**JS Advanced - Exam**

**Problem 3. Unit Testing**

**Your Task**

Using **Mocha** and **Chai** write **JS Unit Tests** to test a variable named **dealership**, 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:

* **newCarCost(oldCarModel, newCarPrice)** - A function that accepts string and a number:
  + the function checks if you are returning your old car to the dealers or not
  + if you are returning your old car, you will get a fixed amount of money deducted from your new car price
  + then the function returns the price for the car
* **carEquipment(extrasArr, indexArr)** - A function that accepts two arrays:
  + the first arrays includes the available extras for your car ([‘heated seats’, ‘sliding roof’, ‘sport rims’, ‘navigation’, etc.])
  + the second array includes the index of the extras you would want ([0, 3, 4])
  + every given index in the indexArr will be valid
  + then the function returns an array with all the selected extras
* **euroCategory(category)** - A function that accepts a single parameter (number):
  + the function checks the ecology of your new car
  + then the function returns a message regarding the final price you will have to pay, depending on your car eco category

**JS Code**

To ease you in the process, you are provided with an implementation which meets all of the specification requirements for the **pizzUni** object:

|  |
| --- |
| **dealership.js** |
| let dealership = {      newCarCost: function (oldCarModel, newCarPrice) {          let discountForOldCar = {              'Audi A4 B8': 15000,              'Audi A6 4K': 20000,              'Audi A8 D5': 25000,              'Audi TT 8J': 14000,          }          if (discountForOldCar.hasOwnProperty(oldCarModel)) {              let discount = discountForOldCar[oldCarModel];              let finalPrice = newCarPrice - discount;              return finalPrice;          } else {              return newCarPrice;          }      },      carEquipment: function (extrasArr, indexArr) {          let selectedExtras = [];          indexArr.forEach(i => {              selectedExtras.push(extrasArr[i])          });          return selectedExtras;      },      euroCategory: function (category) {          if (category >= 4) {              let price = this.newCarCost('Audi A4 B8', 30000);              let total = price - (price \* 0.05)              return `We have added 5% discount to the final price: ${total}.`;          } else {              return 'Your euro category is low, so there is no discount from the final price!';          }      }  } |

**Submission**

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

const dealership = require('./unit');

const { assert } = require('chai');

describe("Dealership …", function () {

    it("returnCar", function () {

        assert.equal(dealership.newCarCost('Audi A4 B8', 30000), 15000);

        assert.equal(dealership.newCarCost('Audi X', 30000), 30000);

    });

    it("returnExtras", function () {

        assert.deepEqual(dealership.carEquipment(['a', 'b', 'c'], [0]), ['a']);

    });

    it("above or equal to 4 standart", function () {

        assert.equal(dealership.euroCategory(4), `We have added 5% discount to the final price: 14250.`)

    });

    it("below 4 standart", function () {

        assert.equal(dealership.euroCategory(1), `Your euro category is low, so there is no discount from the final price!`)

    });

});