COLLEGE OF COMPUTER STUDIES

(Applications Development and Emerging Technologies)

TECHNICAL-SUMMATIVE ASSESSMENT

1

PHP OUTPUT, VARIABLE FAMILIARIZATION, OPERATORS AND CONTROL STRUCTURE

Student Name / Group Name:	Bryan Quintos
Members (if Group):	**Name** / **Role**
Section:	TW24
Professor:	Sir. John Benedict Legaspi

I. PROGRAM OUTCOME/S (PO) ADDRESSED BY THE LABORATORY EXERCISE

- Design, implement and evaluate computer-based systems or applications to meet desired needs and requirements.

II. COURSE LEARNING OUTCOME/S (CLO) ADDRESSED BY THE LABORATORY EXERCISE

- Understand and apply best practices and standards in the development of website.

III. INTENDED LEARNING OUTCOME/S (ILO) OF THE LABORATORY EXERCISE

At the end of this exercise, students must be able to:

- Familiarize various Web Architecture, tools that used in PHP
- The basic understanding before using PHP
- Familiarize in environment of web developing
- Use of comments, variables and Echo / Print
- To understand the different types of operators that are available on PHP.
- To know what is operator precedence and operator associativity in PHP.
- To use escape sequence properly in the program.
- To know the different approach of control structures.
- To know the fundamentals syntax for conditional and looping structures.
- To properly use the compound expression using the logical operators.
- To know the rules of break, continue, and goto statements.

IV. BACKGROUND INFORMATION

Running PHP Scripts

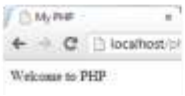
```
<!-- welcome.php -->
<html>
<head>
<title>My PHP</title>
</head>
<body>

<?php
    echo 'Welcome to PHP';
?>

</body>
<html>
```

All php scripts must enclosed with
<?php … ?> (standard tag)
<? … ?> (short open tag)
<script language="php">
….
</script> (script tag)
<% … %> (asp style tag)

OUTPUT:



An introduction to PHP web programming

Using ‘ ‘ (pair single quote), “ “ (pair double quotes) and . (dot character) PHP

Example:

```
<?php
    $name = "Juan";
    echo "$name's Store"."<br/>";
    echo '"$name\'s Store"'."<br/>";
    //statements below will produced errors
    //echo "displaying double quote" ";
    //echo "dispalying single quote ' ";
?>
```

Output:

PHP Operators and Control Structures

Arithmetic Operator

Example	Label	Outcome
$a + $b	Addition	Sum of $a and $b
$a - $b	Subtraction	Difference of $a and $b
$a * $b	Multiplication	Product of $a and $b
$a / $b	Division	Quotient of $a and $b
$a % $b	Modulus	Remainder of $a divided by $b

Beginning PHP and MySQL 3rd Edition by: W. Jason Gilmore pp.124

Assignment Operator

Example	Label	Outcome
$a = 5	Assignment	$a equals 5
$a += 5	Addition-assignment	$a equals $a plus 5
$a *= 5	Multiplication-assignment	$a equals $a multiplied by 5
$a /= 5	Division-assignment	$a equals $a divided by 5
$a .= 5	Concatenation-assignment	$a equals $a concatenated with 5

