Introduction to PHP

Overview

This lab walks you through using PHP to create simple applications. PHP is popular for many Web applications, so becoming comfortable with the syntax of PHP will help you diagnose and identify potential security issues.

Learning Outcomes:

At the completion of the lab you should be able to:

1. Execute PHP scripts within the AWS Cloud VM
2. Create simple PHP applications comprised of basic syntax, variables, strings, selection statements and repetition statements.

Lab Submission Requirements:
After completing this lab, you will submit a word (or PDF) document that meets all of the requirements in the description at the end of this document. In addition, your PHP file should be submitted. You should submit multiple files in a zip file.

Virtual Machine Account Information
Your Virtual Machine has been preconfigured with all of the software you will need for this class. You have previously connected to this machine in the previous labs. Reconnect again using the Remote Desktop connection, your Administrator username and password.

Part 1 – Execute PHP scripts at the shell prompt within the AWS Cloud VM

The Virtual Machine already has PHP installed. It is also configured to run properly on your Apache web server. This exercise will walk through creating a simple PHP script and running from a Web browser. We will use the notepad++ text editor to create the PHP file.

1. Assuming you have already launched and logged into your AWS Cloud VM, click on the notepad++ icon found on the left side of the screen of your VM as shown in figure 1.

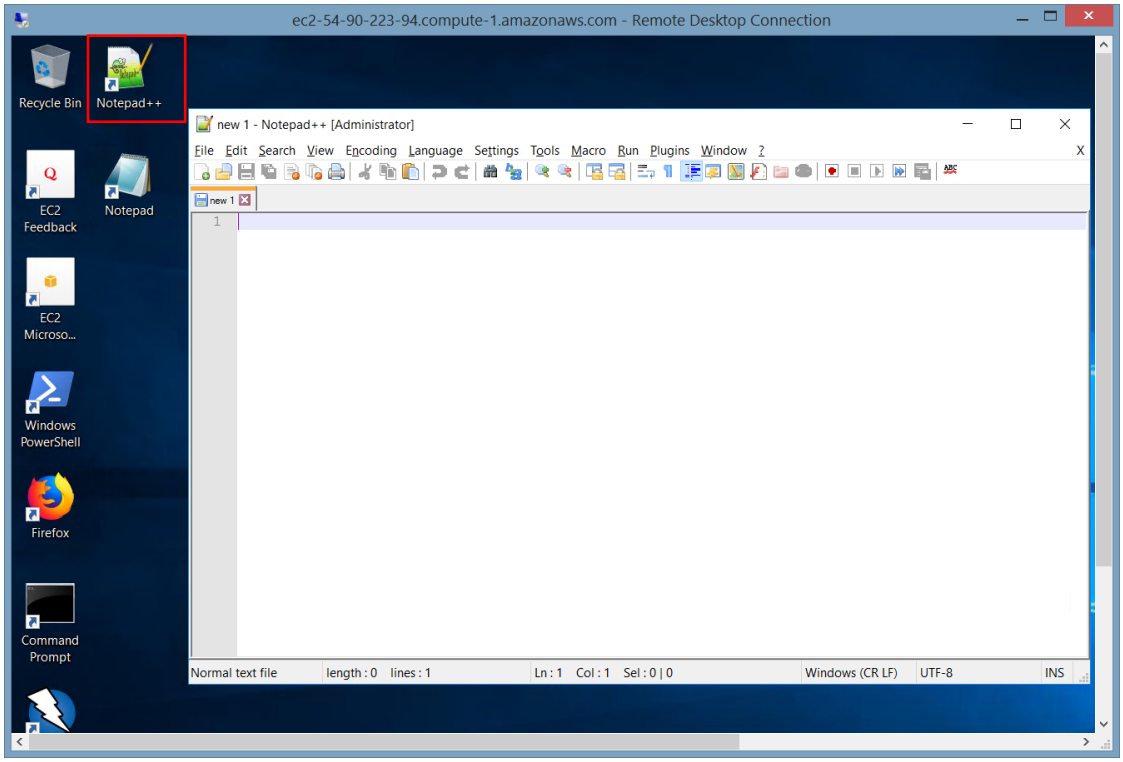

Figure 1 Open Notepad++ on you VM

2. To create a new document just begin typing or copying and pasting the PHP code shown below:

```
<!-- Simple Hello, World PHP Script
 Date: Jan 01, XXXX
 Author: Dr. Robertson
 Title: HelloSDEV300.php
 description: Print Hello greeting
 -->
 <!DOCTYPE html>
<!-- HelloPHP.html -->
<!-- Jan 22, XXXX -->
<html>
<head>
  <title>My First PHP Script </title>
</head>
<body>
<h1>Welcome to SDEV 300. </h1>
<h1>The following greeting is from PHP </h1>
<?php
    echo "Hello, SDEV 300 students and class!</br>";
    echo "The current time is " . date("g:i:h a"); ?>
<p>
</body>
</html>
```

Save the file in the C:\Bitnami\wampstack-7.1.16-0\apache2\htdocs\SDEV300 folder in a file named HelloSDEV300.php. Recall the C:\Bitnami\wampstack-7.1.16-0\apache2\htdocs\is the location of the Apache web server html files. (See figure 2).

