Lab 07 – Introduction to Server-side Programming with PHP

Aims:

- Develop an understanding of the basic use of *variables and expressions* in PHP.
- To be able to use various control structures and develop your own functions.
- Use PHP predefined 'superglobal' variables to get data from a form.
- Gain some of the knowledge and skills needed to complete Assignments.

Task 1: Functions, GET, and while statements

In this task we will apply

- User-defined functions
- In-built functions
- **`if`** selection statements and loops

Step 1: Creating functions

Unzip the file **`lab07.zip`** into **`lab07`** folder.
Open the file **`mathfunctions.php`** in a text editor and read through the file.
The *user-defined* function **`factorial`** accepts a positive integer and returns its factorial value. A factorial of a non-negative integer n, denoted by n! is the product of all positive integers less than or equal to n. For example,

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

Add a function **`isPositiveInteger`** to **`mathfunctions.php`** as set out below. This user-defined function will use the *in-built* functions **`is_numeric()`** and **`floor()`** to check if a variable is an integer.

```
function isPositiveInteger ($n) {   // declare the function that takes a parameter
     $result = false;
       if (is_numeric($n))    //use the inbuilt function is_numeric which returns boolean
          if ($n == floor($n))
             if ($n >0 )
                $result = true;
     return $result;            // function execution will end here if -ve or non-integer
}
```

***Aside*:** Why aren't braces needed after the if condition **`if (is_numeric($n))`**?
While in this example braces are not used, it is good practice to always use them, as it can make the code easier to read and edit.

Step 2: Call the functions

Create an HTML/PHP file **`factorial.php`** that will *include* the file **`mathfunctions.php`** in order to access the defined functions in the file. The code below needs to be embedded in the HTML <body> of factorial.php.

You will need to write the rest of the HTML around the <body>

```
<body>
<?php
  include ("mathfunctions.php");    //makes functions in the included file available
?>
<h1>Creating Web Applications - Lab 8</h1>
<?php
     $num = 5;                //enter some different values here to test  - remove later
        if (isPositiveInteger($num)) { // call the function we defined in the previous step
           echo "<p>", $num, "! is ", factorial ($num), ".</p>";  //ditto for factorial
        } else {                // number is not an integer
           echo "<p>Please enter a positive integer.</p>";
        }
     echo "<p><a href='factorial.html'>Return to the Entry Page</a></p>";
?>
</body>
```

Note: At this stage, in the code above, the value of the variable **`$num = 5`** is *hard-coded* for testing purposes. We will replace this with code to accept input from a form in the next task.

Step 3: Test and validate

Use WinSCP or similar to copy the file to your **`lab07`** folder on Mercury.

Test it in the browser, and check that the delivered page is valid HTML5 (from the browser, view page source. Copy and paste the generated HTML code to https://validator.w3.org/).

Test the output. The answer should be "5! is 120".

Initialise **`$num`** to different values, and test that you get the correct output. Try:

- Different integers – positive and negative
- Non-integer numbers - e.g 3.4
- Non numbers

Task 2: Passing parameters to a PHP file from a form

In this task we will modify **`factorial.php`** to receive values from an HTML form.
This will involve two steps:

1. Writing a PHP program that accepts input.
2. Writing a client HTML form that sends input to the PHP program on the server.

Step 1: Modify the PHP to accept input

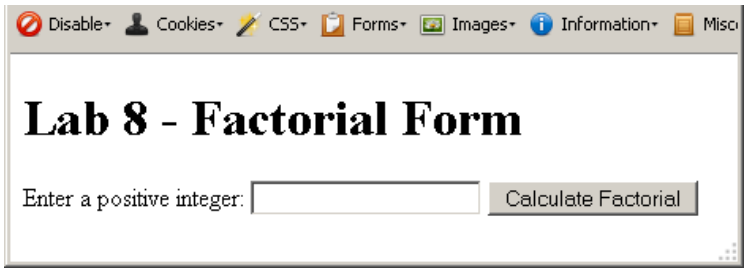
Re-name your file **`factorial.php`**. to **`factorial_with_get.php`**

Replace the section of the PHP that has the hard-coded input **`$num = 5;`** with the following code.
The script will now accept input submitted from a form, that has a form control with **`name="number"`**.

```
<?php
     if(isset($_GET["number"])) {        // check if form data exists
        $num = $_GET["number"];          // obtain the form data
        if(isPositiveInteger($num)) {    // call the function
            . . . other code
        }
     }
     echo "<p><a href='factorial.html'>Return to the Entry Page</a></p>";
?>
```

Step 2: Create the form

Create a file **`factorial.html`** that contains a form with a single text box (as shown below) that allows a user to enter a number, and submit it to **`factorial_with_get.php`** using **`method="get"`**.



Remember: forms send information to the server in `name-value` pairs so make sure the **`name`** in your form **`input type="text"`** matches the **`number`** parameter of the **`$_GET`** statement in **`factorial_with_get.php`**

Test in the browser, and check that the delivered web page is valid HTML5.

[IMPORTANT] Send your tutor the link to your web page running on the Mercury server to be marked off.