# Exam Preparation II

Submit your solutions here: <https://alpha.judge.softuni.org/contests/exam-preparation-ii/4513>

## 1. Unit Test: Fruits

Test a given method which takes in **a <string, int> dictionary representing fruits** and **a string fruit name**. It returns **the quantity of the fruit**.

The method is found in the Fruits.cs file:

A computer screen shot of a computer code

Description automatically generated

You are given a **test** **file** FruitsTests.cs containing **5 empty tests**. Implement all tests:

A screenshot of a computer program

Description automatically generated

When you are ready make sure your **tests run:**

A screenshot of a computer

Description automatically generated

**IMPORTANT:** **DO NOT REMOVE OR CHANGE ANY NAMESPACES AND USINGS.**

## 2. Unit Test: ToDo

You are given a **folder of 2 classes -** TaskItemandToDoList. The TaskItem **class** is just a helper class:

A screen shot of a computer code

Description automatically generated

The ToDoList class holds a **list of tasks** and **methods** for **using the list** that you will **test**:

A computer screen shot of a program

Description automatically generated

A computer screen shot of text

Description automatically generated

A computer screen shot of a program

Description automatically generated

You will need to use the test file ToDoListTests.cs, inside they are **5 empty tests with a setup method:**

A screenshot of a computer program

Description automatically generated

When you are ready make sure your **tests run:**

A screenshot of a computer

Description automatically generated

**IMPORTANT:** **DO NOT REMOVE OR CHANGE ANY NAMESPACES AND USINGS.**

**3. JS Unit Testing**

After opening the project, run **npm install** in the console.

**Your Task**

Using **Mocha** and **Chai** write **JS Unit Tests** to test a variable named **chooseYourCar**, which represents an object. You may use the following code as a template:

|  |
| --- |
| describe(**"*Tests* …"**, **function**() {  describe(**"*TODO* …"**, **function**() {  ***it***(**"*TODO …*"**, **function**() {  *//* ***TODO:*** …  });  });  *//* ***TODO:*** …  }); |

The object that should have the following functionality:

* **choosingType (type, color,** **year) -** A function that accepts **three** parameters: **string**, **string**, and **number**.
* If the **year** is **less** than 1900 and the **year** is **more** than 2022, **throw** an error: **"Invalid Year!"**
* If the value of the string **type** is different from "**Sedan**", **throw** an error: **"This type of car is not what you are looking for."**
* To be picked, the **car** must meet the **following requirement**:
  + If the **year** of the car is **greater** or **equal** to **2010**, **return** the string:

**"This ${color} ${type} meets the requirements, that you have."**

* Otherwise, if the above conditions are **not** met, **return** the following message:

**"This ${type} is too old for you, especially with that ${color} color."**

* There is **no** need for **validation** for the **input**, you will always be given two strings, and number.
* **brandName** **(brands, brandIndex) -** A function that accepts an **array** and **number**. The **brands** array will store the brand names (["**BMW**", "**Toyota**", "**Peugeot**"…]).
  + You must **remove** an **element** (brand) from the **array** that is located on the **index** specified as a parameter.
  + Finally, **return** the changed array of brands as a string, joined by a comma and a space.
  + There is a **need for validation** for the input, an **array** and index may not always be valid. In case of submitted **invalid** parameters, **throw** an error **"Invalid Information!"**:
    - If passed **brands** parameteris **not** an array.
    - If the **index** is not a number and is outside the limits of the array.
* **CarFuelConsumption (distanceInKilometers, consumptedFuelInLitres) -** A function that accepts two parameters: **number, number**.
* You test drive the car to find out what its consumption is.
* You need to **calculate** liters per 100 kilometers consumption by **dividing** the fuel consumption by 100 and then **multiply** by distance.
  + - **The result must be formatted to the second digit after the decimal point.**
* If the liters/100km is **less** or **equal** to 7L. **return** the following message:

**"The car is efficient enough, it burns ${litersPerHundredKm} liters/100 km."**

* Else, **return** the following message:

**"The car burns too much fuel - ${litersPerHundredKm} liters!"**

* You **need to validate** the input, if the **distanceInKilometers** and **consumptedFuelInLitres** are not **numbers**,or are **negative** numbers, **throw** an error: **"Invalid Information!"**.

**JS Code**

To ease you in the process, you are provided with an implementation that meets all of the specification requirements for the **chooseYourCar** object.

**Submission**

Submit your tests inside a **describe()** statement, as shown above.