Assignment #2

Server-Side User Input Validation

Goal

Use these starting files. Failure to do so will result in a mark of 0 for this assignment.

Complete the server-side portion of the web application for the WD1 Project 3 (Form Validation). This means you will set the action of your order form to a new PHP file which you will create.

A couple notes:

- When checking the "cart", just validate the quantity inputs
  - As it's just a cart simulation using Javascript, it won't completely work without it, so just test those inputs for content
- If you are using a solution with Javascript, that's ok! But the same stipulations would remain about checking the quantity inputs, you just won't have to manually add the years to that select element

Based on data passed to the server using the superglobal POST variable, process the data so that an invoice is produced similar to the results achieved in Project 2.

Your script will then generate an invoice showing the correct number of <tr> elements, which are dependent on the items the customer ordered.

If an item is NOT ordered, then it shouldn't be displayed on the invoice.

Data Validation

The data submitted should be validated according to the following rules using PHP's *filter_input* function.

- The email must be a valid email address.
- The postal code must be validated using a regular expression to be a valid Canadian postal code. (You can use the `FILTER_VALIDATE_REGEXP` filter with filter_input(); or, you can use this PHP function to do regex checks: http://php.net/manual/en/function.preg-match.php
  - A tool like this is good for regex testing: https://www.regexpal.com/

- The credit card number must be an integer.
- The credit card number must be exactly 10 digits.
- The credit card month must be an integer from 1 to 12.
- The credit card year must be an integer with a minimum value of the current year and a maximum value of five years from the current year.
- The credit card type must have at least one selection (the POST will show a value of "on")
- The full name, card name, address and city must not be blank.
- The province must be one of the two digit abbreviations from the form's select options.
- All quantities must be integers. (Unordered products can have a blank quantity.)

If any of your validations fail you should display an error message stating that the form could not be processed. You do not need to redirect the user back to the form.
Extra challenge (optional): Display a list of all of the failed validations.

NOTE: *Yes, these validations duplicate the javascript validations you already have in place. Remember, it's very easy to* *disable JavaScript in a browser. You always need to pair client-side validations with equivalent server-side ones. The easiest way to test your server-side validations will be to disable JavaScript in your browser. You may also remove the JavaScript from your order form.*

Other Requirements

- Your output should match the structure shown in the "Sample Output" section.
- Your invoice markup should be as similar as possible to what is generated if you point your form to: https://www.stungeye.com/school/webdev/thankyou.php
- The markup of your invoice should validate as valid HTML5.
- When a validation error occurs, your markup should still be valid.

