**Student ID:** \_Thanh Dat Nguyen\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Full Names:** \_610761\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Web Application Programming

(CS472)

(45 points)

(April 2020)

Instructor: O. Kalu

Midterm Exam

1. The exam duration is 2 hours.
2. The exam is an online, computer-based exam; so you may use a computer for both the Part 1 (theory) and Part 2 (coding) tasks.
3. **This exam paper document is a copyrighted material and so it must not be copied or reproduced or transferred or shared or distributed**.
4. You are expected to use an IDE or any Code Editor tool of your choice to implement your solutions for the questions in the Part 2 (Web Application Coding).
5. During the exam, if you have any question, please **send it via a Chat to me** in the MS Teams app.
6. Upon completion, put your entire Exam (including the projects/folders with your **source code** and this document with your typed-in answers**)** into a single zip file named **MidtermExam.zip**, and submit to Sakai, under the Assignment titled, “Midterm Exam”.
7. **NOTE**: ***If you fail to submit your exam to Sakai because it has past the Submission due time, and you then email it instead, then be aware that your maximum possible score will be 83%,and only if your work scores up to that level.***

--------------------------------------------------------------------------------------------------------------------

Make sure to include the screenshots of your results, where it is required.

--------------------------------------------------------------------------------------------------------------------

(CS472 - WAP)

(April 2020)

Part 1 - Theory (10 points)

**Part I – Theory (True/False, Short answers, Multiple choice questions):** (10 points)

**Note:** *For these questions, please follow the instructions given for each individual question and do type all your answers right here in this document.*

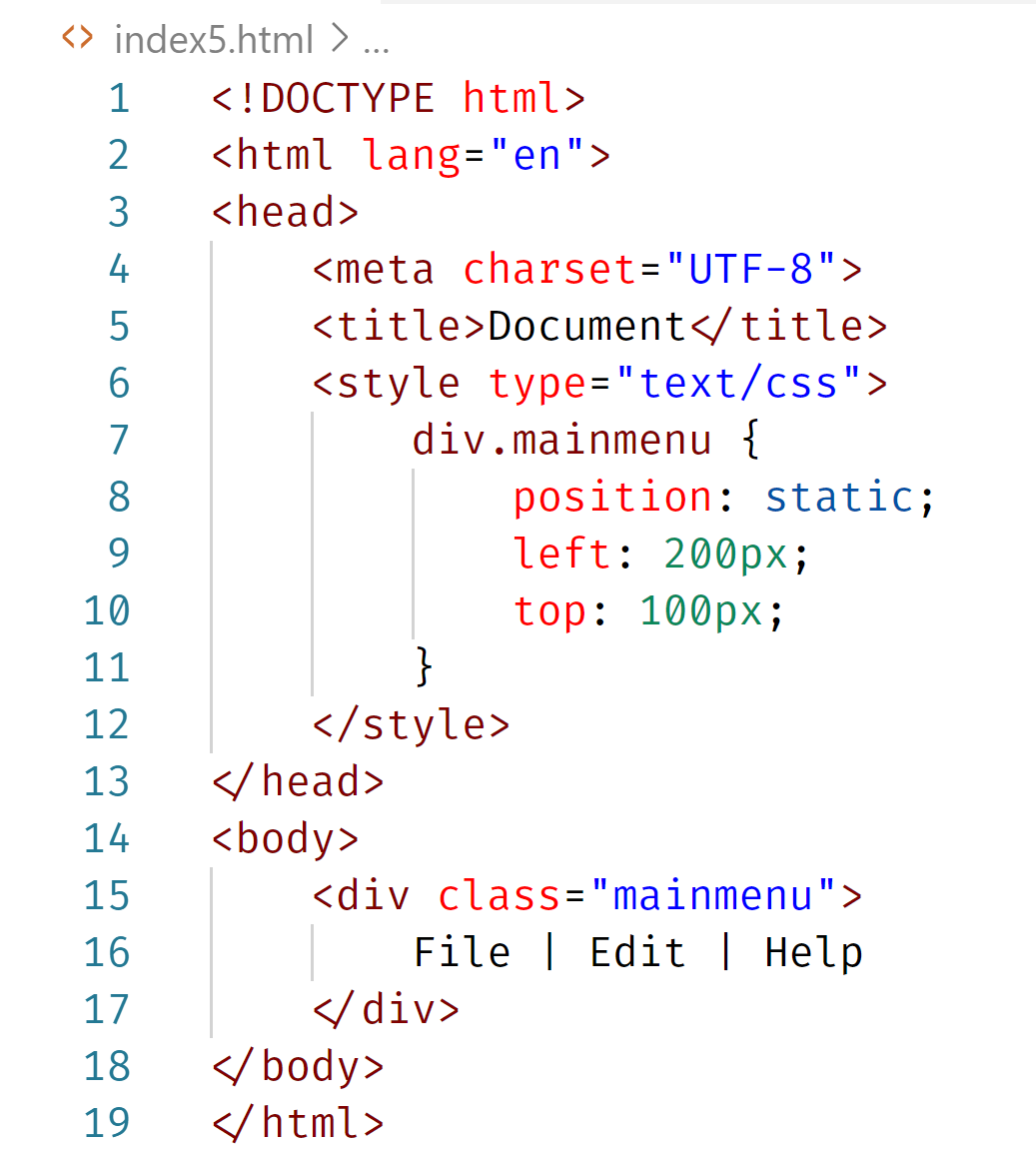
1. (8 points) Answer the following questions with True or False. For each answer, give a rationale (i.e. If True state how, if False state why. No rationale, earns you just half of the points if your True/False answer is correct, and zero point if your True/False answer is incorrect).
   1. (2 points) In order to create a sub-form inside a Web-form, an HTML <form> element can be contained or nested inside another HTML <form> on a web page.

