *getMin*

It should accept **arbitrary** number of integer arguments and returns the one with the smallest value.

**Tip**: since **arguments** is like array, you can use simple iteration over it  
and use arguments[ i ] to get the argument of a given index  
**For example**:

getMin(3, 0, -3); //=> -3

**Task #4**

Write function – *makeNumber*

It should accept a string with different symbols and return string of numbers from passed argument.

For example:

makeNumber('erer384jjjfd123'); //=>384123

makeNumber('123098h76gfdd'); //=>12309876

makeNumber('ijifjgdj'); //=> should return empty string ->''

**Task #5**

Write a function - *countNumbers*

It should accept string with different symbols and return object which contains counts of each numbers.

**Tip**: consider reusing *makeNumber* function

For example:

countNumbers('erer384jj4444666888jfd123');

// => {'1': 1, '2': 1, '3': 2, '4': 5, '6': 3, '8': 4}

countNumbers('jdjjka000466588kkkfs662555');

// => {'0': 3, '2': 1, '4': 1, '5': 4, '6': 4, '8': 2}

countNumbers(''); // => {}

**Task #6**

Write a function - *pipe*

It should accept a number as a first argument and arbitrary amount of functions after. The number should be passed to each function in sequence. The number passed to the next function is the returned result of previous function.

**Tip**: you need to use **arguments** to access all passed functions  
**For example**:

function addOne(x) {

return x + 1;

}

pipe(1, addOne); //=> 2

pipe(1, addOne, addOne); //=> 3

**Task #7**

Write a function - *isLeapYear*

It should accept a number of millisecond or string of date as an argument. This function checks if passed argument is a Leap Year. If it is a Leap Year, function should return a string – ‘ *“year”* is a leap year’ and if it isn’t, should return string - ‘ *“year”* is not a leap year’.(‘year’ means number of year passed as an argument)

Passed argument should be in an appropriate format (which will be valid for Date object).

In case when passed invalid argument, function should return string 'Invalid Date'.

**Tip**: need to use Date object here  
**For example**:

isLeapYear('2020-01-01 00:00:00'); // => ‘2020 is a leap year’

isLeapYear('2020-01-01 00:00:00777'); // => ‘Invalid Date’

isLeapYear('2021-01-15 13:00:00'); // => ‘2021 is not a leap year’

isLeapYear('2200-01-15 13:00:00'); // => ‘2200 is not a leap year’

isLeapYear(1213131313135465656654564646542132132131); // => ‘Invalid Date’

isLeapYear(1213131313); ); // => ‘1970 is not a leap year’

## RESTRICTIONS

* Usage of **Math object** is forbidden;

**BEFORE SUBMIT**

* Verify that all functionality is implemented according to requirements;
* Format your code (remove redundant spaces, lines of code etc.);
* Validate code via eslint;
* Add comments if necessary, delete non-relevant comments;
* In order to use npm you should install nodejs (https://nodejs.org/ );
* Install eslint to check your code (npm install -g eslint);
* open a terminal (or cmd);
* go to src folder;
* run eslint;
* Code should be without ‘errors’;

**SUBMIT**

* The folder should be uploaded to github repository '**FL12**' into **master** branch