Arrays

1. In the following code, what are the final `length` values for `array1`, `array2`, `array3`, `array4`, and `array5`?

```
let array1 = [1, 2, undefined, 4];

let array2 = [1];
array2.length = 5;

let array3 = [];
array3[-1] = [1];

let array4 = [1, 2, 3, 4, 5];
array4.length = 3;

let array5 = [];
array5[100] = 3;
```

My Answer:

```
let array1 = [1, 2, undefined, 4]; // output 4

let array2 = [1];
array2.length = 5; // output 5

let array3 = [];
array3[-1] = [1]; // output 1

let array4 = [1, 2, 3, 4, 5];
array4.length = 3; // output 3

let array5 = [];
array5[100] = 3; // output 3
```

2. Log all of the even values from `myArray` to the console.
3.

```
let myArray = [1, 3, 6, 11, 4, 2,
               4, 9, 17, 16, 0];
```

My Answer:

```
function evenNums(array) {
    let arr = [];
    for (let i=0; i<array.length; i++) {
    if (array[i] % 2 === 0) {
        arr.push(array[i]);
        }
    }
    return arr;
};
let myArray = [1, 3, 6, 11, 4, 2, 4, 9, 17, 16, 0];
console.log(evenNums(myArray)); // output (6) [6, 4, 2, 4, 16, 0]
```

4. Let's make the problem a little harder. In this problem, we're again interested in even numbers, but this time the numbers are in nested arrays in a single outer array.

```
let myArray = [
  [1, 3, 6, 11],
  [4, 2, 4],
  [9, 17, 16, 0],
];
```

My Answer:

```
let myArray = [
    [1, 3, 6, 11],
    [4, 2, 4],
    [9, 17, 16, 0],
  ];
let mergedArray=[].concat.apply([], myArray);
console.log(mergedArray.filter(a=>a%2==0)); // output (6) [6, 4, 2, 4, 16, 0]
```

5. Let's try another variation on the even-numbers theme.
    a. We'll return to the simpler one-dimensional array. In this problem, we want to use the `map` function to create a new array that contains one element for each element in the original array. If the element is an even value, then

the corresponding element in the new array should contain the string `'even'`; otherwise, the element in the new array should contain `'odd'`.

Example

```
let myArray = [
  1, 3, 6, 11,
  4, 2, 4, 9,
  17, 16, 0,
];
// Desired result:
//  [
//    'odd', 'odd', 'even', 'odd',
//    'even', 'even', 'even', 'odd',
//    'odd', even', 'even',
//  ]
```

If you have trouble using `map` to accomplish this, try it using a regular for loop instead.

My Answer:

```
let myArray = [
    1, 3, 6, 11,
    4, 2, 4, 9,
    17, 16, 0,
  ];

  function callback(n) {
    if (n % 2 === 0) {
        return "even"
    } else {
        return "odd"
    }
};

const map2 = myArray.map(callback);
console.log(map2);
// output: //  [ 'odd', 'odd', 'even', 'odd', 'even', 'even', 'even', 'odd',
'odd', even', 'even', ]
```

```
// FYI using: (x => x%2 === 0); as the callback returns true and false instead of
even and odd
```

6. Write a function that takes in an array and logs all of the values from beginning to end. Then, it logs each value from end to the beginning.
   E.g. arrayPrinter([“a”,”b”,”c”]) ->
   “a”
   “b”
   “c”
   “c”
   “b”
   “a”

My Answer:

```
const guinArray = ["green-eyes", "brown-hair",
"confused-by-JavaScript","adores-design"];

function endToBegin(arr) {
    for (let i = arr.length - 1; i >= 0; i--) {
        console.log(arr[i]);
    }
};
function beginToEnd(arr) {
    for (let j = 0; j < arr.length; j++) {
        console.log(arr[j]);
    }
};

beginToEnd(guinArray); // output:
endToBegin(guinArray); // output:
// green-eyes
// brown-hair
// confused-by-JavaScript
// adores-design
// adores-design
// confused-by-JavaScript
// brown-hair
// green-eyes
```

7. Use the `filter` method to implement a function that takes in an array and returns a new array with all of the integers from the input array. It should ignore all non-integer values from the input.

```
let array = [1, 'a', '1', 3, NaN, 3.1415, -4, null, false];
let newArray = removeNonInteger(array);
console.log(newArray); // => [1, 3, -4]
```

You can use `Number.isInteger(value)` to determine whether a numeric `value` is an integer. It returns `true` if the value is an integer, `false` otherwise.

```
 function removeNonInteger(array) {
     return array.filter(function(element) {
          return Number.isInteger(element);
});
}
```