Beginning PHP and MySQL 3rd Edition by: W. Jason Gilmore pp.126

PHP Operators and Control Structures

PHP Control Structures: Conditional Statements

if statement

syntax:

```
if(expression) {
        statement...
}
```

else statement

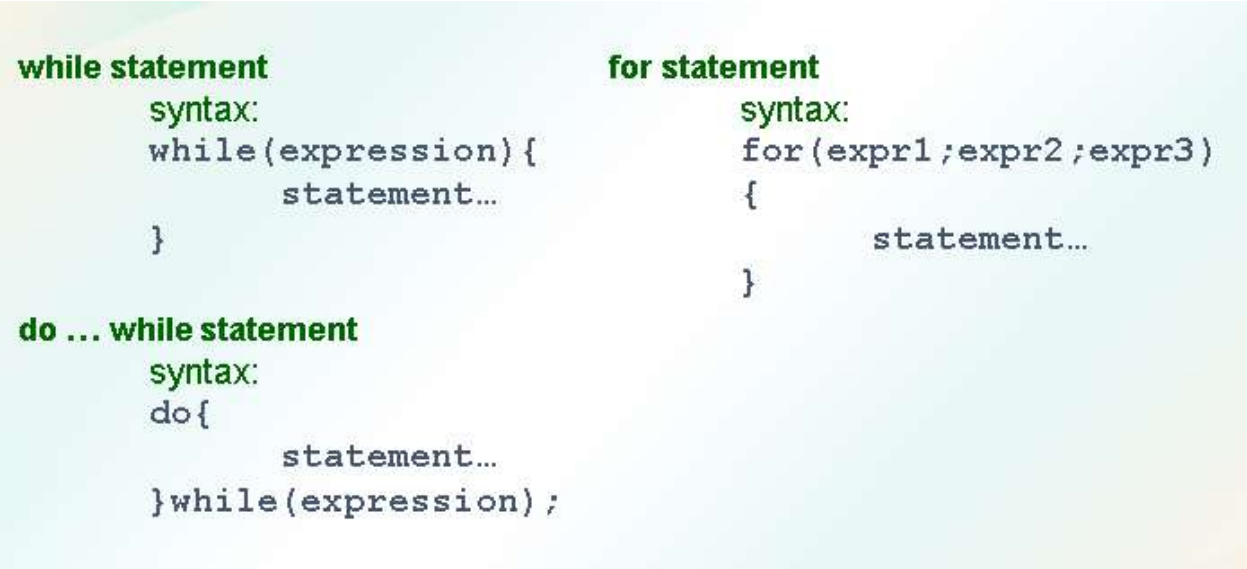
syntax:

```
if(expression) {
        statement...
} else {
        statement
}
```

elseif statement

syntax:

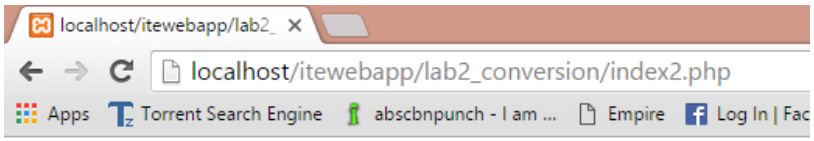
```
if(expression) {
        statement...
}
elseif(expression) {
        statement...
} else {
        statement...
}
```



V. GRADING SYSTEM / RUBRIC (please see separate sheet)

VI. LABORATORY ACTIVITY

1. Create a php program that will display the same output below. Use control structures to display the multiplication table with alternating color.



Multiplication Table

0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10
0	2	4	6	8	10	12	14	16	18	20
0	3	6	9	12	15	18	21	24	27	30
0	4	8	12	16	20	24	28	32	36	40
0	5	10	15	20	25	30	35	40	45	50
0	6	12	18	24	30	36	42	48	54	60
0	7	14	21	28	35	42	49	56	63	70
0	8	16	24	32	40	48	56	64	72	80
0	9	18	27	36	45	54	63	72	81	90
0	10	20	30	40	50	60	70	80	90	100

VII. QUESTION AND ANSWER

1. What is a variable?
   - A variable is a container like a storage box used to store date in a program. It can hold different types of data, such as strings, integers, floats, arrays or objects. In our PHP program, I use variables like $row, $col, and $value to store current row number, column number and the multiplication result. These variables did help generate the multiplication table dynamically.
2. What are the rules in creating a variable?

- In creating variables in PHP, we should make sure they begin with a letter or an underscore and not a number. They can only contain letters, numbers, and underscores. Also, variable names are case-sensitive, so $row and $Row would be treated as different variables.

