Lab 1

Due Sep 15, 2022 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip **Available** after Sep 8, 2022 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.

You can download the lab code starting template here: **Lab1_stub.zip (https://sit.instructure.com/courses/61549/files/10115862?wrap=1)** **(https://sit.instructure.com/courses/61549/files/10115862/download?download_frd=1)**

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
   return //return result
}

const questionTwo = function questionTwo(num) {
    // Implement question 2 here
    return //return result
}

const questionThree = function questionThree(text) {
    // Implement question 3 here
    return //return result
}

const questionFour = function questionFour(num) {
    // Implement question 4 here
    return //return result
}

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([1, 2, 3])); // should return and then output [false, true, true]

console.log(lab1.questionTwo(2, 10, 4)); //Returns and then outputs 2222

console.log(lab1.questionFour("hello world", "o")); //Returns and then outputs 2
```

Functions to implement

questionOne(arr);

For your first function, you will calculate if each and every element in the array is a prime number. Your function will return an array that has boolean values corresponding to the original index of the input. That means that in `lab1.test.js`, running `lab1.questionOne([5, 3, 10])` would return `[true, true, false]`.

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 then outputs [true, true, false]
console.log(lab1.questionOne([2, 1, 2])); // Returns and then outputs[true, false, true]
console.log(lab1.questionOne([512, 1007, 17389])); //Returns and then outputs [false, false, true]
console.log(lab1.questionOne([0, 14159, 785])); //Returns and then outputs [false, true, false]
console.log(lab1.questionOne([11, 4])); //Returns and then outputs [true, false]
```

questionTwo(startingNumber, commonRatio, numberOfTerms);

This function will sum a **Geometric Series** (https://www.mathsisfun.com/algebra/sequences-sums-geometric.html)

`startingNumber` can be positive, negative, decimal but CANNOT be 0. If 0 is passed into the function as the `startingNumber` your function should return 0.

`commonRatio` can be positive, negative, decimal but CANNOT be 0. If 0 is passed into the function as the `commonRatio` your function should return 0.

`numberOfTerms` can only be a positive whole number greater than 0. If `numberOfTerms` is <=0 or is a decimal, your function should return `NaN`

That means that in `lab1.test.js`, running `lab1.questionTwo(2,10,4)` would return `2222`.

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(5, 3, 10));  // Returns and then outputs 147620
console.log(lab1.questionTwo(2, 0, 2)); // Returns and then outputs 0
console.log(lab1.questionTwo(512, 1007, -5)); //Returns and then outputs NaN
console.log(lab1.questionTwo(2, 10, 4)); //Returns and then outputs 2222
console.log(lab1.questionTwo(175, 3, 5)); //Returns and then outputs 21175
```

questionThree(str)

This function will return the number of consonants contained in the value `str`.

```
console.log(lab1.questionThree("How now brown cow"));  // Returns and then outputs 10
console.log(lab1.questionThree("Welcome to CS-546")); // Returns and then outputs 7
console.log(lab1.questionThree("JavaScript is fun!")); //Returns and then outputs 10
```

questionFour(fullString, substring)

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

For example, calling `questionFour("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 `questionFour("Hellllllo, class!", "ll")`; should return `3` since `"ll"` appears 3 times.

```
console.log(lab1.questionFour("hello world", "o"));  // Returns and then outputs 2
console.log(lab1.questionFour("Hellllllo, class!", "ll")); // Returns and then outputs 3
```

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

Lab 1 Rubric

Criteria	Ratings		Pts
questionOne(arr); Test cases used for grading will be different from assignment examples.	**25 to >0.0 pts** **5 Points Per Test Case** The function is implemented correctly, and test cases pass.	**0 pts** **All Test Cases Failed** Incorrect implementation or none of the test cases pass.	25 pts
questionTwo(startingNumber, commonRatio, numberOfTerms); Test cases used for grading will be different from assignment examples.	**25 to >0.0 pts** **5 Points Per Test Case** The function is implemented correctly, and test cases pass.	**0 pts** **All Test Cases Failed** Incorrect implementation or none of the test cases pass.	25 pts
questionThree(str); Test cases used for grading will be different from assignment examples.	**25 to >0.0 pts** **5 Points Per Test Case** The function is implemented correctly, and test cases pass.	**0 pts** **All Test Cases Failed** Incorrect implementation or none of the test cases pass.	25 pts
questionFour(fullString, substring); Test cases used for grading will be different from assignment examples.	**25 to >0.0 pts** **5 Points Per Test Case** The function is implemented correctly, and test cases pass.	**0 pts** **All Test Cases Failed** Incorrect implementation or none of the test cases pass.	25 pts
			Total Points: 100