JS Advanced Exam

Problem 3. Unit Testing

Your Task

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

`isItExpensive (issue)` - A function that accepts one parameter: **string**.

- If the value of the parameter **`issue`** is equal to "**`Engine`**" or "**`Transmission`**", return: **`` `The issue with the car is more severe and it will cost more money` ``**
- Otherwise, if the above conditions are not met, **return** the following message:

  `` `The overall price will be a bit cheaper` ``
- There is **no** need for **validation** for the **input**, you will always be given a string.

`discount (numberOfParts, totalPrice)` - A function that accepts two parameters: **number** and **number**.

- Percentage of discount based on the **`numberOfParts:`**
  - **`15%`** when **`numberOfParts`** is higher than 2 and smaller or equal to 7
  - **`30%`** when **`numberOfParts`** is higher than 7
- You need to **calculate** and **return** the **price** you will save, depending on the **discount**.

- If the **`numberOfParts`** is smaller or equal to **2,** return**:**

  `"You cannot apply a discount"`
- Otherwise, calculate the discount and **return** the following message:

  `` `Discount applied! You saved ${result}$` ``
- You need to validate the input, if the **`numberOfParts`** and **`totalPrice`** are not a **number**, **throw** an error: "**`Invalid input`**"

`partsToBuy(partsCatalog, neededParts)` **-** A function that accepts two arrays.

- The **`partsCatalog`** array will store the parts and the price for them ([{ **part**: "**blowoff valve**", **price**: **145** }, { **part: "coil springs", price: 230** } ...])
- The **`neededParts`** array will store the parts that you need to buy (["**blowoff valve**", "**injectors**" ...])
- You must iterate through both the arrays and calculate the **total price** of the **parts** that are equal to the **`neededParts`**.
- If **`partsCatalog`** is empty**,** return **`0`**
- Finally, **return** the total price of all parts needed.
- There is a need for validation for the input, may not always be valid. In case of submitted **invalid** parameters, **throw** an error "**`Invalid input`**":
  - If passed **`partsCatalog`** or **`neededParts`** parameters are not an arrays.

JS Code

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

carService.js

```
const carService = {
  isItExpensive(issue) {
    if (issue === "Engine" || issue === "Transmission") {
      return `The issue with the car is more severe and it will cost more money`;
    } else {
      return `The overall price will be a bit cheaper`;
    }
  },
  discount(numberOfParts, totalPrice) {
    if (
```

```
      typeof numberOfParts !== "number" ||
      typeof totalPrice !== "number"
    ) {
      throw new Error("Invalid input");
    }

    let discountPercentage = 0;

    if (numberOfParts > 2 && numberOfParts <= 7) {
      discountPercentage = 15;
    } else if (numberOfParts > 7) {
      discountPercentage = 30;
    }

    let result = (discountPercentage / 100) * totalPrice;

    if (numberOfParts <= 2) {
      return "You cannot apply a discount";
    } else {
      return `Discount applied! You saved ${result}$`;
    }
  },
  partsToBuy(partsCatalog, neededParts) {
    let totalSum = 0;

    if (!Array.isArray(partsCatalog) || !Array.isArray(neededParts)) {
      throw new Error("Invalid input");
    }
    neededParts.forEach((neededPart) => {
      partsCatalog.map((obj) => {
        if (obj.part === neededPart) {
          totalSum += obj.price;
        }
      });
    });

    return totalSum;
  },
};
```

Submission

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