Question 2: JSON Server Testing (25 pts)

You are working at the **Apple Store** as a test engineer and they have just started work on a new inventory management system. You will write some tests for the server they are developing. They require you to use the *Mocha* testing framework along with *Chai* assertions for the tests. **Note**: only test the functionality specifically requested or you will run out of time! Your tests will also be graded on coding style.

You are given the following file:

- `storeServerVB.js` contains the Apple Store server. This contains practice data for store inventory and has the interfaces to be tested. This **should** not to be modified. I will run your tests against this running server.

You will create a `test` subdirectory and within it a `storeTest.js` file containing the following tests. **Note**: the server will not necessarily pass all tests!

(a) Inventory API Testing (10 pts)

Test that the GET `/inventory` interface which returns the current store inventory, i.e., the current items available from the store.

- Returns an object
- The "inventory" object only contains items of the types: *grannySmith*, *fuji*, *gravenstein*, *honeyCrisp*.

(b) Stock API Testing (5 pts)

The POST `/stock` interface is used when items are **added** to the store inventory. This takes JSON items of the form:

```
{
  "grannySmith": 200,
  "fuji": 430
}
```

or

```
{
  "honeyCrisp": 1100
}
```

Note that not all items are necessarily stocked at the same time.

Create a test to check that the inventory increases after a POST to this interface. Note that you only have to check this for one of the items.

(c) Fulfill API Testing (5 pts)

The POST `/fulfill/` interface is used when fulfilling an order, i.e., taking it out of inventory. The format of the sent data is exactly the same as that use in the *stock* API but now the inventory will decrease.

Create a test to check that inventory can never go *negative* for an item even if we submit a request for an arbitrarily large number of items.

(d) Bad Input Testing (5 pts)

Create a test that checks if the *stock* API rejects negative values and issues an error/status code of 400.