KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
Pollachi Main Road, Eachanari Post, Coimbatore - 641021
(For the candidates admitted from 2016 onwards)
DEPARTMENT OF CS, CA&IT

Semester – VI
16CTU601A **PHP PROGRAMMING** **4H – 4C**

Instruction Hours / week: L: 3 T: 0 P: 0 **Marks:** Int : **40** Ext : **60** Total: **100**

SCOPE

This course design focuses on the structure of the website including the information architecture, the layout or the pages and the conceptual design with branding. PHP helps the students for developing dynamic web pages.

OBJECTIVES

- To work with open source applications that deal with database and website development.
- Establish a working environment for PHP web page development
- Very familiar with GUI, coded, modified controls

UNIT-I

Introduction to PHP:
PHP introduction, inventions and versions, important tools and software requirements (like Web Server, Database, Editors etc.) -PHP with other technologies, scope of PHP
Basic Syntax, PHP variables and constants-Types of data in PHP, Expressions, scopes of a variable (local, global)-PHP Operators: Arithmetic, Assignment, Relational, Logical operators, Bitwise , ternary and MOD operator.-PHP operator Precedence and associativity

UNIT-II

Handling HTML form with PHP:
Capturing Form Data -GETand POST form methods- Dealing with multi value fields -Redirecting a form after submission **-PHP conditional events and Loops:** PHP IF Else conditional statements (Nested IF and Else) -Switch case, while ,For and Do While Loop -Goto , Break ,Continue and exit

UNIT-III

PHP Functions:
Function, Need of Function, declaration and calling of a function-PHP Function with arguments, Default Arguments in Function-Function argument with call by value, call by reference-Scope of Function Global and Local

UNIT-IV

String Manipulation and Regular Expression: (3L) Creating and accessing String , Searching & Replacing String-Formatting, joining and splitting String , String Related Library functions -Use and advantage of regular expression over inbuilt function-Use of preg_match(), preg_replace(), preg_split() functions in regular expression.

UNIT-V
Array:
Anatomy of an Array ,Creating index based and Associative array, Accessing array-Looping with Index based array, with associative array using each() and foreach() -Some useful Library function

Suggested Readings

1. Steven Holzner. (2007). PHP: The Complete Reference. New Delhi: McGraw Hill Education (India).
2. Timothy Boronczyk., & Martin, E. Psinas. (2008). PHP and MYSQL (Create-Modify-Reuse). New Delhi: Wiley India Private Limited.
3. Robin Nixon. (2014). Learning PHP, MySQL, JavaScript, CSS & HTML5 (3rd ed.). O'reilly.
4. Luke Welling.,& Laura Thompson.(2008). PHP and MySQL Web Development (4th ed.). Addition Paperback, Addison-Wesley Professsional.
5. David Sklar., & Adam Trachtenberg. PHP Cookbook: Solutions & Examples for PHP.

Websites

W1: https://www.phptpoint.com/php-tutorial/
W2:https://www.tutorialspoint.com/php/
W3:https://www.w3schools.com/php/php_forms.asp
W4:https://www.geeksforgeeks.org/php-functions/
W5:https://www.guru99.com/
W6: https://www.oreilly.com/library/view/php-and-mysql/9780133038644/ch04.html
W7:https://www.tutorialrepublic.com/php-tutorial/
W8:https://www.techrepublic.com/article/17-useful-functions-for-manipulating-arrays-in-php/

PATTERN OF EXAMINATION (CIA & ESE)

Continuous Internal Assessment

S.No	Category	Marks
1	Assignment	5
2	Attendance	5
3	Seminar	5
4	CIA I	8
5	CIA II	8
6	CIA III	9
Total Marks		**40**

End Semester Examination
Marks Allocation

1	**Part A** 20 X 1 = 20 Online Examination	20
2	**Part B** 5 X 2 = 10	10
3	**Part C** 5 X 6 = 30 Either '**A**' OR '**B**' Choice	30
	Total Marks	**60**

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
Pollachi Main Road, Eachanari Post, Coimbatore - 641021
(For the candidates admitted from 2016 onwards)
DEPARTMENT OF INFORMATION TECHNOLOGY

SUBJECT : PHP Programming
SEMESTER : VI
SUBJECT CODE: 16CTU601A **CLASS : III B.SC (CT)**

LECTURE PLAN

S.NO	LECTURE DURATION (Hour)	TOPICS TO BE COVERED	SUPPORT MATERIALS
UNIT I			
1.	1	**Introduction to PHP:** ➢ PHP introduction	S1:1,W1
2.	1	➢ Inventions and versions	S1:1
3.	1	➢ Important tools and software requirements	W1, S3:15-43
		➢ PHP with other technologies	S3:5-10
4.	1	➢ Scope of PHP	S3:66-70
5.	1	➢ Basic Syntax	S3:49
		➢ PHP variables and constants	S1:26-30,35-36, S3:50-55,63-64
6.	1	➢ Types of data in PHP	S3:62,W2
		➢ Expressions, scopes of a variable	S3:73-75
7.	1	➢ PHP Operators:	S1:41-52
8.	1	➢ PHP Operator Precedence and associativity	S1:53-55, S3:76-80

9.	1	**Recapitulation of Unit I**	
		Discussion of important Questions	
	Total no. of Hours Planned for Unit - I		9 hrs
UNIT II			
1.	1	**Handling HTML form with PHP:** ➢ Capturing Form Data	S1:161-195, W3,W4
2.	1	➢ GET and POST form methods	
3.	1	➢ Dealing with multi value fields	
4.	1	➢ Redirecting a form after submission	
5.	1	**PHP conditional events and Loops:** ➢ PHP IF Else conditional statements (Nested IF and Else)	S1:55-67, S3:84-87
		➢ Switch case	S1:67-69, S3:88-91
6.	1	➢ While, For and Do While Loop	S1:69-76
7.	1	➢ Goto , Break ,Continue and exit	S3:97-98
8.	1	**Recapitulation of Unit II**	
		Discussion of important Questions	
		Total no. of Hours Planned for Unit – II	8 Hrs
UNIT III			
1.	1	**PHP Functions:** ➢ Function, Need of Function	S1:123-125 S3:103-106
2.	1	➢ Declaration and calling of a function	S1:125-127,W4
3.	1	➢ PHP Function with arguments	S1:127-130,W4
4.	1	➢ Default Arguments in Function	S1:132-133

5.	1	➢ Function argument with call by value	S1:133-135
		➢ Function argument with call by reference	S1:130-132
6.	1	➢ Scope of Function Global and Local	S1:145, S3:111
7.	1	**Recapitulation of Unit III**	
		Discussion of important Questions	
		Total no. of Hours Planned for Unit - III	7 Hrs
		UNIT IV	
1.	1	**String Manipulation and Regular Expression** ➢ Creating and accessing String , Searching & Replacing String	S1:83-84 S4:115,S5:1-8
2.	1	➢ Formatting, joining and splitting String	S5:10-18 S4:116,W6
3.	1	➢ String Related Library functions	
4.	1	➢ Use and advantage of regular expression over inbuilt function	S3:396-397,W2
5.	1	➢ Use of preg_match(), preg_replace ()	S5:651-657, S5:12-14,W5,W7
6.	1	➢ preg_split() functions in regular expression.	S5:25-27,W5,W7
7.	1	**Recapitulation of Unit IV**	
		Discussion of important Questions	
		Total no. of Hours Planned for Unit - IV	7 Hrs
		UNIT V	
1.	1	**Array:** ➢ Anatomy of an Array	S3:131,W7
2.	1	➢ Creating index based	S3:131-134
3.	1	➢ Associative array ➢ Accessing array	

4.	1	➢ Looping with Index based array	S3:131,W7
5.	1	➢ associative array using each() and for each()	S3:135-137
6.	1	➢ Some useful Library function	S3:140-144,W8
7.	1	**Recapitulation of Unit V**	
		Discussion of important Questions	
8.	1	**Previous Year ESE Questions Discussion**	
9.	1	**Previous Year ESE Questions Discussion**	
		Total no. of Hours Planned for Unit - V	9 Hrs

Suggested Readings

1. Steven Holzner. (2007). PHP: The Complete Reference. New Delhi: McGraw Hill Education (India).
2. Timothy Boronczyk., & Martin, E. Psinas. (2008). PHP and MYSQL (Create-Modify-Reuse). New Delhi: Wiley India Private Limited.
3. Robin Nixon. (2014). Learning PHP, MySQL, JavaScript, CSS & HTML5 (3rd ed.). O'reilly.
4. Luke Welling.,& Laura Thompson.(2008). PHP and MySQL Web Development (4th ed.). Addition Paperback, Addison-Wesley Professsional.
5. David Sklar., & Adam Trachtenberg. PHP Cookbook: Solutions & Examples for PHP.

Website

W1: https://www.phptpoint.com/php-tutorial/
W2:https://www.tutorialspoint.com/php/
W3:https://www.w3schools.com/php/php_forms.asp
W4:https://www.geeksforgeeks.org/php-functions/
W5:https://www.guru99.com/
W6: https://www.oreilly.com/library/view/php-and-mysql/9780133038644/ch04.html
W7:https://www.tutorialrepublic.com/php-tutorial/
W8:https://www.techrepublic.com/article/17-useful-functions-for-manipulating-arrays-in-php/

UNIT-I

Introduction to PHP:

PHP introduction, inventions and versions, important tools and software requirements (like Web Server, Database, Editors etc.) -PHP with other technologies, scope of PHP

Basic Syntax, PHP variables and constants-Types of data in PHP , Expressions, scopes of a variable (local, global)-PHP Operators: Arithmetic, Assignment, Relational, Logical operators, Bitwise , ternary and MOD operator.-PHP operator Precedence and associativity

PHP INTRODUCTION:

Introduction to PHP

PHP is a simple yet powerful language designed for creating HTML content. This chapter covers essential background on the PHP language. It describes the nature and history of PHP, which platforms it runs on, and how to configure it. This chapter ends by showing you PHP in action, with a quick walkthrough of several PHP programs that illustrate common tasks, such as processing form data, interacting with a database, and creating graphics.

What is PHP

An established server-side, crossplatform embedded HTML scripting language for creating dynamic Web pages. PHP provides many features that commercial entities are looking for

What Does PHP Do?

PHP can be used in three primary ways:

1. Server-side scripting

PHP was originally designed to create dynamic web content, and it is still best suited for that task. To generate HTML, you need the PHP parser and a web server through which to send the coded documents. PHP has also become popular for generating XML documents, graphics, Flash animations, PDF files, and so much more.

2. Command-line scripting

PHP can run scripts from the command line, much like Perl, awk, or the Unix shell.

You might use the command-line scripts for system administration tasks, such as backup and log parsing; even some CRON job type scripts can be done this way (nonvisual PHP tasks).

3. Client-side GUI applications

Using PHP-GTK, you can write full-blown, cross-platform GUI applications in PHP.

Characteristics of PHP

Five important characteristics make PHP's practical nature possible:

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

PHP Advantages

- Exceptionally short learning curve
- Quick development time
- Very high performance
- It supports all major platforms (UNIX, Windows and even mainframes).
- Features native support for most popular databases

INVENTION AND VERSIONS OF PHP

The first version of what came to be known as PHP was created in 1995 by a man named Rasmus Lerdof. Rasmus, now an engineer at Yahoo!, needed something to make it easier to create content on his web site, something that would work well with HTML, yet give him power and flexibility beyond what HTML could offer him. Essentially, what he needed was an easy way to write scripts that would run on his web server both to create content, and handle data being passed back to the server from the web browser. Using the Perl language, he created some technology that gave him what he needed and decided to call this technology "Personal Home Page/Forms Interpreter". The technology provided a convenient way to process web forms and create content.

The name "Personal Home Page/Forms Interpreter" was later shortened to PHP/FI and eventually renamed to represent "PHP: Hypertext Preprocessor". The name is said to be recursive because the full name also includes the acronym "PHP" - an odd geeky joke that is common in technology circles when people have trouble naming things. GNU is another recursive name that represents "GNU's Not Unix".

PHP/FI version 1.0 was never really used outside of Rasmus' own web site. With the introduction of PHP/FI 2.0 this began to change. When PHP 3 was released in 1997, adoption of PHP exploded beyond all belief.

PHP 3 Hits the Big Time

By the time 1997 arrived the number of web sites on the internet was growing exponentially and most of these web sites were being implemented using the Apache web server. It was around this time that Andy Gutmans and Zeev Suraski launched the PHP 3 project, a project designed to take PHP to the next level. One of the key achievements of the PHP 3 project was to implement PHP as a robust Apache Module.

PHP 3 was implemented using a modular approach that made it easy for others to extend functionality, and also introduced the first elements of object-orientation that would continue to evolve through subsequent releases.

The combination of PHP 3 and Apache quickly lead to the widespread adoption of PHP, and it is commonly estimated that, at its peak adoption level, PHP3 was used to power over 10% of all web sites on the internet.

PHP 4 - Optimization, Scalability and More

With PHP 4 Andi Gutmans and Zeev Suraski once again re-architected PHP from the ground up. PHP 4 was built upon a piece of technology called the Zend Engine. The move to the Zend Engine brought about a number of key improvements in PHP:

- Support for other web servers (Microsoft's Internet Information Server (IIS) being of particular significance).
- Improved memory handling to avoid memory leaks (one of the most difficult types of problems to isolate in a program).
- Improved efficiency and performance to support large scale, complex, mission critical enterprise application development using PHP.

In addition PHP 4 also built on the earlier Object Oriented Programming features of PHP 3 with the introduction of classes.

PHP 5 - Object Orientation, Error Handling and XML

The main, though far from only, feature of PHP 5 is the improved support for Object Oriented Programming (OOP). In addition, PHP 5 introduced some features common in other languages such as Java like try/catch error and exception handling.

PHP 5 also introduced new extensions aimed at easing the storage and manipulation of data. Significant new features include SimpleXML for handling XML documents, and SQLite, an embedded basic and easy to use database interface.

PHP 6 : In this version, ICU (International Components for Unicode) library wasembedded into the program. But due to several other reasons this version was abandoned and was not launched in the market.

PHP 7 : This is the newest version of the php programming language having several features which were not present in the previous versions such as 64-bit integer support, return and scalar type declarations etc. It is powered by Zend Engine 3.

IMPORTANT TOOLS AND SOFTWARE REQUIREMENTS

The three types of tools for PHP:

- Tools for writing code
- A file transfer program
- A local Web server

Software Requirements

PHPKB knowledge base software is very simple to install and easy to configure PHP script on your web server. If required, we also offer a free installation service of our knowledge management software on your web-server. It requires the following:

User Requirements

- Operating System: Windows, Mac or Linux
- Web Browser: All industry standard web browsers (Internet Explorer, Mozilla Firefox, Google Chrome, Apple Safari)

Web Server Requirements

You need to be running PHP 5.3+ & MySQL 4.1+ on a web server (Apache/IIS). PHPKB knowledge base software has been deployed successfully on both Apache and IIS web servers.

