JavaScript for QA Problem Set 1 Solutions

1 Basic Logic & Syntax

Problem 1.1: Check if input is a valid number.

```
function isValidNumber(input) {
    return typeof input === 'number' && !isNaN(input);
}
```

Problem 1.2: Return "Pass" if score is 50 or above, otherwise "Fail".

```
function checkPassFail(score) {
    return score >= 50 ? 'Pass' : 'Fail';
}
```

Problem 1.3: Convert Celsius to Fahrenheit.

```
function celsiusToFahrenheit(c) {
    return (c * 9/5) + 32;
}
```

2 Arrays and Looping

Problem 2.1: Count the number of failed tests (status = 'fail').

```
function countFailed(tests) {
    return tests.filter(t => t.status === 'fail').length
        ;
}
```

Problem 2.2: Return only names from an array of user objects.

```
function extractNames(users) {
    return users.map(u => u.name);
}
```

Problem 2.3: Find duplicate elements in an array.

```
function findDuplicates(arr) {
    let seen = {};
    let duplicates = [];
    for (let val of arr) {
        if (seen[val]) {
            duplicates.push(val);
        } else {
            seen[val] = true;
        }
    }
    return [...new Set(duplicates)];
}
```

3 Functions and Validation

Problem 3.1: Check if a user object has all required fields.

```
function isValidUser(user) {
    return user.name && user.email && user.password;
}
```

Problem 3.2: Validate if a string is a valid email (basic check).

```
function isEmail(email) {
    const regex = /^\S+@\S+\.\S+$/;
    return regex.test(email);
}
```

Problem 3.3: Return the longest word in a sentence.

```
function longestWord(sentence) {
    return sentence.split(' ').reduce((a, b) => a.length
        > b.length ? a : b);
}
```

4 Object Handling and Assertions-Like Logic

Problem 4.1: Check if API response contains key fields.

```
function checkApiResponse(response, fields) {
    return fields.every(field => response.hasOwnProperty
       (field));
}
```

Problem 4.2: Compare two objects for equality.

```
function shallowEqual(obj1, obj2) {
    let keys1 = Object.keys(obj1);
    let keys2 = Object.keys(obj2);
    if (keys1.length !== keys2.length) return false;
    return keys1.every(key => obj1[key] === obj2[key]);
}
```

Problem 4.3: Count keys in an object where value is null or undefined.

```
function countEmptyFields(obj) {
    return Object.values(obj).filter(val => val == null)
       .length;
}
```

5 Intermediate Logic and Async Concepts

Problem 5.1: Simulate a fake fetch and handle its result.

```
function fakeFetch() {
    return new Promise((resolve) => {
        setTimeout(() => resolve({ status: 200, data: '
           OK' }), 1000);
    });
}

fakeFetch().then(res => {
    console.log(res.status === 200 ? "Success" : "Fail")
       ;
});
```

Problem 5.2: Remove falsy values from an array.

```
function removeFalsy(arr) {
    return arr.filter(Boolean);
}
```

Problem 5.3: Return how many users have emails ending in '.com'.

```
function countDotComEmails(users) {
    return users.filter(user => user.email.endsWith('.
       com')).length;
}
```

6 Bonus QA-Specific Challenges

Problem 6.1: Normalize API response keys to lowercase.

```
function normalizeKeys(response) {
    let normalized = {};
    for (let key in response) {
        normalized[key.toLowerCase()] = response[key];
    }
    return normalized;
}
```

Problem 6.2: Detect if any value in an array is a duplicate (boolean).

```
function hasDuplicate(arr) {
    return new Set(arr).size !== arr.length;
}
```

Problem 6.3: Check if a password is strong

upper and lower).

```
// For a password to be strong we need a minimum of 8
   characters, a number, upper and lower characters as
   well. (sometimes a special character)

function isStrongPassword(pw) {
    const regex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d).{8,}$
       /;
    return regex.test(pw);
}
```