COSC 360
Lab 7 Instructions – Introduction to PHP

In this lab, you will to explore the basics of PHP. It will provide an overview of the basic building blocks of the PHP language. The goal is to thoroughly study the code examples and experiment with them in order to understand the underlying concepts.

As you are now working with server-side scripting, a server will be required. **You will need to use a webserver to host your page for this lab.**

Information:

This assignment has multiple parts. Part I through IV are a series of exercises to familiarize you with the basics of PHP. Complete these activities and make notes as you work through the exercise as you will be responsible for knowing this material; **no submission is required for these parts**. Part V will involve converting an HTML file into a PHP file to help develop your PHP skills. **Submit your files from part V through GitHub classroom and the paste your repository link in Canvas**. As this lab involves setting up your web development stack you will have to manage your time in the event you need assistance from your TA. You will still only have one lab period to work on this activity as next week will have a new PHP lab so plan accordingly.

You can find further practice for PHP at https://www.w3schools.com/php/.

Instructions:

Part 0: Clone Your Repo

Using the invitation link for lab 7, clone the repository for Lab 7 to your local machine. Make sure that you place your completed files back in the repo folder when committing and pushing your final lab results.

Part I: Apache, PHP and MySQL for Local Development

In order to develop and build with PHP, you will need a webserver application (in this case Apache) as well as PHP interpreter. In a production environment, these resources would sit on your deployed server, making development and testing more challenging as you would need to continually upload changes to your remote server and then view the changes with a web browser. In addition to being an awkward process, it potentially exposes your developed to the public network. Using a toolset such as the LAMP (Linux, Apache, MySQL/MariaDB and PHP) web development stack allows these services to run on a local computer, allowing for a more streamlined development process. You will need to have this in place to complete future labs and the tool is available for Windows, OSX and Linux (**www.apachefriends.org**).

This will be a required tool for the remainder of the labs as well as for the development of your project. From the stack, you will be using the A and P components this week, being Apache (the web server which processes http requests and serves pages to browsers, in addition to interacting with the PHP interpreter) and PHP (the PHP language interpreter that will process a `.php` file and provide the results to Apache to be served). The details of the interaction between Apache and PHP can be found in section 11.1 pp. 506-507 in your textbook. Please take the time to review this entire section as you will be responsible for getting it running in order to accomplish your development.

The goal is to ensure that your stack is running correctly. If you have not already installed the tool, find and download the correct version for your OS and install it. Details can be found in section 11.1 and consult your TA for assistance. There are also numerous resources available online and generally is a straight forward installation.

Once the stack is running, it can be tested by opening your web browser and connection to the localhost. By typing into your URL bar in your browser `http://localhost/index.php`, this will instruct your browser to connect to the local web server and access the file `index.php`. Depending on the flavor of the stack (i.e. Windows, OSX, Linux) this may be slightly different and if you are having troubles ask your TA. You will know it is working when it will display the welcome page for XAMPP. If you are encountering issues, you may have conflicting ports or firewall issues; the text offers some quick solutions, but please ask your TA for assistance.

The install location of the Apache will vary depending on your OS and ensure that you are able to locate it. Inside the folder locate the `htdocs` folder. This is the folder where you will start to place your PHP files and the default location the webserver will look for named resources. Instead of using `.html` files, you will now use `.php` even if they contain html.

Create a new file called `lab7_test.php` and save it in the `htdocs` folder. This will be your first PHP file. Once the file is created, open it in your editor of choice and enter the following code:

```
<!DOCTYPE html>
<html>
	<head></head>
	<body>
		<?php
		 echo "<p>Hello, world</p>";
		?>
	</body>
</html>
```

and save the file. While this file may look similar to an HTML file, it contains new tags which are `<?php` and `?>`. These are the opening and closing tags used to indicate that what is contained within the tags is PHP code and needs to be interpreted and executed. Any other code located outside these tags is echoed directly to the client. Another important thing to note that each line in PHP is terminated with a semi-colon (`;`) and can be a common source of errors in your PHP code. Once this file has been saved, you will be able to view the file by entering

http://localhost/lab7_test.php into your browsers address bar. If everything is working, you should see the results as:

Congratulations! You've written your first PHP file. If you inspect the source files from your web browser, what you will find is strictly HTML. The PHP interpreter has interpreted your code and emitted the results to be served to your web browser. The `echo` command in PHP will echo what is ever located between the quotations, allowing you to create dynamic html. Alternately, you can use `print`. Section 11.2.4 expands on writing output. Please consult this section for further details.

To complete the rest of the lab, please review chapter 11, which will provide further background on PHP syntax.

Part II: PHP variables, Data types and Constants

Like other languages, PHP has variables that can be used to manipulate and store data. PHP is a dynamically typed language but PHP also has predefined data types that can be used to describe the type of data being stored in variable. Additionally, all PHP variables begin with a `$`. This is a must and your code will not work without it.

