Position: JavaScript Developer
Level: Junior to Mid-Level

Type: Remote/On-site**Application Deadline**: 03/12/2024

About Us
We are a dynamic and fast-growing tech company looking for a JavaScript developer with a solid understanding of working with arrays, objects, and conditions. This task is part of our interview process to assess problem-solving and coding skills.

Interview Task: JavaScript Problem-Solving

Objective: Demonstrate your ability to manipulate data structures (arrays and objects) and implement logic using JavaScript.

Task Description:

Write a JavaScript function to process and analyze a dataset of user information.

Dataset:

An array of objects representing users:

javascript
Copy code

```
const users = [
    { id: 1, name: "Alice", age: 25, isActive: true, scores: [85, 92,
88] },
    { id: 2, name: "Bob", age: 30, isActive: false, scores: [70, 75,
80] },
    { id: 3, name: "Charlie", age: 35, isActive: true, scores: [95,
90, 93] },
    { id: 4, name: "Diana", age: 28, isActive: true, scores: [60, 65,
70] },
    { id: 5, name: "Eve", age: 40, isActive: false, scores: [80, 85,
88] }
];
```

Requirements:

1. **Filter Active Users**
    - Write a function to filter out only users who are `isActive`.
2. **Calculate Average Score**
    - For each active user, calculate their average score and add it as a new key (`averageScore`) to their object.
3. **Find Top Performer**
    - Identify the active user with the highest average score.
4. **Group Users by Age Range**
    - Group all users (active and inactive) into age ranges:
        - `Under 30`
        - `30 and Above`
    - Return an object with the users categorized into these two groups.

Output Format

Create a function named `processUsers()` that returns an object structured as follows:

javascript
Copy code

```
{
    activeUsers: [ /* array of active users with averageScore */ ],
    topPerformer: { /* user object of top performer */ },
    ageGroups: {
        under30: [ /* users under 30 */ ],
        over30: [ /* users 30 and above */ ]
    }
}
```

Submission Guidelines

- **Code Quality**: Use meaningful variable names, comments, and proper formatting.
- **Technology**: Write in plain JavaScript (no frameworks or libraries).
- **Delivery**: Submit the solution as a GitHub repository link or a file via email.

Bonus Points:

- If you write unit tests for your function.
- If you handle edge cases (e.g., empty datasets, users with no scores).

Evaluation Criteria

- Accuracy of the solution.
- Code readability and structure.
- Problem-solving approach.

We look forward to seeing your solution! If you have any questions, feel free to reach out.

Good luck!

==========

Task Title: Employee Performance Tracker

Objective:
You need to create a JavaScript program that processes employee performance data and determines their eligibility for a bonus.

Instructions:

1. **Input Data:**
   You are given an array of employee objects. Each object contains the following properties:
   - `id` **(unique employee ID)**
   - `name` **(employee name)**
   - `tasksCompleted` **(number of tasks the employee completed)**
   - `rating` **(employee performance rating out of 5)**

```
const employees = [

  { id: 1, name: 'Alice', tasksCompleted: 50, rating: 4.8 },

  { id: 2, name: 'Bob', tasksCompleted: 30, rating: 3.9 },

  { id: 3, name: 'Charlie', tasksCompleted: 70, rating: 4.5 },

  { id: 4, name: 'Diana', tasksCompleted: 20, rating: 3.2 },
```

```
];
```

2. **Requirements:**
   Write a function called `calculateBonuses` that:
   - Loops through the array of employees.
   - Checks if the employee is eligible for a bonus based on these conditions:
     - The employee must have completed at least 40 tasks.
     - The employee must have a rating of 4.0 or higher.
   - If eligible, calculate the bonus amount as `tasksCompleted * 10`.
3. **Output:**
   The function should return a new array of objects for eligible employees. Each object should include:
   - `id`
   - `name`
   - `bonus`

Example Output:
For the given `employees` array, the output should be:

```
[
  { id: 1, name: 'Alice', bonus: 500 },
  { id: 3, name: 'Charlie', bonus: 700 }
]
```

Bonus Challenge:

Add a function to calculate the average rating of all employees and log a message indicating if the company's overall performance is excellent (average rating >= 4.5), good (average rating >= 4.0), or needs improvement (average rating < 4.0).