True or False?

\_\_\_\_\_False\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Rationale:** \_ We can have multi form in page but we can’t have nested form because if we have nested form and action lead to submit event we can’t decide which form will submit\_\_\_\_\_\_\_\_\_\_\_

* 1. (2 points) Consider the HTML markup and CSS styling code presented in the Figure below.



Based on the code, the <div> with class named, mainmenu, will be displayed on the webpage, positioned at 200pixels from the left margin and 100pixels from the top margin.

True or False?

\_\_\_\_\_\_\_False\_\_\_\_\_\_\_\_\_\_

**Rationale**: because this div is position static we can’t use offset here if we want to use offset it should change to position relative to use it

* 1. (2 points) Consider the HTML markup and CSS styling code presented in the Figure below.



Based on the code, the two paragraphs inside the article will be displayed with background-color of lightgreen.

True or False?

\_\_\_\_\_\_\_False\_\_\_\_\_\_\_\_

**Rationale**: Because I use > so that mean p have to be a directed child of article.blogpost but in this code section is a directed child of article.blogpost. so not thing happen.

* 1. (2 points) Consider the HTML markup and CSS styling code presented in the Figure below.



Based on the code, the two paragraphs inside the article will be displayed with background-color of lightgreen.

True or False?

\_\_\_\_\_True\_\_\_\_\_\_\_\_\_\_

**Rationale**: because it the selector is find the p inside article have class blogpost.

1. (2 points) What is the difference between the 2 CSS properties settings,

**visibility: hidden** versus **display: none**?

Display: none will the page render as though the element doesn’t exist.

Visibility: hidden it will hide an element but the element still take up the space it would if it was fully visible.

(CS472 - WAP)

(April 2020)

Part 2 - Coding (35 points)

