SACTIVITY ANSWER SHEET

Name	Gladys D. Tagsip
Section:	BS-IT3R1

Instructions:

1. Push your output on your GITHUB repository.
2. Use the answer sheet provided save it as PDF file then push it to your GitHub.
3. Answer the ff. problems write it on the answer sheet.
4. Late submissions will no longer be accepted.
5. Caught copying outputs of others will be given sanctions.
6. Failure to follow these instructions will be given sanctions.

Activity 1: Control Structures

1. Write down the syntax in PHP for the ff.

1. if	if (condition) {
	    code to be executed if condition is true;
	}

2. if…else	if (condition) {
	    code to be executed if condition is true;
	} else {
	    code to be executed if condition is false;
	}

3. if…else if…else	if (condition) {
	    code to be executed if this condition is true;
	} else if (condition) {
	    code to be executed if first condition is false and this condition is true;
	} else {
	    code to be executed if all conditions are false;
	}

4. switch…case	switch (n) {
	    case label1:
	        code to be executed if n=label1;
	        break;
	    case label2:
	        code to be executed if n=label2;
	        break;
	    case label3:
	        code to be executed if n=label3;
	        break;
	    ...
	    default:
	        code to be executed if n is different from all labels;
	}

5. for loop	for (init counter; test counter; increment counter) {
	    code to be executed for each iteration;
	}

6. do while loop	do {
	    code to be executed;
	} while (condition is true);

7. while loop	while (condition is true) {
	    code to be executed;
	}

8. foreach loop	foreach ($array as $value) {
	  code to be executed;
	}

9. break statement	Break;
10. continue statement	Continue;

11. try…catch	try {
	    // run your code here
	}
	catch (exception $e) {
	    //code to handle the exception
	}
	finally {
	    //optional code that always runs
	}

2. Solve the ff. problem using PHP.

a. Write a program that checks if value is a number (integer).

Sample input: ‘1’ Sample input: 1
Expected output: Not a number Expected output: A number

```
<?php
$t = 1;

if (is_integer($t)) {
   echo "a number";
} else {
   echo "not a number";
}
?>
```

b. Write a program that checks if a value is positive or negative and odd or even.

Sample input: 0 Sample input: -1
Expected output: Positive & Even Expected output: Negative and Odd

```
<?php
$t = -1;

if($t > 0 ){
   if($t%2 == 0){
      echo "positive and even";
   }else{
      echo "positive and odd";
   }
}else if ($t < 0){
   if($t%2 == 0){
      echo "negative and even";
   }else{
      echo "negative and odd";
   }
}
?>
```

c. Write a program that checks if a value is palindrome.

Sample input: Anna Sample input: Bogart
Expected output: Palindrome Expected output: Not a Palindrome

```
<?php
   $string = "anna";
   if(strrev($string) == $string){
      echo "palindrome";
   }else{
      echo "not palindrom";
   }
?>
```

d. Write a program to calculate and print the factorial of a number using a for loop.

Sample input: 4
Expected output: 24

```
<?php
   $number =4 ;
   $increment = 1;
   $fact = 1;
   for($increment = 1;$increment<$number;$increment++){
      $fact = $fact * ($increment+1);
   }

   echo $fact;
?>
```

e. Write a PHP program to generate and display the first n lines of a Floyd triangle.

Sample input: 3

Sample output:

```
1
2 3
4 5 6
```

```
<?php
   $lineNum = 3;
   $count = 1;
   $s = 0;
   $i =0;

   for($i = $lineNum;$i > 0;$i--){
      for($j = $i; $j < $lineNum;$j++){
         printf("%4s", $count);
         $count++;
      }
      echo nl2br("\n");
   }
?>
```

Activity 2: PHP Built-in Functions

Write down the functionalities of the ff. built-in functions in PHP.

Array	-Array is a special variable, which can hold more than one value at a time. array_fill() -Fills an array with values array_fill_keys() -Fills an array with values, specifying keys array_filter() -Filters the values of an array using a callback function array_flip() -Flips/Exchanges all keys with their associated values in an array array_intersect() -Compare arrays, and returns the matches (compare values only)
Calendar	-Calendar extension contains functions that simplifies converting between different calendar formats.