My Answer:

```
const removeStuff = (arr) => {
    return arr.filter(x=(item)=>{
        return Number.isInteger(item);
    });
};
let numbersAndStuff = ["apple", 1, "orange", "watermelon", 25, Infinity,
undefined, null, 42, 50, 60, 8.09, 9.08]
console.log(removeStuff(numbersAndStuff));
// output (5) [1, 25, 42, 50, 60]
```

8. Use `map` and `filter` to first determine the lengths of all the elements in an array of string values, then discard the even values (keep the odd values).

```
let arr = ['a', 'abcd', 'abcde', 'abc', 'ab'];
console.log(oddLengths(arr)); // => [1, 5, 3]
```

My Answer:

```
let arr = ['a', 'abcd', 'abcde', 'abc', 'ab'];
let lengths = arr.map(function(item){
    return item.length
  });
console.log(lengths); // output (5) [1, 4, 5, 3, 2]
let odds = lengths.filter(x => x%2);
```

```
console.log(odds); // output (3) [1, 5, 3]
```

9. Without using a `for`, `while`, or `do/while` loop, write some code that checks whether the number `3` appears inside these arrays:

```
let numbers1 = [1, 3, 5, 7, 9, 11];
let numbers2 = [];
let numbers3 = [2, 4, 6, 8];
```

Return `true` or `false` depending on each result.

My Answer:

```
let numbers1 = [1, 3, 5, 7, 9, 11];
numbers1.some(function(num){return num === 3}); // output true
let numbers2 = [];
numbers2.some(function(num){return num === 3}); // output false
let numbers3 = [2, 4, 6, 8];
numbers3.some(function(num){return num === 3}); // output false
```

10. Write some code to extract the word `'mem'` from the following nested array:

```
let arr = [["test", "hello", "world"], ["example", 6, "mem", null],
[4, 8, 12]];
```

My Answer:

```
let arr = [["test", "hello", "world"], ["example", 6, "mem", null], [4, 8, 12]];
let newArr = arr.reduce(function(prev, curr) {
    return prev.concat(curr);
  });
console.log(newArr); // ["test", "hello", "world", "example", 6, "mem", null, 4,
8, 12]
newArr.indexOf("mem"); // output 5
console.log(newArr[5]); // output mem
```

Maps

1. Write a map object to store the following information:
   Our family needs to keep track of our pets. We have 2 dogs, 1 hamster, 3 cats, and 1 fish.

My Answer:

```
const petMap = new Map([
    [1, '1 hamster'],
    [2, '1 fish'],
    [3, '2 dogs'],
    [4, '3 cats']
]);
console.log(petMap); // output Map(4) {1 => "1 hamster", 2 => "1 fish", 3 => "2 dogs", 4 => "3 cats"}
```

2. Given the following code:

```
let myMap = new Map();
myMap.set('pizza', 'delicious');
myMap.set('broccoli', 'important');
myMap.set('pizza', 'unhealthy');
```

Write code to obtain the value stored at the key ‘pizza’. What is it?

My Answer:

```
let myMap = new Map();
myMap.set('pizza', 'delicious');
myMap.set('broccoli', 'important');
myMap.set('pizza', 'unhealthy');
console.log(myMap); // output Map(2) {"pizza" => "unhealthy", "broccoli" => "important"}
console.log(myMap.get('pizza')); // output unhealthy
```

3. Write a function that takes in a Map as input. Using a loop, it then cycles through each key in the map and prints out the values corresponding to that key.

My Answer:

```
function mapValues(map) {
    let mapIter = map.keys();
    for(i=0;i<map.size;i++) {
        let value = map.get(mapIter.next().values)
        console.log(value);
    }
};
console.log(petMap);
```

4. Write a function that takes in two arrays as input, keys and values. The input arrays are of equal length, and each entry in them corresponds to a single key and value pair. Return a map containing all of the information in the original arrays.
   E.g. arraysToMap([‘chicken’,’dog’], [3,’fish’]) ->
   Map { ‘chicken’ => 3, ‘dog’ => 'fish’ }

My Answer:

```
array1 = [101,102,103,104];
array2 = ['programming-foundations','html','css','javascript'];
function arrToMap(keysArr,valuesArr) {
    // since we're returning to a map, we need to create a map
    let result = new Map();
    for(let i=0;i<keysArr.length;i++){
        // get a key from one array & a value from the other
        let key = keysArr[i];
        let value = valuesArr[i]
        //add a key-value pair to map
        result.set(key,value)
    }
    return result;
};
arrToMap(array1,array2);
```

5. Let's make a basic password system in two parts.
    a. That is, it takes in two strings as input: one representing a username and one representing a password. The function stores the username and password combination in a global object as plain text. There is no output.
    b. Write a function to act as a password reminder. That is, it takes in a string as input representing a username. If the username already has a password stored, the function returns the password. If the username hasn't been stored, the function returns false and logs a message to the console letting them know about the issue.

