Lab 02: Input and Output Using PHP

In this lab, you will get yourself more familiar with PHP codes to receive input from the users and write output to the web browser.

1.1 Writing Output to Web Browser

Similar to CLI desktop application you learnt previously on C#.NET, you use a built-in command to push the text into the output interface (which in this case is the client-side web browser). Keyword echo is commonly used for this purpose.

In addition to being used to write standard string to the web browser, echo can also be combined with a variable. Refer to examples below.

Note that echo does not behave like a function, hence, may not appear in between other functions requesting a return value from echo.

```php
echo "Hello World";

echo "This spans
multiple lines. The newlines will be
output as well";

echo "This spans\nmultiple lines. The newlines will be\noutput as well.";

echo "Escaping characters is done \"Like this\".";

// You can use variables inside of an echo statement
$foo = "foobar";
$bar = "barbaz";

echo "foo is $foo"; // foo is foobar

// You can also use arrays
$baz = array("value" => "foo");

echo "this is {$baz['value']} !"; // this is foo !
```

```php
// Using single quotes will echo the variable name, not the value
echo 'foo is $foo'; // foo is $foo

// If you are not using any other characters, you can just echo variables
echo $foo;          // foobar
echo $foo,$bar;     // foobarbarbaz

// Strings can either be passed individually as multiple arguments or
// concatenated together and passed as a single argument
echo 'This ', 'string ', 'was ', 'made ', 'with multiple parameters.', chr(10);
echo 'This ' . 'string ' . 'was ' . 'made ' . 'with concatenation.' . "\n";

echo <<<END
This uses the "here document" syntax to output
multiple lines with $variable interpolation. Note
that the here document terminator must appear on a
line with just a semicolon. no extra whitespace!
END;
```

```php
// Because echo does not behave like a function, the following code is invalid.
($some_var) ? echo 'true' : echo 'false';

// However, the following examples will work:
($foo) ? print 'true' : print 'false'; // echo is also a construct, but
                                       // it behaves like a function, so
                                       // it may be used in this context.
echo $bar ? 'true': 'false'; // changing the statement around
```

```php
$count = 10;

if ($count == 1) { echo "Small quantity"; }
else { echo "Large quantity"; }
```

1.2 Receiving Input from the Users

Unlike in desktop application where user's interaction with the application is direct and immediate, web application does not work in the same fashion. It is detached, by-request and the users have no direct access to the server side.

Users submit data from web browser to the web application residing in the server by means of data transfer across OSI 7 layers.

There are 2 (two) common methods of such data submission, i.e. POST and GET. The data to be transmitted needs to be enveloped in a HTML Form tag at the client side interface.

Example is shown below using POST method. For a GET method, simply replace keyword POST with GET.

```html
<form action="personaldetails.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
```

Figure 1: Client-side HTML Form tag using POST method

```php
<?php

echo "Name: ", $_POST["name"];

$email_address = $_POST["email"];
echo "E-mail: ", $email_address;

?>
```

Figure 2: Server-side receiving transmitted data

TODO: Read further on the differences between POST and GET.

-- END --