Forms and Two-way Binding Lab

Remember when we enabled and disabled the "Mark as shipped" button based on the order lines' picked properties? We're now going make that work with the actual order line statuses.

1. Edit ship-order.component.ts. Create a method called isReadyToShip(order). Note: It should receive an order object.
2. It should do something like this:

```
return order.lines.every(line => line.picked);
```

   This returns true if every order line is picked.
3. Now edit ship-order.component.html. Find the button and do an Angular property binding. Bind the "disabled" property to the return value of the isReadyToShip(order) function. (Hint: use square brackets. Another hint: The "!" character makes a true false and a false true).
4. Run and test. Since none of the lines are picked yet, you'll see that the button is disabled.
5. Now edit ship-order.component.html. Find your "Got it" checkbox. Using banana-in-a-box, 2-way bind it to the order line's "picked" property.
6. Run and test. When you've marked every line as picked, the "Mark as shipped" button should enable. (HInt: if you get errors, remember that you have to import the FormsModule).

Receive Component

When the user enters a package tracking number we want to make sure it is valid. (Hint: everything in this section can be done 100% in the template. No need to work in the class.)

7. Edit receive-product.component.html. Notice that there's a pattern attribute on the tracking number input box.
8. Add an ngClass directive to the <section> where the user enters a tracking number. If the tracking number is a valid format, make the background of the section turn green by applying the bg-success and text-success classes. If it is invalid, apply the bg-danger and text-danger classes.
9. Run and test with these valid numbers

FedEx	122816215025810
UPS	1Z1234590291980793
USPS	9400 1057 3346 4567 2475 01
USPS	9205 5000 3865 8954 7543 00

10. In the Events Lab you make the button click call a saveTrackingNumber() function. Now, have that function console.log() the tracking number just entered.
11. Bonus!! If you have time, change your code such that the success and danger classes only show if the user has touched the field.

Recording the items received

Let's allow the user to receive items. We want to read in the quantity and productID and add it to the array of products being received when he/she hits the "Receive Product" button.

12. Add a private array called receivedProducts to the component class. This will be an array to keep track of all the products and quantities that are being received.
13. Notice that you have an input box for the productID being received. Bind that to a variable called productID.
14. There's another input box for the quantity. Bind that to a variable called quantity.

15. When the user hits the receive product button you're calling the receiveProduct() method. Modify that to pass the productID and quantity to that method. Modify the method to add the productID and quantity to the receivedProducts array.
16. In the starter HTML you were given, there's a hardcoded <table> of all the items being received. Convert that to an *ngFor, iterating receivedProducts and show all the productIDs and quantities in a table. (Note: later on we'll get an entire product but for now it should just be the productID.)
17. Run and test. When you click the button you should see your new product/quantity added to the table. Cool, right?

18. Bonus!! Note that after you add your product to the array you still have values in the productID and quantity textboxes. Can you make the values clear? Go ahead and do that.