CA650 XUnit Assignment

For this assignment, I chose to write my code in JavaScript as I have prior knowledge of how to code in JavaScript and test the same.

I use the Jest which is a JavaScript testing framework to write the test suite for the Queue class code and test it by various test cases.

Following is the 1st draft Queue code in JavaScript:

Queue code:

```
class Queue {
 constructor(size) {
   if(size > 5) {
     throw new Error("Queue size should not be more than 5");
   }
   this.queue = []; // Array to store the queue elements
   this.size = size;
 }

 add(element) {
   if (this.isFull()) {
     throw new Error("Queue is full");
   }

   if (!(typeof element == null) && (!Number.isInteger(element))) {
     throw new Error('Element is not a integer'); // Throw an error if the element
is not a integer
   }

   if ((Math.sign(element) === -1) || (Math.sign(element) === 0)) {
     throw new Error('Element should be a positive integer'); // Throw an error if
the element is not positive integer
   }

   return this.queue.push(element); // Enqueue the element to the end of the queue
 }

 remove() {
   if (this.isEmpty()) {
     throw new Error('Queue is empty'); // Throw an error if the queue is empty
   }

   return this.queue.shift(); // Dequeue the element from the front of the queue
 }

 howMany() {
   return this.queue.length; // Get the number of elements in the queue
 }
```

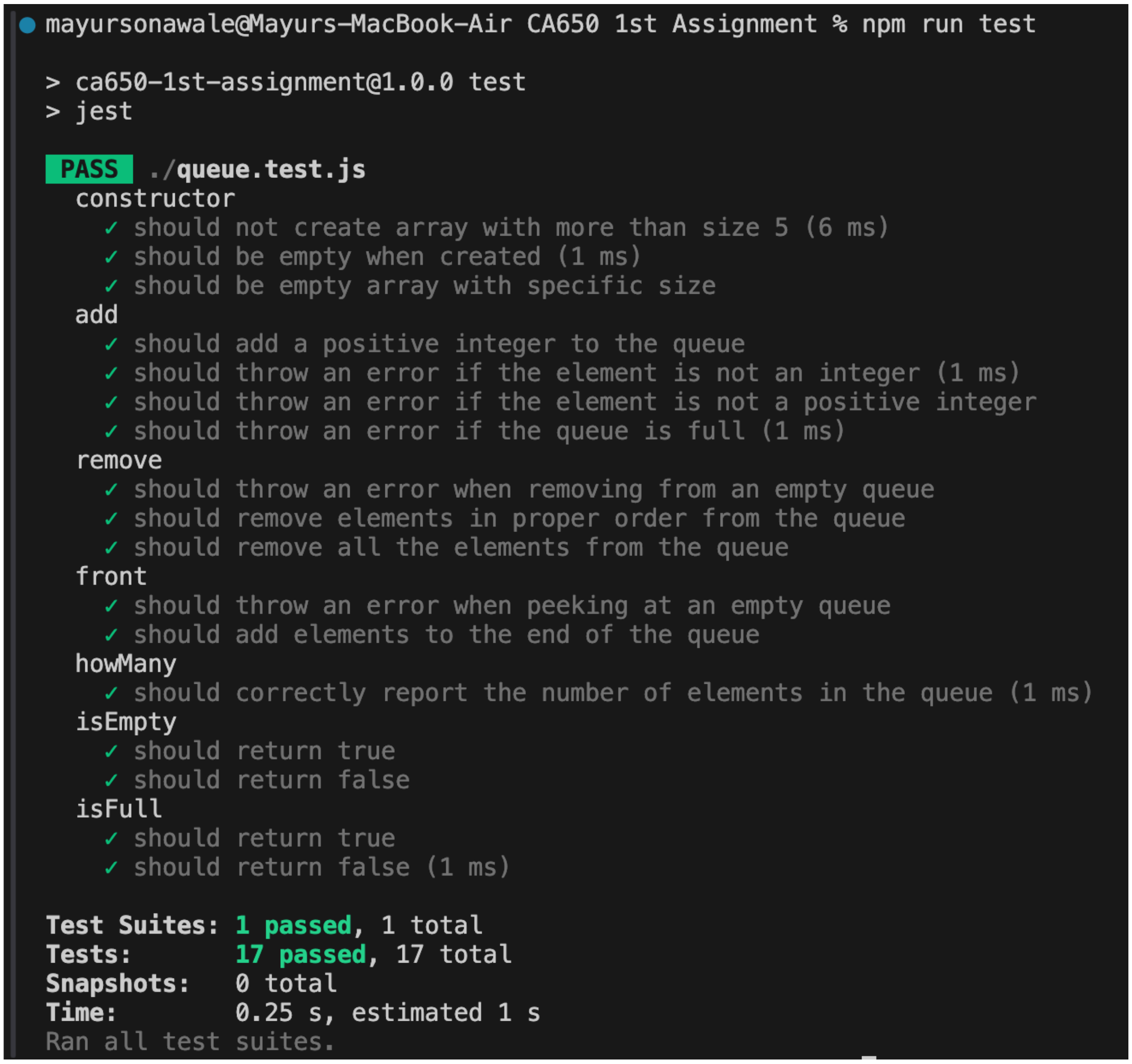
```
  isEmpty() {
    return this.howMany() === 0; // Check if the queue is empty
  }

  isFull() {
    return this.howMany() == this.size; // Check if the queue is full
  }

  front() {
    if (this.isEmpty()) {
      throw new Error('Queue is empty'); // Throw an error if the queue is empty
    }

    return this.queue[0]; // Return the front element without removing it
  }
};

module.exports = Queue;
```

