Lab 1

New Attempt

Due Sep 16 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip
Available after Sep 9 at 12am

CS-546 Lab 1

An Intro to Node

For this lab, you will be creating and running several functions to practice JavaScript syntax.

For this lab, you will make two files: `lab1.js` and `lab1.test.js` and submit them in a zip file that's named `LastName_FirstName.zip` . For example: Hill_Patrick.zip

You **should not** have any folders inside the zip file.

You **must** submit your files with the format specified, named as specified otherwise points will be deducted.

lab1.js

In this file, you will update the content of the functions and update the `firstName`, `lastName`, and `studentId` with the appropriate information. The function specifications are listed in the section below.

```
const questionOne = function questionOne(arr) {
    // Implement question 1 here
}

const questionTwo = function questionTwo(arr) {
    // Implement question 2 here
}

const questionThree = function questionThree(arr) {
    // Implement question 3 here
}

const questionFour = function questionFour(num1, num2, num3) {
    // Implement question 4 here
}

module.exports = {
    firstName: "YOUR FIRST NAME",
    lastName: "YOUR LAST NAME",
    studentId: "YOUR STUDENT ID",
    questionOne,
    questionTwo,
    questionThree,
    questionFour
};
```

lab1.test.js

```
const lab1 = require("./lab1");

console.log(lab1.questionOne([2]));
// should return and output: {'3': true}

console.log(lab1.questionTwo([1, 2, 3, 2, 1]));
// should return and output: [1, 2, 3]

console.log(lab1.questionThree(["bar", "car", "car", "arc"]);
// should return and output: { acr: ["car", "arc"] }

console.log(lab1.questionFour(1, 3, 2));
// should return and output: 4
```

Functions to implement

questionOne(arr)

For your first function, you will calculate each element with the following equation `x^2 - 7` and determine if the `Math.abs` of the `result` is prime or not. You will return an `object` with the result as the key and true/false as the value That means that in `lab1.test.js`, running `lab1.questionOne([5, 3, 10])` would return `{18: false, 2: true, 93: false}`. (please note that when you console.log the return value it will have quotes around the keys, that is fine like so: `{'18': false, '2': true, '93': false}` is the same as: `{18: false, 2: true, 93: false}`

If an empty array is passed in or if the function is called without any input parameters, just return an empty object. You do not have to worry about dealing with different data types passed in. You can assume only arrays with numbers as elements and empty arrays will be passed in to your function (we get to type checking and error handling in lecture 2)

To test this function, you will log the result of 5 calls to `lab1.questionOne([x, y, z])` with different inputs, like so:

```
console.log(lab1.questionOne([5, 3, 10]));
//returns and outputs: {'18': false, '2': true, '93': false}

console.log(lab1.questionOne([2]));
// returns and outputs: {'3': true}

console.log(lab1.questionOne([]));
// returns and outputs: {}
```

```
console.log(lab1.questionOne());
// returns and outputs: {}
```

questionTwo(arr);

This function will return a new array that contains no duplicated values.

Note: '1' and 1 are not considered the same value.

If an empty array is passed in, just return an empty array `[]`. You do not have to worry about dealing with different data types passed in. You can assume only arrays are passed in. In the arrays, numbers and strings are the only element types that are contained (we get to type checking and error handling in lecture 2).

To test this function, you will log the result of 5 calls to `lab1.questionTwo([x, y, z])` with different inputs, like so:

```
console.log(lab1.questionTwo([1, 1, 1, 1, 1, 1]));
//returns and outputs: [1]

console.log(lab1.questionTwo([1, '1', 1, '1', 2]));
// returns and outputs: [1, '1', 2]

console.log(lab1.questionTwo([3, 'a', 'b', 3, '1']));
// returns and outputs: [3, 'a', 'b', '1']

console.log(lab1.questionTwo([]));
//returns and outputs: []
```

questionThree(arr)

This function will take in an array of strings and you will be finding the anagrams of each other. You will create an `object` that contains the sorted word as the key, and the value will be an array of the strings that match those exact letters. You will return an `object` whose values are greater than 1 (meaning, there are anagrams that are grouped together). If all values in the object only have a length 1, return an empty object.

Note: If the array contains the same words, ie. ["foo", "foo"], consider foo to one, meaning you have to check the value to see if the array already contains the string.

If an empty array is passed in, just return an empty object. You do not have to worry about dealing with different data types passed in. You can assume only an array of strings or an empty array will be passed in to your function (we get to type checking and error handling in lecture 2).

To test this function, you will log the result of 5 calls to `lab1.questionThree(str)` with different inputs, like so:

```
console.log(lab1.questionThree(["cat", "act", "foo", "bar"]));
// returns and outputs: { act: ["cat", "act"] }

console.log(lab1.questionThree(["race", "care", "foo", "foo", "foo"]));
// returns and outputs: { acer: ["race", "care"] }
```

```
console.log(lab1.questionThree(["foo", "bar", "test", "Patrick", "Hill"]));
// returns and outputs: {}

console.log(lab1.questionThree([]));
// returns and outputs: {}
```

questionFour(num1, num2, num3)

This function will take three number inputs and for each input, you will calculate the factorials for each input, add all the results up and divide it by the average of the 3 original inputs. You can assume that all input parameters will be supplied (none missing) and that they will all be numbers. You will return the `Math.floor` of the final result.

Note: We will not be testing any values that are imaginary, you can assume that the values we will be testing with will return a number.

To test this function, you will log the result of 5 calls to `lab1.questionFour(num1, num2, num3)` with different inputs, like so:

```
console.log(lab1.questionFour(1, 3, 2));
//returns and outputs: 4

console.log(lab1.questionFour(2, 5, 6));
//returns and outputs: 194
```

Requirements

1. You will have to write each function
2. You must submit all files, zipped up, not contained in any folders
3. You must not use any npm dependenices in this lab