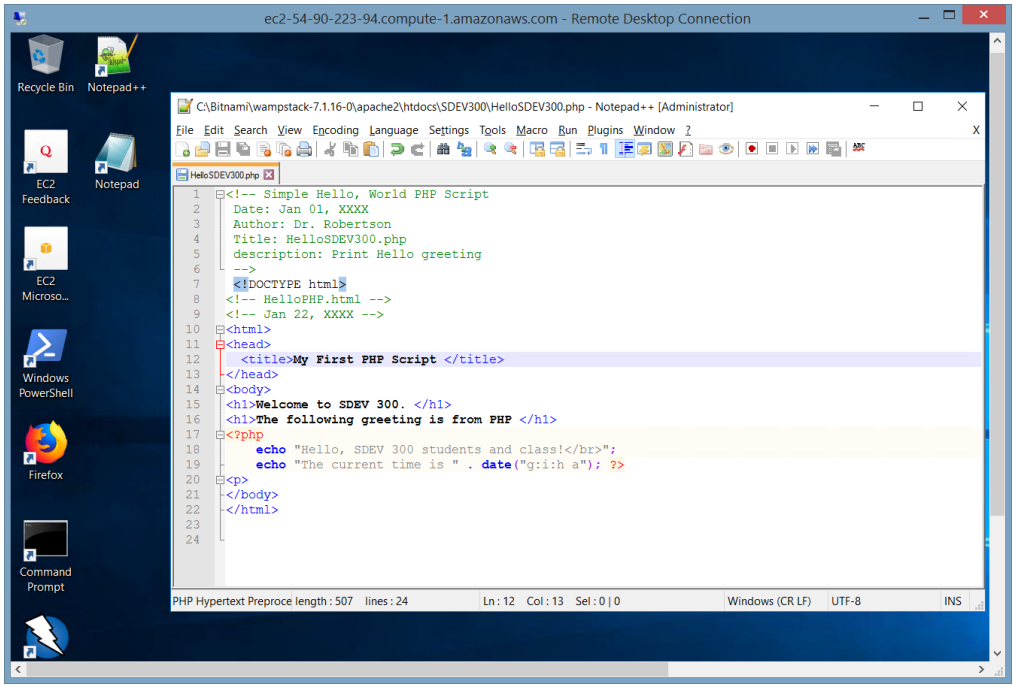

Figure 2 Saving the Hello, World PHP File

As shown in figure 3, launch the Firefox browser and run your home page by entering the following URL: localhost/SDEV300/HelloSDEV300.php.