XUnit code:

```
const Queue = require('./queue');

describe('constructor', () => {
  it('should not create array with more than size 5', () => {
    expect(() => new Queue(6)).toThrowError('Queue size should not be more than 5');
  });

  it('should be empty when created', () => {
    const queue = new Queue();
    expect(queue.isEmpty()).toBe(true);
  });

  it('should be empty array with specific size', () => {
    const queue = new Queue(2);
    expect(queue).toEqual({ queue: [], size: 2 });
  });
});

describe('add', () => {
  it('should add a positive integer to the queue', () => {
    const queue = new Queue();
    queue.add(5);
    expect(queue.queue).toEqual([5]);
  });

  // Try adding other data-types values
  it('should throw an error if the element is not an integer', () => {
    const queue = new Queue();
    expect(() => queue.add(null)).toThrowError('Element is not a integer');
    expect(() => queue.add('string')).toThrowError('Element is not a integer');
    expect(() => queue.add(true)).toThrowError('Element is not a integer');
    expect(() => queue.add(3.14)).toThrowError('Element is not a integer');
  });

  // Try adding non-integer values
  it('should throw an error if the element is not a positive integer', () => {
    const queue = new Queue();
    expect(() => queue.add(-1)).toThrowError('Element should be a positive integer');
    expect(() => queue.add(0)).toThrowError('Element should be a positive integer');
  });

  it('should throw an error if the queue is full', () => {
    const queue = new Queue(2);
    queue.add(1);
    queue.add(2);
    expect(() => queue.add(3)).toThrowError('Queue is full');
  });

});

describe('remove', () => {
  test('should throw an error when removing from an empty queue', () => {
    const queue = new Queue(1);
    queue.add(2);
    queue.remove();
    expect(() => queue.remove()).toThrowError('Queue is empty');
  });

  test('should remove elements in proper order from the queue', () => {
    const queue = new Queue(2);
    queue.add(1);
    queue.add(2);
    queue.remove();
    expect(queue.front()).toBe(2);
  });

  test('should remove all the elements from the queue', () => {
    const queue = new Queue(2);
    queue.add(1);
    queue.add(2);
    queue.remove();
    queue.remove();
    expect(queue.queue).toEqual([]);
  });
});

describe('front', () => {
  test('should throw an error when peeking at an empty queue', () => {
    const queue = new Queue();
    expect(() => queue.front()).toThrowError('Queue is empty');
  });

  test('should add elements to the end of the queue', () => {
    const queue = new Queue(2);
    queue.add(1);
    queue.add(2);
    queue.remove();
    queue.add(3);
    queue.remove();
    expect(queue.front()).toBe(3);
  });
```

```
});

describe('howMany', () => {
 test('should correctly report the number of elements in the queue', () => {
   const queue = new Queue(2);
   expect(queue.howMany()).toBe(0);
   queue.add(1);
   expect(queue.howMany()).toBe(1);
   queue.add(2);
   expect(queue.howMany()).toBe(2);
   queue.remove();
   expect(queue.howMany()).toBe(1);
 });
});

describe('isEmpty', () => {
 test('should return true', () => {
   const queue = new Queue(1);
   queue.add(1);
   expect(queue.isEmpty()).toBe(false);
 });

 test('should return false', () => {
   const queue = new Queue(1);
   queue.add(1);
   queue.remove();
   expect(queue.isEmpty()).toBe(true);
 });
});

describe('isFull', () => {
 test('should return true', () => {
   const queue = new Queue(1);
   queue.add(1);
   expect(queue.isFull()).toBe(true);
 });

 test('should return false', () => {
   const queue = new Queue(1);
   queue.add(1);
   queue.remove();
   expect(queue.isFull()).toBe(false);
 });
});
```