3. Is it important to know HTML and CSS before using PHP? Explain
   - Yes, it is important. PHP is often used to generate HTML dynamically, as we did in our program. The multiplication table is created using PHP, but the structure (like <table>, <tr>, and <td>) is HTML, and the styling (like alternating colors and borders) is done with CSS. Without understanding HTML and CSS, it would be difficult to create visually appealing and functional web pages using PHP.
4. What is the difference between operator precedence and operator associativity?
   - Operator Precedence determines the order in which operations are performed. For example, in our program, if we wrote $row + $col * 2, the multiplication (*) would happen before the addition (+) because * has higher precedence. On the other hand, Operator Associativity determines the order of evaluation when operators have the same precedence. For example, in $a = $b = 5, the assignment operator (=) is right-associative, so $b = 5 is evaluated first, then $a = $b. Understanding these concepts ensures that our calculations, like $row * $col, are performed correctly.
5. What are the different control structures? Explain each
   - Control structures are used to control the flow of a program. The main types are:

     1. Conditional Statements:
   - *if*: Executes a block of code if a condition is true.
   - *if-else*: Executes one block of code if a condition is true, and another if it is false.
   - *if-elseif-else*: Allows multiple conditions to be checked.
   - *switch*: Executes one block of code based on the value of a variable.

     2. **Loops**:
   - *for*: Repeats a block of code a specific number of times.
   - *while*: Repeats a block of code as long as a condition is true.
   - *do-while*: Executes a block of code at least once, then repeats as long as a condition is true.
   - *foreach*: Iterates over elements in an array.

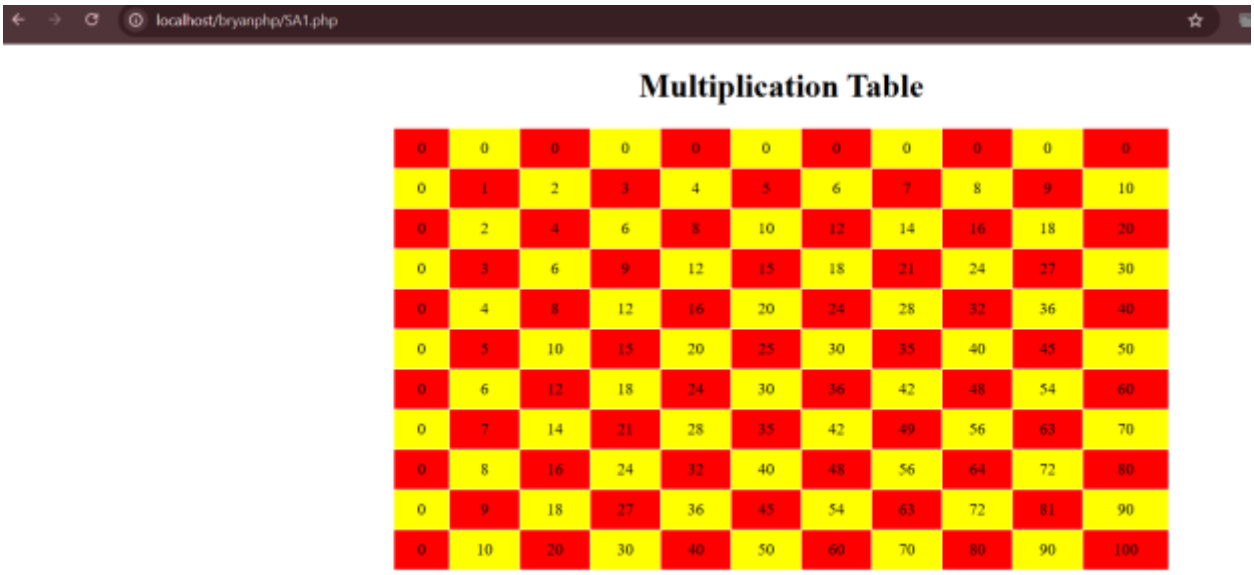
     3. Jump Statements:
   - *break*: Exits a loop or a switch statement.
   - *continue*: Skips the current iteration of a loop and moves to the next iteration.
6. Explain the rules of break, continue, and goto statements.
   - *break*: Exits a loop or switch statement. For example, if we wanted to stop generating rows after a certain condition, we could use break.
   - *continue*: Skips the current iteration of a loop and moves to the next one. For example, if we wanted to skip generating cells for certain rows or columns, we could use continue.