- **Operating System**: Linux, Unix, Windows
- **Web Server**: Apache Web Server, LigHTTPD, IIS (with ISAPI_Rewrite installed)
- **PHP Version**:
  - For MySQL Editions, PHP 5.3 or above with PHP XML extension enabled.
  - For SQL Server Editions, PHP 5.3 or above with PHP XML extension enabled and Microsoft SQL Server Driver for PHP

- **Database**: MySQL 4.1 or above, Microsoft SQL Server 2005 or above.
- Optional Requirements
    - Apache's mod_rewrite() extension to allow for SEO friendly URLs.
    - PHP LDAP library is required to use LDAP functionality in the Enterprise edition
    - AntiWord (free) is required to index and search Microsoft Word 2003/XP documents
    - ppthtml (free) is required to index and search Microsoft PowerPoint 2003/XP documents
    - PHP zip library (free) is required to index and search Microsoft Word/PowerPoint 2007 documents
    - pdftohtml (free) is required to index and search Adobe PDF documents

Requirement Details

Apache (Recommended)

PHPKB will work on Apache 1.3 or Apache 2.x hosted on UNIX/Linux, OS X, or Windows. The majority of PHPKB development and deployment is done on Apache, so there is more community experience and testing performed on Apache than on other web servers. You can use the Apache 'mod_rewrite' extension to allow for SEO friendly URLs.

Microsoft IIS

PHPKB will work using IIS 5, IIS 6, IIS 7 or later, if PHP is configured correctly. To achieve SEO friendly URLs you may need to use a third party product. For IIS7 (or later), you can use the Microsoft URL Rewrite Module or a third party solution such as ISAPI_Rewrite. ISAPI_Rewrite is Apache mod_rewrite compatible URL rewriter for Microsoft IIS.

Disk Space

A minimum base installation requires at least 8MB of disk space but you should assume that your actual disk space will be somewhat higher. For example, if you upload file attachments and images for articles, the actual disk space for your knowledge base could easily be 20 MB or more (exclusive of database content, media, backups and other files).

The total file size of your PHPKB installation will depend on what you add to your knowledge base, but core files alone will take up approximately 7 to 9 MB uncompressed. The exact size depends on the edition of PHPKB you have installed.

Browsers

All modern browsers are supported (IE7+, Firefox, Safari, Google Chrome, Opera) that support CSS and JavaScript.

Recommended Hardware Requirements

PHPKB knowledge base software works well on any web-server that meets its software requirements specified above. This is definitely not a processor-hungry or memory-hungry application. Numerous instances of PHPKB knowledge base software are running on shared web hosting (virtual hosting) systems and are working perfectly fine. It means you can install it on a shared hosting server too. If you would like to use a dedicated hosting or host it on a local server, you can have a look at the hardware requirements and example configurations for a reference.

PHP Server

The PHP Community Provides Some types of Software Server solution under The GNU (General Public License).

These are the following:

1. WAMP Server
2. LAMP Server
3. MAMP Server
4. XAMPP Server

All these types of software automatic configure inside operating system after installation it having PHP, MySQL, Apache and operating system base configuration file, it doesn't need to configure manually.

Server	Stands for
WAMP	Microsoft window o/s, Apache, Mysql, PHP
LAMP	Linux Operating System, Apache, Mysql, PHP
MAMP	Mac os, Apache, Mysql, PHP,
XAMPP	x-os(cross operating system), Apache, Mysql, PHP,Perl

SCOPE OF PHP

PHP Visibility over internet : securing php future

- If you surf net, you can easily find out many websites with extension '.php'. This vast availability of php applications on internet can make you think about its popularity.
- Currently PHP pages are a common part of web applications, and one of the most popular languages for web development used by developers worldwide.
- If we surf internet we can see millions of websites built with php and mysql.

PHP in blogging: a secure scope of php

- Excellent blog webistes are built using wordpress, which is also designed by PHP.
- Blogs are the most common contents of internet in today's world. People with minimum technical knowledge operates their Blogs, this is possible by simple language like php.

CMS Supporting PHP: a Technical Future Move

- Thousands of CMS that support PHP are freely available to download and use like Zend, Codeigniter, YUii, symfony, Joomla, Magento, Drupal and Cake php etc.
- CMS in support to PHP provides the efficient way to publish websites easily, which is a very good move for the future of PHP.

PHP and MySQL: providing vast range application

- The combination of PHP and MySQL provides a vast range of web application development.
- Web application development for even non-technical persons have become simple with combo of PHP and MySQL database.

PHP simplicity uplifting its use in future

- PHP is so simple and easy to use, that makes its future bright as working on other languages is more complex than PHP.
- PHP came in lame light after over taking many powerful languages in web development, this justifies that PHP is going to be in market for long lasting**.**

Regular updation in PHP: a way to maintain future existence

- Day by day modification for using PHP is being done by developing new technologies, new frameworks.
- New frameworks are coming in existence to support PHP coding, this is a measure for PHP future.

- Object oriented based frameworks like zend is in market that is working like a booster for PHP web application development.
- CMS that supports working on PHP provides extendable plug-ins to extend the features for using PHP.

Recommendations: increase in popularity and future scope of PHP

- Many professionals from top companies prefer using PHP for web application development.
- PHP is used over other languages because it's above mentioned benefits.
- PHP is also a low cost web development tool, developers charge very less or we may say low development charge is allotted. As a result companies prefer it more.
- For the developers who are fresher to PHP, for them also PHP is a very good and effective tool to start with.
- If you want to continue your job as a developer itself, then PHP or any other tool/language is just a technology, more focus should be on your logics.
- Logics are the important tool for your survival as the code developer

BASIC PHP SYNTAX

A PHP script can be placed anywhere in the document.

Canonical PHP Tags:

The script starts with **<?php** and ends with **?>** . These tags are also called 'Canonical PHP tags'.

A PHP script starts with <?php and ends with ?>:

<u>Syntax</u>

```
<?php
// PHP code goes here
?>
```

The default file extension for PHP files is ".php".

A PHP file normally contains HTML tags, and some PHP scripting code.

Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

Semicolons

PHP commands ended with a semicolon, like this:

```
$x += 10;
```

Probably the most common cause of errors you will encounter with PHP is forgetting this semicolon. This causes PHP to treat multiple statements like one statement, which it is unable to understand, prompting it to produce a Parse error message.

Example

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

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

The $ symbol

The $ symbol has come to be used in many different ways by different programming languages. For example, if you have ever written in the BASIC language, you will have used the $ to terminate variable names to denote them as strings.

In PHP, however, you must place a $ in front of *all* variables. This is required to make the PHP parser faster, as it instantly knows whenever it comes across a variable.

Example:Three different types of variable assignment

```
<?php
$mycounter = 1;
$mystring = "Hello";
$myarray = array("One", "Two", "Three");
?>
```

COMMENTS IN PHP

A comment is something which is ignored and not read or executed by PHP engine or the language as part of a program and is written to make the code more readable and understandable. These are used to help other users and developers to describe the code and what it is trying to do. It can also be used in documenting a set of code or part of a program. You must have noticed this in above sample programs.
PHP supports two types of comment:

Single Line Comment: As the name suggests these are single line or short relevant explanations that one can add in there code. To add this, we need to begin the line with (//) or (#).

Example:

```
<?php
// This is a single line comment
// These cannot be extended to more lines
  echo "hello world!!!";
  # This is also a single line comment
?>
```

Multi-line or Multiple line Comment: These are used to accomodate multiple lines with a single tag and can be extended to many lines as required by the user. To add this, we need to begin and end the line with (**/*…*/**)

```
<?php
/* This is a multi line comment
In PHP variables are written
by adding a $ sign at the beginning.*/

$geek = "hello world!";
echo $geek;
?>
```

PHP VARIABLES

A variable in PHP is a name of memory location that holds data. A variable is a temporary storage that is used to store data temporarily.

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number

- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)

Syntax of declaring a variable in PHP is given below:

```
$variablename=value;
```

PHP VARIABLE: DECLARING STRING, INTEGER AND FLOAT

```
<?php
$str="hello string";
$x=200;
$y=44.6;
echo "string is: $str <br/>";
echo "integer is: $x <br/>";
echo "float is: $y <br/>";
?>
```

Output:

```
string is: hello string
integer is: 200
float is: 44.6
```

PHP Variable: Sum of two variables

```
<?php
$x=5;
$y=6;
$z=$x+$y;
echo $z;
?>
```

Output:

```
11
```

PHP Variable: case sensitive

In PHP, variable names are case sensitive. So variable name "color" is different from Color, COLOR, COLor etc.

File: variable3.php

```
<?php
$color="red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
```

Output:

```
My car is red
Notice: Undefined variable: COLOR in C:\wamp\www\variable.php on line 4
My house is
Notice: Undefined variable: coLOR in C:\wamp\www\variable.php on line 5
My boat is
```

PHP Variable: Rules

PHP variables must start with letter or underscore only.

PHP variable can't be start with numbers and special symbols.

File: variablevalid.php

```
<?php
$a="hello";//letter (valid)
$_b="hello";//underscore (valid)

echo "$a <br/> $_b";
?>
```

Output:

```
hello
hello
```

PHP Data Types

PHP data types are used to hold different types of data or values. PHP supports 8 primitive data types that can be categorized further in 3 types:

1. Scalar Types
2. Compound Types
3. Special Types

PHP Data Types: Scalar Types

There are 4 scalar data types in PHP.

1. boolean
2. integer
3. float
4. string

PHP Data Types: Compound Types

There are 2 compound data types in PHP.

1. Array
2. Object

PHP Data Types: Special Types

There are 2 special data types in PHP.

1. resource
2. NULL

PHP Booleans

Booleans are like a switch it has only two possible values either 1 (true) or 0 (false).

Example

```
<?php
// Assign the value TRUE to a variable
$show_error = true;
var_dump($show_error);
?>
```

PHP Integers

Integers are whole numbers, without a decimal point (..., -2, -1, 0, 1, 2, ...). Integers can be specified in decimal (base 10), hexadecimal (base 16 - prefixed with 0x) or octal (base 8 - prefixed with 0) notation, optionally preceded by a sign (- or +).

```
<?php
```

```
$a = 123; // decimal number
var_dump($a);
echo "<br>";

$b = -123; // a negative number
var_dump($b);
echo "<br>";

$c = 0x1A; // hexadecimal number
var_dump($c);
echo "<br>";

$d = 0123; // octal number
var_dump($d);
?>
```

PHP Floating Point Numbers or Doubles

Floating point numbers (also known as "floats", "doubles", or "real numbers") are decimal or fractional numbers, like demonstrated in the example below.

Example

```
<?php
$a = 1.234;
var_dump($a);
echo "<br>";

$b = 10.2e3;
var_dump($b);
echo "<br>";

$c = 4E-10;
var_dump($c);
?>
```

PHP Strings

Strings are sequences of characters, where every character is the same as a byte.

A string can hold letters, numbers, and special characters and it can be as large as up to 2GB (2147483647 bytes maximum). The simplest way to specify a string is to enclose it in single quotes (e.g. 'Hello world!'), however you can also use double quotes ("Hello world!").

Example

```
<?php
$a = 'Hello world!';
echo $a;
echo "<br>";

$b = "Hello world!";
echo $b;
echo "<br>";

$c = 'Stay here, I\'ll be back.';
echo $c;
?>
```

PHP Arrays

An array is a variable that can hold more than one value at a time. It is useful to aggregate a series of related items together, for example a set of country or city names.

An array is formally defined as an indexed collection of data values. Each index (also known as the key) of an array is unique and references a corresponding value.

Example

```
<?php
$colors = array("Red", "Green", "Blue");
var_dump($colors);
echo "<br>";

$color_codes = array(
    "Red" => "#ff0000",
    "Green" => "#00ff00",
    "Blue" => "#0000ff"
);
var_dump($color_codes);
?>
```

PHP Objects

An object is a data type that not only allows storing data but also information on, how to process that data. An object is a specific instance of a class which serve as templates for objects. Objects are created based on this template via the new keyword.

Every object has properties and methods corresponding to those of its parent class. Every object instance is completely independent, with its own properties and methods, and can thus be manipulated independently of other objects of the same class.

Here's a simple example of a class definition followed by the object creation.

Example

```
<?php
// Class definition
class greeting{
    // properties
    public $str = "Hello World!";
        // methods
    function show_greeting(){
        return $this->str;
    }
}

// Create object from class
$message = new greeting;
var_dump($message);
?>
```

Tip: The data elements stored within an object are referred to as its properties and the information, or code which describing how to process the data is called the methods of the object.

PHP Resources

A resource is a special variable, holding a reference to an external resource.

Resource variables typically hold special handlers to opened files and database connections.

Example

```
<?php
// Open a file for reading
$handle = fopen("note.txt", "r");
var_dump($handle);
echo "<br>";

// Connect to MySQL database server with default setting
$link = mysql_connect("localhost", "root", "");
var_dump($link);
?>
```

PHP NULL

The special NULL value is used to represent empty variables in PHP. A variable of type NULL is a variable without any data. NULL is the only possible value of type null.

Example

```
<?php
$a = NULL;
var_dump($a);
echo "<br>";

$b = "Hello World!";
$b = NULL;
var_dump($b);
?>
```

When a variable is created without a value in PHP like $var; it is automatically assigned a value of null. Many novice PHP developers mistakenly considered both $var1 = NULL; and $var2 =

""; are same, but this is not true. Both variables are different – the $var1 has null value while $var2 indicates no value assigned to it.

PHP Variables Scope

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

- local
- global
- static

Global and Local Scope

A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function:

Example

```
<?php
$x = 5; // global scope

function myTest() {
   // using x inside this function will generate an error
   echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

```
<?php
function myTest() {
   $x = 5; // local scope
```

```
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

PHP The global Keyword

The global keyword is used to access a global variable from within a function.

To do this, use the global keyword before the variables (inside the function):

Example

```
<?php
$x = 5;
$y = 10;
function myTest() {
    global $x, $y;
    $y = $x + $y;
}
myTest();
echo $y; // outputs 15
?>
```

PHP also stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

The example above can be rewritten like this:

Example

```
<?php
$x = 5;
$y = 10;
```

```
function myTest() {
   $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
?>
```

PHP The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

To do this, use the static keyword when you first declare the variable:

Example

```
<?php
function myTest() {
   static $x = 0;
   echo $x;
   $x++;
}

myTest();
myTest();
myTest();
?>
```

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators

- String operators
- Array operators

PHP Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power (Introduced in PHP 5.6)

Example Program

```
<?php
  // variable 1
$x = 29;
  // variable 2
$y = 4;
  // some arithmetic operations on
// these two variables
echo ($x + $y), "\n";
echo($x - $y), "\n";
echo($x * $y), "\n";
echo($x / $y), "\n";
echo($x % $y), "\n";

?>
```

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the valu of the assignment expression on the right.

