At this point in the project, we'll be implementing the useContext hook to act as our application's shopping cart.

Why is useContext a good choice here?

Context will make the shopping cart and its functions widely available to components without having to manually write the props.

Start by creating a CartContext.js file in your `src` folder.

The imports you'll need in this same file are the following.

```
import { createContext, useState } from "react";
import { productsArray, getProductData } from "./productStore";
```

Then create an exportable context that works as an *interface like* object literal that describes what functions are to be implemented and given access to via your provider.

In the context you create, you'll need a handful of properties. One should be called `items` set to an empty array. Then in the same context have one of each functions,

```
getProductQuantity, addOneToCart,
removeOneFromCart, deleteFromCart, getTotalCost
```

Then set up an `exportable provider component` that returns a CartContext component that passes each of the soon-to-be-implemented functions within a prop by the name of *value*.

Hint: *pass the functions as a group not individually. What data structure can you use to pass as a group?*

Once this is set up, implement each of the functions according to their name.

Here are a couple of hints, `items` should hold a value managed by the useState hook (an array works best).

Moreover, the structure of the data that we're storing in the state mentioned above is the following,

```
{ id: 1, quantity: 2 }
```

id is a representation of the product id, and *quantity* is how many products the user would like to purchase. This data structure is what your functions will be analyzing.

I recommend using high-order functions to help you work with the state.

Side note: Reach out to Fido if the implementation of the functions get's too difficult.

Once all functions are implemented make your `provider component` available for importing to the other files in your project by using the `export default` syntax.

Import the component into your App.js file and wrap it around your most external container.

This will give all the sub-components access to the context and functions you've implemented.

Now inside your ProductCard.js file, test the context you've created.

Your imports inside this mentioned file should look something like this,

```
import { CartContext } from '../CartContext';
import { useContext } from 'react';
```

Once your imports are set up, set the `onClick` attribute of your `<Button></Button>` component to execute the `addOneToCart()` function when the button is clicked.

The function should be accessible through the useContext hook, and the CartContex you imported.