University of Colombo School of Computing
SCS 1311 - Internet and Web Technologies

Lab Sheet - 08

1. Run PHP Code in WAMP Server

Step 1:

- Download **WAMP Server** from the official website: https://www.wampserver.com/en/
- Install WAMP by following the setup instructions.
- Choose your **default browser** when prompted.

Step 2:

- Start WAMP Server

Step 3: Locate the 'www' Folder

- Open File Explorer.
- Navigate to the WAMP installation directory
  - Default Path: C:\wamp64\www\ (for 64-bit) or C:\wamp\www\ (for 32-bit).
- Create a new folder (e.g., myproject) inside www.

Step 4: Write Your PHP Code

- Inside the myproject folder, create a new PHP file:
  - Example: index.php
- Open index.php in a text editor (Notepad++, VS Code, Sublime Text).
- Write the following PHP code

```
<?php
        echo "Hello, PHP is running on WAMP!";
?>
```

- Save the file as index.php

Step 5: Run the PHP Code in a Browser

- Open your browser (Chrome, Firefox, Edge).
- In the address bar, type
  http://localhost/myproject/index.php
- Press **Enter**, and you should see
  Hello, PHP is running on WAMP!

For the given activities, please follow the instructions below for submission:

1. Create a separate folder for each question.
   - Inside each folder, place the PHP source code file (.php) and screenshot(s) of the output.
2. Make sure to organize your files properly.
   - Each folder should be named according to the respective question (e.g., "Activity1.Q1", "Activity1.Q2", etc.).
   - The folder should contain both the code and the screenshots for that specific question.
3. Compress (zip) all the folders into a single ZIP file.
4. Submit the ZIP file containing all the folders and files.

Activity 1:

1. Execute the following code to see the output.

```
<!DOCTYPE html>
<html>
<body>
<h1> <?php echo "Welcome to My Site"; ?> </h1>
<p> <?php echo "This is a paragraph generated using PHP."; ?> </p>
</body>
</html>
```

2. Execute the following code to see the output.

```
<?php
// Defining a constant
define("SITE_NAME", "PHP Example");
// Global variable
$globalVar = "I am a global variable";
function exampleFunction() {
   // Local variable
   $localVar = "I am a local variable";
   // Using the global keyword
   global $globalVar;
   echo "Inside function:\n";
   echo $localVar . "\n"; // Accessing local variable
```

```
    echo $globalVar . "\n"; // Accessing global variable using global keyword

    // Static variable

    static $count = 0;

    $count++;

    echo "Static count: " . $count . "\n";

}

// Calling function multiple times to show static behavior

exampleFunction();

exampleFunction();


// Using concatenation

$name = "Upeksha";

$greeting = "Hello, " . $name . "! Welcome to " . SITE_NAME . ".";

echo $greeting . "\n";

?>
```

3. Create a PHP script that demonstrates the following concepts:
    a. Use echo to display output.
    b. Include single-line and multi-line comments explaining your code.
    c. Define a constant using the define() function and print its value.
    d. Declare a global variable and access it inside a function using the global keyword.
    e. Declare a local variable inside a function and print its value.
    f. Use the static keyword inside a function and show its behavior by calling the function multiple times.
    g. Perform string concatenation using the . operator to join multiple strings.

    Your script should include a function that:

    h. Accepts a name as a parameter.
    i. Uses a global variable inside the function.
    j. Uses a static variable to count how many times the function has been called.

k. Returns a greeting message using concatenation that includes the constant value.

Run the script and capture the output.

Submit the following:

l. The PHP script as a .php file.
m. A screenshot of the output.

Activity 2:

1. Execute the following code to see the output.

```
<?php

// Arithmetic Operators

$num1 = 10;

$num2 = 5;

echo "Arithmetic Operations:\n";

echo "Addition: " . ($num1 + $num2) . "\n";  // 15

echo "Subtraction: " . ($num1 - $num2) . "\n"; // 5

echo "Multiplication: " . ($num1 * $num2) . "\n"; // 50

echo "Division: " . ($num1 / $num2) . "\n"; // 2

echo "Modulus: " . ($num1 % $num2) . "\n"; // 0

echo "\n";

// Assignment Operators

$x = 10;

echo "Assignment Operations:\n";

$x += 5;  // Equivalent to $x = $x + 5

echo "x += 5: " . $x . "\n"; // 15

$x -= 3;  // Equivalent to $x = $x - 3
```

```
echo "x -= 3: " . $x . "\n"; // 12
$x *= 2;  // Equivalent to $x = $x * 2
echo "x *= 2: " . $x . "\n"; // 24
$x /= 4;  // Equivalent to $x = $x / 4
echo "x /= 4: " . $x . "\n"; // 6
$x %= 3;  // Equivalent to $x = $x % 3
echo "x %= 3: " . $x . "\n"; // 0
echo "\n";
// Comparison Operators
$a = 10;
$b = "10";
echo "Comparison Operations:\n";
echo "Equal (==): " . var_export($a == $b, true) . "\n";  // true
echo "Identical (===): " . var_export($a === $b, true) . "\n"; // false
echo "Not equal (!=): " . var_export($a != $b, true) . "\n"; // false
echo "Greater than (>): " . var_export($a > 5, true) . "\n"; // true
echo "Less than (<): " . var_export($a < 20, true) . "\n"; // true
echo "\n";
// Logical Operators
$val1 = true;
$val2 = false;
echo "Logical Operations:\n";
echo "AND (&&): " . var_export($val1 && $val2, true) . "\n"; // false
echo "OR (||): " . var_export($val1 || $val2, true) . "\n"; // true

echo "NOT (!): " . var_export(!$val1, true) . "\n"; // false

?>
```

2. Create a PHP script that

- Declares two numeric variables and performs Arithmetic operations (+, -, *, /, %).
- Uses Assignment operators (+=, -=, *=, /=, %=) to modify one of the variables.
- Compares two values using Comparison operators (==, ===, !=, >, <) and prints the results.
- Uses Logical operators (&&, ||, !) in if conditions to check multiple conditions.