Xunit code run result:



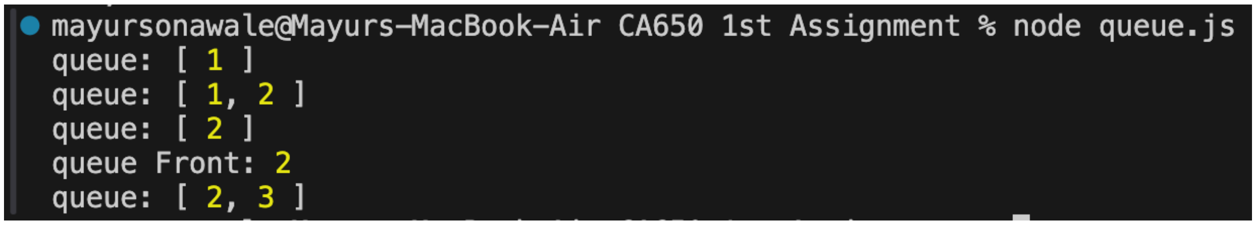
```
● mayursonawale@Mayurs-MacBook-Air CA650 1st Assignment % npm run test

  > ca650-1st-assignment@1.0.0 test
  > jest

   PASS  ./queue.test.js
    constructor
      ✓ should not create array with more than size 5 (6 ms)
      ✓ should be empty when created (1 ms)
      ✓ should be empty array with specific size
    add
      ✓ should add a positive integer to the queue
      ✓ should throw an error if the element is not an integer (1 ms)
      ✓ should throw an error if the element is not a positive integer
      ✓ should throw an error if the queue is full (1 ms)
    remove
      ✓ should throw an error when removing from an empty queue
      ✓ should remove elements in proper order from the queue
      ✓ should remove all the elements from the queue
    front
      ✓ should throw an error when peeking at an empty queue
      ✓ should add elements to the end of the queue
    howMany
      ✓ should correctly report the number of elements in the queue (1 ms)
    isEmpty
      ✓ should return true
      ✓ should return false
    isFull
      ✓ should return true
      ✓ should return false (1 ms)

  Test Suites: 1 passed, 1 total
  Tests:       17 passed, 17 total
  Snapshots:   0 total
  Time:        0.25 s, estimated 1 s
  Ran all test suites.
```

Sample runs:

1. 1st sample:

```
a. const queue = new Queue(3);
b. queue.add(1);
c. console.log("queue:", queue.queue);
d. queue.add(2);
e. console.log("queue:", queue.queue);
f. queue.remove();
g. console.log("queue:", queue.queue);
h. console.log("queue Front:", queue.front());;
i. queue.add(3);
j. console.log("queue:", queue.queue);
```

Output:

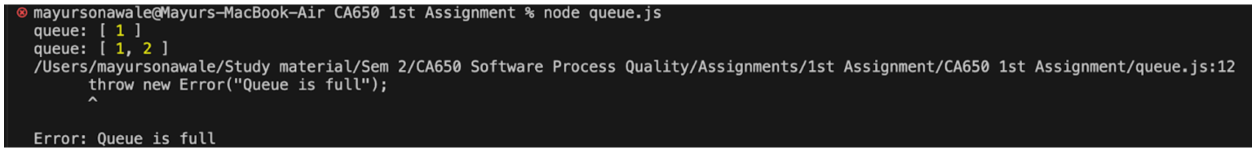


```
mayursonawale@Mayurs-MacBook-Air CA650 1st Assignment % node queue.js
queue:  [ 1 ]
queue:  [ 1, 2 ]
queue:  [ 2 ]
queue Front: 2
queue:  [ 2, 3 ]
```

2. 2nd sample:

```
a.    const queue = new Queue(2);
b.    queue.add(1);
c.    console.log("queue:", queue.queue);
d.    queue.add(2);
e.    console.log("queue:", queue.queue);
f.    queue.add(3);
```

Output:



```
mayursonawale@Mayurs-MacBook-Air CA650 1st Assignment % node queue.js
queue: [ 1 ]
queue: [ 1, 2 ]
/Users/mayursonawale/Study material/Sem 2/CA650 Software Process Quality/Assignments/1st Assignment/CA650 1st Assignment/queue.js:12
      throw new Error("Queue is full");
      ^

Error: Queue is full
```

