Postman Problem Set 1 Solutions

Useful examples of test scripts can be found here:
https://learning.postman.com/docs/tests-and-scripts/write-scripts/test-examples/

1 Basic Tests and Assertions

Problem 1.1: Check if status code is 200

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
```

Problem 1.2: Check if JSON response has a specific key

```
pm.test("Response has 'message'", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("message");
});
```

Problem 1.3: Check if a string is contained in response body

```
pm.test("Body includes 'success'", function () {
    pm.expect(pm.response.text()).to.include("success");
});
```

2 Data Validation and Chaining

Problem 2.1: Save token from login response to environment variable

```
var jsonData = pm.response.json();
pm.environment.set("authToken", jsonData.token);
```

Problem 2.2: Assert nested user data is correct

```
pm.test("Check user email", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.user.email).to.eql("qa@example.
       com");
});
```

Problem 2.3: Verify array response contains at least 3 items

```
pm.test("Array has at least 3 items", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.items.length).to.be.at.least(3);
});
```

3 Intermediate Flow Control and Loops

Problem 3.1: Loop through all items and ensure each has an id

```
pm.test("All items have an id", function () {
    var jsonData = pm.response.json();
    jsonData.items.forEach(item => {
        pm.expect(item).to.have.property("id");
    });
});
```

Problem 3.2: Check that one item has a specific value

```
pm.test("At least one item has status active", function
   () {
    var jsonData = pm.response.json();
    const active = jsonData.items.some(item => item.
       status === "active");
    pm.expect(active).to.be.true;
});
```

Problem 3.3: Store a dynamic user ID for future requests

```
var jsonData = pm.response.json();
pm.environment.set("userId", jsonData.user.id);
```

4 Advanced Validations

Problem 4.1: Validate response time is below 500ms

```
pm.test("Response time is < 500ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(500)
        ;
});
```

Problem 4.2: Validate token format using regex

```
pm.test("Token is alphanumeric", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.token).to.match(/^[a-zA-Z0-9]+$/)
        ;
});
```

Problem 4.3: Validate object has all required fields

```
pm.test("User has all required fields", function () {
    var user = pm.response.json().user;
    ["id", "email", "created_at"].forEach(field => {
        pm.expect(user).to.have.property(field);
    });
});
```

5 Bonus Challenges

Problem 5.1: Chain login -¿ get profile -¿ assert name

Assume you got a token from a login call and used it in a second request. Validate that the user name from ‘/profile‘ is correct.

```
pm.test("User name is John QA", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.name).to.eql("John QA");
});
```

Problem 5.2: Check that all users have valid emails

```
pm.test("All users have valid emails", function () {
    var jsonData = pm.response.json();
    const emailRegex = /^\S+@\S+\.\S+$/;
    jsonData.users.forEach(user => {
        pm.expect(user.email).to.match(emailRegex);
    });
});
```

Problem 5.3: Check custom header is present in response

```
pm.test("X-Custom-Header exists", function () {
    pm.response.to.have.header("X-Custom-Header");
});
```