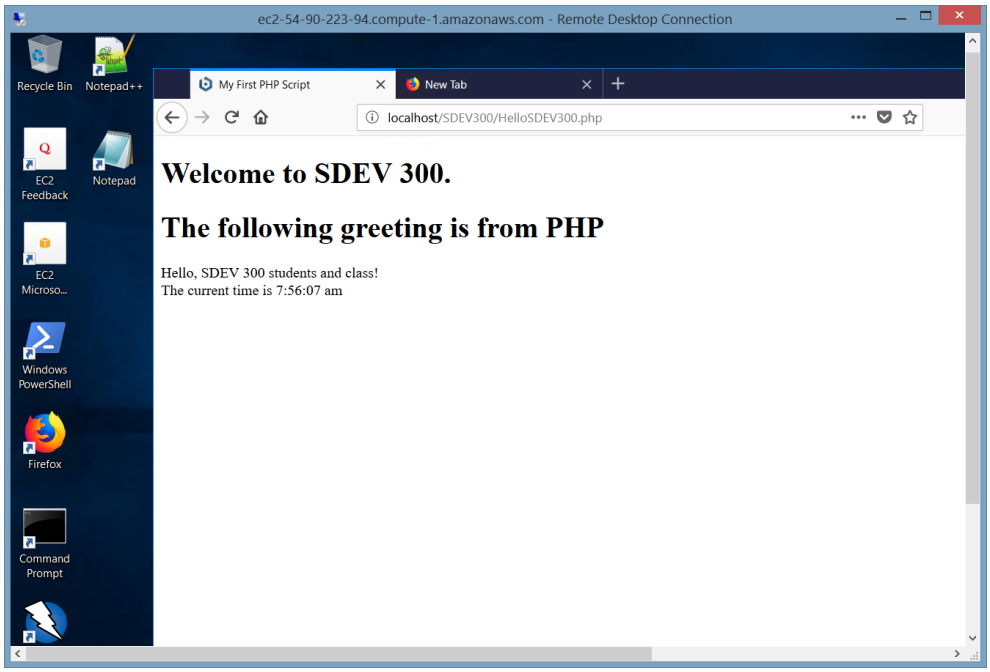

Figure 3 Launching the PHP Application

3. We can take the Web page we created in week 2 and add some additional PHP components. Copy and paste the following PHP file into your text editor and save the file as CShome.php in the SDEV300 folder.

```
<!DOCTYPE html>
<!-- CShome.php -->
<!-- Jan 22, XXXX -->
<html>
<head>
  <title>Computer Security Home Page </title>
</head>
<body>
<h1>Welcome to Computer Security Consultants! </h1>
<p>
<?php
    echo "Hello, SDEV 300 students and class!</br>";
    echo "The current time is " . date("g:i:h a");
?>
<!-- Add Table of Hyperlinks -->
<p>
Click on any link in the table below to see some of our current customers:
</p>
<table border = "1">
<tr><td>Site</td><td>Web Address</td></tr>
<tr><td>UMUC</td><td><a href="http://umuc.edu">UMUC</a></td></tr>
<tr><td>Oracle</td><td><a href="http://oracle.com">Oracle</a></td></tr>
<tr><td>Microsoft</td><td><a href="http://www.
microsoft.com">Microsoft</a></td></tr>
<tr><td>Twitter</td><td><a href="http://www.
twitter.com">Twitter</a></td></tr>
</table>

<!-- Add some images in a table -->
<p>
Check out our latest Mars photos:
</p>
<table>
<tr><td>Description</td><td>Photo</td></tr>
<tr><td>Mars Near Darwin</td><td><img
src="http://mars.jpl.nasa.gov/msl/images/mars-curiosity-rover-mount-sharp-
pia19083-Sol387-br2.jpg" width="300"
height="150"/></td></tr>
<tr><td>Mars Parhump Hills</td><td><img
src="http://mars.jpl.nasa.gov/msl/images/mars-curiosity-rover-pahrump-hills-
rock-outcrop-pia19075-br2.jpg" width="300"
height="150"/></td></tr>
</table>

<p>
We offer the following products:
<ul>
<li>Security Consulting </li>
<li>Apache security monitoring</li>
<li>Software Penetration Testing</li>
```

```
<li>Threat Modeling and Risk Managements </li>
</ul>
</p>

<!-- Add a Form -->
<p> Tell us about yourself and what you are interested in doing:
<form action="" method="post">
Name: <input type="text" name="username"></br>
E-Mail: <input type="text" name="e-mail"><br/>
Interest: <select name="sport">
<option>Apache Security Monitoring</option>
<option>Security Consulting</option>
<option>Software Penetration Testing</option>
<option>Threat Modeling and Risk Management</option>
</select>
<br/><br/>

<input type="submit" value="Click to Submit"/>
<input type="reset" value="Reset"/>
</form>
</p>

</body>
</html>
```

4. Launching the PHP file within Browser will result in the following output.

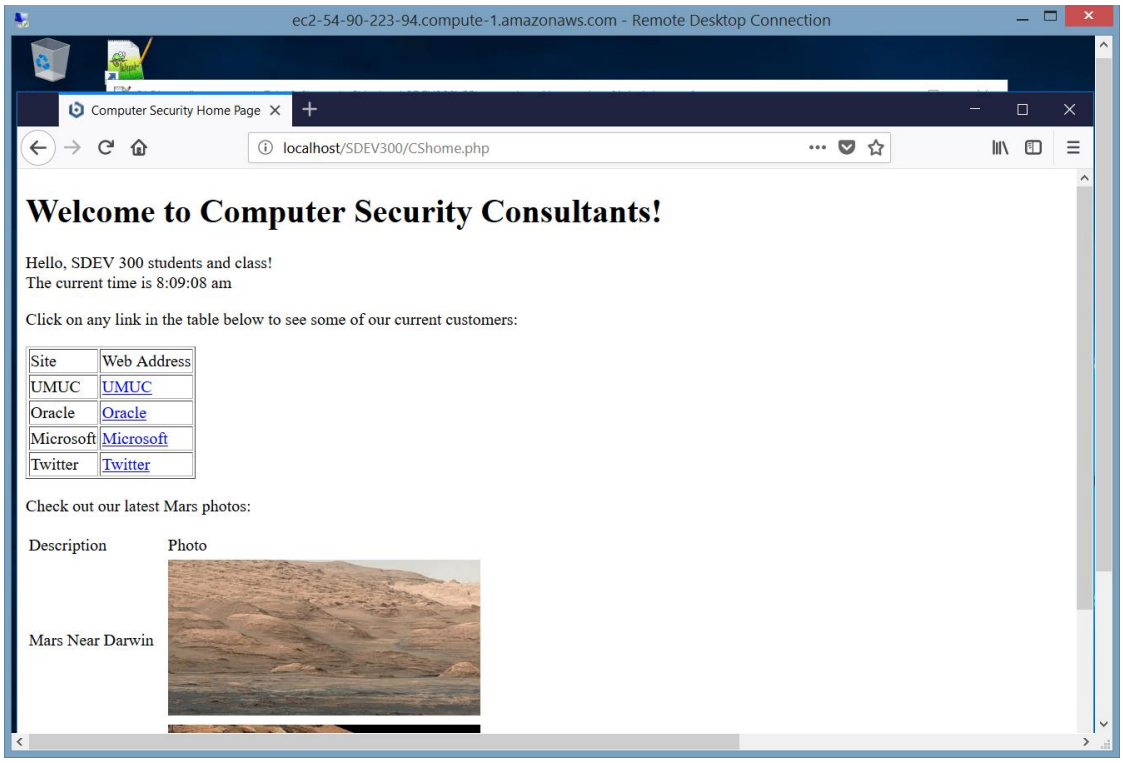

Figure 4 Combining PHP and HTML

Notice the Web page we created from previous week has the PHP welcome message at the top of the page. Combining PHP code and HTML to make dynamic web pages is just that easy. The key is to make sure you properly tag your PHP components. Notice, the use of the following PHP tags:

```
<?php

?>
```

The <?php starts the PHP code, while ?> ends the PHP code. Be sure all of your PHP specific commands are inside of those tags. Those tags tell PHP to start and stop interpreting the code between them. Everything outside of a pair of opening and closing tags is ignored by the PHP parser. Now, you can easily integrate those tags within existing HTML to add life to your web pages.

The next section of this lab, illustrates additional PHP functionality. Don't let the newness of PHP bother you. You have coded in Java and probably other languages already. PHP is just another programming language with similar functionality and slightly different syntax.