**Part II – JavaScript/Web Application Programming skills:** (35 points)

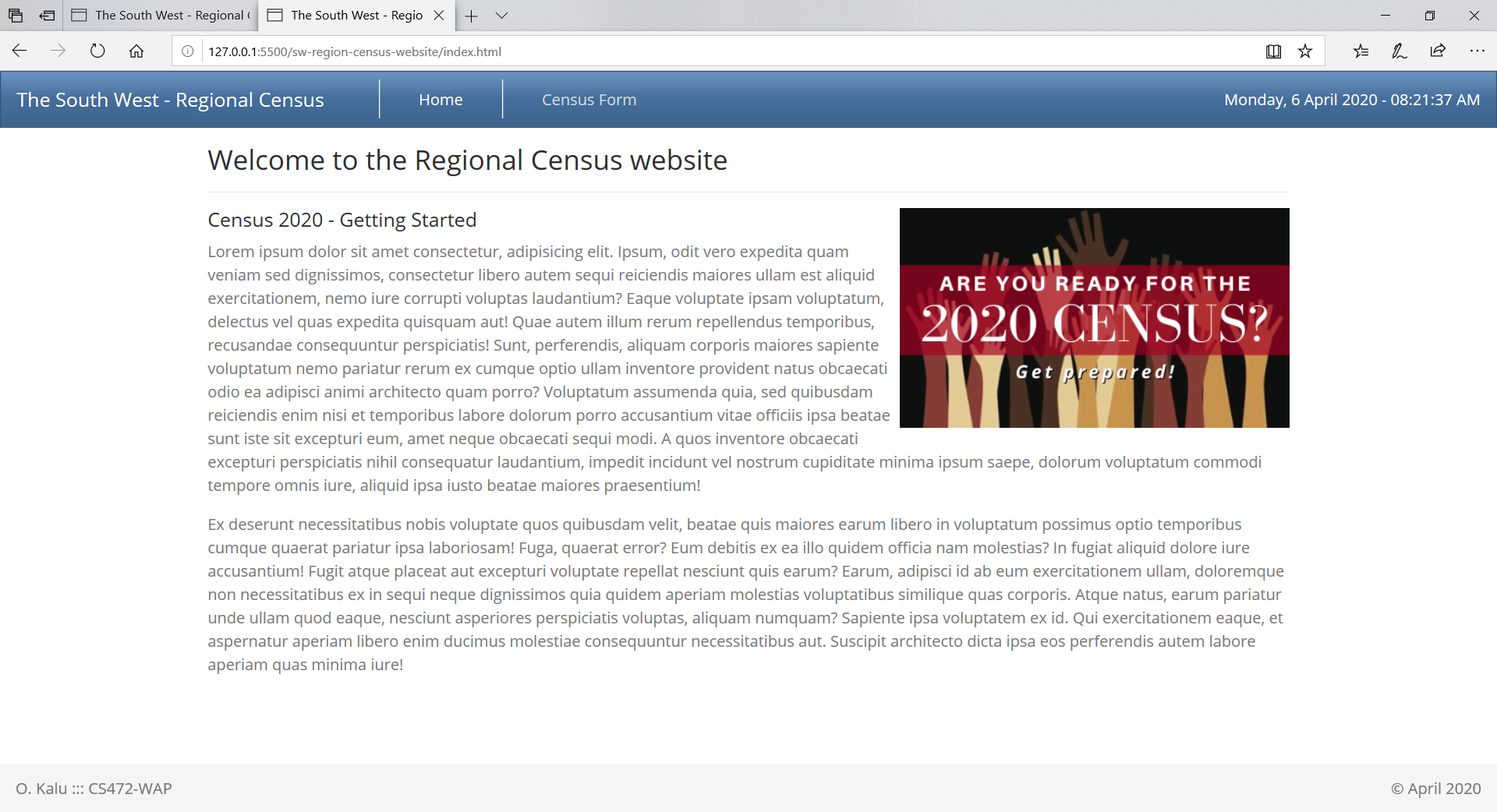
**Note:** *For the tasks in these questions, where applicable, you are expected to take screenshot(s) of your web UI(s), save into a .png or .jpg image file, placed inside a folder named, screenshots and include these in the MidtermExam.zip file, you submit or copy/paste your screenshots to the bottom of the associated question(s) right here in this document.*

