**JS Advanced Retake Exam**

**Problem 3. Unit Testing**

**Your Task**

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

**ageRestrictions (movieRating) -** A function that accepts one parameter: **string**.

* If the value of the parameter **movieRating** is equal to "**G**", return:

**`All ages admitted to watch the movie`**

* If the value of the parameter **movieRating** is equal to "**PG**", return:

**`Parental guidance suggested! Some material may not be suitable for pre-teenagers`**

* If the value of the parameter **movieRating** is equal to "**R**", return:

**`Restricted! Under 17 requires accompanying parent or adult guardian`**

* If the value of the parameter **movieRating** is equal to "**NC-17**", return:

**`No one under 17 admitted to watch the movie`**

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

**`There are no age restrictions for this movie`**

* There is **no** need for **validation** for the **input**, you will always be given a string.

**moneySpent (tickets, food, drinks) -** A function that accepts three parameters: **number, array** and **array**.

* Calculate the **total cost** you are going to pay depending on the **tickets** count, purchased **food** and **drinks:**
  + Each **ticket** costs **15** levs
* The theater offers **two** options for **foods** and **drinks**:
  + The two options for foods are:
    - **Nachos,** which costs **6** levs
    - **Popcorn,** which costs **4.50** levs
  + The two options for drinks are:
    - **Soda**, which costs **2.50** levs
    - **Water**, which costs **1.50** levs
* If the **total cost** is bigger than **50** a **discount** of **20%** shouldbeapplied**.** Then **return** the following message with the cost **rounded** to the **second** **digit** after the decimal point**:**

**`The total cost for the purchase with applied discount is ${totalCost}`**

* Otherwise, return:

**`The total cost for the purchase is ${totalCost}`**

* You need to validate the input, if the **tickets, food** and **drinks** are not a **number, array and array**, **throw** an error: "**Invalid input**"

**reservation (rowsArray, neededSeatsCount) -** A function that accepts **array** and **number**.

* + The **rowsArray** array will store the rows and the free seats for them ([{ **rowNumber**: **1**, **freeSeats**: **7** }, { **rowNumber: 2, freeSeats: 5** }...])
  + The **neededSeatsCount** will be a number higher than **0**, representing the count of the needed seats for the reservation.
  + You must iterate through both the **rowsArray** and extract the **numbers** of the **rows** that have enough free spaces
* Finally, **return** the row with the largest number.
  + 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 **rowsArray** or **neededSeatsCount** parameters are not an array and number.

**JS Code**

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

|  |
| --- |
| **movieTheater.js** |
| const movieTheater = {    ageRestrictions(movieRating) {      switch (movieRating) {        case "G":          return "All ages admitted to watch the movie";        case "PG":          return "Parental guidance suggested! Some material may not be suitable for pre-teenagers";        case "R":          return "Restricted! Under 17 requires accompanying parent or adult guardian";        case "NC-17":          return "No one under 17 admitted to watch the movie";        default:          return "There are no age restrictions for this movie";      }    },    moneySpent(tickets, food, drinks) {      if (        typeof tickets !== "number" ||        !Array.isArray(food) ||        !Array.isArray(drinks)      ) {        throw new Error("Invalid input");      }      let bill = 0;      bill += tickets \* 15;      food.forEach((element) => {        switch (element) {          case "Nachos":            bill += 6;            break;          case "Popcorn":            bill += 4.5;            break;        }      });      drinks.forEach((element) => {        switch (element) {          case "Soda":            bill += 2.5;            break;          case "Water":            bill += 1.5;            break;        }      });      if (bill > 50) {        bill -= bill \* 0.2;        return `The total cost for the purchase with applied discount is ${bill.toFixed(          2        )}`;      } else {        return `The total cost for the purchase is ${bill.toFixed(2)}`;      }    },    reservation(rowsArray, neededSeatsCount) {      if (!Array.isArray(rowsArray) || typeof neededSeatsCount !== "number") {        throw new Error("Invalid input");      }      let availableRows = [];      rowsArray.forEach((row) => {        if (row.freeSeats >= neededSeatsCount) {          availableRows.push(row.rowNumber);        }      });      return Math.max(...availableRows);    },  }; |

**Submission**

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