==========

Task Title: Student Grades Tracker

Objective:
Create a JavaScript program to evaluate students' grades, determine their pass/fail status, and calculate their average scores.

Instructions:

1. **Input Data:**
   You are given an array of student objects. Each object contains the following properties:
   - `id` (unique student ID)
   - `name` (student name)
   - `scores` (an array of numbers representing scores in different subjects)

```
const students = [
  { id: 1, name: 'John', scores: [85, 78, 92] },
  { id: 2, name: 'Sara', scores: [62, 70, 68] },
  { id: 3, name: 'Emma', scores: [90, 95, 94] },
  { id: 4, name: 'Tom', scores: [50, 48, 55] },
];
```

2. **Requirements:**
   Write the following functions:
   a) **`calculateAverage(scores)`**
   - Input: An array of scores (e.g., `[85, 78, 92]`).
   - Output: The average score of the array (e.g., `85`).
3. **b)** **`evaluateStudents(students)`**
   - Input: The `students` array.
   - Output: A new array of objects containing:
     - `id`
     - `name`
     - `averageScore`
     - `status` (`"Pass"` if `averageScore >= 60`, otherwise `"Fail"`)

Output:
The output should look like this:

```
[
  { id: 1, name: 'John', averageScore: 85, status: 'Pass' },
  { id: 2, name: 'Sara', averageScore: 66.67, status: 'Pass' },
  { id: 3, name: 'Emma', averageScore: 93, status: 'Pass' },
  { id: 4, name: 'Tom', averageScore: 51, status: 'Fail' },
];
```

4. **Bonus Challenge:**
    - Write a function `getTopScorer(students)` that returns the student object with the highest average score.
    - Add a function `classPerformance(students)` that evaluates the class's overall performance:
        - Return `"Excellent"` if the average of all students' average scores is >= 80.
        - Return `"Good"` if the average is >= 60 and < 80.
        - Return `"Needs Improvement"` if the average is < 60.

Task Title: Shopping Cart Manager

Objective:
Create a JavaScript program to simulate a basic shopping cart system with functionalities like adding items, removing items, and calculating totals.

Instructions:

1. **Input Data:**
   You are given an array of product objects, each containing:
   - `id` (unique product ID)
   - `name` (product name)
   - `price` (product price per unit)
   - `quantity` (available stock of the product)

```
const products = [
  { id: 1, name: 'Laptop', price: 800, quantity: 10 },
  { id: 2, name: 'Phone', price: 500, quantity: 15 },
  { id: 3, name: 'Headphones', price: 100, quantity: 20 },
  { id: 4, name: 'Charger', price: 25, quantity: 50 },
];
```

2. **Requirements:**
   Write the following functions:
   a) `addToCart(cart, productId, quantity)`
   - Input:
     - `cart` (an array representing the shopping cart)
     - `productId` (ID of the product to add)
     - `quantity` (number of units to add)
   - Output:
     - Add the product to the cart if it exists in `products` and there's enough stock.
     - Update the `quantity` in the cart if the product is already added.

        - Reduce the stock of the product in `products`.
3. **b) `removeFromCart(cart, productId)`**
    - Input:
        - `cart` (the shopping cart)
        - `productId` (ID of the product to remove)
    - Output:
        - Removes the product from the cart if it exists.
        - Restores the stock of the product in `products`.
4. **c) `calculateTotal(cart)`**
    - Input:
        - `cart` (the shopping cart)
    - Output:
        - Returns the total price of items in the cart.

Example Usage:

```
const cart = [];


addToCart(cart, 1, 2); // Add 2 Laptops

addToCart(cart, 3, 1); // Add 1 Headphone

console.log(cart);

// Output: [{ id: 1, name: 'Laptop', price: 800, quantity: 2 }, { id:
3, name: 'Headphones', price: 100, quantity: 1 }]


console.log(calculateTotal(cart));

// Output: 1700


removeFromCart(cart, 1); // Remove Laptop from the cart

console.log(cart);

// Output: [{ id: 3, name: 'Headphones', price: 100, quantity: 1 }]
```

5. **Bonus Challenge:**
    - Write a function `checkout(cart)` that:
        - Empty the cart.
        - Logs a message confirming the purchase with the total cost.
    - Ensure stock in `products` cannot drop below 0.