1. (20 points) **Implementing a client-side JavaScript Web Application**

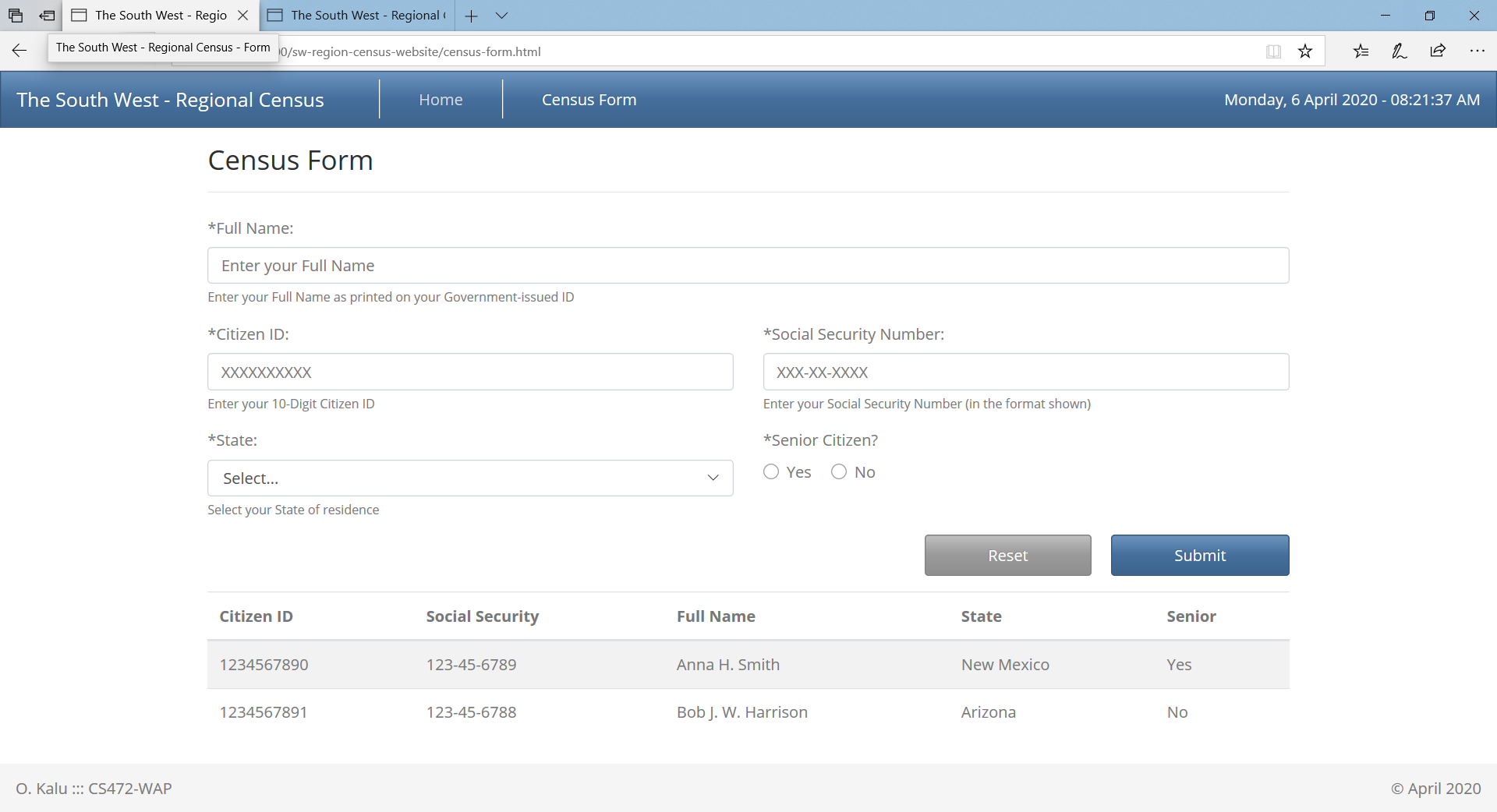
Assume you have been hired by the Regional Office of Census Administration of the South West states, USA. And they have tasked you to design and build a website for the 2020 Census, including just a homepage and a Census Data Collection form (see the sample web UIs shown below).

Using HTML, CSS and JavaScript, implement the website, as shown in the UI screenshots below, with the features and functionalities, specified/listed further below:

**Homepage**:



**Census Form**:



* 1. Code the User interface of the web application using standards-compliant, semantically-correct HTML5 markup, including all the web pages and form fields as shown in the UI screenshots above.
  2. Apply styling using Bootstrap or you may apply your own custom CSS styling to produce the same or similar look and layout. **Note:** Your UI does NOT necessarily have to be exactly the same as the sample shown above. But it should have all the necessary form data input fields, labels, images and buttons etc.
  3. Citizen ID, Social Security Number, Full Name, State and Senior Citizen are all required data, to submit the Census form.
  4. Add validation using appropriate regular expression to ensure that each Citizen ID and Social Security Number entered must be in the specified format of, XXXXXXXXXX and XXX-XX-XXXX, respectively, as shown on the Census form UI above. And where X is a numeric digit.
  5. The user can record citizens’ data for the following 3 states of the South West region – Arizona, New Mexico and Texas. Therefore, your drop-down list data field on the Census form should contain these 3 options.
  6. Using JQuery and/or plain-vanilla JavaScript and DOM API, implement code to submit the 2 sample citizens’ data given below, and also display/append the submitted data in a table at the bottom of the census form, as shown in the screenshot above. The user should be able to submit many citizens’ data and see them listed on the webpage.

**Sample Citizens’ Data for submission to test/demonstrate your webapp**:

Citizen 1:

Citizen ID: 1234567890,

Social Security Number: 123-45-6789,

Full Name: Anna H. Smith,

State: New Mexico,

Senior Citizen: Yes

Citizen 2:

Citizen ID: 1234567891,

Social Security Number: 123-45-6788,

Full Name: Bob J. Harrison,

State: Arizona,

Senior Citizen: No

(**Please note:** Points will be awarded based on your adherence to Web programming recommended best practices such as use of standards-compliant, semantically-correct HTML5 markup, unobtrusive CSS, unobtrusive JavaScript etc).