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

Example Program

```
<?php
// simple assign operator
$y = 75;
echo $y, "\n";
 // add then assign operator
$y = 100;
$y += 200;
echo $y, "\n";
// subtract then assign operator
$y = 70;
$y -= 10;
echo $y, "\n";
// multiply then assign operator
$y = 30;
$y *= 20;
echo $y, "\n";
  // Divide then assign(quotient) operator
$y = 100;
$y /= 5;
echo $y, "\n";
  // Divide then assign(remainder) operator
$y = 50;
$y %= 5;
echo $y;
?>
```

PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result
==	Equal	\$x == \$y	Returns true if \$x is equal to \$y
===	Identical	\$x === \$y	Returns true if \$x is equal to \$y, and they are of the same type
!=	Not equal	\$x != \$y	Returns true if \$x is not equal to \$y
<>	Not equal	\$x <> \$y	Returns true if \$x is not equal to \$y
!==	Not identical	\$x !== \$y	Returns true if \$x is not equal to \$y, or they are not of the same type
>	Greater than	\$x > \$y	Returns true if \$x is greater than \$y
<	Less than	\$x < \$y	Returns true if \$x is less than \$y
>=	Greater than or equal to	\$x >= \$y	Returns true if \$x is greater than or equal to \$y
<=	Less than or equal to	\$x <= \$y	Returns true if \$x is less than or equal to \$y

Example Program

```
<?php
$a = 80;
$b = 50;
$c = "80";
  // Here var_dump function has been used to
// display structured information. We will learn
// about this function in complete details in further
// articles.
var_dump($a == $c) + "\n";
var_dump($a != $b) + "\n";
var_dump($a <> $b) + "\n";
var_dump($a === $c) + "\n";
var_dump($a !== $c) + "\n";
var_dump($a < $b) + "\n";
var_dump($a > $b) + "\n";
var_dump($a <= $b) + "\n";
var_dump($a >= $b);

  ?>
```

PHP Increment / Decrement Operators

The PHP increment operators are used to increment a variable's value.

The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

Example Program

```
<?php

$x = 2;
echo ++$x, " First increments then prints \n";
echo $x, "\n";

$x = 2;
echo $x++, " First prints then increments \n";
echo $x, "\n";

$x = 2;
echo --$x, " First decrements then prints \n";
echo $x, "\n";

$x = 2;
echo $x--, " First prints then decrements \n";
echo $x;

?>
```

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

Example Program

```
<?php
  $x = 50;
$y = 30;
  if ($x == 50 and $y == 30)
    echo "and Success \n";
  if ($x == 50 or $y == 20)
    echo "or Success \n";
if ($x == 50 xor $y == 20)
    echo "xor Success \n";
  if ($x == 50 && $y == 30)
    echo "&& Success \n";
  if ($x == 50 || $y == 20)
    echo "|| Success \n";
  if (!$z)
    echo "! Success \n";
  ?>
```

PHP String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

Example Program

```
<?php

$x = "Geeks";
$y = "for";
$z = "Geeks!!!";
echo $x . $y . $z, "\n";
  $x .= $y . $z;
echo $x;
?>
```

PHP Array Operators

The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

Example Program

```
<?php
 $x = array("k" => "Car", "l" => "Bike");
$y = array("a" => "Train", "b" => "Plane");
 var_dump($x + $y);
var_dump($x == $y) + "\n";
var_dump($x != $y) + "\n";
var_dump($x <> $y) + "\n";
var_dump($x === $y) + "\n";
var_dump($x !== $y) + "\n";
?>
```

Conditional or Ternary Operators

These operators are used to compare two values and take either of the result simultaneously, depending on whether the outcome is TRUE or FALSE. These are also used as shorthand notation for if...else statement that we will read in the article on decision making.

Syntax:

```
$var = (condition)? value1 : value2;
```

Here, condition will either evaluate to true or false. If the condition evaluates to True, then value1 will be assigned to the variable $var otherwise value2 will be assigned to it.

Operator	Name	Operation
?:	Ternary	If condition is true ? then $x : or else $y. This means that if condition is true then left result of the colon is accepted otherwise the result on right.

Example Program

```
<?php
  $x = -12;
  echo ($x > 0) ? 'The number is positive' : 'The number is negative';
  ?>
```

PHP operator Precedence and associativity

An *expression* is a bit of PHP that can be evaluated to produce a value. The simplest expressions are literal values and variables. A literal value evaluates to itself, while a variable evaluates to the value stored in the variable. More complex expressions can be formed using simple expressions and operators.

An *operator* takes some values (the operands) and does something (for instance, adds them together). Operators are written as punctuation symbols—for instance, the + and - familiar to us from math. Some operators modify their operands, while most do not.

Table summarizes the operators in PHP, many of which were borrowed from C and Perl. The column labeled "P" gives the operator's precedence; the operators are listed in precedence order, from highest to lowest. The column labeled "A" gives the operator's associativity, which can be L (left-to-right), R (right-to-left), or N (non-associative).

PHP operators

P	A	Operator	Operation
19	N	new	Create new object
18	R	[	Array subscript
17	R	!	Logical NOT
	R	~	Bitwise NOT

	R	++	Increment
	R	--	Decrement
	R	(int), (double), (string), (array), (ob ject)	Cast
	R	@	Inhibit errors
16	L	*	Multiplication
	L	/	Division
	L	%	Modulus
15	L	+	Addition
	L	-	Subtraction
	L	.	String concatenation
14	L	<<	Bitwise shift left
	L	>>	Bitwise shift right
13	N	<, <=	Less than, less than or equal
	N	>, >=	Greater than, greater than or equal
12	N	==	Value equality
	N	!=, <>	Inequality
	N	===	Type and value equality
	N	!==	Type and value inequality
11	L	&	Bitwise AND
10	L	^	Bitwise XOR
9	L	|	Bitwise OR
8	L	&&	Logical AND
7	L	||	Logical OR

6	L	?:	Conditional operator
5	L	=	Assignment
	L	+=, -=, *=, /=, .=, %=, &=, |=, ^=, ~=, <<=, >>=	Assignment with operation
4	L	and	Logical AND
3	L	xor	Logical XOR
2	L	or	Logical OR
1	L	,	List separator

POSSIBLE QUESTIONS

Part B (2 Marks)

1. What is PHP?
2. What are the tools required for PHP Programming?
3. Write the syntax of PHP?
4. What are the differences between echo and print in PHP?
5. What is the use of "echo" in php?

Part C (6 Marks)

1. Discuss about PHP inventions and versions.
2. How to declare a variable in PHP? What are the rules to be followed to declare the variable?
3. What is Operator? Explain the operators in PHP with examples.
4. Explain on different types of PHP variables.
5. Illustrate the Pros and Cons of PHP.

KARPAGAM ACADEMY OF HIGHER EDUCATION

DEPARTMENT OF COMPUTER TECHNOLOGY

Subject Name PHP PROGRAMMIN

Class: III B.SC CT

SUBCODE: 16CTU601A

Sem:VI

S. No	Question	Choice1	Choice2	Choice3	Choice4	Answer Key
1	PHP stands for ___________________	Program Holding Process	Hypertext Preprocessor	Personal HypertextProgramming	program process	Hypertext Preprocessor
2	The part that interprets and executes PHP code	zend engine	search engine	web browser	interpreter	zend engine
3	PHP is used for building___________________websites	static	dynamic	none	a&b	dynamic
4	PHP program are run on________________	web browser	interpreter	web server	compiler	web server
5	PHP is devised by_____________	Tim Berners-Lee	Rasmus Lerdorf	Robert Caillau	Richard Fairly	Rasmus Lerdorf
6	PHP programs run via_________	Web browser	interpreter	web server	none of these	Web browser
7	HTML stands for_______________	Home Tool Management Logic	HyperText Managing Language	HyperText Markup Language	none of these	HyperText Markup Language
8	PHP was devised in the year	1996	1990	1986	1994	1994
9	HTML is used for building_______________websites	dynamic	static	both	none	static
10	CLI stands for________	Command Line Interpreter	Common Line Integrator	Common Line Interface	Command Line Interface	Command Line Interface

11	CLI was introduced in____________	PHP4	PHP2	PHP5	none of these	PHP4
12	GTK stands for________________	General Tool Kit	Gnome Track Kit	Gnome Tool Kit	General Tab Kit	Gnome Tool Kit
13	PEAR stands for_____________	PHP Extension and Application Report	PHP Extension and Application Repository	PHP Exchange and Application Report	none of these	PHP Extension and Application Repository
14	MIME stands for__________	Multipurpose Internal Mail Extension	Multiple Interface Multiple Extension	Multiple Interface and Mail Extension	Multipurpose Internet Mail Extension	Multipurpose Internet Mail Extension
15	ODBC stands for_______________	Open Database Connectivity	Object Declaration and Basic Commands	Open Database Control	none of these	Open Database Connectivity
16	_____________ is a programming language designed to work with HTML	c	c++	PHP	COBOL	PHP
17	______________ is the language or format for communications from browser to web	XML	HTTP	HTML	JAVA	HTTP
18	___________provides programmatic functionality with in web pages.	Java script	XML	JSP	none of these	Java script
19	PERL stands for________________	PHP Extension and Reporting Language	Practical Extraction and Reporting Language	PHP Extraction and Resource Language	none of these	Practical Extraction and Reporting Language
20	HTTP stands for___________	Hyper Tool Transfer Protocol	Hypertext Transmission Protocol	Home Tool Transfer Protocol	Hypertext Transfer Protocol	Hypertext Transfer Protocol

21	___________is a method by which results are stored temporarily	Caching	packing	compiling	none of these	Caching
22	RPM stands for ___________	Redhat Protocol Manager	Removable Program Memory	Redhat Package Manager	none of these	Redhat Package Manager
23	___________function is used to send a string value to the browser.	Echo	cookie	cout	none of these	Echo
24	IIS stands for _______________	Information Integrated Service	Internet Information Standard	Internet Information Server	Internal Information Service	Internet Information Server
25	__________ are special characters that indicates where the data starts and stop.	Modifiers	Limiters	Identifiers	Delimiters	Delimiters
26	In PHP code statements end with _________	Colon	Semicolon	Comma	Star operator	Semicolon
27	___________ are enclosed in curly braces.	Functions	Data	Code blocks	none of these	Code blocks
28	________file is parsed when PHP is first loaded and executed.	php.ini	php.exe	php.awt	php.init	php.ini
29	_________are programmatic capabilities that add to or enhance PHP's built-in capabilities.	PHP standard	PHP extension	PHP loader	none of these	PHP standard
30	SAPI stands for ___________	Server And Programmers Interaction	Standard Analysis Program Interface	Server Application Programming Interface	none of these	Server Application Programming Interface
31	PWS stands for___________	Personal Web Service	Programed Window Standard	Personal Web Server	Powered Window Service	Personal Web Server
32	___________ means the processing engine reads the individual commands and checks for syntax errors.	Parsing	Caching	Storing	Checking	Parsing

33	_________ function is used to determine whether a search string exists with in a searched string.	Strstr( )	Chr( )	Strlen( )	Strpos( )	Strpos( )
34	_______function gets any part of a string that is after the first instance of a particular character or string with in a string.	Chr( )	Strstr( )	Strlen( )	Strpos( )	Strstr( )
35	_________function returns a string character value corresponding to the decimal ASCII value entered as the argument.	Chr()	Strstr( )	Strlen()	Strpos( )	Chr()
36	________ function finds the length of a string.	Strstr( )	Chr( )	Strlen()	none of these	Strlen()
37	DSO stands for _________	Digital Service Operator	Dynamic Shared Object	Domain Service Operator	Dynamic Standard Object	Dynamic Shared Object
38	__________can be used between string values to join them together.	Concatenation operator	Star operator	Colon	Plus operator	Concatenation operator
39	SGML stands for ___________	Standard Geometric Manipulation Logic	Static Generalized Making logic	Standard Generalized Markup Language	none of these	Standard Generalized Markup Language
40	PHP automatically makes a few variables called ________	Predefined variables	Defined variables	Library variables	User defined Variables	Predefined variables
41	Predefined variables are also called __________	Global variables	Super variables	Local variables	Superglobal variables	Superglobal variables
42	________ function prints out information about variables	echo	print_r( )	cout	none of these	print_r( )
43	XML stands for _____	External Markup language	Extended Markup Logic	Extended Markup Language	none of these	Extended Markup Language
44	____________ is a series of statements in plain language that are logically	Pseudo code	function code	method code	standard code	Pseudo code

45	__________ is very important to control for structures.	Control logic	Control structures	Boolean logic	none of these	Boolean logic
46	__________ devised Boolean Algebra.	Tim Berners Lee	Robert Boole	Richard Fairly	George Boole	George Boole
47	_________ function is used to view the value of the pointer.	set( )	Current( )	seek( )	tell( )	Current( )
48	_________indicates the element that is currently being used by the script.	object	function	Pointer	none of these	Pointer
49	__________function used to find out index value.	Key( )	index( )	value( )	indexkey( )	Key( )
50	_________ is the another name for index .	Coordinate	key	id	none of these	key
51	________ functions are used to find out the index value of a new element added to an array.	next ( ) and before( )	after( ) and previous( )	next( ) and previous	none of these	next( ) and previous
52	________ functions are used to return the elements in the array that contain data.	list( ) and sort( )	list ( ) and each( )	sort( ) and each( )	none of these	list ( ) and each( )
53	_____function takes the content of the array and sort them in alphabetic order.	Sort( )	list( )	each( )	assort( )	Sort( )
54	_______________takes arrays created with a string index and sort them according to their contents.	sort( )	each( )	asort( )	list( )	asort( )
55	________ returns the appending element in reverse alphabetical order.	sort( ) and rsort( )	sort( ) and list( )	arsort( ) and list( )	rsort( ) and arsort( )	rsort( ) and arsort( )
56	________ sorts the contents of an associated array according to index.	ksort( )	rsort( )	sort( )	list( )	ksort( )
57	_________function sorts multiple arrays or multidimensional arrays.	sort( )	array_multisort( )	list( )	ksort( )	array_multisort( )

58	________ are controls that typically display several items in a list.	Text box	List box	Text area	none of these	List box
59	________ tells the server which page to go to once the user has click the submit button on the form.	Function attribute	Shift attribute	Action attribute	none of these	Action attribute
60	________ attribute controls the way that the information is sends to the server.	Method attribute	Action attribute	Shift attribute	Function attribute	Method attribute

UNIT-II

Handling HTML form with PHP:

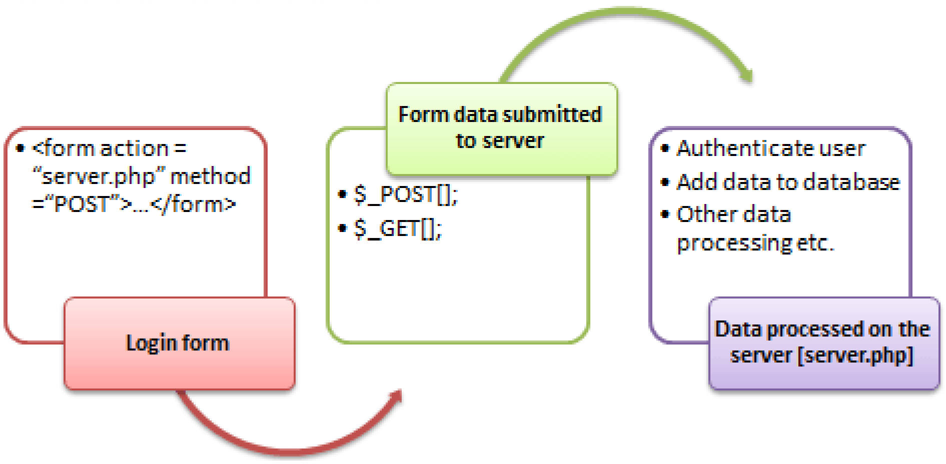
Capturing Form Data -GETand POST form methods- Dealing with multi value fields -Redirecting a form after submission -PHP conditional events and Loops: PHP IF Else conditional statements (Nested IF and Else) -Switch case, while ,For and Do While Loop -Goto , Break ,Continue and exit

Capturing Form Data

Form

Forms are used to get input from the user and submit it to the web server for processing.

