-

CMR INSTITUTE OF TECHNOLOGY

USN | 1 | C | | | | | | | | |

Internal Assessment Test 1 – March 2018

Sub:	Advanced Web Programming							Code:	16MCA42
Date:	12-03-18	Duration:	90 mins	Max Marks:	50	Sem:	IV	Branch:	MCA

Note: Answer any 5 questions. All questions carry equal marks. Total marks: 50

		Marks	OBE CO	OBE RBT
1a.	What is an array. What are the different types of arrays in PHP. What are the different sorting functions in PHP.	[10]	CO1	L4

An array is a data structure that stores one or more similar type of values in a single value. For example if you want to store 100 numbers then instead of defining 100 variables its easy to define an array of 100 length.

There are three different kind of arrays and each array value is accessed using an ID c which is called array index.

- **Numeric array** – An array with a numeric index. Values are stored and accessed in linear fashion.
- **Associative array** – An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.
- **Multidimensional array** – An array containing one or more arrays and values are accessed using multiple indices

Numeric Array

These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.

Example

Following is the example showing how to create and access numeric arrays.Here we have used **array()** function to create array. This function is explained in function reference.

```
<html>
  <body>

    <?php
      /* First method to create array. */
      $numbers = array( 1, 2, 3, 4, 5);

      foreach( $numbers as $value ) {
        echo "Value is $value <br />";
      }

      /* Second method to create array. */
```

```
      $numbers[0] = "one";
      $numbers[1] = "two";
      $numbers[2] = "three";
      $numbers[3] = "four";
      $numbers[4] = "five";

      foreach( $numbers as $value ) {
        echo "Value is $value <br />";
      }
    ?>

  </body>
</html>
```

Associative Arrays

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.

```
<html>
  <body>

    <?php
      /* First method to associate create array. */
      $salaries = array("mohammad" => 2000, "qadir" => 1000, "zara" => 500);

      echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
      echo "Salary of qadir is ".  $salaries['qadir']. "<br />";
      echo "Salary of zara is ".  $salaries['zara']. "<br />";

      /* Second method to create array. */
      $salaries['mohammad'] = "high";
      $salaries['qadir'] = "medium";
      $salaries['zara'] = "low";

      echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
      echo "Salary of qadir is ".  $salaries['qadir']. "<br />";
      echo "Salary of zara is ".  $salaries['zara']. "<br />";
    ?>

  </body>
</html>
```

- sort() - sort arrays in ascending order
- rsort() - sort arrays in descending order
- asort() - sort associative arrays in ascending order, according to the value
- ksort() - sort associative arrays in ascending order, according to the key
- arsort() - sort associative arrays in descending order, according to the value
- krsort() - sort associative arrays in descending order, according to the key

Sort Array in Ascending Order - sort()

```
<!DOCTYPE html>
<html>
<body>
```

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);

$clength = count($cars);
for($x = 0; $x <  $clength; $x++) {
   echo $cars[$x];
   echo "<br>";
}
?>

</body>
</html>
```

O/P

BMW
Toyota
Volvo

Sort Array in Descending Order - rsort()

The following example sorts the elements of the $cars array in descending alphabetical order:

```
<!DOCTYPE html>
<html>
<body>

<?php
$numbers = array(4, 6, 2, 22, 11);
rsort($numbers);

$arrlength = count($numbers);
for($x = 0; $x <  $arrlength; $x++) {
     echo $numbers[$x];
     echo "<br>";
}
?>

</body>
</html>
```

O/P

22
11
6
4
2

Sort Array (Ascending Order), According to Value - asort()

The following example sorts an associative array in ascending order, according to the value:

```
<!DOCTYPE html>
<html>
<body>

<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

```
asort($age);

foreach($age as $x => $x_value) {
     echo "Key=" . $x . ", Value=" . $x_value;
     echo "<br>";
}
?>

</body>
</html>
```

O/P

Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43

Sort Array (Ascending Order), According to Key - ksort()

The following example sorts an associative array in ascending order, according to the key:

```
<!DOCTYPE html>
<html>
<body>

<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
ksort($age);

foreach($age as $x => $x_value) {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
}
?>

</body>
</html>
```

O/P
Key=Ben, Value=37
Key=Joe, Value=43
Key=Peter, Value=35

Sort Array (Descending Order), According to Value - arsort()

The following example sorts an associative array in descending order, according to the value:

```
<!DOCTYPE html>
<html>
<body>

<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
arsort($age);

foreach($age as $x => $x_value) {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
}
?>

</body>
```

</html>

O/P
Key=Joe, Value=43
Key=Ben, Value=37
Key=Peter, Value=35

Sort Array (Descending Order), According to Key - krsort()

The following example sorts an associative array in descending order, according to the key:

```
<!DOCTYPE html>
<html>
<body>

<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
krsort($age);

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>

</body>
</html>
```