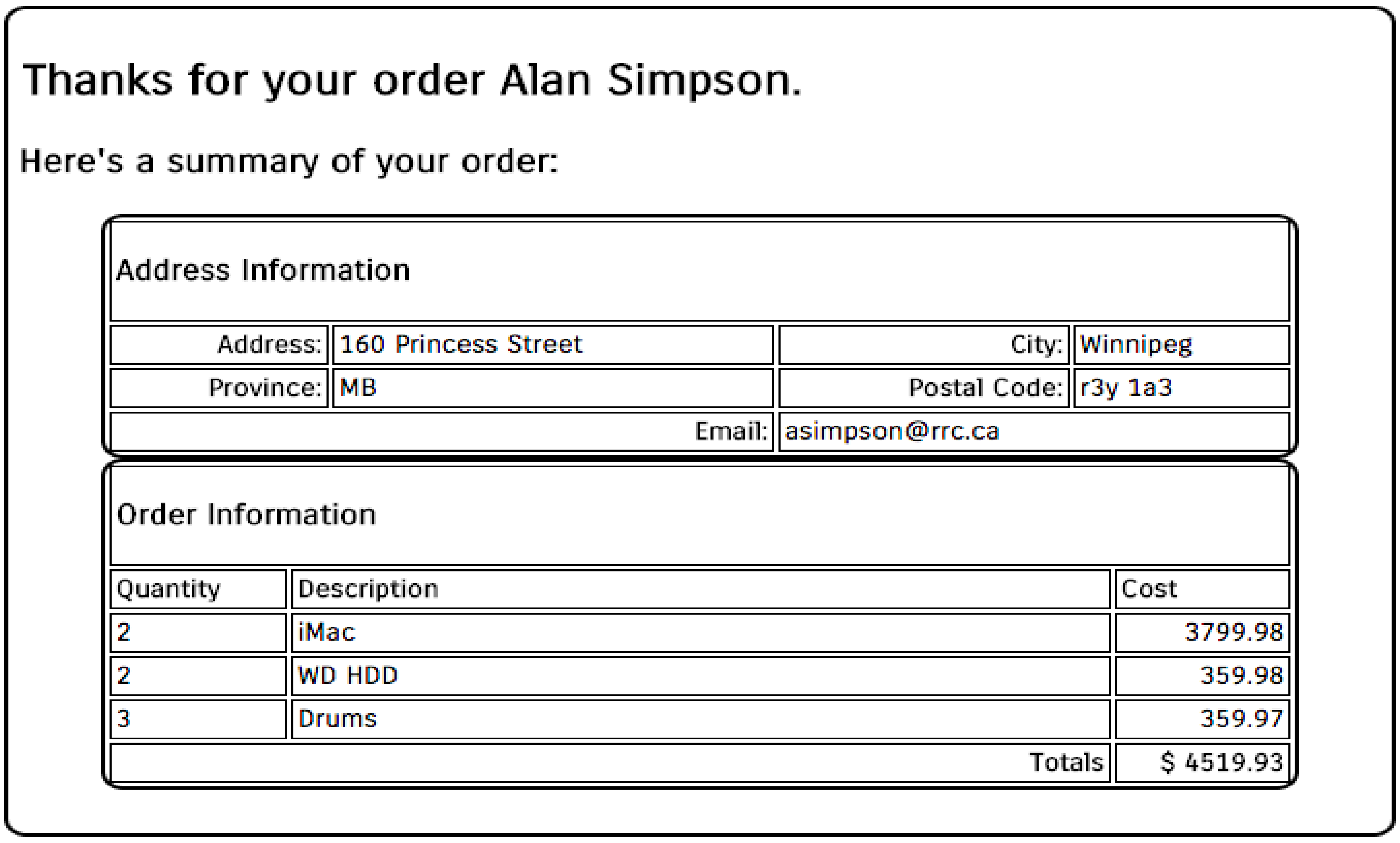
***Note:** Feel free to use the same CSS used by the script hosted on stungeye.com. You can also use the HTML output by the hosted script as a template for your output, although you may need to fix some of the markup to make the document valid HTML5.*

Easter Egg Bonus

The script hosted on stungeye used with your Web Dev 1 Project 2 contains an easter egg. As a bonus, replicate this easter egg or a similar one of your own design.

If you didn't find the Project 2 easter egg, ask around. Other students will tell you how to find it.

Sample Output



Thanks for your order Alan Simpson.

Here's a summary of your order:

Address Information			
Address:	160 Princess Street	City:	Winnipeg
Province:	MB	Postal Code:	r3y 1a3
		Email:	asimpson@rrc.ca

Order Information		
Quantity	Description	Cost
2	iMac	3799.98
2	WD HDD	359.98
3	Drums	359.97
	Totals	$ 4519.93

Rubric

Starting with a mark of **10**:

- Deduct **2** marks for each bullet point in the “Data Validation” section that is incomplete or incorrectly implemented.
- Deduct **2** marks if any of the address information is missing or incorrect when an invoice is generated.
- Deduct **2** marks if any of the order information is missing or if any of the calculated costs/totals are incorrect when an invoice is generated.
- Deduct **3** marks if the markup is not valid HTML5 when an invoice is generated.
- Deduct **3** marks if the markup is not valid when a validation error message is triggered.
- Deduct **5** marks if the PHP generates a notice, warning or error message at any point.
- Add **1** mark if the user implemented an easter egg in their invoice similar to the one that existed in Web Dev 1 Project 2.