Scenario:

- Assume a student's total marks and passing marks.
- Perform calculations and check whether the student has passed.
- Use logical conditions to print different messages.

Run the script and note the output.

Submit the PHP script as a .php file.

Include a screenshot of the output.

3. Execute the following code to see the output.

```
<?php

// Initial value

$num = 5;

echo "Initial Value: $num\n";

// Pre-increment (++$num) - Increments first, then returns the value

echo "Pre-increment: " . (++$num) . "\n"; // 6

// Post-increment ($num++) - Returns the value first, then increments
```

```
echo "Post-increment: " . ($num++) . "\n"; // 6 (but num becomes 7)
echo "After Post-increment: $num\n"; // 7
// Pre-decrement (--$num) - Decrements first, then returns the value
echo "Pre-decrement: " . (--$num) . "\n"; // 6

// Post-decrement ($num--) - Returns the value first, then decrements
echo "Post-decrement: " . ($num--) . "\n"; // 6 (but num becomes 5)
echo "After Post-decrement: $num\n"; // 5
?>
```

4. Execute the following code to see the output.

```
<?php
// Concatenation using . operator
$firstName = "Upeksha";
$lastName = "Sandamini";
$fullName = $firstName . " " . $lastName; // Joining two strings
echo "Full Name: " . $fullName . "\n"; // Output: Upeksha Sandamini
// Concatenation assignment using .= operator
$greeting = "Hello, ";
$greeting .= "how are you today?"; // Appends string to the original variable
echo $greeting . "\n"; // Output: Hello, how are you today?
?>
```

Activity 3:

1. Execute the following code to see the output.

```php
<?php
// Creating an indexed array
$fruits = array("Apple", "Banana", "Cherry");
// Accessing array elements
echo "First fruit: " . $fruits[0] . "\n"; // Output: Apple
// Adding a new element at the end
$fruits[] = "Mango"; // Appends Mango to the array
echo "After adding a fruit: " . $fruits[3] . "\n"; // Output: Mango
// Modifying an array element
$fruits[1] = "Blueberry"; // Changes Banana to Blueberry
echo "After modifying: " . $fruits[1] . "\n"; // Output: Blueberry
// Removing an array element (using unset)
unset($fruits[2]); // Removes the Cherry element
echo "After removal: " . implode(", ", $fruits) . "\n"; // Output: Apple, Blueberry, Mango
// Counting the number of elements in the array
echo "Number of fruits: " . count($fruits) . "\n"; // Output: 3
?>
```

2. Execute the following code to see the output.

```
<?php
// Using escape characters within a string
$sentence = "She said, \"Hello, how are you?\"";
echo $sentence . "\n"; // Output: She said, "Hello, how are you?"
$path = "C:\\Program Files\\PHP";
echo $path . "\n"; // Output: C:\Program Files\PHP
// Escaping single quotes
$quote = 'It\'s a beautiful day!';
echo $quote . "\n"; // Output: It's a beautiful day!
?>
```

Activity 4:

1. Execute the following code to see the output.

   ```
   <?php
   // Declaring a variable
   $age = 18;
   // Using an if-else statement
   if ($age >= 18) {
       echo "You are an adult.\n"; // This will be printed because age is 18
   } else {
       echo "You are a minor.\n";
   }
   ```

```
// Using elseif to check another condition
$time = 14; // 24-hour format (2 PM)
if ($time < 12) {
    echo "Good Morning!\n";
} elseif ($time < 18) {
    echo "Good Afternoon!\n"; // This will be printed because the time is 14
} else {
    echo "Good Evening!\n";
}
?>
```

2. Use conditional statements to check whether a person is eligible to vote based on their age.

Instructions:

Create a PHP script that

- Declares a variable $age representing the person's age.
- If the age is 18 or older, print "You are eligible to vote."
- If the age is less than 18, print "You are not eligible to vote."

Optional
Add another condition to check if the person is exactly 18 years old and print a special message: "Congratulations on turning 18!"

3. Use conditional statements to provide a greeting based on the time of day.

Instructions:

Create a PHP script that

- ○ Declares a variable $time representing the current hour in a 24-hour format (e.g., 7 for 7 AM, 14 for 2 PM).
- ○ Use if, elseif, and else statements to check:
    - ■ If the time is less than 12, print "Good Morning!"
    - ■ If the time is between 12 and 18, print "Good Afternoon!"
    - ■ Otherwise, print "Good Evening!"

Optional
Print a message for Good Night if the time is between 20 and 23.

4. Create a PHP script that

- Create an indexed array named $grades that contains the grades of 5 students. For example: [85, 90, 78, 92, 88].
- Calculate and print the average grade.
- Find and print the highest grade in the array.
- Find and print the lowest grade in the array.
- Use a foreach loop to print each student's grade along with their index number (e.g., "Student 1: 85").