O/P

Key=Peter, Value=35
Key=Joe, Value=43
Key=Ben, Value=37

2a.	Explain cookies and sessions in PHP with neat example This is a suitable method for tracking users. Web servers are stateless entities. **Cookies :-** Cookies provide a way for a server to store information about a user on the user's machine. This enables to maintain the user's visit to the site , so that they can track their movement through the site , or to store information such as their user name . Cookies allow data to be stored in the form of a name/value pair. Both the name and the value are set at choice. It contains the following syntax Name=value;expires=expiration date gmt; path=url(optional) To create a cookie Setcookie(name,value,expires,path); Cookie name/value pair: The first section of the cookie defines the name of the cookie and the value assigned to the cookie. Both the name and value settings can be any value as per the users choice. Cookie Expiration time: The optional expires= section specifies the date on which the associated cookie should expire. The PHP time() can be used to obtain and manipulate dates for this purpose Example:	[10]	CO1	L4

```
<?php
Setcookie('username','abcd',time()+4800);
Echo "cookie has been set";
?>
```

The cookie will expire after 4800 seconds.

Cookies are sent in the HTTP headers in pages sent by the browser . Once the cookies has been set they can be accessed on the next page load with the $_COOKIE array. This is an associative array where the name of the cookie provides the index into the array to extract the corresponding value of the name/value pair.

Example:

```
<?php
Echo "cookie value is ".$_COOKIE['username'];
?>
```

To delete a cookie

Cookies are deleted by calling the setcookie() function with the cookie name , a null for the value and an expiration date in the past.

```
<?php
Setcookie("username","",time()-4800);
?>
```

Sessions

A session is a way to store information (in variables) to be used across multiple pages. A session creates a file in a temporary directory on the server where registered session variables and their values are stored. The data will be available to all pages on the site during the visit.

A session like a cookie provides a way to track data for a user over a series of pages.

The main difference between cookie and session is that cookie stores the data on the client side in the web browser whereas the session data is stored on the server.

Sessions are generally more secure, because the data is not transmitted back and forth between the client and server repeatedly.

Sessions let you store more information than you can in cookie.
When session starts, PHP generates a random session id , a reference to that particular session and its stored data.

To start a session

```
Session_start();
```

Session variables are set with the PHP global variables $_SESSION.

```
<?php
Session_start();
$_SESSION["username"] = "abc";
?>
```

To delete the session variables , we unset $_SESSION array

```
Unset($_SESSION["username"]);
```

	To remove session data from the server where it is stored in the temporary file, we use Session_destroy();			

3a.	Explain with a neat example how PHP handles XML. Insert the values using MySQL and display them
	
	Student.xml
	<?xml version=”1.0” encoding=”utf-8”?>
	<student>
	<stud>
	<usn>1cr12mca01</usn>
	<name>ABC</name>
	<college>CMRIT</college>
	</stud>
	</student>
	
	Student.php
	<?php
	Mysql_connect(“localhost”,”root”,””);
	Mysql_select_db(“student1”);
	$q = simplexml_load_file(“Student.xml”);
	Foreach($q as $res)
	{
	$usn = $res->usn;
	$name = $res->name;
	$college = $res->college;
	$q = mysql_query(“insert into student values(‘$usn’,’$name’,’$college’)”);
	}
	$query = Mysql_query(“select * from student”);
	While($result = mysql_fetch_array($query))
	{
	Echo $result[“usn”];
	Echo $result[“name”];
	Echo $result[“college”];
	}
	?>
		[10]	CO1	L4

4a.	What are super global variables. Explain any six super global variables in PHP Several predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special. The PHP superglobal variables are: • $GLOBALS • $_SERVER • $_REQUEST • $_POST • $_GET • $_FILES • $_ENV • $_COOKIE • $_SESSION $_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.	[6]	CO1	L4

PHP $_REQUEST is used to collect data after submitting an HTML form.The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_REQUEST to collect the value of the input field

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
 Name: <input type="text" name="fname">
 <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   // collect value of input field
   $name = $_REQUEST['fname'];
   if (empty($name)) {
      echo "Name is empty";
   } else {
      echo $name;
   }
}
?>

</body>
</html>
```

PHP $_POST is widely used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_POST to collect the value of the input field:

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
 Name: <input type="text" name="fname">
 <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   // collect value of input field
   $name = $_POST['fname'];
   if (empty($name)) {
      echo "Name is empty";
```

	} else {
	echo $name;
	}
	}
	?>
	
	</body>
	</html>
	
	PHP $_GET can also be used to collect form data after submitting an HTML form with method="get".
	
	$_GET can also collect data sent in the URL.
	