Part 2 Create simple PHP applications

The reading for this week covered the basic syntax of PHP, variable definitions, operators and expressions, decision and loops and strings. As you review these readings, be sure to copy and paste code to create files to test functionality. Also, be sure to modify, enhance and experiment with the code to better understand what each line of code is doing. Use the php.net manual to better understand additional options and functionality available in PHP. The examples provided are just a subset of what PHP can do. With experimentation and time, you will become quite comfortable understanding how to use and analyze PHP code.

Here are some critical PHP syntax components and examples to help expedite learning the language:

1. Use a semi-colon (;) to terminate PHP statements:

```
<?php
    echo 'Statements end with a semi-colon';
?>
```

2. Comments are important for documenting the code you create in any programming language. C, Java and C++ style comments as well as Unix-shell Perl comments are accepted. Consider the following examples:

```
// A one-line c++ style comment

/* A multi line comment

       That is too long for one line. */

# A one-line UNIX-shell-style comment
```

Consistency in programming style is important. I recommend using primarily the single or multi-line C/C++/Java commenting style.

3. Types are loosely defined in PHP but useful for providing some consistency for variables. The table below lists the more common types in PHP along with an example

Type	Description	Example
boolean	Expresses a truth value. (TRUE or FALSE.) values are case insensitive	`$isValid = True;`

integer	Expresses a whole number. Decimal, hexadecimal (0x), octal (0) or binary (0b) notation are available. May be optionally preceded by a sign (- or +).	`$count = 6;`
		`$octcount = 05;`
		`$bincount = 0b0101;`
		`$hexcount = 0xf3;`

float	Expresses a real number (float, double, real). May use e notation.	`$avg = 4.12e5;` `$std = 16.432;`
string	A series of characters. Does not offer Unicode support. May be single of double quoted.	`$lastname = 'Robertson';` `$firstname = "Jim";`

array	An ordered map aligning key pairs. Can be a simple list, trees, hash tables, collections and other data structures. Arrays of arrays are also possible.	`$majors = array("CMSC", "SDEV", "CMIT", "CMST");`
		`$capstones = array(`
		`"CMSC" => "Capstone A",`
		`"SDEV" => "Capstone B",`
		`);`

object	An object to store functions and variables.	`class car {`
		`var $speed=20.2;`
		
		`function get_speed() {`
		`return $this->speed;`
		`}`
		`}`
		`// Create and use the Car`
		`$mycar = new car;`
		`$myspeed = $mycar->get_speed();`
		`echo 'myspeed is ' . $myspeed;`

When experimenting with PHP types, be sure to include the code within the PHP start and end tags. If not, the PHP won't know what to process. For example, for the car object, the following makes the code useful within a Web file or stand-along PHP command script.

```
<?php
class car {
    var $speed=20.2;

  function get_speed() {
    return $this->speed;
  }
}

$mycar = new car;
$myspeed = $mycar->get_speed();
echo 'myspeed is ' . $myspeed;
?>
```

Arrays can be declared and initialized with data.
For a single dimensional array, the declaration and initialization is fairly straight forward:

```
$numbers = array( 11,43,4,5,7,10);
```

For a multi-dimensional associative array the syntax is trickier:

```
$gpa=array(
       array(
         "student"=>"Joe Smith",
          "grade" =>"A"
       ),
       array(
         "student"=>"Mary Jones",
          "grade" =>"A"
       ),
        array(
         "student"=>"John Perry",
          "grade" =>"C"
       ),
     );
```

Notice the use of a nested array statements and use of => to associate a value for array element.

4. Variables in PHP start with a dollar sign ($). Variables are case sensitive and must start with a letter or underscore, followed by any number of letters, numbers, or underscores.

Global variables are possible within PHP but are discouraged from being used. A variable defined within a function has scope to the function where it was defined. Variables defined outside functions are available to all other functions within PHP defined class or file but maybe extended to other included files as well.

5. Constants are identifiers whose values cannot be changed after being defined initially. Constants may be defined using the “define” reserved word or the const as shown below:

```
define("SCHOOLNAME", "UMUC");
define("AVAGADROS", 6.022e-23);
const HI = 'Hello World\n';
```

To use the constants just type in the defined constant name along with the print or display option:

```
echo SCHOOLNAME;
$mymole = 10.2*AVAGADROS;
echo $mymole;
print HI;
```

6. An operator takes one or more values and provides another value. Operators include arithmetic, assignment, relational, comparison, logical and others. The operator precedence is similar to most other modern programming languages with parenthesis taking highest priority, followed by increment and decrement, logical not, arithmetic operators and more. The details are found in the php.net manual available here:

http://php.net/manual/en/language.operators.precedence.php

In all cases, logic and functionality should always be fully tested to make sure complex nested operators are being interpreted as you believe.

Arithmetic operators include +, -, *, / and %.

Table 2 illustrates an example of each Arithmetic operator.

Operator	Example
Addition (+)	`$num3 = $num1 + $num2;`
Subtraction (-)	`$num3 = $num1 - $num2;`
Multiplication (*)	`$num3 = $num1 * $num2;`
Division (/)	`$num3 = $num1 / $num2;`
Modulo (%)	`$num3 = $num1 % $num2; (Returns the remainder of $num1/$num2)`
Exponent (**)	`$num3 = $num1 ** $num2;  (Raises $num1 to $num2 power)`

Assignment operators use the equals (=) sign. For example:

```
$num3 = $num1 + $num2;
```

Shortcuts are also available providing the combination of assignments and an operator. For example, the following code will takes the existing value of $num2 and adds $num1 to it.

```
$num2 += $num1;
```

This is equivalent to:

```
$num2 = $num2 + $num1;
```

Identical functionality is available for other arithmetic operators as well as most other binary operators.