	cal_days_in_month() Returns the number of days in a month for a specified year and calendar **easter_days()** Returns the number of days after March 21, that the Easter Day is in a specified year **frenchtojd()** Converts a French Republican date to a Julian Day Count **gregoriantojd()** Converts a Gregorian date to a Julian Day Count **jddayofweek()** Returns the day of the week
Date	-Date() function formats a local date and time, and returns the formatted date string. **checkdate()** Validates a Gregorian date **date_add()** Adds days, months, years, hours, minutes, and seconds to a date **date_create_from_format()** Returns a new DateTime object formatted according to a specified format **date_create()** Returns a new DateTime object **date_date_set()** Sets a new date
Directory	-Directory functions allow you to retrieve information about directories and their contents. **getcwd()** Returns the current working directory **opendir()** Opens a directory handle **readdir()** Returns an entry from a directory handle **rewinddir()** Resets a directory handle **scandir()** Returns an array of files and directories of a specified directory
Error	-Error functions are used to deal with error handling and logging. -Error functions allow us to define own error handling rules, and modify the way the errors can be logged. display_startup_errors "0" log_errors "0" log_errors_max_len "1024" ignore_repeated_errors "0" ignore_repeated_source "0"
File System	File system functions allow you to access and manipulate the file system. **allow_url_fopen** "1" Allows fopen()-type functions to work with URLs PHP_INI_SYSTEM **allow_url_include** "0" (available since PHP 5.2) PHP_INI_SYSTEM **user_agent** NULL Defines the user agent for PHP to send (available since PHP 4.3) PHP_INI_ALL **default_socket_timeout** "60" Sets the default timeout, in seconds, for socket

	based streams (available since PHP 4.3) PHP_INI_ALL **sys_temp_dir** "" (available since PHP 5.5) PHP_INI_SYSTEM
Filter	Filters is used to validate and filter data coming from insecure sources, like user input. **filter_id()** Returns the filter ID of a specified filter name **filter_input()** Gets an external variable (e.g. from form input) and optionally filters it **filter_input_array()** Gets external variables (e.g. from form input) and optionally filters them **filter_list()** Returns a list of all supported filter names **filter_var()** Filters a variable with a specified filter
FTP	FTP functions are used to open, login and close connections, as well as upload, download, rename, delete, and get information on files from file servers. **ftp_login()** Logs in to the FTP connection **ftp_mdtm()** Returns the last modified time of a specified file **ftp_mkdir()** Creates a new directory on the FTP server **ftp_mlsd()** Returns the list of files in the specified directory **ftp_nb_continue()** Continues retrieving/sending a file (non-blocking)
Libxml	Libxml functions and constant are used together with SimpleXML, XSLT and DOM functions. **libxml_clear_errors()** Clears the libxml error buffer **libxml_disable_entity_loader()** Enables the ability to load external entities **libxml_get_errors()** Gets the errors from the the libxml error buffer **libxml_get_last_error()** Gets the last error from the the libxml error buffer **libxml_set_external_entity_loader()** Changes the default external entity loader
Mail	Mail function allows you to send emails directly from a script. **ezmlm_hash()** Calculates the hash value needed by EZMLM **mail()** Allows you to send emails directly from a script
Math	Math function can handle values within the range of integer and float types.

	decbin() Converts a decimal number to a binary number **dechex()** Converts a decimal number to a hexadecimal number **decoct()** Converts a decimal number to an octal number **deg2rad()** Converts a degree value to a radian value **exp()** Calculates the exponent of e
Misc	The misc. functions were only placed here because none of the other categories seemed to fit. **defined()** Checks whether a constant exists **die()** Alias of exit() **eval()** Evaluates a string as PHP code **exit()** Prints a message and exits the current script **get_browser()** Returns the capabilities of the user's browser
MySQLi	MySQLi functions allows you to access MYSQL database server. **errno()** Returns the last error code for the most recent function call **error()** Returns the last error description for the most recent function call **error_list()** Returns a list of errors for the most recent function call **fetch_all()** Fetches all result rows as an associative array, a numeric array, or both **fetch_array()** Fetches a result row as an associative, a numeric array, or both
Network	The Network functions contains various network function and let you manipulate information sent to the browser by the Web server, before any other output has been sent. **getprotobyname()** Returns the protocol number for a given protocol name **getprotobynumber()** Returns the protocol name for a given protocol number **getservbyname()** Returns the port number for a given Internet service and protocol **getservbyport()** Returns the Internet service for a given port and protocol **header_register_callback()** Calls a header function
SimpleXML	SimpleXML is an extension that allows us to easily manipulate and get XML data. **getDocNamespaces()** Returns the namespaces declared in document **getName()** Returns the name of an element **getNamespaces()** Returns the namespaces used in document **registerXPathNamespace()** Creates

	a namespace context for the next XPath query **saveXML()** Alias of asXML()

Stream	Streams are the way of generalizing file, network, data compression, and other operations which share a common set of functions and uses.
	stream_context_get_options()
	stream_context_get_params()
	stream_context_set_default()
	stream_context_set_options()
	stream_context_set_params()