The diagram below illustrates the form handling process.


• <form action = "server.php" method ="POST">...</form>
Login form
Form data submitted to server
• $_POST[];
• $_GET[];
• Authenticate user
• Add data to database
• Other data processing etc.
Data processed on the server [server.php]

A form is an HTML tag that contains graphical user interface items such as input box, check boxes radio buttons etc.

The form is defined using the <form>...</form> tags and GUI items are defined using form elements such as input.

Use of Forms

- Forms come in handy when developing flexible and dynamic applications that accept user input.
- Forms can be used to edit already existing data from the database

Example

All Web forms start with an opening <form> tag, and end with a closing </form> tag:

<form action="myscript.php" method="POST">

```
    <!-- Contents of the form go here -->
</form>
```

Form attributes

There are two attributes within the opening <form> tag:

action - tells the browser where to send the form data. This should either be an absolute URL or a relative URL.

method - tells the browser how to send the form data. You can use two methods: GET is for sending small amounts of data and makes it easy for the user to resubmit the form, and POST can send much larger amounts of form data.

PHP Form Text Field

A text input field allows the user to enter a single line of text.

Value attribute

We can optionally prefill the field with an initial value using the value attribute. To leave it blank, specify an empty string for the value attribute, or leave the attribute out altogether.

```
<label for="textField">A text input field</label>
<input type="text" name="textField" id="textField" value="" />
```

Example

The following code is for index.htm file and it has a text field and submit button.

```
<html>
<body>
  <form action="index.php" method="get">
    <input type="text" name="user" />
    <input type="submit" value="hit it!" />
   </form>
</body>
</html>
```

Name the following script as index.php and put it into the same folder as above index.htm file. It accepts the value from the text field by using field name user.

```
<?php
  print "Welcome <b>" . $_GET ['user'] . "</b><br/>";
```

```
?>
```

PHP Form Textarea

A text area field is similar to a text input field, but it allows the user to enter multiple lines of text. Unlike most other controls, an initial value is placed between the <textarea> ... </textarea> tags, rather than in a value attribute.

A textarea element must include attributes for the height of the control in rows (rows) and the width of the control in columns (cols):

```
<label for="textAreaField">A text area field</label>
<textarea name="textAreaField" id="textAreaField" rows="4"  cols="50"></textarea>
```

Example

Name the following script as index.htm. It has a text area and a submit button.

```
<html>
<body>
  <form action="index.php" method="get">
    <textarea name="address" rows="5" cols="40"></textarea>
    <input type="submit" value="hit it!" />
   </form>
</body>
</html>
```

Name the following script as index.php. It reads the value from the textarea from the form above.

```
<?php
  print "Your address is: <br/><b>" . $_GET ['address'] . "</b>";
?>
```

PHP Form CheckBox

A checkbox field is a simple toggle button. It can be either on or off.

The value attribute should contain the value that will be sent to the server when the checkbox is selected. If the checkbox isn't selected, nothing is sent.

```
<label for="checkboxField">A checkbox field</label>
<input type="checkbox" name="checkboxField" id="checkboxField" value="yes" />
```

You can preselect a checkbox by adding the attribute checked="checked" to the input tag:

<input type="checkbox" checked="checked" ... />.

By creating multiple checkbox fields with the same name attribute, you can allow the user to select multiple values for the same field.

Example

The following script is for index.htm. It has several checkboxes.

```
<html>
<body>
<form action ="index.php">

<ul>
  <li><input type ="checkbox" name ="chkFries" value ="11.00">Fries</li>
  <li><input type ="checkbox" name ="chkSoda"  value ="12.85">Soda</li>
  <li><input type ="checkbox" name ="chkShake" value ="1.30">Shake</li>
  <li><input type ="checkbox" name ="chkKetchup" value =".05">Ketchup</li>
</ul>
<input type ="submit">
</form>

</body>
</html>
```

The following code is for index.php and it accepts the value from the checkboxes.

```
<?PHP
print "chkFries:" .  $chkFries . "<br/>";
print "chkSoda:" $chkSoda . "<br/>";
print "chkShake:" . $chkShake . "<br/>";
print "chkKetchup" . $chkKetchup . "<br/>";

$total = 0;

if (!empty($chkFries)){
  print ("You chose Fries <br>");
  $total = $total + $chkFries;
}

if (!empty($chkSoda)){
  print ("You chose Soda <br>");
  $total = $total + $chkSoda;
}

if (!empty($chkShake)){
  print ("You chose Shake <br>");
  $total = $total + $chkShake;
}

if (!empty($chkKetchup)){
  print ("You chose Ketchup <br>");
  $total = $total + $chkKetchup;
}

print "The total cost is \$$total";

?>
```

PHP Form Select

Form Select as Pull-down menu

A pull-down menu allows users to choose a single item from a predefined list of options. The size attribute's value of 1 tells the browser that you want the list to be in a pull-down menu format.

Within the select element, you create an option element for each of your options.

Place the option label between the <option> ... </option> tags.

Each option element can have an optional value attribute, which is the value sent to the server if that option is selected.

If you don't include a value attribute, the text between the <option> ... </option> tags is sent instead:

```
<label for="pullDownMenu">A pull-down menu</label>
<select name="pullDownMenu" id="pullDownMenu" size="1">
```

```
    <option value="option1">Option 1</option>
    <option value="option2">Option 2</option>
    <option value="option3">Option 3</option>
</select>
```

Form Select as List Box

A list box works like a pull-down menu, except that it displays several options at once.

To turn a pull-down menu into a list box, change the size attribute from 1 to the number of options to display at once:

```
<label for="listBox">A list box</label>
<select name="listBox" id="listBox" size="3">
    <option value="option1">Option 1</option>
    <option value="option2">Option 2</option>
    <option value="option3">Option 3</option>
</select>
```

Form Select as multi-select list box

A multi-select list box allows the user to select multiple items at once by holding down Ctrl or Command key.

To turn a normal list box into a multi-select box, add the attribute multiple with a value of "multiple" to the select element.

If the user selects more than one option, all the selected values are sent to the server:

```
<label for="multiListBox">A multi-select list box</label>
<select name="multiListBox" id="multiListBox" size="3"  multiple="multiple">
  <option value="option1">Option 1</option>
  <option value="option2">Option 2</option>
  <option value="option3">Option 3</option>
</select>
```

We can preselect an option in any type of select element by adding the attribute selected="selected" to the relevant <option> tag - for example: <option value="option1" selected="selected">.

Example

Name the following script as index.php. It has a multi-select list box.

```
<html>
<body>
<div>
<form action="index.php" method="post">
<p><select name="products[]" multiple="multiple">
  <option>A</option>
  <option>B</option>
  <option>C</option>
  <option>D</option>
</select></p>
<p><input type="submit" value="hit it!" /></p>
</form>
</div>
</body>
</html>
```

The following script is for index.php.

```
<?php/*from ww w . j ava 2 s  .  c o  m*/
  if (is_array ( $_POST ['products'] )) {
    print "<p>Your product choices are:</p>";
    print "<ul>";
    foreach ( $_POST ['products'] as $value ) {
      print "<li>$value</li>\n";
    }
    print "</ul>";
  }
?>
```

PHP Form RadioButton

The radio buttons are for single choice from multiple options. All radio buttons in the group have the same name attribute.

Only one button can be selected per group. As with checkboxes, use the value attribute to store the value that is sent to the server if the button is selected.

The value attribute is mandatory for checkboxes and radio buttons, and optional for other field types.

```
<label for="radioButtonField1">A radio button field</label>
<input type="radio" name="radioButtonField" id="radioButtonField1"  value="radio1" />
<label for="radioButtonField2">Another radio button</label>
<input type="radio" name="radioButtonField" id="radioButtonField2"  value="radio2" />
```

We can preselect a radio button using the same technique as for preselecting checkboxes.

Example

The following script is for index.htm. It has a group of radio buttons.

```
<html>
 <body>
  <form action="index.php" method="post">
   <b>Please select your favorite color wine:</b> <br>
   <input type="radio" name="color" value="white">  White
   <input type="radio" name="color" value="rose">   Rose
   <input type="radio" name="color" value="red">    Red <br>
   <input type="submit" value="Submit This Form">
 </form>
 </body>
</html>
```

The following script is for index.php. It reads the data from the form above.

```
<?php
  $color = $_POST['color'];
  if( ( $color != null ) )
  {
    $msg .= "a nice $color ";
    echo( $msg );
  }
?>
```

PHP Form Hidden Field

A hidden field is not displayed on the page.

It simply stores the text value specified in the value attribute.

Hidden fields are great for passing additional information from the form to the server.

```
<label for="hiddenField">A hidden field</label>
<input type="hidden" name="hiddenField" id="hiddenField" value="" />
```

Example

The following form uses hidden field to store number of tries.

```
<?php
  $num_to_guess = 42;
  $message = "";
  if (! isset ( $_POST ['guess'] )) {
    $message = "Welcome!";
  } else if ($_POST ['guess'] > $num_to_guess) {
    $message = $_POST ['guess'] . " is too big!";
  } else if ($_POST ['guess'] < $num_to_guess) {
    $message = $_POST ['guess'] . " is too small!";
  } else {
    $message = "Well done!";
  }
  $guess = ( int ) $_POST ['guess'];
  $num_tries = ( int ) $_POST ['num_tries'];
  $num_tries ++;
  ?>
<html>
<body>
 <?php print $message?>
 Guess number: <?php print $num_tries?><br />
<form method="post" action="<?php
```

```
print $_SERVER ['PHP_SELF']?>">
<input type="hidden" name="num_tries"
  value="<?php
  print $num_tries?>" /> Type your guess here: <input type="text" name="guess" value="<?php
  print $guess?>" />
</form>
</body>
</html>
```

PHP Form Validation

Basic input validation in PHP is done using the following functions.

is_string(),
is_numeric(),
is_float(),
is_array(), and
is_object().

Each of these functions take a variable and return true if that variable is of the appropriate type.

The three basic validation checks we should do.

- whether we have required variables,
- whether the variables have a value assigned, and
- whether they have the the type we are expecting.

After the basic check, we can do more check, such as integer value range, the string length, count of array elements, etc.

CTYPE functions

There are eleven CTYPE functions in total. they work in the same way as is_numeric().

Function	**Meaning**
ctype_alnum()	Matches A?Z, a?z, 0?9
ctype_alpha()	Matches A?Z, a?z

Function	Meaning
ctype_cntrl()	Matches ASCII control characters
ctype_digit()	Matches 0?9
ctype_graph()	Matches values that can be represented graphically
ctype_lower()	Matches a?z
ctype_print()	Matches visible characters (not whitespace)
ctype_punct()	Matches all non-alphanumeric characters (not whitespace)
ctype_space()	Matches whitespace (space, tab, new line, etc.)
ctype_upper()	Matches A?Z
ctype_xdigit()	Matches digits in hexadecimal format

```
<?PHP
$var = "123456789a";
print (int)ctype_digit($var);
?>
```

Example

The following code checks whether the $Age variable has a numeric value between 18 and 30.

```
if (isset($Age)) {
      if (is_numeric($Age)) {
            if (($Age > 18) && ($Age < 30)) {
                  print " input is valid";
            } else {
                print "Sorry, you're not the right age!";
            }
        } else {
            print "Age is incorrect!"
        }
} else {
        print "Please provide a value for Age.";
}
```

PHP Form File Upload

A file select field allows the users to choose a file on their hard drive for uploading to the server. The value attribute is usually ignored by the browser:

```
<label for="fileSelectField">A file select field</label>
<input type="file" name="fileSelectField" id="fileSelectField" value="" />
```

Information

PHP creates a superglobal array called $_FILES containing various pieces of information about the file or files.

Each file is described by an element in the $_FILES array keyed on the name of the field that was used to upload the file.

For example, say your form contained a file select field called photo :

```
  <input type="file" name="photo" value="" />
```

If the user uploaded a file using this field, its details would be accessible via the following PHP array element:

```
$_FILES["photo"]
```

This array element is itself an associative array that contains information about the file. For example, you can find out the uploaded file ' s filename like this:

Fields of file information

Here's a full list of the elements stored in each nested array within the $_FILES array:

Array Element	**Description**
name	The filename of the uploaded file.
type	The MIME type of the uploaded file. For example, a JPEG image would probably have a MIME type of image/jpeg , whereas a QuickTime movie file would have a MIME type of video/quicktime.
size	The size of the uploaded file, in bytes.
tmp_name	The full path to the temporary file on the server that contains the uploaded file.
error	The error or status code associated with the file upload.

Error

Error element contains an integer value that corresponds to a built-in constant that explains the error.

Possible values include:

Constant	**Value**	**Meaning**
UPLOAD_ERR_OK	0	The file was uploaded successfully.
UPLOAD_ERR_INI_SIZE	1	The file is bigger than the allowed file size specified in the upload_max_filesize directive in the php.ini file.
UPLOAD_ERR_FORM_SIZE	2	The file is bigger than the allowed file size specified in the MAX_FILE_SIZE

Constant	Value	Meaning
		directive in the form.
UPLOAD_ERR_NO_FILE	4	No file was uploaded.
UPLOAD_ERR_NO_TMP_DIR	6	PHP doesn ' t have access to a temporary folder on the server to store the file.
UPLOAD_ERR_CANT_WRITE	7	The file couldn ' t be written to the server ' s hard disk for some reason.
UPLOAD_ERR_EXTENSION	8	The file upload was stopped by one of the currently loaded PHP extensions.

```
$filename = $_FILES["photo"]["name"];
```

File Size

PHP allows you to limit the size of uploaded files.

First, you can add or edit a directive called upload_max_filesize in the php.ini file:

```
; Maximum allowed size for uploaded files.
upload_max_filesize = 32M
```

If a user tries to upload a file larger than this value (32 megabytes in this example), the file upload is cancelled and the corresponding error array element is set to UPLOAD_ERR_INI_SIZE.

Second, you can add a hidden form field called MAX_FILE_SIZE that specifies the maximum allowed size in bytes of an uploaded file. This should be placed before the file upload field:

```
<input type="hidden" name="MAX_FILE_SIZE" value="10000" />
<input type="file" name="fileSelectField" id="fileSelectField" value="" />
```

If the uploaded file is larger than this figure, the upload is cancelled and the corresponding error array element is set to UPLOAD_ERR_FORM_SIZE.

