Redux In a Nutshell Lab

It's unfortunate that we don't have time to really get into Redux because it is a fantastic library that complements React so well. But at least we got a high-level introduction and we are hoping that after the course you'll make time to dig into understanding Redux enough to actually write your own Redux code. To help with that, we are providing you with a prewritten implementation of Redux that we will use for subsequent labs.

Install Redux

1. Open a command window and cd to your "client" project. Install Redux.

```
npm install redux
```

2. Take a look in package.json. Make sure redux is listed as a dependency.
3. Look in node_modules. Make sure you can see the redux library.

Of course there is no way for you to learn Redux in just one lecture, so we won't ask you to write your own redux store, reducers, action creators, and middleware. Instead we've built them for you. Let's see what we can learn by reading through them.

Copy the prebuilt Redux files

It is a best practice to keep your Redux things together in a folder so let's group them.

4. Go to our course's github site (or the copy that you've already cloned/downloaded) and look in the starters folder. Notice that there's a folder in there called 'store'. This is a pre-built Redux store.
5. Copy all of starters/store to your project under src. Something like this might work:

```
cp myProject/starters/store myProject/client/src
```

   (Hint: don't actually execute that command. It is just trying to tell you to copy the entire store folder to your src folder under your project.)
6. Go ahead and look around in that folder. You'll find certain files. Let's examine them.

store.js

7. Crack open store.js in your IDE. Discuss these questions with your pair programming partner.
8. What is being imported from redux? _____
9. What are each of them for? What will they do? _____

Notice that we're importing from ./reducers.js and ./middleware.js. Let's take a look at them.

Looking at the reducers

10. Still in store.js, see if you and your partner can start with combinedReducer and follow through the reducer composition, going backwards through the file, tracing each sub-reducer's part back to its origin.
11. Take a look at the files in the reducers folder. Edit mainReducer.js. Notice the shape of the function being exported. What is being received and returned?
    It is _____
    Note that before we return a new state, we are altering the state.

Notice that we're importing from ./actions.js. Let's look at what's in there.

actions.js

This file holds an enumeration of sorts of all of the types of actions that can be taken. All it does is prevent hard-to-debug typos and enables your IDE to prompt you with intellisense.

12. Scan through this file with your partner. Discuss why it helps. What if you didn't have it? What might happen? Doing this allows us to write cleaner code. It isn't required to use Redux, but it is a best practice.
13. Notice also the actions that are being exported. These are action creators.
14. What signature do each of these functions have?
    _____
15. Can you and your partner see what they're doing?
    _____

Like that actions enumeration, action creators isn't required, but it is a great way to write cleaner code.

middleware.js

16. Open this file and read through. Notice first that it imports actions.js and action-types.js. We'll look through actions.js in a minute or two.
17. Take a look at one or two of the middleware functions that are being exported. Discuss with your partner how they work.
18. Notice the shape of every middleware function. They have the same required shape. What is that shape?
    _____

This is a very advanced subject, so don't fret if you don't understand it. It really takes days to get a deep enough understanding of Redux to be able to apply it. Maybe you can come back for a multiple-day intro at a later time.

19. Notice how fetchCityAndStateMiddleware is fetching some data from an API and in the promise callback, it is dispatching a couple of actions. Where are those actions coming from?
    _____
20. Let's look at that file.

They're simply returning an object that has a type and usually a bit of data which we'd call the payload. This is what is dispatched! These actions are eventually used by the reducer. to alter the state.