1. (6 points) **Implementing** **JavaScript objects and Inheritance** – RedHondaAccord IS-A Car:
   1. (3 points)

Define a base object named, Car, as JavaScript object literal, with the following three properties:

**make** - set its initial value as ‘default’

**model** - set its initial value as ‘default’

**color** - set its initial value as ‘default’

and include the following two methods:

**drive** - takes a parameter named, speed, and prints-out the following message to

the console – “The [color] [make] [model] is driving at [speed] mph.

**stop** - prints-out the following message to the console – “The [color] [make] [model] is stopping.

Using inheritance, create a derived object named, **redHondaAccord**, which inherits the properties and methods from the Car object (**Note**: Use Object.create(…) method).

Assign the values, ‘red’, ‘Honda’ and ‘Accord’ to the respective properties of color, make, and model of the RedHondaAccord object.

Invoke the drive(…) method on the RedHondaAccord object, passing it a speed value of 200.

const Car = {

make: "default",

model: "default",

color: "default",

drive: function(speed) {

console.log(`The ${this.color} ${this.make} ${this.model} is driving at ${speed} mph`);

},

stop: function() {

console.log(`The ${this.color} ${this.make} ${this.model} is stopping`);

},

};

const redHondaAccord = Object.create(Car);

redHondaAccord.color = "red";

redHondaAccord.make = "Honda";

redHondaAccord.model = "Accord";

redHondaAccord.drive(200);

Open the Web browser console and take a screenshot of your result and save it to a filenamed, P2Q2-1.png.

* 1. (3 points)

Re-implement the above described inheritance structure, but this time, using a Constructor function. i.e. define a constructor function named Car, with the properties - make, model and color.

Then add to it, the drive(…) method and the stop() method.

Next, based on your Constructor function, create a derived object named RedHondaAccord, with the values, ‘red’, ‘Honda’ and ‘Accord’ for color, make and model, respectively.

Finally, invoke the RedHondaAccord object’s drive(…) method, passing in the value of speed, 300.

Open the Web browser console and take a screenshot of your result and save it to a filenamed, P2Q2-2.png.

function car(make, model, color) {

this.make = make;

this.model = model;

this.color = color;

}

car.prototype.drive = function(speed) {

console.log(`The ${this.color} ${this.make} ${this.model} is driving at ${speed} mph`);

};

car.prototype.stop = function(speed) {

console.log(`The ${this.color} ${this.make} ${this.model} is stopping`);

};

const RedHondaAccord = new car("Honda", "Accord", "Red");

RedHondaAccord.drive(200);

1. (6 points)

By applying the Modules Pattern, implement code for a module named, arrayUtils.

In your arrayUtils module, implement the following 2 utility functions:

3.1 Function named, **multiplesCount**, that takes as input, an array of numbers, **nums**, and a base number, **b**, and returns how many of those numbers contained in the array, nums, are multiples of b.

For example:

**multiplesCount([1,2,3,4,5,6], 3)** should return 2; **multiplesCount([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15], 5)** should return 3;

and so on.

Implement this **multiplesCount** function using JavaScript functional programming style and arrow functions, only. So, no use of ‘for…loop’ or ‘while…loop’ or ‘if’ statements allowed].

3.2 Function named, **reverseInPlace,** that takes as input an array of numbers, nums, and it reverses the order of the elements in the array, in-place, without declaring/creating a new empty array and pushing the elements into it, in reverse order.

For example:

**reverseInPlace ([1,2,3,4,5,6])** should return [6,5,4,3,2,1];

const arrayUtils = (function() {

function multiplesCount(nums, b) {

const multiplesOf = nums.filter(number => number % b === 0);

return multiplesOf.length;

}

function reverseInPlace(nums) {

return nums.reverse();

}

return {

multiplesCount: multiplesCount,

reverseInPlace: reverseInPlace

};

})();

1. (3 points) Implement code to add a new method named, printTheName, to the JavaScript Array object. The printTheName method should take as input parameter, a string representing a name and it simply prints-out the name to the console.

For example:

const array = [];

**array.printTheName(“Anna H. Smith”)** should print the name, “Anna H. Smith” to the console.

**array.printTheName(“Bob J. Harrison”)** should print the name, “Bob J. Harrison” to the console; and so on.

Array.prototype.printTheName = function(name) {

console.log(name);

};

**//-- The End --//**