We can check the size of an uploaded file manually and reject it if it's too large:

```
if ( $_FILES["photo"]["size"]> 10000 ) die( "File too big!" );
```

Move an Uploaded File

The uploaded file is automatically stored in a temporary folder on the server. To use the file, you need to move it out of the temporary folder using PHP's move_uploaded_file() function.

move_uploaded_file() function takes two arguments: the path of the file to move, and the path to move it to.

You can determine the existing path of the file using the tmp_name array element of the nested array inside the $_FILES array.

move_uploaded_file() returns true if the file was moved successfully, or false if there was an error (such as the path to the file being incorrect).

Here's an example:

```
if ( move_uploaded_file( $_FILES["photo"]["tmp_name"], "/home/matt/photos/photo.jpg" ) ) {
  echo "Your file was successfully uploaded.";
} else {
  echo "There was a problem uploading your file - please try again.";
}
```

Example

Here is an example HTML form that allows users to select a file for uploading to your server.

```
<form enctype="multipart/form-data" method="post" action="upload.php">
     Send this file: <input name="userfile" type="file" /><br />
     <input type="submit" value="Send File" />
</form>
```

We give the new file element the name userfile.

If there are file uploads, PHP puts information in the superglobal $_FILES for each one in the form of an array. If you run var_dump() on $_FILES.

```
<?PHP
$filename = $_FILES['userfile']['name'];
$filesize = $_FILES['userfile']['size'];
print "Received $filename  - its size is $filesize";
?>
```

If you find files over a certain size aren't being uploaded properly, you may need to increase the upload_max_filesize setting in your php.ini file.

Example

To uploaded files using move_uploaded_file() function.

The first parameter is the name of the uploaded file. This corresponds to $_FILES['userfile']['tmp_name']. The second parameter is the target file path name.

If all goes well, PHP returns true, and the file will be where you expect it.

Here is the whole operation in action:

```
if (move_uploaded_file($_FILES['userfile']['tmp_name'], "/place/for/file"))
{
    print "Received {$_FILES['userfile']['name']} - its size is {$_FILES['userfile']['size']}";
} else {
    print "Upload failed!";
}
```

Example

The move_uploaded_file() function is the same as the rename() function, plus it checks if the file was just uploaded by the PHP script.

To perform this check yourself, use the is_uploaded_file() function. It returns true if the file was uploaded by the script and false if not.

Here is a simple example:

```
if (is_uploaded_file($somefile)) {
    copy($somefile, "/var/www/userfiles/$somefile");
}
```

Example 4

A File Upload Script

```
    <html>
     <body>
    <?php
     if ( isset( $_POST["sendPhoto"] ) ) {
      processForm();
     } else {
      displayForm();
     }

     function processForm() {
      if ( isset( $_FILES["photo"] ) and $_FILES["photo"]["error"] == UPLOAD_ERR_OK ) {
        if ( $_FILES["photo"]["type"] != "image/jpeg" ) {
          echo "<p>JPEG photos only, thanks!</p>";
        } elseif ( !move_uploaded_file( $_FILES["photo"]["tmp_name"], "photos/" . basename(
$_FILES["photo"]["name"] ) ) ) {
          echo "<p>Sorry, there was a problem uploading that photo.</p>" . $_FILES["photo"]["error"]
;
        } else {
          displayThanks();
        }
      } else {
        switch( $_FILES["photo"]["error"] ) {
          case UPLOAD_ERR_INI_SIZE:
            $message = "The photo is larger than the server allows.";
            break;
          case UPLOAD_ERR_FORM_SIZE:
            $message = "The photo is larger than the script allows.";
            break;
          case UPLOAD_ERR_NO_FILE:
            $message = "No file was uploaded. Make sure you choose a file to upload.";
            break;
          default:
            $message = "Please contact your server administrator for help.";
        }
        echo "<p>Sorry, there was a problem uploading that photo. $message</p>";
   }
  }

  function displayForm() {
  ?>
     <p>Please enter your name and choose a photo to upload, then click
  Send Photo.</p>

     <form action="photo_upload.php" method="post" enctype="multipart/form-data">
       <input type="hidden" name="MAX_FILE_SIZE" value="50000" />

       <label for="visitorName">Your name</label>
       <input type="text" name="visitorName" id="visitorName" value="" />

       <label for="photo">Your photo</label>
       <input type="file" name="photo" id="photo" value="" />

       <input type="submit" name="sendPhoto" value="Send Photo" />
     </form>
  <?php
  }

  function displayThanks() {
  ?>
```

```
        <p>Thanks for uploading your photo
           <?php if ( $_POST["visitorName"] ) echo ", " . $_POST["visitorName"] ?>!
         </p>
        <p>Here's your photo:</p>
        <p><img src="photos/<?php echo $_FILES["photo"]["name"] ?>" alt="Photo"/></p>
      <?php
      }
      ?>


       </body>
      </html>
```

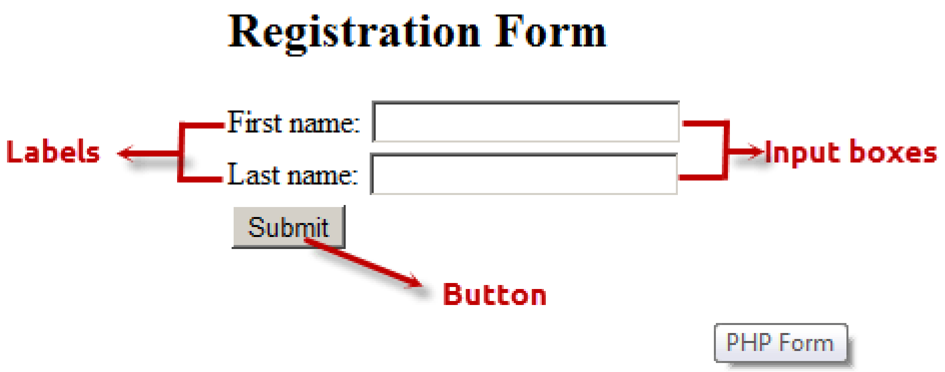
Example Program

```
<form action="someform.php" method="post">
  Name: <input type="text" name="Name" value="Jim" /><br />
  Password: <input type="password" name="Password" /><br />
  Age: <input type="text" name="Age" /><br />
  <input type="submit" />
</form>
```

The code below creates a simple registration form

```
<html>
<head>
            <title>Registration Form</title>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>


   <h2>Registration Form</h2>


   <form action="registration_form.php" method="POST"> First name:
        <input type="text" name="firstname"> <br> Last name:

        <input type="text" name="lastname">

        <input type="hidden" name="form_submitted" value="1" />

        <input type="submit" value="Submit">

    </form>
</body>
</html>
```

Viewing the above code in a web browser displays the following form.



HERE,

- <form...>...</form> are the opening and closing form tags
- action="registration_form.php" method="POST"> specifies the destination URL and the submission type.
- First/Last name: are labels for the input boxes
- <input type="text"...> are input box tags
- <br> is the new line tag
- <input type="hidden" name="form_submitted" value="1"/> is a hidden value that is used to check whether the form has been submitted or not
- <input type="submit" value="Submit"> is the button that when clicked submits the form to the server for processing

Submitting the form data to the server

The action attribute of the form specifies the submission URL that processes the data. The method attribute specifies the submission type.

GET and POST

Superglobal Array

To read the data from a form, we can use the following three superglobal variables.

Superglobal Array	Description
$_GET	Contains all the field names and values sent by a form using the get method
$_POST	Contains all the field names and values sent by a form using the post method
$_REQUEST	Contains the values of both the $_GET and $_POST arrays combined, along with the values of the $_COOKIE superglobal array

- When defining the method to send information to the PHP script, you either use GET or POST. Both send variables across to a script, but they do so in different ways.
- The GET method sends its variables in the web browsers URL, which makes it easy to see and possibly change the information that was sent. So this method should not be used when sending passwords or other sensitive information. It also should not be used for any actions that cause a change in the server, such as placing an order or updating a database. However, because the variables are displayed in the URL, it is possible to bookmark the page.
- The GET method has a limit on the amount of information than can be sent. As a result, if you send long variables using GET, you are likely to lose large amounts of them.
- The POST method sends its variables behind the scenes and has no limits on the amount of information to be sent. Because the variables are not displayed in the URL, it is not possible to bookmark the page.

The GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the ?character.

http://www.test.com/index.htm?name1=value1&name2=value2

It has the following syntax.

```
<?php
$_GET['variable_name'];
?>
```

HERE

The GET method produces a long string that appears in your server logs, in the browser's Location: box.

The GET method is restricted to send upto 1024 characters only.

Never use GET method if you have password or other sensitive information to be sent to the server.

GET can't be used to send binary data, like images or word documents, to the server.

The data sent by GET method can be accessed using QUERY_STRING environment variable.

The PHP provides $_GET associative array to access all the sent information using GET method.

Try out following example by putting the source code in test.php script.

```
<?php
  if( $_GET["name"] || $_GET["age"] ) {
    echo "Welcome ". $_GET['name']. "<br />";
    echo "You are ". $_GET['age']. " years old.";

    exit();
  }
?>
<html>
  <body>

    <form action = "<?php $_PHP_SELF ?>" method = "GET">
      Name: <input type = "text" name = "name" />
      Age: <input type = "text" name = "age" />
      <input type = "submit" />
    </form>

  </body>
</html>
```

It will produce the following result −

The POST Method

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

The POST method does not have any restriction on data size to be sent.

```
<?php
$_POST['variable_name'];
?>
```

The POST method can be used to send ASCII as well as binary data.

The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.

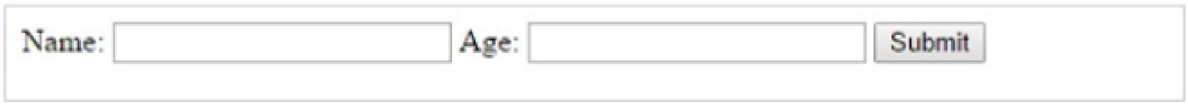
The PHP provides $_POST associative array to access all the sent information using POST method.

Try out following example by putting the source code in test.php script.

```
<?php
  if( $_POST["name"] || $_POST["age"] ) {
    if (preg_match("/[^A-Za-z'-]/",$_POST['name'] )) {
      die ("invalid name and name should be alpha");
    }
    echo "Welcome ". $_POST['name']. "<br />";
    echo "You are ". $_POST['age']. " years old.";

    exit();
  }
?>
<html>
  <body>

    <form action = "<?php $_PHP_SELF ?>" method = "POST">
```

```
        Name: <input type = "text" name = "name" />
        Age: <input type = "text" name = "age" />
        <input type = "submit" />
      </form>

   </body>
</html>
```

It will produce the following result –

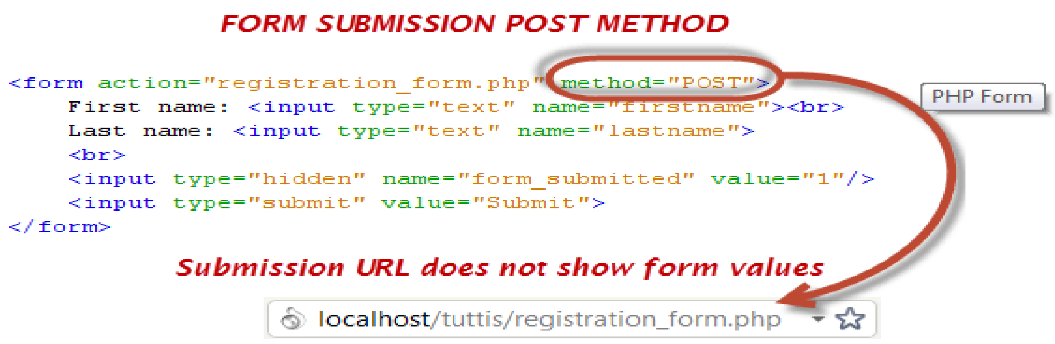


POST Vs GET METHOD

POST	GET
Values not visible in the URL	Values visible in the URL
Has not limitation of the length of the values since they are submitted via the body of HTTP	Has limitation on the length of the values usually 255 characters. This is because the values are displayed in the URL. Note the upper limit of the characters is dependent on the browser.
Has lower performance compared to Php_GET method due to time spent encapsulation the Php_POST values in the HTTP body	Has high performance compared to POST method dues to the simple nature of appending the values in the URL.
Supports many different data types such as string, numeric, binary etc.	Supports only string data types because the values are displayed in the URL
Results cannot be book marked	Results can be book marked due to the visibility of the values in the URL

The below diagram shows the difference between get and post

POST method

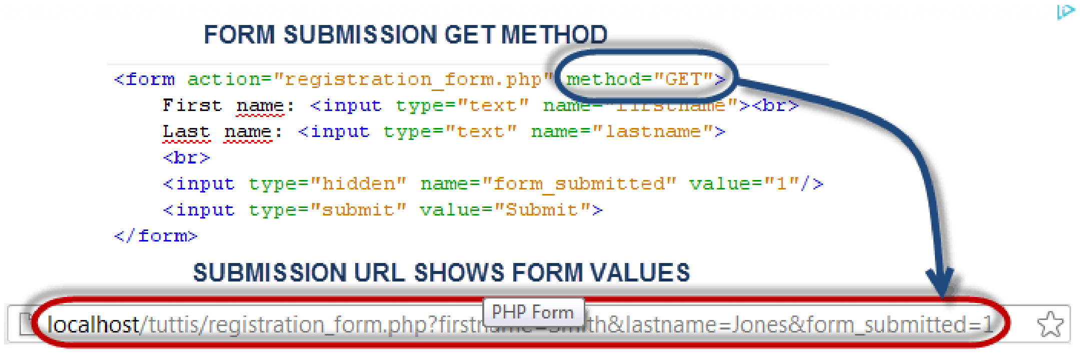

FORM SUBMISSION POST METHOD

```
<form action="registration_form.php" method="POST">
    First name: <input type="text" name="firstname"><br>
    Last name: <input type="text" name="lastname">
    <br>
    <input type="hidden" name="form_submitted" value="1"/>
    <input type="submit" value="Submit">
</form>
```

Submission URL does not show form values

localhost/tuttis/registration_form.php

GET Method


FORM SUBMISSION GET METHOD

```
<form action="registration_form.php" method="GET">
    First name: <input type="text" name="firstname"><br>
    Last name: <input type="text" name="lastname">
    <br>
    <input type="hidden" name="form_submitted" value="1"/>
    <input type="submit" value="Submit">
</form>
```

SUBMISSION URL SHOWS FORM VALUES

localhost/tuttis/registration_form.php?firstname=Smith&lastname=Jones&form_submitted=1

FORM ELEMENT LIST

The following table lists HTML elements in forms.

Element	Description
input type="checkbox"	A checkbox that lets users select multiple options.
input type="file"	A text box plus a button that opens a file selection dialog.
input type="hidden"	A hidden form element.
input type="password"	A password text box.
input type="radio"	A radio button.
input type="reset"	A button to clear the form.
input type="submit"	A button to submit the form.
input type="text"	A text box.
option	An option in a SELECT element.
select	A listbox; can also be a drop-down list box.
textarea	Multiline text box.