Constants can also be defined in your PHP file and generally will be located near the top of your file and need to be located between PHP tags. The syntax for defining a constant is

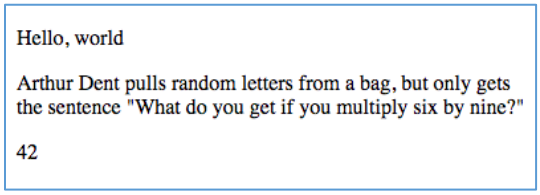
```
Define("<NAME_OF_CONSTANT>", "<the_value>")
```

Note that constants do not use $ preface. Update your file to contain some variables and constants as shown:

```
<!DOCTYPE html>
<html>
      <head></head>
      <body>

            <?php
                  define("THE_CONSTANT", "42");

                  $first_name = "Arthur";
                  $second_name = "Dent";

                   echo "<p>Hello, world</p>";
                   echo "<p>".$first_name." ".$second_name." pulls
                   random letters from a bag, but only gets the
                   sentence \"What do you get if you multiply six
                   by nine?\"<p>";
                   echo "<p>".THE_CONSTANT."</p>";
            ?>
      </body>
```

```
</html>
```

This code when interpreted should result in the following:



With this, you have been able to introduce variable data into your PHP code. Two things to notice; the escape character in PHP is `\` which is required if you want to print quotations. Secondly, in order to print variables, they must appear outside the quotations but be concatenated to the string. The concatenation operator in PHP is the period (`.`).

Part III: Program Control

As with many other languages, PHP has a number of conditional and iteration type constructs that can be used for flow control such as `for`, `if-else`, `while`, `do-while`, and `switch`. Section 11.3 expands on these concepts and will appear familiar to what you have seen previously in other languages. You can use these constructs to create context-specific content. Create a new file called `lab7_test2.php`, save it in the `htdocs` folder and enter the following code:

```
<!DOCTYPE html>
<html>
	<head></head>
	<body>
		<?php
			$logged_in = false;
			 echo "<p>Hello, world</p>";
			 if ($logged_in == true){
			   echo "<p>Welcome user!<p>";
			 }
			 else {
			   echo "<p>Logged out<p>";
			 }
		?>
	</body>
</html>
```

In this code, you can use the variable `$logged_in` (which is a Boolean and in PHP it can be `true` or `false`) to change the state of the page. Experiment by changing the value of `$logged_in` to `true` and refresh the page, making note of what happens. Take the time to explore what else can be done with the control structures.

Part IV: Functions

You can improve the organization of your code through the use of functions. PHP contains many built in functions but you can also create user-defined functions. In PHP, you can create reusable sections of name code as functions as well as pass data in through parameters and return results. Functions can exist on their own but must be within scope. Functions are defined by the keyword `function`. While a return type does not have to be specified, the `return` keyword is used to return a value from a function. Parameters can be used to pass data into a function and are listed in the `()` in the function definition. The body of the function is enclosed within `{}`. Section 11.4 expands on details for functions. Modify `lab7_test2.php` as follows:

```
<!DOCTYPE html>
<html>
	<head></head>
	<body>
		<?php
			$logged_in = true;
			$user = "Arthur";
			 echo "<p>Hello, world</p>";
			 echo login_message($logged_in);

			 function login_message($is_logged_in = false)
			 {
			   global $user;
			   if ($is_logged_in == true){
			     return "<p>Welcome ".$user."</p>";
			   }
			   else {
			     return "<p>Logged out<p>";
			   }
			 }
		?>
	</body>
</html>
```

In the example, the function has been used to return the appropriate login message based on the value of the variable passed into the function. One thing to note is that a function does not need to be defined before it is used in PHP as long as it is within scope. A function can be called directly by name.

As with Java and C, a variable defined within a function has function scope, meaning that they are not accessible outside the function. Variables declared in the main script body have `global` scope but unlike other languages, they are not accessible within functions. To access a global variable from within a function, it must be preceded by the keyword `global` as shown in the example code.

Part IV: Include Files

One important feature with PHP is that it has the ability to include or insert content from other files into another which helps to organize code. While included files appear to offer some level of encapsulation, this is not the case with PHP as including a file is essentially the same as copy-

pasting the included file directly into your code and will be expanded on further in class. To create and utilize an included file, create a new file called `login_msg.php` in your `htdocs` directory and place your `login_message` function in the file. The file should appear as:

```
<?php
function login_message($is_logged_in = false)
 {
   global $user;
   if ($is_logged_in == true){
     return "<p>Welcome ".$user."<p>";
   }
   else {
     return "<p>Logged out<p>";
   }
 }
?>
```

Update `lab7_test2.php` to utilize the include. To include a file, use the keyword `include` followed by the include file name. The file should appear as:

```
<!DOCTYPE html>
<html>
      <head></head>
      <body>
            <?php
                  include "login_msg.php";
                  $logged_in = true;
                  $user = 'Arthur';
                   echo "<p>Hello, world</p>";
                   echo login_message($logged_in);
            ?>
      </body>
</html>
```

You can see that this has significantly cleaned up and organized the code. Further details on includes can be found in section 11.3.6 of the textbook.