My Answer:

```
function passwordStorer(username,password) {
    let thatMap = new Map(); // create the map
    thatMap.set(username,password) // "set" or push the keys and values into the
map
    return thatMap; // return the new map
}

function passwordReminder(username,map) {
    if(map.has(username)){
        let password = map.get(username); // .get() method used with a key
returns the assigned value
        return password;
    } else {
        console.log("Error: no password");
        return false;

    }
};

let database = passwordStorer('gwhite123','gooberberries');
passwordReminder('gwhite123', database);
passwordReminder('gwhite123', database); // output "gooberberries"
```

6. Write a function that takes in two maps and merges them, returning a single map representing the combination of the two. That is, it does the following:
    a. If a key in the first map does not exist in the second map (or vice versa), add the key and value to the map.

b. If a key exists in BOTH maps, the new value for the key should be an array containing BOTH values from the two maps.
E.g.

```
var first = new Map([[1, 'apple'],[2, 'banana'],[3, 'cherry']])
var second = new Map([[3, 'watermelon'],[4, 'pear'] ])
var combined = mapCombiner(first, second);
console.log(combined);
/*
Map { 1 => 'apple', 2 => 'banana', 3 => ['cherry','watermelon'], 4 =>
'pear' }
*/
```

My Answer: NOT DONE YET

7. A kidnapper wrote a ransom note, but now he is worried it will be traced back to him through his handwriting. He found a magazine and wants to know if he can cut out whole words from it and use them to create an untraceable replica of his ransom note. The words in his note are *case-sensitive* and he *must* use only whole words available in the magazine. He *cannot* use substrings or concatenation to create the words he needs.

   Given the words in the magazine and the words in the ransom note, print Yes if he can replicate his ransom note *exactly* using whole words from the magazine; otherwise, print No.
   For example, the note is "Attack at dawn". The magazine contains only "attack at dawn". The magazine has all the right words, but there's a case mismatch. The answer is “No”. The inputs are a list of strings representing the magazine and a list of strings representing the note.

My Answer: NOT DONE YET

8. Given two strings, determine if they share a common substring. A substring may be as small as one character.
   For example, the words "a", "and", "art" share the common substring “a”. The words "be" and "cat" do not share a substring.
   Hint: consider a sequence of nested maps

My Answer: NOT DONE YET

Friday 7/10 Coding Exercises
Loops

1. Write a for loop that writes the numbers 1-5 out to the document. 1a. Write a while loop that writes the numbers 1-5 out to the console.

My Answer:

```
function forLoop() {
    for(let i=0;i<6;i++){
        console.log(i);
    }
}
forLoop();
// output
// 0
// 1
// 2
// 3
// 4
// 5

function whileLoop() {
    i=0;
    while(i<6) {
        console.log(i);
        i++;
    }
}
whileLoop();
// output
// 0
// 1
// 2
// 3
// 4
// 5
```

2. Write a for loop that logs the numbers 5-1 (in descending order) out to the console.
2a. Write a while loop that writes the numbers 5-1 out to the document.

My Answer:

```
function backwardFor() {
    let i = 0;
    for (i = 5; i > 0; i--){
      console.log(i);
    }
};
backwardFor();
// output
// 5
// 4
// 3
// 2
// 1

function backwardWhile() {
    let i=5;
    while(i > 0) {
        console.log(i);
        i--;
    }
}
backwardWhile();
// output
// 5
// 4
// 3
// 2
// 1
```

3 - Andrew is worried that when he's walking on the sidewalk and steps on a crack, it'll break his mom's back. 3a. Create an array of steps and on step 3 of 5 (index 3) signify the crack that'll break his mom's back (i.e crack/broken). On the remaining steps signify

that his mom is safe (i.e no crack/safe). Write a function using a for loop that writes his steps out to the document. 3b. Using the same array, write a function using a while loop that stops right before he steps on the crack.

My Answer:

```
steps = ["step 0-safe","step 1-safe","step 2-safe",
"step 3 crack-broken","step 4-safe","step 5-safe"];

function stepCounter(array) {
    for (let i=0; i<array.length; i++) {
        console.log(array[index]);
    }
};
// output (6) ["step 0-safe", "step 1-safe", "step 2-safe", "step 3
crack-broken", "step 4-safe", "step 5-safe"]

function stopStepping(array,substr) {
    for (let i = 0; i < array.length; i++){
    let stopValue = array.findIndex(item => item.includes(substr));
    console.log(stopValue - 1) // test to see if this is working as expected
    if (stopValue - 1) {
        console.log("Stop before you break your mother's back!")
        break;
        }
    }
}
stopStepping(steps,"crack");
// output
// 2
// Stop before you break your mother's back!
```