DEALING WITH MULTI VALUE FIELDS

- Treat the field as a nested array in the super global arrays
- Form fields can send multiple values, rather than a single value.

Form Select as Pull-down menu

A pull-down menu allows users to choose a single item from a predefined list of options. The size attribute's value of 1 tells the browser that you want the list to be in a pull-down menu format.

Within the select element, you create an option element for each of your options.

To place the option label between the <option> ... </option> tags.

Each option element can have an optional value attribute, which is the value sent to the server if that option is selected.

If you don't include a value attribute, the text between the <option> ... </option> tags is sent instead:

<label **for="pullDownMenu"**>A pull-down menu</label>

```
<select name="pullDownMenu" id="pullDownMenu" size="1">
   <option value="option1">Option 1</option>
   <option value="option2">Option 2</option>
   <option value="option3">Option 3</option>
</select> //j  av  a 2  s . com
```

Form Select as List Box

A list box works like a pull-down menu, except that it displays several options at once.

To turn a pull-down menu into a list box, change the size attribute from 1 to the number of options to display at once:

```
<label for="listBox">A list box</label>
<select name="listBox" id="listBox" size="3">
   <option value="option1">Option 1</option>
   <option value="option2">Option 2</option>
   <option value="option3">Option 3</option>
</select> /*j a  v a 2s.  c  o m*/
```

Form Select as multi-select list box

A multi-select list box allows the user to select multiple items at once by holding down Ctrl or Command key.

To turn a normal list box into a multi-select box, add the attribute multiple with a value of "multiple" to the select element.

If the user selects more than one option, all the selected values are sent to the server:

```
<label for="multiListBox">A multi-select list box</label>
<select name="multiListBox" id="multiListBox" size="3"  multiple="multiple">
  <option value="option1">Option 1</option>
  <option value="option2">Option 2</option>
  <option value="option3">Option 3</option>
</select> /*from  j ava2 s .  c  o m*/
```

You can preselect an option in any type of select element by adding the attribute selected="selected" to the relevant <option> tag - for example: <option value="option1" selected="selected">.

Example

Name the following script as index.htm. It has a multi-select list box.

```
<html>/*  j a v  a 2  s.  c  o m*/
<body>
<div>
<form action="index.php" method="post">
<p><select name="products[]" multiple="multiple">
  <option>A</option>
  <option>B</option>
  <option>C</option>
  <option>D</option>
</select></p>
<p><input type="submit" value="hit it!" /></p>
</form>
</div>
</body>
</html>
```

The following script is for index.php.

```
<?php/*from java 2  s.  com*/
  if (is_array ( $_POST ['products'] )) {
    print "<p>Your product choices are:</p>";
    print "<ul>";
    foreach ( $_POST ['products'] as $value ) {
      print "<li>$value</li>\n";
    }
    print "</ul>";
  }
?>
```

A checkbox field is a simple toggle button. It can be either on or off.

The value attribute should contain the value that will be sent to the server when the checkbox is selected. If the checkbox isn't selected, nothing is sent.

```
<label for="checkboxField">A checkbox field</label>
```

```
<input type="checkbox" name="checkboxField" id="checkboxField" value="yes" />
```

Preselect a checkbox by adding the attribute checked="checked" to the input tag:

```
<input type="checkbox" checked="checked" ... />.
```

By creating multiple checkbox fields with the same name attribute, you can allow the user to select multiple values for the same field.

Example

The following script is for index.htm. It has several checkboxes.

```
<html>
<body>
<form action ="index.php">

<ul>
  <li><input type ="checkbox" name ="chkFries" value ="11.00">Fries</li>
  <li><input type ="checkbox" name ="chkSoda"  value ="12.85">Soda</li>
  <li><input type ="checkbox" name ="chkShake" value ="1.30">Shake</li>
  <li><input type ="checkbox" name ="chkKetchup" value =".05">Ketchup</li>
</ul>
<input type ="submit">
</form>

</body>
</html>
```

The following code is for index.php and it accepts the value from the checkboxes.

```
<?PHP/*java2 s  .c  om*/
print "chkFries:" .  $chkFries . "<br/>";
print "chkSoda:" $chkSoda . "<br/>";
print "chkShake:" . $chkShake . "<br/>";
print "chkKetchup" . $chkKetchup . "<br/>";

$total = 0;
```

```
if (!empty($chkFries)){
  print ("You chose Fries <br>");
  $total = $total + $chkFries;
}

if (!empty($chkSoda)){
  print ("You chose Soda <br>");
  $total = $total + $chkSoda;
}

if (!empty($chkShake)){
  print ("You chose Shake <br>");
  $total = $total + $chkShake;
}

if (!empty($chkKetchup)){
  print ("You chose Ketchup <br>");
  $total = $total + $chkKetchup;
}

print "The total cost is \$$total";

?>
```

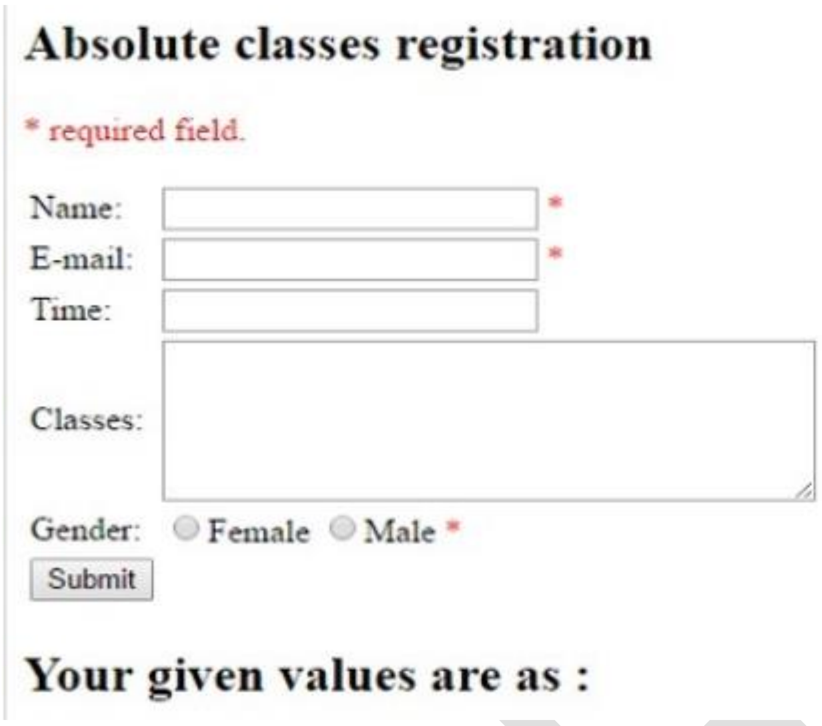
For Example Program

```
<html>
      <head>
      <style>
         .error {color: #FF0000;}
      </style>
   </head>

   <body>
```

```
<?php
 // define variables and set to empty values
 $nameErr = $emailErr = $genderErr = $websiteErr = "";
 $name = $email = $gender = $comment = $website = "";

 if ($_SERVER["REQUEST_METHOD"] == "POST") {
   if (empty($_POST["name"])) {
     $nameErr = "Name is required";
   }else {
     $name = test_input($_POST["name"]);
   }

   if (empty($_POST["email"])) {
     $emailErr = "Email is required";
   }else {
     $email = test_input($_POST["email"]);

     // check if e-mail address is well-formed
     if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
       $emailErr = "Invalid email format";
     }
   }

   if (empty($_POST["website"])) {
     $website = "";
   }else {
     $website = test_input($_POST["website"]);
   }

   if (empty($_POST["comment"])) {
     $comment = "";
   }else {
     $comment = test_input($_POST["comment"]);
   }

   if (empty($_POST["gender"])) {
     $genderErr = "Gender is required";
   }else {
     $gender = test_input($_POST["gender"]);
   }
 }

 function test_input($data) {
   $data = trim($data);
   $data = stripslashes($data);
   $data = htmlspecialchars($data);
   return $data;
 }
?>

<h2>Absolute classes registration</h2>

<p><span class = "error">* required field.</span></p>

<form method = "post" action = "<?php
 echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
 <table>
   <tr>
     <td>Name:</td>
     <td><input type = "text" name = "name">
       <span class = "error">* <?php echo $nameErr;?></span>
     </td>
   </tr>
```

```
<tr>
  <td>E-mail: </td>
  <td><input type = "text" name = "email">
    <span class = "error">* <?php echo $emailErr;?></span>
  </td>
</tr>

<tr>
  <td>Time:</td>
  <td> <input type = "text" name = "website">
    <span class = "error"><?php echo $websiteErr;?></span>
  </td>
</tr>

<tr>
  <td>Classes:</td>
  <td> <textarea name = "comment" rows = "5" cols = "40"></textarea></td>
</tr>

<tr>
  <td>Gender:</td>
  <td>
    <input type = "radio" name = "gender" value = "female">Female
    <input type = "radio" name = "gender" value = "male">Male
    <span class = "error">* <?php echo $genderErr;?></span>
  </td>
</tr>

<td>
  <input type = "submit" name = "submit" value = "Submit">
</td>
```

```
      </table>

    </form>

    <?php
      echo "<h2>Your given values are as:</h2>";
      echo $name;
      echo "<br>";

      echo $email;
      echo "<br>";

      echo $website;
      echo "<br>";

      echo $comment;
      echo "<br>";

      echo $gender;
    ?>

  </body>
</html>
```


Absolute classes registration

* required field.

Name:	*
E-mail:	*
Time:	
Classes:	
Gender:	◯ Female ◯ Male *

Your given values are as :

PHP REDIRECT AFTER A FORM SUBMISSION

- Redirection is done by outputting a Location using PHP using the header() function.
- Here's how to redirect to a page called thanks.html:
- header("Location: thanks.html");
- Don't output any content to the browser via echo() or print(), or by including HTML markup outside the <?php ... ?> tags before calling header().

When a browser receives information from the web server it receives an HTTP header (which the header() function outputs the redirect to), and then outputs the body of the HTTP response after the HTTP header.

```
<?php
// Process form
// If processing was successful, redirect
if($success)
{
// Replace your redirection URL here
   header("Location: http://www.coolcakes.net/");
}
```

```
?>
```

Example

Here's a quick example of a form handler script that redirects to a thank - you page:

```
<?php
   if ( isset( $_POST["submitButton"] ) ) {
     // (deal with the submitted fields here)
     header( "Location: thanks.html" );
     exit;
   } else {
     displayForm();
   }
   function displayForm() {
   ?>
   <html>
     <body>
       <form action="index.php" method="post">
           <label for="firstName">First name</label>
           <input type="text" name="firstName" id="firstName" value="" />
           <label for="lastName">Last name</label>
           <input type="text" name="lastName" id="lastName" value="" />
           <input type="submit" name="submitButton" id="submitButton" value= "Send Details" />
       </form>
     </body>
   </html>
   <?php
   }
   ?>
```

PHP Conditional Statements

Conditional statements are used to perform different actions based on different conditions.

In PHP we have the following conditional statements:

1. if statement - executes some code if one condition is true

2. if...else statement - executes some code if a condition is true and another code if that condition is false
3. if...elseif....else statement - executes different codes for more than two conditions
4. switch statement - selects one of many blocks of code to be executed

PHP - The if Statement

The if statement executes some code if one condition is true.

Syntax

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

The example below will output "Have a good day!" if the current time (HOUR) is less than 20:

Example

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
}
?>
```

PHP - The if...else Statement

The if....else statement executes some code if a condition is true and another code if that condition is false.

Syntax

```
if (condition) {
    code to be executed if condition is true;
} else {
    code to be executed if condition is false;
}
```

The example below will output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

Example

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

PHP - The if...elseif....else Statement

The if....elseif...else statement executes different codes for more than two conditions.

Syntax

```
if (condition) {
    code to be executed if this condition is true;
} elseif (condition) {
    code to be executed if this condition is true;
} else {
    code to be executed if all conditions are false;
}
```

The example below will output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

Example

```
<?php
$t = date("H");

if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

PHP - The switch Statement

The switch statement is used to perform different actions based on different conditions.

The PHP switch Statement

Use the switch statement to select one of many blocks of code to be executed.

Syntax

```
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;
}
```

This is how it works: First we have a single expression n (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use break to prevent the code from running into the next case automatically. The default statement is used if no match is found.

Example

```
<?php
$favcolor = "red";

switch ($favcolor) {
   case "red":
      echo "Your favorite color is red!";
      break;
   case "blue":
      echo "Your favorite color is blue!";
      break;
   case "green":
      echo "Your favorite color is green!";
      break;
   default:
      echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

PHP Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal code-lines in a script, we can use loops to perform a task like this.

In PHP, we have the following looping statements:

1. while - loops through a block of code as long as the specified condition is true
2. do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true
3. for - loops through a block of code a specified number of times
4. foreach - loops through a block of code for each element in an array

The PHP while Loop

The while loop executes a block of code as long as the specified condition is true.

Syntax

```
while (condition is true) {
    code to be executed;
}
```

The example below first sets a variable $x to 1 ($x = 1). Then, the while loop will continue to run as long as $x is less than, or equal to 5 ($x <= 5). $x will increase by 1 each time the loop runs ($x++):

Example

```
<?php
$x = 1;

while($x <= 5) {
    echo "The number is: $x <br>";
    $x++;
}
?>
```

do...while Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax

```
do {
    code to be executed;
} while (condition is true);
```

The example below first sets a variable $x to 1 ($x = 1). Then, the do while loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

Example

```
<?php
$x = 1;
do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

do while loop the condition is tested AFTER executing the statements within the loop. This means that the do while loop would execute its statements at least once, even if the condition is false the first time.

The example below sets the $x variable to 6, then it runs the loop, and then the condition is checked:

Example

```
<?php
$x = 6;
do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

The PHP for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (init counter; test counter; increment counter) {
    code to be executed;
}
```

Parameters:

- init counter: Initialize the loop counter value
- test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- increment counter: Increases the loop counter value
- The example below displays the numbers from 0 to 10:

Example

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

PHP foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

```
foreach ($array as $value) {
    code to be executed;
}
```

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

The following example demonstrates a loop that will output the values of the given array ($colors):

Example

```
<?php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
    echo "$value <br>";
}
?>
```

POSSIBLE QUESTIONS

Part B (2 Marks)

1. Difference between GET and POST Methods.
2. What are form elements PHP?
3. List out the multivalue fields in PHP.
4. Compare While and for while loop.
5. What it Goto and Break Statement.

Part C (6 Marks)

1. How to handle HTML forms in PHP with example.
2. Describe on multi value fields in PHP
3. How to redirect a form after submission with example program.
4. Explain in detail about conditional statements in PHP with example.
5. Discuss on different types of looping statements in PHP.