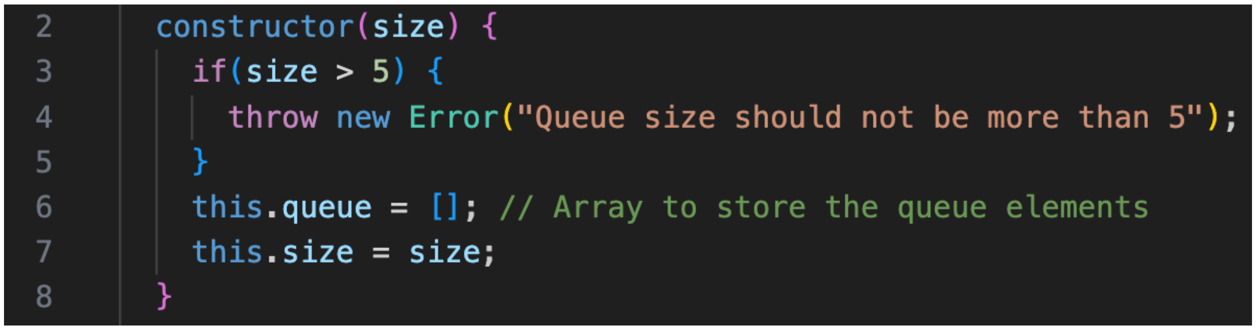
Reflection on the errors, faults, and failures discovered

1. **Failure: Infinite input**

In the first draft of the code, there was only a constructor that created an array without any specific size.

In this case, if an attacker finds out that a code is taking all the elements passed by the user without any threshold that could be the easiest way to crash the program by overflowing the stack by adding infinite values to the queue.

To solve this issue, I have taken the input from the user which should not be exceeding than a specific value(5). By doing this user can create a queue for only 5 elements and it would not cause to crash the program by overflowing the stack.



```
constructor(size) {
  if(size > 5) {
    throw new Error("Queue size should not be more than 5");
  }
  this.queue = []; // Array to store the queue elements
  this.size = size;
}
```

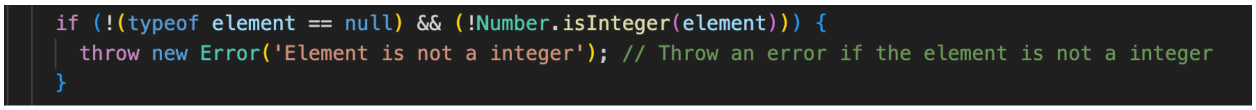
2. **Fault : Different Data types**

Another scenario I encounter while testing the add functionality.

Because queue is an array, so by in nature it would take any data type as a element such as String, Float, or Boolean and if think about worst case scenario it would also take a Null value.

So to avoid this case, I strictly check for the data type of the element push to the queue and allow it only if it is a integer.

