**Problem 3. Online Store**

**Your Task**

Using Mocha and Chai, write JavaScript unit tests to test an object named **onlineStore**. You may use the following code as a template:

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

The **onlineStore** object represents an online shopping platform and contains the following functionality:

**isProductAvailable(product, stockQuantity) -**  A function that accepts two parameters: a **string** representing a product and a **number** representing the stock quantity.

* If the **stockQuantity** is **less** than or **equal** to 0, and the product is considered out of stock, the function should **return** a message:

**`Sorry, ${product} is currently out of stock.`**

* If the **stockQuantity** is **greater** than 0, the product is available, and the function should return:

**`Great! ${product} is available for purchase.`**

There is a need for validation for the input, the **product** parameter should be an **string**, and the **stockQuantity**  should be a **number**. In case of invalid parameters, the function should **throw an error**:

**"Invalid input."**

* **canAffordProduct(productPrice, accountBalance)-** A function that accepts two parameters: a **number** representing the product price and a **number** representing the account balance.
* The function should calculate if the user can afford to buy the product by **subtracting** the **product** price from the account **balance**.
* If the result is **less** than 0, the user doesn't have enough funds, and the function should **return**:

**"You don't have sufficient funds to buy this product."**

* If the result is **greater** than or **equal** to 0, the purchase is successful, and the function should **return**:

**`Product purchased. Your remaining balance is $${remainingBalance}.`**

* You need to validate the input; if **productPrice** and **accountBalance** are **not** numbers, the function should **throw an error:**

**"Invalid input."**

* **getRecommendedProducts(productList, category)** A function that accepts two parameters: an **array** of **objects** representing products and a **string** representing a category.
  + The **productList** array stores objects with product names and categories (e.g., **[{ name: "Camera", category: "Photography" }, ...])**.
* The function should find and **return** product names that match the specified **category** in the format:  
  **`Recommended products in the ${category} category: ${product}, ${product}`**
  + If there are no recommended products in the specified category, the function should return:

**`Sorry, we currently have no recommended products in the ${category} category.`**

There is a need for validation for the input, the **productList** parameter should be an **array**, and the **category** should be a **string**. In case of invalid parameters, the function should **throw an error**:

**"Invalid input."**

**JS Code**

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

|  |
| --- |
| onlineStore.js |
| const onlineStore = {      isProductAvailable(product, stockQuantity) {      if (typeof product !== "string" || typeof stockQuantity !== "number") {              throw new Error("Invalid input.");          }          if (stockQuantity <= 0) {          return `Sorry, ${product} is currently out of stock.`;        } else {          return `Great! ${product} is available for purchase.`;        }      },      canAffordProduct(productPrice, accountBalance) {        if (typeof productPrice !== "number" || typeof accountBalance !== "number") {          throw new Error("Invalid input.");        }          let remainingBalance = accountBalance - productPrice;          if (remainingBalance < 0) {          return "You don't have sufficient funds to buy this product.";        } else {          return `Product purchased. Your remaining balance is $${remainingBalance}.`;        }      },      getRecommendedProducts(productList, category) {        let recommendedProducts = [];          if (!Array.isArray(productList) || typeof category !== "string") {          throw new Error("Invalid input.");        }          productList.forEach((product) => {          if (product.category === category) {            recommendedProducts.push(product.name);          }        });          if (recommendedProducts.length === 0) {          return `Sorry, we currently have no recommended products in the ${category} category.`;        } else {          return `Recommended products in the ${category} category: ${recommendedProducts.join(", ")}`;        }      },    }; |

**Submission**

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