Part V: Building HTML with PHP

The goal is to convert specific elements of an HTML file to PHP. The results should look similar to Figure 1 (but not necessarily exactly the same error text). You are provided with the html file for this page as well as the corresponding style files. Only certain components of this page will be converted to PHP. As a side note, this file contains some styling from the bootstrap framework.

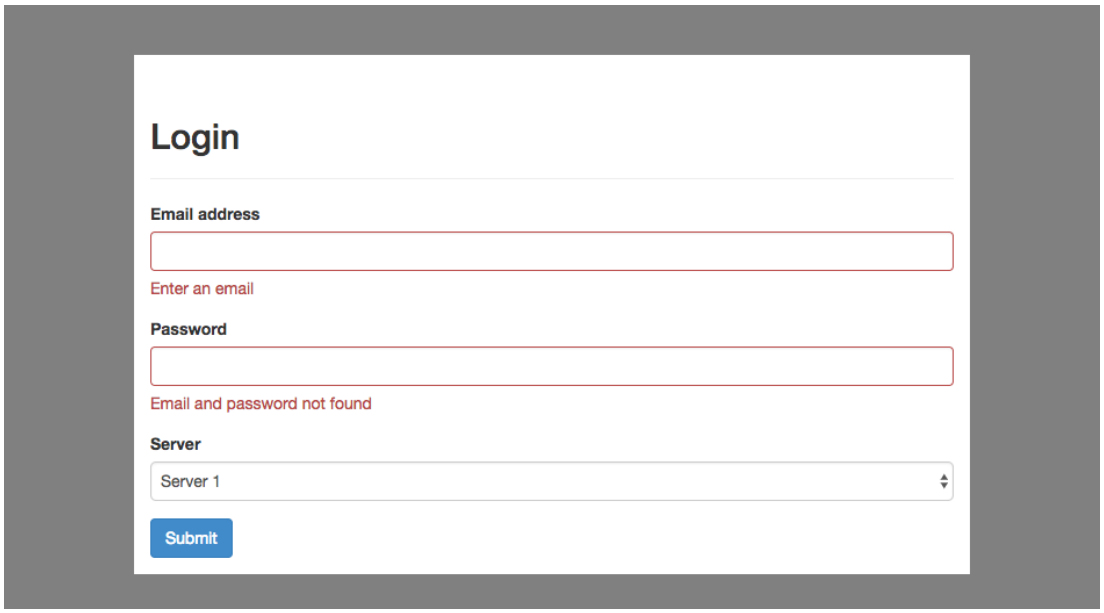


Figure 1- Lab7.html

1. Download the provided HTML5, PHP code and CSS. Create a folder called `lab7` in your `htdocs` directory. Unzip and place the provided files in your `lab7` directory. Ensure that you can view the page by going to `http://localhost/lab7/lab7.html`. Examine the `lab7.html` file to understand the structure of the code. Make a copy of this file and called it `lab7.php`. This is the file you will be working with.
2. Use the PHP `include()` function to include the file `lab7-data.php` in `lab7.php`. This file sets the values of two variables: `$email` and `$password`. Don't forget to put your PHP code within the PHP tags.
3. Using a `for` loop in PHP, replace the `<option>` elements as noted in the file.
4. Modify the code so that the values of the email input element and password input element are provided by the variables `$email` and `$password`.
5. Use an `if-else` to display an error message under the input fields for username (as shown in Figure 1) if the `$email` variable is left blank (by default, the values of the variables in `lab7-data.php` are empty). You can check to see if a variable is empty in PHP using the `empty()` function where you pass in the variable you want to check as a parameter and it will return `true` or `false`. Add the `has-error` CSS class to the appropriate `<div class="form-group">` element if `$email` is empty. This can be done by adding a PHP statement into the class attribute value that contextually changes from `'has-error'` when there is an error (i.e. empty email address) to `''` when there is a value. The selector is already provided for you in the css files, you just need to have the PHP file generate the correct value for the attribute.

6. Repeat step 5 for the `$password` variable.

Testing:

1. Test the form in your browser to make sure fields function correctly. Recall that you simply can't load a PHP file into your browser but it must be loaded by requesting the

page from your web server. The page can be loaded as http://localhost/lab7/lab7.php. Check the page with the default values for $password and $email (being empty).

2. Modify the values of `$password` and `$email` manually in `lab7.php` and verify that the logic of the page generation is correct.

Commit your PHP to your repository and push back to GitHub as well as submit your repository url via canvas.