- *goto*: Jumps to a labeled part of the code. This is rarely used because it can make the code harder to read.
- In my program, I didn't need break, continue, or goto because the for loops and conditional logic were sufficient to generate the table

VIII. REFERENCES

1. https://www.w3schools.com/php/func_string_echo.asp
2. https://www.w3schools.com/css/
3. https://www.w3schools.com/html/
4. https://www.w3schools.com/php/php_variables.asp
5. https://www.w3resource.com/php/operators/arithmetic-operators.php
6. https://www.tutorialspoint.com/php/php_arithmatic_operators_examples.htm
7. https://www.w3schools.com/php/php_if_else.asp
8. https://www.w3schools.com/php/php_switch.asp
9. https://www.w3schools.com/php/php_looping.asp
10. https://www.w3schools.com/php/php_looping_while.asp
11. https://www.w3schools.com/php/php_looping_do_while.asp
12. https://www.w3schools.com/php/php_looping_for.asp
13. https://www.w3schools.com/php/php_looping_foreach.asp
14. https://www.w3schools.com/php/php_looping_break.asp

SNIPPETS:


localhost/bryanphp/5A1.php
Multiplication Table
0 0 0 0 0 0 0 0 0 0 0
0 1 2 3 4 5 6 7 8 9 10
0 2 4 6 8 10 12 14 16 18 20
0 3 6 9 12 15 18 21 24 27 30
0 4 8 12 16 20 24 28 32 36 40
0 5 10 15 20 25 30 35 40 45 50
0 6 12 18 24 30 36 42 48 54 60
0 7 14 21 28 35 42 49 56 63 70
0 8 16 24 32 40 48 56 64 72 80
0 9 18 27 36 45 54 63 72 81 90
0 10 20 30 40 50 60 70 80 90 100

SA1.php

```php
<!DOCTYPE html>
<html>
<head>
    <title>Multiplication Table</title>
    <style>
        table {
            border-collapse: collapse;
            width: 50%;
            margin: auto;
            text-align: center;
            border: 2px solid white;
        }
        th, td {
            border: 1px solid white;
            padding: 10px;
        }
        .red {
            background-color: red;
        }
        .yellow {
            background-color: yellow;
        }
    </style>
</head>
<body>
```

```php
<body>
    <h1 style="text-align: center;">Multiplication Table</h1>
    <table>
        <?php
        for ($row = 0; $row <= 10; $row++) {
            echo "<tr>";
            for ($col = 0; $col <= 10; $col++) {
                $value = $row * $col;
                $class = ($row + $col) % 2 == 0 ? "red" : "yellow";
                echo "<td class='$class'>$value</td>";
            }
            echo "</tr>";
        }
        ?>
    </table>
</body>
</html>
```

Note: The following rubrics/metrics will be used to grade students' output.

Program (100 pts.)	(Excellent)	(Good)	(Fair)	(Poor)

Program execution (20pts)	Program executes correctly with no syntax or runtime errors **(18-20pts)**	Program executes with less than 3 errors **(15-17pts)**	Program executes with more than 3 errors **(12-14pts)**	Program does not execute **(10-11pts)**
Correct output (20pts)	Program displays correct output with no errors **(18-20pts)**	Output has minor errors **(15-17pts)**	Output has multiple errors **(12-14pts)**	Output is incorrect **(10-11pts)**
Design of output (10pts)	Program displays more than expected **(10pts)**	Program displays minimally expected output **(8-9pts)**	Program does not display the required output (**6-7pts)**	Output is poorly designed **(5pts)**
Design of logic (20pts)	Program is logically well designed **(18-20pts)**	Program has slight logic errors that do no significantly affect the results **(15-17pts)**	Program has significant logic errors **(3-5pts)**	Program is incorrect **(10-11pts)**
Standards (20pts)	Program code is stylistically well designed **(18-20pts)**	Few inappropriate design choices (i.e. poor variable names, improper indentation) **(15-17pts)**	Several inappropriate design choices (i.e. poor variable names, improper indentation) **(12-14pts)**	Program is poorly written **(10-11pts)**
Delivery (10pts)	The program was delivered on time. **(10pts)**	The program was delivered a day after the deadline. **(8-9pts)**	The program was delivered two days after the deadline. **(6-7pts)**	The program was delivered more than two days after the deadline. **(5pts)**