Bitwise operators of &,|, ^, ~, <<, and >> are also available for AND, OR, Xor, Not, Left-shift and right-shift; respectively. Bitwise operators work at the bit level. Bitwise operators are handy for fast arithmetic computation.

Comparison operators allow you to compare two variables and determine their equality or lack thereof.

Table 3 shows the commonly used comparison operators in PHP. Note the use of !== and === that include the test to see if the two variables are of the same type as well as the same value.

Comparison	Example	Results
Equal	`$a==$b`	TRUE if $a is equal to $b after type juggling.
Identical	`$a===$b`	TRUE if $a is equal to $b, and they are of the same type.
Not Equal	`$a!=$b or $a<>$b`	TRUE if $a is not equal to $b after type juggling.
Not identical	`$a!==$b`	TRUE if $a is not equal to $b, or they are not of the same type.
Less than	`$a<$b`	TRUE if $a is strictly less than $b.
Greater than	`$a>$b`	TRUE if $a is strictly greater than $b.

Less than or Equal	`$a<=$b`	TRUE if $a is less than or equal to $b.
Greater than or Equal	`$a>=$b`	TRUE if $a is greater than or equal to $b.

Pre- and post-increments as well as pre- and post-decrements are available using the ++ and -- operators placed either before or after the variable name. These operators work identically in most programming languages. The following examples further clarifies the functionality.

`++$a` Pre-increment- Increments $a by one, then returns $a.

`$a++` Post-increment- Returns $a, then increments $a by one.

`--$a` Pre-decrement- Decrements $a by one, then returns $a.

`$a--` Post-decrement-Returns $a, then decrements $a by one.

Logical operators are used for comparison and include AND, OR, NOT, and XOR. Both symbolic and string versions are available as shown in table 3.

Table 3. Logical Operators

Logical	Example	Results
And	`$a and $b`	TRUE if both $a and $b are TRUE.
Or	`$a or $b`	TRUE if either $a or $b is TRUE.
XOR	`$a xor $b`	TRUE if either $a or $b is TRUE, but not both.
Not	`! $a`	`TRUE if $a is not TRUE.`
&&	`$a && $b`	TRUE if both $a and $b are TRUE.
||	`$a || $b`	TRUE if either $a or $b is TRUE.

Note the precedence of &&, || is higher than And, Or. In most cases it is recommended to use the symbolic representation of && and ||.

String operators include "." and ".=". Both are useful for concatenating two strings. For example the following PHP code results in concatenating "Welcome " and "to SDEV 300."

```
$str1 = "Welcome ";
$str2 = "to SDEV 300.";
$str3 = $str1 . $str2;
```

7. Control Structures such selection and repetitions statements are available within PHP.

There are multiple options to select when using control structures. For consistency in style it is recommended to use the structures you are most comfortable if they provide the desired functionality. For example, for loops are very functional and could be used instead of while loops. Table 4 shows

popular control structures along with an example. Experiment with these and other control structures available in the PHP manual.

Table 4. PHP control structures

Control	Example

if/else	`if ($a > $b) {`
	`  echo "a is greater than b";`
	`} else {`
	`  echo "a is NOT greater than b";`
	`}`

else if/else	`if ($a > $b) {`
	`  echo "a is bigger than b";`
	`} elseif ($a == $b) {`
	`  echo "a is equal to b";`
	`} else {`
	`  echo "a is smaller than b";`
	`}`

while	`$i = 1;`
	`while ($i <= 10) {`
	`  echo $i++;`
	`}`

do-while	`$i = 0;`
	`do {`
	`  echo $i;`
	`} while ($i > 0);`

for	`for ($i = 1; $i <= 10; $i++) {`
	`  echo $i;`
	`}`

foreach	`$a = array(1, 2, 3, 17);`
	
	`foreach ($a as $v) {`
	`  echo "Current value of \$a: $v.\n";`
	`}`

switch	`switch ($i) {`
	`  case 0:`
	`  echo "i equals 0";`
	`  break;`
	`  case 1:`
	`  echo "i equals 1";`
	`  break;`
	`  case 2:`
	`  echo "i equals 2";`
	`  break;`
	`}`

The functionality is not unlike what you have used in Java and C in previous courses. However; be sure to use `$` for the variable names as this syntax looks different than other languages although the functionality is the same.

8. Functions are critical to any programming language. PHP includes both user-defined and built-in functions. In both cases, the functions may use input parameters and return values.

Functions help organize your code and provide more portability and potential future code reuse in other projects or applications. When possible, use functions regardless of the programming language.

The following is a user defined function to calculate the area of a rectangle:

```
function calcRectArea($len, $width)
{
    echo "Calling Rectangle Area.\n";
    return $len*$width;
}
```

To call the function the following PHP code would work:

```
$myLen = 4.2;
$myWidth = 8.7;
$myArea = calcRectArea($myLen, $myWidth);
```

Function names are case insensitive, however for consistency, programmers should follow a programming style for naming functions. Similar to other programming languages, recursive functions are possible and used as appropriate.

