# JavaScript for QA Problem Set 2

## 1 1. Data Structures and Logic

### Problem 1.1: Deep Equality Check Between Two Objects

Write a function that deeply compares two JavaScript objects and returns true if they are structurally and value-wise equal.

### Problem 1.2: Extract All Unique Keys from Nested JSON

Given a deeply nested JSON object, extract all unique keys and return them as an array.

### Problem 1.3: Flatten a Nested Array to a Single Level

Write a function that flattens a nested array to a one-dimensional array.

# 2 2. String Data Validation

### Problem 2.1: Validate an Email Format with Regex

Create a function that takes a string and returns true if it matches a valid email format.

### Problem 2.2: Detect Repeated Words in a Paragraph

Write a function that scans a paragraph and highlights any repeated word used consecutively (case insensitive).

### Problem 2.3: Mask Credit Card Number Except Last 4 Digits

Write a function that takes a credit card number and masks it, keeping only the last four digits visible.

# 3 3. Arrays, Maps, and Filters

### Problem 3.1: Group Items in an Array by Type

Given an array of objects like `[ {type: 'bug', ...}, {type: 'task', ...} ]`, return an object grouped by `type`.

### Problem 3.2: Find Duplicates and Their Count in an Array

Return an object where each key is a duplicated item and value is the number of occurrences.

### Problem 3.3: Merge and Deduplicate Two Arrays

Write a function to merge two arrays and remove duplicates (numbers or strings).

# 4 4. Async and Promises

### Problem 4.1: Create a Delayed Logger using Promises

Write a function that logs a message after a given delay using `setTimeout` wrapped in a promise.

### Problem 4.2: Sequential API Call Simulator

Simulate calling 3 APIs (mock URLs or functions) one after the other and collecting their results.

### Problem 4.3: Retry Logic with Timeout

Write a function that retries a failed promise-based operation up to 3 times with delay between attempts.

# 5 5. Test Logic Simulation (QA-centric)

### Problem 5.1: Simulate an Assertion Framework

Create a mini version of `expect()` and `toBe()`, allowing basic test assertions like `expect(2 + 2).toBe(4)`.

### Problem 5.2: Field Validator for User Registration

Validate a JS object { `email, password, confirmPassword` }. Return a list of error messages.

### Problem 5.3: Simulate Backend Response Validator

Given a mock JSON response, write a function that verifies if required fields exist and meet certain conditions (e.g., length, value).