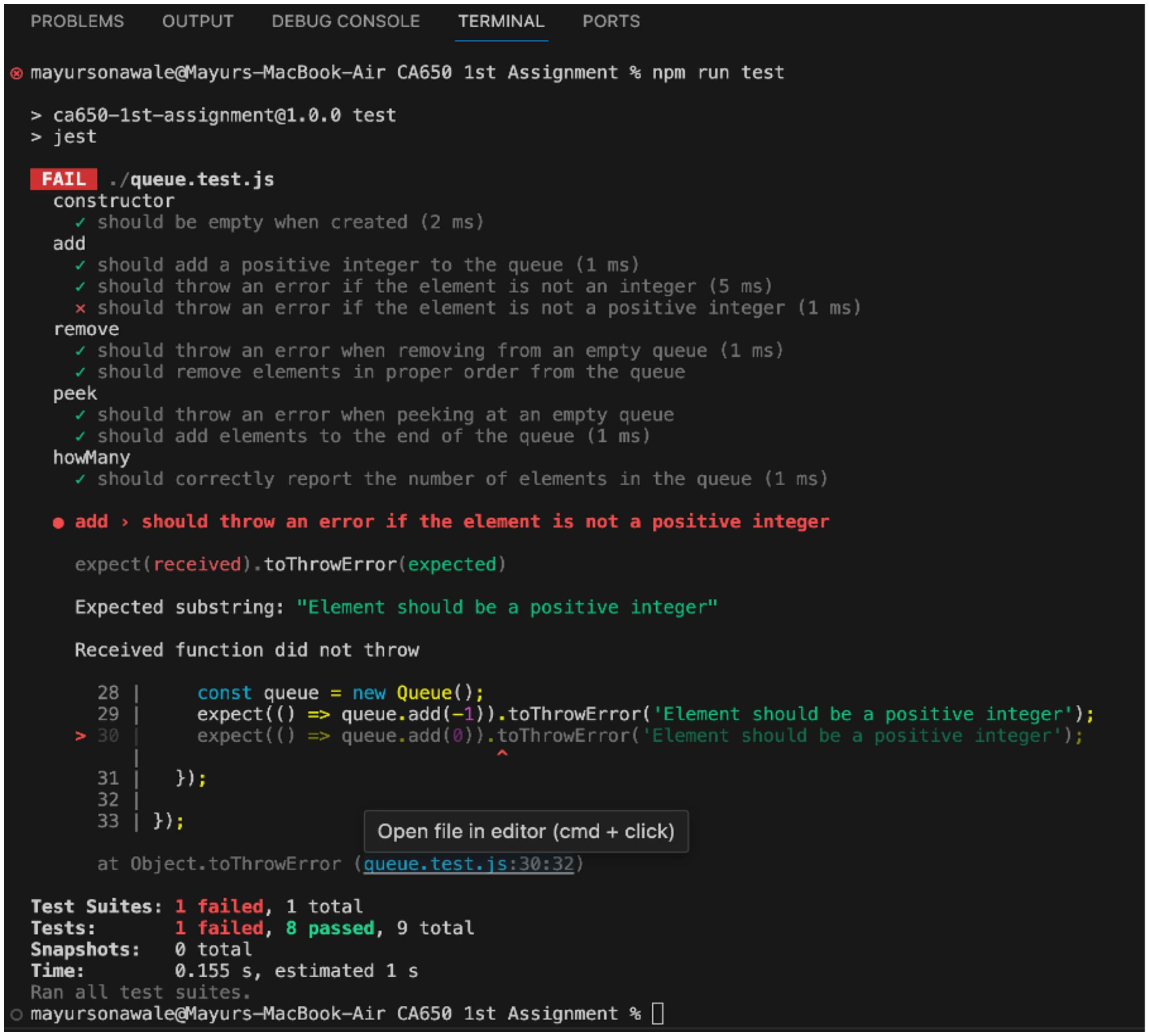
I resolve this issue by adding a condition as follows:



```
if (!(typeof element == null) && (!Number.isInteger(element))) {
  throw new Error('Element is not a integer'); // Throw an error if the element is not a integer
}
```

3. Error: Missing scenario

While testing the code it failed for one simple test case when trying to enqueue the -1 and 0.



```
⊗ mayursonawale@Mayurs-MacBook-Air CA650 1st Assignment % npm run test

> ca650-1st-assignment@1.0.0 test
> jest

 FAIL  ./queue.test.js
  constructor
    ✓ should be empty when created (2 ms)
  add
    ✓ should add a positive integer to the queue (1 ms)
    ✓ should throw an error if the element is not an integer (5 ms)
    × should throw an error if the element is not a positive integer (1 ms)
  remove
    ✓ should throw an error when removing from an empty queue (1 ms)
    ✓ should remove elements in proper order from the queue
  peek
    ✓ should throw an error when peeking at an empty queue
    ✓ should add elements to the end of the queue (1 ms)
  howMany
    ✓ should correctly report the number of elements in the queue (1 ms)

  ● add › should throw an error if the element is not a positive integer

    expect(received).toThrowError(expected)

    Expected substring: "Element should be a positive integer"

    Received function did not throw

      28 |     const queue = new Queue();
      29 |     expect(() => queue.add(-1)).toThrowError('Element should be a positive integer');
    > 30 |     expect(() => queue.add(0)).toThrowError('Element should be a positive integer');
         |                                ^
      31 |   });
      32 |
      33 | });

      at Object.toThrowError (queue.test.js:30:32)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 8 passed, 9 total
Snapshots:   0 total
Time:        0.155 s, estimated 1 s
Ran all test suites.
○ mayursonawale@Mayurs-MacBook-Air CA650 1st Assignment %
```

The basic idea was that the code should throw an error when supplying -1 and 0 to the program as they both are non-positive integers.

The test case was passed for -1 but failed for 0 as it was required to handle it explicitly in the code.

I resolve this issue by adding one condition as follows:

```
if ((Math.sign(element) === -1) || (Math.sign(element) === 0)) {
  throw new Error('Element should be a positive integer'); // Throw an error if the element is not positive integer
}
```