Multiple built-in functions are available within the core PHP installation and additional useful functions are available as packages are installed. Although just a minor subset of String and Math built-in functions, table 5 provides a list of useful built-in functions:

Table 5. PHP built-in function examples

Function	Example call
strlen – get the string length	`$str = 'SDEV300';` `$myStr =  strlen($str);`
strrev — Reverse a string	`$str = 'SDEV300';` `strrev("Hello world!");`
strtoupper — Make a string uppercase	`$str = 'Welcome Students!';` `$str = strtoupper($str);`
trim — Strip whitespace (or other characters) from the beginning and end of a string	`$str = '   Welcome Students!  ';` `$trimmed = trim($text);`
str_shuffle — Randomly shuffles a string	`$str = 'SDEV300';` `$shuffled = str_shuffle($str);`
floor — Round fractions down	`echo floor(8.6); // Prints 8`
dechex — Decimal to hexadecimal	`echo dechex(16);`
sqrt — Square root	`echo sqrt(9);`

You should review and use the php.net manual to look-up additional useful math, string and other PHP functions as needed. For example, the following URLs take you to the String and Math function definitions:

http://php.net/manual/en/ref.math.php

http://php.net/manual/en/ref.strings.php

The following is another PHP function example that calls a user-defined function named cubeIt() and two built-in math functions.

```
<!-- PHP and Functions
 Date: Jan 01, XXXX
 Author: Dr. Robertson
 Title: FunctionsDemo.php
 description: Demo how to use Functions in PHP
 -->
 <!DOCTYPE html>
<html>
<head>
  <title>Functions Demo </title>
</head>
<body>
<h1>PHP Functions Demo </h1>
<?php
// Create a simple array of Degrees
$numbers = array( 15,30,45,75,90);

echo "<h3> Example PHP Functions </h3>";
// Create a table and display the numbers

echo "<table border='1'>";
echo "<tr>
        <th>Degree </th>
        <th> Sqrt(Degree) </th>
        <th> sin(Degree) </th>
        <th> cos(Degree) </th>
        <th> tan(Degree) </th>
         <th> cubeIt(Degree) </th>
      </tr>";
 foreach ( $numbers as $val ) {
    echo "<tr>";
    echo "<td>" . $val . "</td>";
    echo "<td>" . sqrt($val). "</td>";
    echo "<td>" . sin(deg2rad($val)). "</td>";
    echo "<td>" . cos(deg2rad($val)). "</td>";
    echo "<td>" . tan(deg2rad($val)). "</td>";
    echo "<td>" . cubeIt($val). "</td>";
    echo "</tr>";
  }
echo "</table>";

// Simple Cube function
// Return the cube of the input value
 function cubeIt($val) {
     return $val*$val*$val;
    }

?>
</body>
```

```
</html>
```

Reviewing the above code you should note the following:

a. Built-in PHP functions can be used easily by calling the function name and any required parameters. For this example, sqrt(), deg2rad(), sin(), cos() and tan() existing functions were called.
b. PHP functions you create should be of the format:

```
function functionName($parameter1, $parameter2 …) {
    // Code here
     return $returnvalue;
}
```

c. You can create functions with any level of rigor and complexity as needed to solve the computing problem at hand. The simple PHP function provided for this example calculates the cube of the input parameter:

```
function cubeIt($val) {
     return $val*$val*$val;
    }
```

In this exercise we will create a PHP web page that displays the multiplication table using a nested, for loop. HTML table tags will be used to format the data values.

1. Copy and paste the following code into a file named MultiplicationTable.php in the SDEV300 folder within the Apache2 location on your Virtual Machine.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" >
<html>
   <head>
      <title>Multiplication Table</title>
   </head>
   <body>
   <h1> Week 3 PHP and HTML Blending </h1>
   <h2>Multiplication Table</h2>

  <!-- First part of table -->
  <table border="1">
  <tr>
     <td>X</td>
     <td>1</td>
     <td>2</td>
     <td>3</td>
     <td>4</td>
     <td>5</td>
     <td>6</td>
     <td>7</td>
```

```
      <td>8</td>
      <td>9</td>
      <td>10</td>
    </tr>


<!-- Notice interweaving of PHP and HTML -->
     <?php
       $iterations = 10;
       // Nested for loop to calculate product

       for ( $num1=1; $num1 <= $iterations; $num1++ ){
         ?>
         <tr><td><?php echo $num1;?></td>
         <?php
         for ( $num2=1; $num2 <= $iterations; $num2++ ){
          $product = $num1 * $num2;
        ?>

          <td><?php echo $product;?> </td>
      <?php
       }
      ?>
       </tr>
      <?php
      }
      ?>

      </table>

    <p>
    <h3>A quote from Edgar Allan Poe</h3>
    <?php
    // Add a string for manipulation
     $poequote = "I have no faith in human perfectability. I think
   that human exertion will have no appreciable effect upon humanity.
   Man is now only more active - not more happy - nor more wise, than
   he was 6000 years ago.";
   echo $poequote;
   ?>
   <p>
    <h3>Quote modified with ucwords </h3>
    <?php
    // Make Uppercase for first letter
    $newquote = ucwords($poequote);
     echo $newquote;
    ?>