	Assume we have an HTML page that contains a hyperlink with parameters:
	
	<html>
	<body>
	
	Test $GET
	
	</body>
	</html>
	
	<html>
	<body>
	
	<?php
	echo "Study " . $_GET['subject'] . " at " . $_GET['web'];
	?>
	
	</body>
	</html>			

b.	What are the various ways to include files in PHP. Include and require Include If the included file is not found in the folder it gives a warning message and continues execution of the script. Require: If the included file is not found in the folder it gives a fatal error and halts the execution of the script. Similarly include_once and require_once	[4]	CO1	L4
5a.	What are the different types of errors in PHP. **Error is a deviation from accuracy or a mistake caused unintentionally. In PHP, 3 types of basic errors are –** **1.** Notices**: These are small, non-critical errors that PHP encounters while executing a script – for example, accessing a variable that has not yet been defined. By default, such errors are not displayed to the user at all – although the default behavior can be changed.** **2.** Warnings**: Warnings are more severe errors like attempting to include() a file which does not exist. By default, these errors are displayed to the user, but they do not result in script termination.** **3.** Fatal errors**: These are critical errors – for example, instantiating an object of a non-**	[5]	CO1	L4

	existent class, or calling a non-existent function. These errors cause the immediate termination of the script, and PHP's default behavior is to display them to the user when they take place. Different types of errors are : **E_ERROR: A fatal error that causes script termination** **E_WARNING: Run-time warning that does not cause script termination** **E_PARSE: Compile time parse error.** **E_NOTICE: Run time notice caused due to error in code** **E_CORE_ERROR: Fatal errors that occur during PHP's initial startup (installation)** **E_CORE_WARNING: Warnings that occur during PHP's initial startup** **E_COMPILE_ERROR: Fatal compile-time errors indication problem with script.** **E_USER_ERROR: User-generated error message.** **E_USER_WARNING: User-generated warning message.** **E_USER_NOTICE: User-generated notice message.** **E_STRICT: Run-time notices.** **E_RECOVERABLE_ERROR: Catchable fatal error indicating a dangerous error** **E_ALL: Catches all errors and warnings**			
b	Differentiate between GET and POST method in PHP. Both GET and POST create an array (e.g. array(key => value, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user. Both GET and POST are treated as $_GET and $_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special. $_GET is an array of variables passed to the current script via the URL parameters. $_POST is an array of variables passed to the current script via the HTTP POST method. <u>When to use GET?</u> Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases. GET may be used for sending non-sensitive data. **Note:** GET should NEVER be used for sending passwords or other sensitive information! <u>When to use POST?</u> Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send. Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server. However, because the variables are not displayed in the URL, it is not possible to	[5]	CO1	L4

	bookmark the page.			

6	a.	Discuss about database connectivity in PHP. Explain with a neat example how to insert into and display values from Mysql in PHP
		
		<?php
		error_reporting(0);
		mysql_connect("localhost","root","");
		mysql_select_db("employee");
		?>
		
		<?php
		include "dbconnect.php";
		if(($_SERVER["REQUEST_METHOD"]=="POST")\|\|($_SERVER["REQUEST_METHOD"]=="post"))
		{
		$name=$_POST["name"];
		$des=$_POST["des"];
		$dep=$_POST["dep"];
		$pno=$_POST["pno"];
		$email=$_POST["email"];
		$qry="insert
		intodetail(name,des,dep,pno,email)values('$name','$des','$dep','$pno','$email')";mysql_query($qry);
		header("Location:display.php");
		}
		?>
		<html>
		<head> </head>
		<body>
		<form method="post" action="home.php">
		 <table border="5px">
		<caption>EMPLOYEE DETAILS</caption>
		<tr>
		<td>Name:<input type="text" name="name"/></td>
		</tr>
		<tr>
		<td>Destination:<input type="text" name="des"/></td>
		</tr>
		<tr>
		<td>Department:<input type="text" name="dep"/></td>
		</tr>
		<tr>
		<td>Phone no:<input type="text" name="pno"/></td>
		</tr>
		<tr>
		<td>Email:<input type="text" name="email"/></td>
		</tr>
		<tr>
		<td><input type="submit" value="SUBMIT"/></td>
		<td>DISPLAY</td>
		</tr>
		</table>
		</form>
		</body>
		</html>
			[10]	CO1	L4

7	a.	Write a program in PHP to count the number of 0s and 1s in 110010011 <?php function calculate() { $num=1100100100; $n = strlen($num);	[10]	CO1	L4

```
  $ones=0;$zeroes=0;
 while($n>0)
 {
  $r=$num%10;
  if($r==1)
   $ones++;
  else
   $zeroes++;
  $num=$num/10;
        $n--;
 }
  echo "\r\n count of 0s in given number is =",$zeroes;
 echo "<br>count of 1s in given number is =",$ones;
}
calculate();
?>
```

8a. Write a program for file uploading in PHP? [10] CO1 L4

```
<?php
// --------- File Uploading script--------/
error_reporting(0);
if(isset($_POST))//validating whether post data is filled or not
{
        $name       =       $_FILES["fupload"]["name"];
        $size       =       $_FILES["fupload"]["size"];
        $type       =       $_FILES["fupload"]["type"];
        $tmpname    =       $_FILES["fupload"]["tmp_name"];
        $upload = move_uploaded_file($tmpname,"uploads/".$name);
        if($upload)
                echo "Uploaded successfully";
        else
                echo "Upload failed";
}

?>
<html>
<body>
<form method="post" enctype="multipart/form-data" action="">
<input type="hidden" name="MAX_FILE_SIZE" value="102400000"/>
Photo <input type="file" name="fupload"/>
<br/><input type="submit" value="Upload"/>
</form>
</body>
</html>
```