String	The PHP string functions are part of the PHP core. No installation is required to use these functions. **crc32()** Calculates a 32-bit CRC for a string **crypt()** One-way string hashing **echo()** Outputs one or more strings **explode()** Breaks a string into an array **fprintf()** Writes a formatted string to a specified output stream
XML Parser	The XML functions lets you parse, but not validate, XML documents. **xml_get_current_column_number()** Returns the current column number from the XML parser **xml_get_current_line_number()** Returns the current line number from the XML parser **xml_get_error_code()** Returns an error code from the XML parser **xml_parse()** Parses an XML document **xml_parse_into_struct()** Parses XML data into an array
Zip	Zip files function allows you to read ZIP Files. **zip_entry_name()** Returns the name of a ZIP directory entry **zip_entry_open()** Opens a directory entry in a ZIP file for reading **zip_entry_read()** Reads from an open directory entry in the ZIP file **zip_open()** Opens a ZIP file archive **zip_read()** Reads the next file in a open ZIP file archive
Timezones	**PHP Date/Time Functions** **PHP gmdate() Function** **PHP strtotime() Function** **PHP Date and Time** **PHP Tryit Editor v1.1**

Activity 3: Regular Expression

1. Define Regular Expression (RegEx) and provide example programming scenario where you can use (RegEx). Provide example syntax in PHP.
   - A regular expression (rational expression) is a sequence of characters that define a search pattern, mainly for use in pattern matching with strings, or string matching.
   *Programming scenario.
   Regular expressions help in validation of text strings which are of programmer’s interest.
   *Syntax

```
<?php
$string = "abc def";
$test = "/abc/";

if (preg_match($test,$string)){
echo "abc is found";
}
else{
 echo "abcis not found";
}
?>
```

2. Solve the ff. problem using Regular Expressions.

a. Write a PHP script that checks if a string contains another string
Sample String: 'The quick brown fox'
Test input: ‘Fox’
Expected output: Fox is found the string

```
<?php
$string = "The quick brown fox";
$test = "/fox/";

if (preg_match($test,$string)){
   echo "FOX  is found";
}
else{
   echo "FOX is not found";
}

?>
```

b. Write a PHP script that removes the last word from a string.
Sample String: 'The quick brown fox'
Expected output: ‘The quick brown'

```
<?php
$string = "The quick brown fox";
echo preg_replace('/\W\w+\s*(\W*)$/', '$1', $string)."\n";
?>
```

c. Write a PHP script to remove nonnumeric characters except comma and dot.
Sample String: '/$123,34.00A#'
Expected output: 123,34.00

```
<?php
$str = "/$123,34.00A#";
echo preg_replace("/[^0-9,.]/", "", $str)."\n";
?>
```

d. Write a PHP script to extract text (within parenthesis) from a string.
Sample String: ‘The quick brown [fox].’
Expected output: Fox

```
<?php
$str = 'The quick brown [fox].';
preg_match('#\[(.*?)\]#', $str, $match);
print $match[1]."\n";
?>
```

e. Write a PHP script to remove all characters from a string except a-z A-Z 0-9 or " ".
Sample String: ‘abcde$ddfd @abcd)der]’
Expected output: abcdeddfd abcd der

```
<?php
$alphabet = 'abcde$ddfd @abcd  )der]';
$run = preg_replace("/[^A-Za-z0-9 ]/", '', $alphabet);
echo 'Output : '.$run."\n";
?>
```

Activity 4: Error Handling

1. List down the different PHP errors. Provide example code on how to handle these errors.

***Parse Errors**

```
try{
        eval("echo 'toto' echo 'tata'");

}catch(ParseError $p){

   echo $p->getMessage();
}
```

***Fatal Errors**

```
set_error_handler('myErrorHandler');
register_shutdown_function('fatalErrorShutdownHandler');
function myErrorHandler($code, $message, $file, $line) {
  ...
}
function fatalErrorShutdownHandler()
{
  $last_error = error_get_last();
  if ($last_error['type'] === E_ERROR) {
    // fatal error
    myErrorHandler(E_ERROR, $last_error['message'], $last_error['file'], $last_error['line']);
  }
}
```

***Warning Errors**

```
set_error_handler("warning_handler", E_WARNING);
dns_get_record(...)
restore_error_handler();

function warning_handler($errno, $errstr) {
// do something
```

***Notice Errors**

```
<?php

// Turn off all error reporting
error_reporting(0);

// Report simple running errors
error_reporting(E_ERROR | E_WARNING | E_PARSE);

// Reporting E_NOTICE can be good too (to report uninitialized
// variables or catch variable name misspellings ...)
error_reporting(E_ERROR | E_WARNING | E_PARSE | E_NOTICE);

// Report all errors except E_NOTICE
error_reporting(E_ALL & ~E_NOTICE);

// Report all PHP errors (see changelog)
error_reporting(E_ALL);

// Report all PHP errors
error_reporting(-1);

// Same as error_reporting(E_ALL);
ini_set('error_reporting', E_ALL);

?>
```