    </body>
  </html>
```

Figure shows the PHP file created in notepad++ on the AWS Cloud VM.

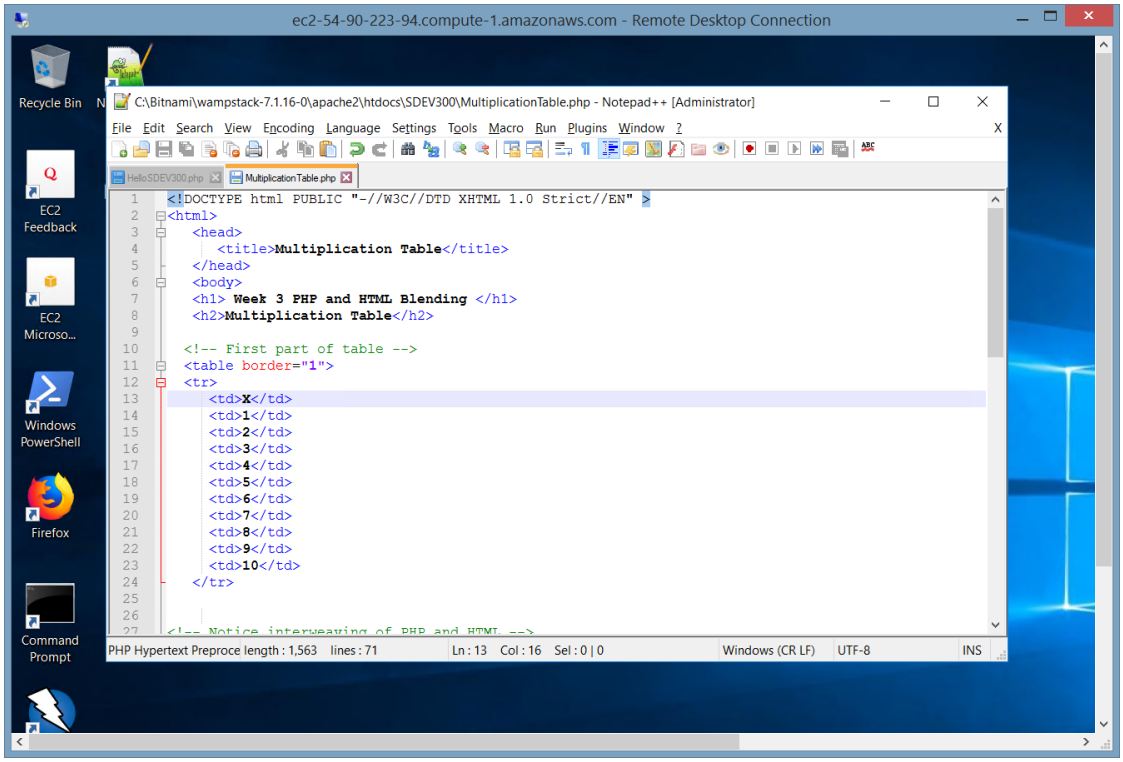

Figure 5 Multiplication Table PHP file.

2. Using your AWS Cloud VM, launch your Firefox browser and run the Web application using the localhost/SDEV300/MultiplicationTable.php

If successful, the resulting output will look similar to the screen shown in figure 6.

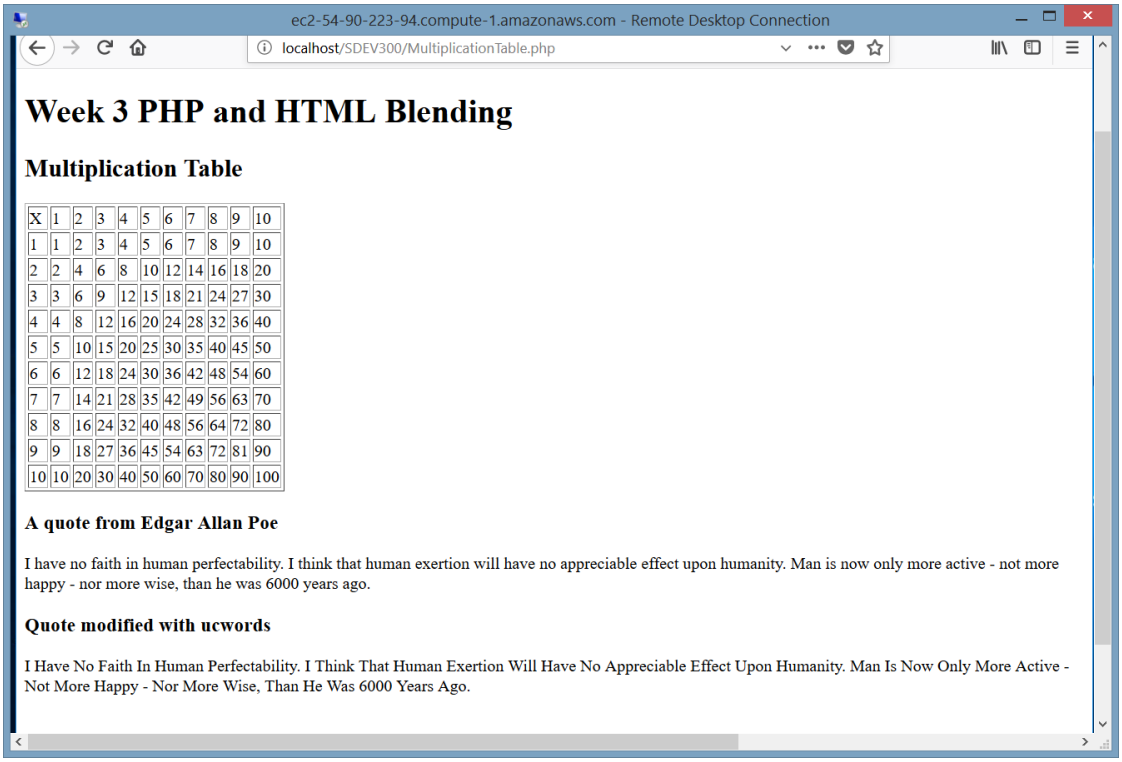

Figure 6 Running the MultiplicationTable.php file

3. Reviewing the code you should note the following:
   d. PHP codes start and stop with `<?php ?>`. This can get tricky to do by hand but for simple applications keeping track of the opening and closing braces is feasible.
   e. HTML code is interleaved between the PHP code. You will need to make sure you have a complete table structure between the PHP code. For example, the following HTML code is within a PHP loop to echo the data into just one cell. `<td><?php echo $product;?></td>`
   f. Adding ending braces for loops can be challenging. If you do this by hand, be sure to write your loops first, and then integrate the HTML tags. This will help you avoid infinite loops. The following is a typical listing for a table built using PHP loops. You should take your time walking through this code to see the complete structure and how PHP is used to provide for looping and dynamic programming.

