**JS Advanced Exam Retake – 02 Aug 2023**

**Problem 3. Unit Testing**

**Your Task**

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

* **isTypeSuitable(type, dietaryRestriction)** This function determines if a recipe type is suitable for a given dietary restriction. It takes in two parameters: a **type** (**string**) representing the recipe type and a **dietaryRestriction** (**string**) representing the dietary restriction.
* If the dietaryRestriction is "**Vegetarian**" and the type is "**Meat**", it **returns** the message:

**"This recipe is not suitable for vegetarians".**

* If the **dietaryRestriction** is "**Vegan**" and the type is either "**Meat**" or "**Dairy**", it **returns** the message:

**"This recipe is not suitable for vegans"**

* For any other combination of **type** and **dietaryRestriction**, it **returns** the message:

**"This recipe is suitable for your dietary restriction"**

* You need to validate the input, if the **type** and **dietaryRestriction** are not a **strings**, **throw** an error: "**Invalid input**".
* **isItAffordable (price, budget) -** A function that accepts two parameters: **number** and **number**.
* It **calculates** the remaining **budget** by **subtracting** the **price** from the **budget**.
* If the remaining **budget** is **less** than 0, it **returns** the message:

**"You don't have enough budget to afford this recipe"**

* Otherwise, it **returns** the message:

**"Recipe ingredients bought. You have ${remainingBudget}$ left"**

* Where **remainingBudget** is the calculated value.
* You need to validate the input, if the **price** and **budget** are not a **number**, **throw** an error: "**Invalid input**".
* **getRecipesByCategory(recipes, category)** This function filters an array of **recipes** based on a desired **category** and **returns** an array of recipe titles. It takes two parameters: **recipes** (**array**) representing the array of recipe objects and **category** (**string**) representing the desired **category.**
  + It filters the **recipes** array based on the **category** and creates a **new** array **filteredRecipes** containing only the **recipes** that match the desired **category**.
  + The **recipes** array will store the titles and the category of its recipes ([{ **title**: " Spicy Tofu Stir-Fry ", **category**: " **Asian** " }, ...])
  + It maps through the **filteredRecipes** array to extract the **titles** of the recipes and **returns** an array of these **titles**.
  + There is a need for validation for the input, an **array** and **string** may not always be valid. In case of submitted **invalid** parameters, **throw** an error "**Invalid input**":
    - If passed **recipes** parameteris not an array.
    - If the **category** is not a string.

**JS Code**

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

|  |
| --- |
| recipeSelection.js |
| const recipeSelection = {    isTypeSuitable(type, dietaryRestriction) {      if (typeof type !== "string" || typeof dietaryRestriction !== "string"){        throw new Error("Invalid input");      }      if (dietaryRestriction === "Vegetarian" && type === "Meat") {        return "This recipe is not suitable for vegetarians";      } else if (dietaryRestriction === "Vegan" && (type === "Meat" || type === "Dairy")) {        return "This recipe is not suitable for vegans";      } else {        return "This recipe is suitable for your dietary restriction";      }    },    isItAffordable(price, budget) {      if (typeof price !== "number" || typeof budget !== "number") {        throw new Error("Invalid input");      }      let remainingBudget = budget - price;      if (remainingBudget < 0) {        return "You don't have enough budget to afford this recipe";      } else {        return `Recipe ingredients bought. You have ${remainingBudget}$ left`;      }    },    getRecipesByCategory(recipes, category) {      if (!Array.isArray(recipes) || typeof category !== "string") {        throw new Error("Invalid input");      }      const filteredRecipes = recipes.filter((recipe) => recipe.category === category);      return filteredRecipes.map((recipe) => recipe.title);    },  }; |

**Submission**

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