Lab 16.2 - Test a Callback-Based API

The labs-2 folder contains a `package.json` file and a `store.js` file.

The `package.json` file contains the following:

```
{
  "name": "labs-2",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "UNLICENSED"
}
```

The `store.js` file contains the following:

```
'use strict'
module.exports = (value, cb) => {
  if (Buffer.isBuffer(value) === false) {
    cb(Error('input must be a buffer'))
    return
  }
  setTimeout(() => {
    const id = Math.random().toString(36).split('.')[1].slice(0, 4)
    cb(null, { id })
```



```
  }, 300)
}
```

This API mimics some kind of async storage mechanism, such as to a database. In some circumstances it is infeasible to check for a specific value (for instance an ID returned from a database). For those cases, we can check for the presence of an ID, or apply some validation. In our case we can at least check that the length of the ID is 4.

Write tests for the `store.js` file. Ensure that when `npm test` is executed with the labs-2 folder as the current working directory the `store.js` file is fully tested.

Any additional dependencies, such as a test harness, may be additionally installed.

Also in the labs-2 folder is a `validate.js` file. The implementation can be checked with `node validate.js`. The implementation is successful if the final output of `node validate.js` is `passed!`

For completeness the following is the `validate.js` file contains the following, but it is not necessary to understand it for the purposes of this exercise:

```
'use strict'
const assert = require('assert').strict
const { spawnSync } = require('child_process')
const { writeFileSync } = require('fs')
const store = require.resolve('./store')
const storeCode =
Buffer.from('27757365207374726963742706a6d6f64756c652e6578706f727473203
d202876616c75652c20636229203d3e207b0a2020696620284275666665722e6973427
5666665722876616c756529203d3d3d2066616c736529207b0a2020202063622845727
26f722827696e707574206d75737420626520612062756666657227292 90a202020207
2657475726e0a20207d0a202073657454696d656f7574282829203d3e207b0a2020202
0636f6e7374206964203d204d6174682e72616e646f6d28292e746f537472696e67283
336292e73706c697428272e27295b315d2e736c69636528302c2034290a20202020636
2286e756c6c2c207b206964207d290a20207d2c20333030290a7d0a', 'hex')

try {
  {
    writeFileSync(store, storeCode)

    const sp = spawnSync('npm', ['test'], {
      stdio: 'ignore'
```



```
    })

    assert.equal(sp.status, 0, 'tests should be successful (is
package.json test script configured?)')
  }

  {
    const badOutput = `'use strict'
      module.exports = (value, cb) => {
        if (Buffer.isBuffer(value) === false) {
          cb(Error('input must be a buffer'))
          return
        }
        setTimeout(() => {
          const id = Math.random().toString(36).split('.')[1].slice(0,
2)
          cb(null, { id })
        }, 300)
      }

    `

    writeFileSync(store, badOutput)

    const sp = spawnSync('npm', ['test'], {
      stdio: 'ignore'
    })

    assert.equal(sp.status, 1, 'output should be tested (id length)')
  }

  {
    const badValidation = `'use strict'
    module.exports = (value, cb) => {
      if (Buffer.isBuffer(value) === true) {
        cb(Error('input must be a buffer'))
        return
      }
      setTimeout(() => {
        const id = Math.random().toString(36).split('.')[1].slice(0,
4)
```



```
        cb(null, { id })
      }, 300)
    }

    `

    writeFileSync(store, badValidation)

    const sp = spawnSync('npm', ['test'], {
      stdio: 'ignore'
    })

    assert.equal(sp.status, 1, 'error case should be tested')
  }

  {
    const unexpectedError = `'use strict'
    module.exports = (value, cb) => {
      cb(Error('input must be a buffer'), {id: '1234'})
    }
    `

    writeFileSync(store, unexpectedError)

    const sp = spawnSync('npm', ['test'], {
      stdio: 'ignore'
    })

    assert.equal(sp.status, 1, 'unexpected errors should be guarded -
e.g. ifError')
  }

  console.log('passed!')
} finally {
  writeFileSync(store, storeCode)
}
```