```
<!-- First part of table -->
  <table border="1">
  <tr>
     <td>X</td>
     <td>1</td>
     <td>2</td>
     <td>3</td>
     <td>4</td>
     <td>5</td>
     <td>6</td>
     <td>7</td>
```

```
      <td>8</td>
      <td>9</td>
      <td>10</td>
    </tr>


<!-- Notice interweaving of PHP and HTML -->
    <?php
      $iterations = 10;
      // Nested for loop to calculate product

       for ( $num1=1; $num1 <= $iterations; $num1++ ){
     ?>
        <tr><td><?php echo $num1;?></td>
        <?php
           for ( $num2=1; $num2 <= $iterations; $num2++ ){
                $product = $num1 * $num2;
        ?>

         <td><?php echo $product;?> </td>
     <?php
      }
     ?>
      </tr>
     <?php
     }
     ?>
     </table>
```

Lab submission details:

As part of the submission for this Lab, you will create your own Web page that uses both HTML and PHP to create several different tables providing specific information. You will get a chance to use most of the concepts you studied so far in this course as you will apply both HTML and PHP code to this exercise. You may enhance your HTML display with CSS style sheets as desired but that is not required.

Specifically, you will create a PHP Web application that provides 2 different tables. You will use your design skills to determine the size and organization of the resulting tables.

The first table should include the results of using PHP to calculate several mathematical and trigonometric functions. Specifically, the following formulas should be implemented as f**unctions** in PHP:

- slope-intercept equation for a line: y = mx+b
- Surface Area of Sphere: $A = 4\pi R^2$
- Distance an object travels for given velocity and time : d = vt

The values used for each of the formulas are as follows:

Shape	Values and parameters
Slope intercept calculate y	x={2,5,8,10} for m=-2; b=0;
Surface Area of Sphere A	R = {2,6,10, 100,1000};
Distance object travels: d	v={10m/s, 30m/s, 327m/s, 1200 m/s} for time from 0 to 10 in steps of 0.5 seconds.

Note, you will have multiple results for each formula. Organize the results in an HTML table of dimensions of your choice. The results should clearly provide the formula used and the input and output results for each formula in the table.

Also, be sure to define the formulas as PHP function and call the functions.

The second table should include a famous quote (or quote that you like) and slightly modified versions of that quote using PHP String functions. You should use existing PHP functionality (e.g. built-in functions) to modify the quote. The modifications are described below:

Note, the quote should be at least 300 characters in length.

Modification	Description
Original	Original quote as is
Capitalize the first letter of each word.	For example: Hello, I Am A Robot With An AI-motivated Brain.
Displays the word length of each word in the quote separated by commas.	For example for "My name is Joe.", the output would be 2, 4, 2, 4 (note the period is counted as part of the last word.
Randomly shuffles each word in the quote.	For example for "My name is Joe", the output might be: yM maen si OJ.e

Create screen captures showing the successful running of your application. Be sure to label your screen captures and fully describe them. Each screen capture should clearly show the AWS Cloud VM was used to run the code.

Be sure to appropriately label the output cells for each table to indicate which formula or String function is being.

For your deliverables, you should submit a winzip file containing your word document (or PDF file) with screen captures of the application running successfully along with your PHP web application file.

You should include the Apache log file. **Submissions without access.log files will not be accepted.**

Include your full name, class number and section, date and the professor's name in the document.
Hints:

1. Make sure your math calculations, formulas and functions are correct.
2. Test everything before submitting.
3. Start with the PHP functions and then build the display output around HTML tables that call those functions.
4. Use PHP arrays to store your datasets.
5. Use for loops (or other repetition) to cycle through the datasets and the String data.
6. Use the built-in functions. (e.g. explode() is perfect for some of the String work you will need to do)
7. Start early on this project. It will take you longer than you think.
8. Ask questions if you get stuck

Grading Rubric:

Attribute	Meets
HTML and PHP	**75 points** Creates a PHP Web application that provides 2 different tables. (10 points) The first table clearly provides the mathematical or trigonometric formula used and the input and output results for each formula and each required set of input values. (20 points) The mathematical and trigonometric functions were implemented as PHP functions. (15 points) The second table include a famous quote (or quote that you like) and specific modified versions of that quote using PHP built-in String functions. (20 points) The quote is at least 300 characters in length. (5 points) Appropriately label the output cells for each table to indicate which formula or String function is being used (5 points) **Does not include access.log (-100)** **Does not use the SDEV AMI. (-100)**
Documentation and submission	**25 points** Submits a winzip file containing your word document with screen captures and access.log, html files, image files, PHP files and other required Web application files. (10 points) Includes labeled, screen captures of the AWS Cloud VM running the Web/PHP page. Screen captures are described with VM IP address clearly visible. (10 points) Title page includes your full name, class number and section, date and the professor's name. Document is neat, well-organized and free from spelling and grammar errors. (5 points)