KARPAGAM ACADEMY OF HIGHER EDUCATION
DEPARTMENT OF COMPUTER TECHNOLOGY

Subject Name PHP PROGRAMMING

Class: III B.SC CT

SUBCODE: 16CTU601A **Sem:VI**

S.No	Question	Choice1	Choice2	Choice3	Choice4	Answer Key
1	_____ is the first step of debugging.	compiling	testing	analyzing	processing	testing
2	_____ occurs when your program produces an incorrect response or answer.	logic error	syntax error	fatal error	semantic error	logic error
3	____ sets the display of error messages to the screen on or off.	logic errors	view_errors	display_errors	Compling	display_errors
4	Runtime errors that stop your program from completing execution are termed ____	fatal	local	runtime error	bugs	fatal
5	_____ is used to set values to variables	=	= =	>=	<=	=
6	_____ is used for comparing values.	=	>=	= =		= =
7	____ function adds slashes wherever it finds string characters.	stripslashes()	addslashes()	putslashes()	string_slashes()	addslashes()
8	____occur when you leave out a semicolon.	semantic error	logic error	syntax error	notice error	syntax error
9	Syntax error is also known as	parse error	report error	notice error	logic error	parse error
10	Square bracket surrounding a pattern of character is called a ____	string class	word class	pattern class	character class	character class
11	URL stands for____	user resource locator	universal resource loader	uniform resource locator	uniform resource location	uniform resource locator
12	ISO stands for____	International standard organization	Indian standard organization	International service organization	Indian standard oganization	International standard organization
13	RAD stands for _____	Random Access Devices	Rapid Application Design	Rapid Application Device	Random Identifier	Rapid Application Design
14	_____ is used to comment a single line of code.	/	/*	**	//	//
15	____ is used to comment a block of code.	/* and */	// and //	* and *	//* and *//	/* and */
16	All data submitted in a browser to your web server is formatted as ____	numbers	character	strings	boolean	strings
17	Trouble shooting works for both ____ & ____ errors.	syntax & runtime	runtime & logic	syntax & logic	semantic & syntax	syntax & logic
18	____ gradually eliminates potential causes of problems until you have found the right	fatal	qualifier	boundaries	trouble shooting	trouble shooting
19	Serious errors cause PHP to simply quit processing, displaying a ______ message on the screen	Notice error	fatal error	warning error	flash	fatal error
20	Errors that are not quite serious may cause ____ message to be displayed.	flash	list error	blinking	warning error	warning error
21	The most recent error messages is available by enabling _____	recent_error	find_error	track_error	built_error	track_error

22	Notice error include _____ and user generated notices.	labels	runtime notices	error report	flash	runtime notices
23	PHP5 has _____ function to handle errors.	error()	track_error	try/catch	throw()	try/catch
24	In PHP _____ displays the value on the screen.	print	echo	disp()	cout	echo
25	____ statement ends all processing.	Exit	End	Break	Stop	Exit
26	____ function changes HTML tags into special characters.	special char()	convert()	HTMLspecial chars()	char_set()	HTMLspecial chars()
27	____functions are quantum leap more powerful when it comes to manipulating data.	Manip()	Regular expression	Compound expression	Exchg()	Regular expression
28	The _____ function is used to look for a string within a string.	substr()	find()	search()	strstr()	strstr()
29	____ function is used to separate out data values in a string.	extract()	exploded()	remove()	find()	exploded()
30	____ are like mini programming language for creating very powerful patterns.	regexps	minexp	exps	cmpexp	regexps
31	PHP's regular expression functions that allow Perl notation are called ____ functions.	PREXP	PCRE	PHPEXP	EXGP	PCRE
32	____ is used to store successfully matched expressions.	exp()	egep()	match()	ereg()	ereg()
33	The symbols that can be used to indicate the location on the string where the match must occur	locators	matcher	anchors	provider	anchors
34	____ anchor appears at the beginning of the pattern anchoring a match to the beginning of the string.	^	&	@	$	^
35	____ anchor appears at the end of the pattern anchoring a match to the end of the string.	#	$	?	%	$
36	When the words may be preceded or followed by a variety of punctuation marks, there are special symbols called ____	delimiters	qualifier	word boundaries	special chars	word boundaries
37	____ operator in regular expression is same as bitwise "or" operator.	either-or	||	OR	AND	either-or
38	____ are used to set limits and ranges on the quantity of characters to be matched.	Delimiter	Match()	Quantifier	Boundaries	Quantifier
39	____ meta character shows the meaning that any one character other than a, b, or c.	[a^b^c]	[^abc]	[a||b||c]	[abc^]	[^abc]
40	____meta character shows a word character.	\c	\wc	\w	\m	\w
41	_____ meta character shows a non-digit.	\D	\d	\nd	/d	\D
42	_____ meta character shows any character.	*	/	.	^	.
43	\s is used to indicate that it is a _____	string	white space	shift	control	white space
44	The meta character \d means that it is a _____	digit	date	non-dight	double	digit
45	_____ function is used to find out index value.	val()	index()	key()	ereg()	key()
46	The error_log function can take up to _____ arguments.	three	five	one	four	four

47	The function transfers any argument values into new variables called ______	parameters	qualifier	meta character	meta data	parameters
48	_____ keyword may be used to pass values back out to the calling code after data processing is complete inside the function.	return	break	carry	pass	return
49	Multiple parameters are separated by _____	semi colon	colon	slashes	commas	commas
50	Calling a function from within itself is known as _____	recursion	looping	branching	nesting	recursion
51	The process of creating and calling functions within functions is known as _____	recursion	nesting	branching	looping	nesting
52	_____ is used to bring external files into the current scripts and run them.	import	accept	extern	include	include
53	A failure of require results in a _____	fatal error	logic error	warning	syntax error	fatal error
54	A failure of include results in a _____	syntax error	fatal error	flash error	warning	warning
55	_____ variables are created outside a function and remain alive until the script ends.	local	global	static	char	global
56	_____ are created inside a function.	fatal	global	local	logical	local
57	The name of the global variable is preceded by _____	underscore	backslash	pound	dot	underscore
58	If we leave out an argument, the function will automatically assume a _____ for numeric argument	three	zero	two	one	zero
59	_____ function is used to determine whether the form has been submitted.	submit()	is_set()	isset()	return()	isset()
60	Inside the function a _____ loop is used to iterate through the fieldnames and values and perform the appropriate processing on them.	foreach	while	for	switch	foreach

UNIT-III

PHP Functions:

Function, Need of Function, declaration and calling of a function-PHP Function with arguments, Default Arguments in Function-Function argument with call by value, call by reference-Scope of Function Global and Local

Function -Introduction

Functions are basically named scripts that can be called upon from any other script to perform a specific task. Values (known as arguments) can be passed into a function so that they can be used in the function script, and functions can, in turn, return results to the location from which they were called.

Need of PHP function

- Better code organization – functions allow us to group blocks of related code that perform a specific task together.
- Reusability – once defined, a function can be called by a number of scripts in our PHP files. This saves us time of reinventing the wheel when we want to perform some routine tasks such as connecting to the database
- Easy maintenance- updates to the system only need to be made in one place.

There are two basic types of functions. Built-in functions and user defined ones. The built-in functions are part of the PHP language.

User Defined Functions

Besides the built-in PHP functions, we can create our own functions.

1. A function is a block of statements that can be used repeatedly in a program.
2. A function will not execute immediately when a page loads.
3. A function will be executed by a call to the function.

Declaration and Calling of a Function-in PHP

A user-defined function declaration starts with the word function:

Syntax

```
function functionName() {
    code to be executed;
}
```

The declaration of a user-defined function start with the word function, followed by the name of the function you want to create followed by parentheses i.e. () and finally place your function's code between curly brackets {}.

Note: A function name can start with a letter or underscore (not a number).

Example

```
<?php
function writeMsg() {
    echo "Hello world!";
}
writeMsg(); // call the function
?>
```

Calling PHP Functions

PHP functions are called by using the name declared when the function was defined, together with any values that need to be passed through as parameters. The following example both defines and then calls our addNumbers() function:

```
<?php
function addNumbers ($arg1, $arg2)
{
     return $arg1 + $arg2;
}
```

```
$var1 = 10;
$var2 = 20;
print addNumbers( $var1, $var2);
?>
```

When loaded into a browser the print statement will display the result returned by the addNumbers() function, in this case the number 30.

PHP Function Arguments

Information can be passed to functions through arguments. An argument is just like a variable.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

Example

```
<?php
function familyName($fname) {
    echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

The following example has a function with two arguments ($fname and $year):

```
<?php
function familyName($fname, $year) {
    echo "$fname Refsnes. Born in $year <br>";
}
familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

PHP Default Argument Value

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

Example

```
<?php
function setHeight($minheight = 50) {
    echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

Function argument with call by value

Call by value means passing the value directly to a function. The called function uses the value in a local variable; any changes to it **do not** affect the source variable.

Example

This is a "global", or automatic global, variable. This simply means that it is available in all scopes throughout a script. There is no need to do global $x; to access it within functions or methods.

```
<?php
//Call by value program
function abc($x)
{
$x=$x+10;
return($x);
}
$a=20;
echo abc($a)."<br>";
echo ($a);
?>
```

Note: Call by value: in the call by value method, the called function creates a new set of variables and copies the values of arguments into them.

Function argument with call by reference

Call by reference means passing the address of a variable where the actual value is stored. The called function uses the value stored in the passed address; any changes to it **do** affect the source variable.

```
<?php
//call by reference program in php
function abc($x)
{
$x=$x-10;
return($x);
}
$a=50;
```

```
echo abc($a)."<br>";
echo ($a);
?>
```

Note: Call by reference: in the call by reference method, instead of passing a value to the function being called a reference/pointer to the original variable is passed.

SCOPE OF FUNCTION GLOBAL AND LOCAL

1. Variables declared outside of functions and classes are global. Global variables are available else where in the script.
2. Function variables are self-contained and do not affect variables in the main script.
3. Variables from the main script are not implicitly made available inside functions.

Example

Take a look at this example:

```
<?PHP
function foo() {
  $bar = "java2s.com";
}
$bar = "PHP";
foo();
print $bar;
?>
```

The code above generates the following result.

PHP

- Execution of the script starts at the $bar = "PHP" line, and then calls the foo() function.
- foo() sets $bar to java2s.com, then returns control to the main script where $bar is printed out.
- Function foo() is called, and, having no knowledge that a $bar variable exists in the global scope, creates a $bar variable in its local scope.

- Once the function ends, all local scopes are gone, leaving the original $bar variable intact.

PHP Global Variables

A global variable can be accessed anywhere in your script, whether inside or outside a function.

In PHP, all variables created outside a function are, in a sense, global in that they can be accessed by any other code in the script that's not inside a function.

To use such a variable inside a function, write the word global followed by the variable name inside the function ' s code block.

```
<?PHP
        $myGlobal = "Hello there!";

        function hello() {
          global $myGlobal;
          echo "$myGlobal\n";
        }
        hello(); // Displays "Hello there!"
?>
```

The code above generates the following result.

```
Hello there!
```

hello() function accesses the $myGlobal variable by declaring it to be global using the global statement. The function can then use the variable to display the greeting.

Example 1

Created a variable outside a function to use it as a global variable. Take a look at the following script:

```
<?PHP
        function setup() {
         global $myGlobal;
         $myGlobal = "Hello there!";
        }
        function hello() {
          global $myGlobal;
          echo "$myGlobal\n";
        }
        setup();
        hello(); // Displays "Hello there!"
?>
```

The code above generates the following result.

Hello there!

In this script, the setup() function is called first. It declares the $myGlobal variable as global, and gives it a value.

Then the hello() function is called. It too declares $myGlobal to be global, which means it can now access its value previously set by setup() and display it.

Example 2

The $GLOBALS array can access global variables within functions. All variables declared in the global scope are in the $GLOBALS array, which you can access anywhere in the script. Here is a demonstration:

```
<?PHP
function foo() {
  $GLOBALS['bar'] = "java2s.com";
}

$bar = "PHP";
foo();
print $bar;
?>
```

The code above generates the following result.

java2s.com

We can read variables in the same way:

```
$localbar = $GLOBALS['bar'];
```

PHP GLOBAL keyword allow a variable to be accessed locally.

```
function myfunc() {
    GLOBAL $foo, $bar, $baz;
    ++$baz;
}
```

The code above reads the global variables $foo, $bar, and $baz. The ++$baz line will increment $baz by 1, and this will be reflected in the global scope.

Note

We can also declare more than one global variable at once on the same line, just separate the variables using commas:

```
function myFunction() {
  global $oneGlobal, $anotherGlobal;
}
```

Be careful with global variables. If you modify the value of a global variable in many different places within your application, it can make it hard to debug your code.

POSSIBLE QUESTIONS

Part B (2 Marks)

1. Define Function.
2. How to declare the function?
3. Compare between Call by value and Call by reference.
4. What is the use of function?
5. Write short notes on scope of function.

Part C (6 Marks)

1. How to declaration and calling of a function in PHP? Explain it.
2. Explain in detail about PHP Function with arguments with example program.
3. Discuss about Default Arguments in Function with example.
4. Describe on Function argument with call by value, call by reference
5. Elucidate on Scope of Function in Global and Local.

KARPAGAM ACADEMY OF HIGHER EDUCATION

DEPARTMENT OF COMPUTER TECHNOLOGY

Subject Name PHP PROGRAMMING

Class: III B.SC CT

SUBCODE: 16CTU601A **Sem:VI**

S.No	Question	Choice1	Choice2	Choice3	Choice4	Answer Key
1	___ is an ordered sequence of bytes stored on hard disk , floppy disk and other	Memory	file	record	disk	file
2	___________is a special type of file that holds the names of other files and directories.	file	memory	directory	disk	directory
3	A file handle is simply an _______ value that is used to identify the file.	integer	double	floating point	string	integer
4	________ function , which used to read data from a file and takes a string argument that holds the path to the file.	file( )	fopen( )	a+	r+	file( )
5	________ is used to open a file.	file()	fopen( )	a+	r+	fopen( )
6	___________ value is used to open file for reading & writing.	file()	fopen( )	a+	r+	r+
7	a+ value is used to open file for ________ &_________.	reading & appending	writing & reading	appending & deleting	writing&appending	reading & appending
8	PHP include the ______function to enable us to capture information about a file by providing the filename as an argument to the function.	stat( )	sort( )	fread( )	open()	stat( )
9	____ function returns an indexed array that contain file statistic and information within each spot in the array.	fread( )	stat( )	fget c( )	sort( )	stat( )
10	__________ function can be used to extract a character string from a file.	fread( )	stat( )	fget c( )	sort( )	fread( )
11	____________ function can be used to read from file one character at a time.	fread( )	stat( )	fget c( )	sort( )	fget c( )
12	__________ function must be given a valid file handle a numerical value higher than the length of each line.	fgetcsv( )	fread( )	stat( )	fwrite( )	fgetcsv( )
13	fputs( ) function is simply an alias for ____________.	fgetcsv( )	fread( )	stat( )	fwrite( )	fwrite( )
14	____________ takes only one argument.	file( ) function	fwrite( ) function	fread( ) function	file mtime( )	file( ) function
15	___________ function is the one to use if all you want to do is read and print the entire file to the web browser.	fpassthru( )	file ctime( )	filemtime( )	fwrite( ) function	fpassthru( )
16	___________ returns the time at which the file was last changed as a UNIX timestamp.	Fpassthru( )	filectime( )	filemtime( )	fwrite( ) function	filectime( )
17	_______ returns the time at which the file was last modified as a UNIX timestamp.	Fpassthru( )	filectime( )	filemtime( )	fread( ) function	filemtime( )
18	_________returns the user ID of the owner of the specified file.	fileowners( )	filegroup( )	file type( )	fwrite( ) function	fileowners( )
19	________returns the group ID of the owner of the specified file.	fileowners( )	filegroup( )	file type( )	fwrite( ) function	filegroup( )
20	__________ returns the type of the specified file.	fileowners( )	filegroup( )	file type( )	fwrite( ) function	file type( )
21	_________ is designed specifically to work with directories.	is_dir( )	is_file()	dir	file	is_dir( )
22	_______ returns true if the given file name refers to a regular file.	is_dir( )	is_file( )	dir	file	is_file( )
23	PHP also enables you to _______ ,________&______ files.	copy ,rename ,delete	copy ,paste, delete	cut ,copy, paste	edit, copy, paste	copy ,rename ,delete
24	_________ function is used to rename a file.	rename( )	rewind dir( )	rmdir( )	chdir( )	rename( )

25	________ function takes single string argument referring to a name of a file you want to delete.	unlink( )	sort( )	fread( )	delete()	unlink( )
26	Many versions of PHP for windows donot support the _______ function.	sort( )	unlink( )	fread( )	delete()	unlink( )
27	________ is called to arrange the array entries in ascending order, and display are accept the current & parent directories.	unlink( )	chdir( )	sort	dir( )	sort
28	_______ function resets PHP internal pointer when you want to move back to the first entry in a given directory while working with it.	mkdir( )	chdir( )	rewinddir( )	rename( )	rewinddir( )
29	_______ function call changes PHP's current directory to the given directory.	mkdir( )	chdir( )	rewinddir( )	rename( )	chdir( )
30	_________ function removes a given directory.	rmdir( )	chdir( )	rewinddir( )	rename( )	rmdir( )
31	The ___________ function creates a directory as specified in its first argument.	rename( )	rmdir( )	chdir( )	mkdir( )	mkdir( )
32	The _________ function returns the directory part of a given filename.	dirname( )	rmdir( )	chdir( )	mkdir( )	dirname( )
33	__________ & _________ are the two properties provided by the dir object.	read( ) & write( )	handle & path	path & time	path&date	handle & path
34	____________ are the three methods supported by the dir object.	read( ),rewind( ) &open	open( ),write( )&close( )	open( ),read( ) & close	read( ),rewind( ) &close( )	read( ),rewind( ) &close( )
35	The _________ function is based on the concept of recursion & traverses the whole directory hierarchy under a specified directory.	recursion_dir( )	is_dir( )	traverse_dir( )	dir()	traverse_dir( )
36	________ function print out the contents of a given file in a new window.	display( )	show()	print	project	display( )
37	_________ are the three different levels of visibility that a member valuable or method can have.	class, object &function	public, private &protected	static, local &global	static,numaric&integer	public, private &protected
38	By creating a special function called ______ we can perform any activities required to instantiate the object.	_construct( )	_set()	_this()	_any()	_construct( )
39	The constructor opens a connection to the database and stores the resource handle in ________.	$this->-hdb	_construct( )	_set( )	_this()	$this->-hdb
40	In _______ we can trap the destruction of the object and take action.	PHP	PHP4	PHP5	PHP6	PHP5
41	The __________ method is changed to keep track of any properties that might have been modified in the variable.	_construct( )	_set( )	_get()	_this()	_set( )
42	____________ is the capability of an application to do different things based on the particular object.	polymorphism	inheritance	encapsulation	object	polymorphism
43	The ability to hide the details of implementation is known as ___________.	polymorphism	inheritence	encapsulation	object	encapsulation
44	_____________ is an important concept in object_oriented programming.	class	inheritence	encapsulation	object	encapsulation
45	Static methods are invoked with the ________ operator.	?	::	?::	$	::
46	The _________ function is also very useful when working with timestamps.	gettime( )	getdata( )	getdate( )	getdisplay()	getdate( )
47	_____________ method returns the user's decision.	confirm	decision	return( )	display	confirm
48	Opened a file or directory to be used using____________ or________.	open( ) ,dir( )	fopen( ) ,dir( )	fopen( ) ,opendir	fclose(),dir()	fopen( ) ,opendir
49	__________, which are the "blueprints" for an object and are the actual code that defines the properties and methods.	function	methods	classes	objects	classes
50	_________,which is the ability to define a class of one kind as being a sub-type of a different kind of class.	polymorphism	encapsulation	object	inheritance	inheritance
51	The ____________ function enables to print the contents of a file without even having to call fopen( ).	readfile( )	closefile( )	writefile( )	openfile( )	readfile( )
52	________ will move the file position indicator associated with fp to a position determined by offset.	ftell( )	flength( )	fseek	fsetpos( )	fseek

53	The beginning of the file + offset is _________.	SEEK_SET	SEEK_CUR	SEEK_END	SEEK_OPEN	SEEK_SET
54	The _________ function returns the next entry listed in the open directory.	closedir( )	dir( )	dir_list	readdir( )	readdir( )
55	_________ occurs when the next entry is a subdirectory.	recursion	function	method	array	recursion
56	The _______ function is invoked automatically when you instantiate a new object of class property object .	_set	_get	_construct	_thik()	_construct
57	The word _______ literally means to place in a capsule, or outer container.	inheritance	encapsulation	polymorphism	abstraction	encapsulation
58	Classes in PHP can now have destructors,through the __________ method.	_destruct( )	_destroy	_delete	_set	_destruct( )
59	Which one of the following is not an OOP benefit?	Code reusability	easy to use	modularity	equal	none of these
60	The absolute path to the home directory of the user is ____________.	dir	shell	gid	gecos	dir

UNIT IV

String Manipulation and Regular Expression: Creating and accessing String , Searching & Replacing String-Formatting, joining and splitting String , String Related Library functions -Use and advantage of regular expression over inbuilt function-Use of preg_match(), preg_replace(), preg_split() functions in regular expression.

STRING MANIPULATION AND REGULAR EXPRESSION

A string is a sequence of letters, numbers, special characters and arithmetic values or combination of all. The simplest way to create a string is to enclose the string literal (i.e. string characters) in single quotation marks ('), like this:

$my_string = 'Hello World';

We can also use double quotation marks ("). However, single and double quotation marks work in different ways. Strings enclosed in single-quotes are treated almost literally, whereas the strings delimited by the double quotes replaces variables with the string representations of their values as well as specially interpreting certain escape sequences.

The escape-sequence replacements are:

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

Here's an example to clarify the differences between single and double quoted strings:

Example

```
<?php

$my_str = 'World';

echo "Hello, $my_str!<br>";      // Displays: Hello World!

echo 'Hello, $my_str!<br>';      // Displays: Hello, $my_str!

echo '<pre>Hello\tWorld!</pre>'; // Displays: Hello\tWorld!
echo "<pre>Hello\tWorld!</pre>"; // Displays: Hello   World!

echo 'I\'ll be back';          // Displays: I'll be back

?>
```

There Are Many Numbers of Functions That Are Directly Concerned With Manipulating Strings

- Search for text within a string
- Calculate the string length
- Break a string down into component parts
- Formatting of strings

Strings Functions

String functions like strlen, strsrt, strpos etc.

Php Strlen() Function

The length of the string using strlen() function.

Syntax of Strlen Function

Strlen($expression);

Strlen Example

```
<?php

 $element = strlen("PHP");

 echo $element."<br>";

 echo strlen("welcome");

?>
```

Searching Strings With Strstr() Function

Find out whether some text occurs within a string or not, using strstr() function ,if string match print all the string from where string matched. If word is not found strstr function return false.

Syntax Of Strstr() Function

Strstr($expression ,search text);

Strsrt Example

```
<?php
 $string1= "Hello world!";
 echo strstr( $string1, "H" )."<br>";
 echo ( strstr( $string1, "xyz" ) ? "Yes" : "No" ) ."<br>";
 $string2= "Welcome to php string";
 echo strstr( $string1, "java" ) ."<br>"; //nothing print
 echo ( strstr( $string2, "php" ) ? "Yes" : "No" ) ."<br>";
 ?>
```

Accessing Characters within A String

To access the each characters of a string. To access a character at a particular position.

Syntax

$char = $str [position] ;

String Example

```
<?php
 $myStr = "Welcome to the php string";
 echo $myStr[0] ."<br>"; // print "W"
 echo $myStr[6] ."<br>"; // print "e"
 $myStr[25] = '?';//Welcome to the php string?
 echo $myStr ."<br>";
 ?>
```

Replacing String

Definition and Usage

- The str_replace() function replaces some characters with some other characters in a string.
- This function works by the following rules:
- If the string to be searched is an array, it returns an array
- If the string to be searched is an array, find and replace is performed with every array element
- If both find and replace are arrays, and replace has fewer elements than find, an empty string will be used as replace
- If find is an array and replace is a string, the replace string will be used for every find value

Syntax

str_replace(find,replace,string,count)

Parameter	Description
find	Required. Specifies the value to find
replace	Required. Specifies the value to replace the value in *find*
string	Required. Specifies the string to be searched
count	Optional. A variable that counts the number of replacements

Example

Using str_replace() with an array and a count variable:

```
<?php

$arr = array("blue","red","green","yellow");

print_r(str_replace("red","pink",$arr,$i));

echo "Replacements: $i";
```

```
?>
```

Formatting Strings

Concatenation

Concatenation is a programming word for adding strings together. In PHP, the concatenation operator is a dot (.). Generally, concatenation is used to combine literal text with variables or values returned from functions.

Example

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Concatenation</title>
</head>
<body>
<h1>Concatenation</h1>
<?php
        $firstName = 'Paul';
        $greeting = 'Hello';
        echo $greeting . ' ' . $firstName . '!';
?>
<h2>Using Double Quotes to Avoid the Concatenation Operator</h2>
<?php
```

```
        echo "$greeting $firstName!";
?>
<h2>Double quotes don't work when concatenating
        the results of a function call</h2>
<?php
        echo $greeting . ' ' . $firstName . '!  Today is ' . date('l') . '.';
?>
</body>
</html>
```

String Manipulation Functions

Trimming Strings	
Function	**Description**
trim()	Removes whitespace at beginning and end of a string.
ltrim()	Removes whitespace at the beginning of a string.
rtrim()	Removes whitespace at the end of a string.

Presentation	
Function	**Description**
htmlentities()	Escapes all HTML entities.
nl2br()	Inserts a <br /> tag before each newline character in a string.
strtoupper()	Converts a string to uppercase.
strtolower()	Converts a string to lowercase.
ucfirst()	Converts the first character of a string to uppercase.
ucwords()	Converts the first character of each word in a string to uppercase.

Converting Strings and Arrays	
Function	**Description**
explode()	Splits a string into an array on a specified character or group of characters.

Converting Strings and Arrays	
Function	**Description**
implode()	Converts an array into a string, placing a specified character or group of characters between each array element.
join()	Same as implode().

Substrings	
Function	**Description**
substr(str,pos)	Returns the substring from the character in position pos to the end of the string.
substr(str,-len)	Returns the substring from len characters from the end of the string to the end of the string.
substr(str,pos,len)	Returns a len length substring beginning with the character in position pos.
substr(str,pos,-len)	Returns a substring beginning with the character in position pos and chopping off the last len characters of the string.
strstr(haystack,needle,before_needle)	If the third argument (before_needle) is false (default), then it returns the part of the haystack from the needle on. If the third argument (before_needle) is true, then it returns the part of the haystack before the needle. The needle can be a string or an integer (or a number that can be converted to an integer).
stristr(haystack,needle,before_needle)	Same as strstr(), but *case insensitive*.
strpos(haystack,needle)	Finds the position of the first occurrence of a specified needle in a haystack (string). The needle can be a string or an integer (or a number that can be converted to an integer).
strrpos(haystack,needle)	Finds the position of the last occurrence of a specified needle in a haystack (string). The needle can be a string or an integer (or a number that can be converted to an integer).
str_replace()	Replaces all occurrences of one string with another string.

Comparing Strings

Function	Description
strcmp()	Compares two strings. Returns < 0 if str1 is less than str2, > 0 if str1 is greater than str2, and 0 if they are equal.
strcasecmp()	Like strcmp() but case *in*sensitive.
strlen()	Returns the length of a string.

JOINING STRING

Joining String means Compensation Operator is used to combine character strings.

Operator	Description
.	The PHP concatenation operator (.) is used to combine two string values to create one string.
.=	Concatenation assignment which appends the argument on the right side to the argument on the left side.

Example

```
<?php
   $name="John";
   $lastName="Travolta";
   echo $name." ".$lastName; // Outputs John Travolta

   $a="Hello";
   $a .= " John!";
   echo $a; // Outputs Hello John!
?>
```

SPLITTING STRING

Definition and Usage

- The split() function will divide a string into various elements, the boundaries of each element based on the occurrence of pattern in string.
- The optional input parameter limit is used to signify the number of elements into which the string should be divided, starting from the left end of the string and working rightward.
- In cases where the pattern is an alphabetical character, split() is case sensitive.

Syntax

array split (string pattern, string string [, int limit])

Return Value

Returns an array of strings after splitting up a string.

Example

Following is the piece of code, copy and paste this code into a file and verify the result.

```
<?php
  $ip = "123.456.789.000"; // some IP address
  $iparr = split ("\.", $ip);
  print "$iparr[0] <br />";
  print "$iparr[1] <br />" ;
  print "$iparr[2] <br />"  ;
  print "$iparr[3] <br />"  ;
?>
```

This will produce the following result –

```
123

456
```

789

000

str_split() Function

The str_split() is an inbuilt function in PHP and is used to convert the given string into an array. This function basically splits the given string into smaller strings of length specified by the user and stores them in an array and returns the array.

Syntax:

array str_split($org_string, $splitting_length)

Parameters:

The function accepts two parameters and are described below:

1. $org_string (mandatory): This refers to the original string that the user needs to split into an array.
2. $splitting_length (optional): This refers to the length of each array element, we wish to split our string into. By default the function accepts the value as 1.

Return Values: The function returns an array. If the length parameter exceeds the length of the original string, then the whole string is returned as a single element. If the length parameter is less than 1, then False is returned. By default length is equal to 1.

Examples:

```
<?php
// PHP program to display the working of str_split()
$string = "Geeks";
// Since second argument is not passed,
// string is split into substrings of size 1.
```

```
print_r(str_split($string));


$string = "GeeksforGeeks";
// Splits string into substrings of size 4
// and returns array of substrings.
print_r(str_split($string, 4))
?>
```

Output:

```
Array
(
    [0] => G
    [1] => e
    [2] => e
    [3] => k
    [4] => s
)
Array
(
    [0] => Geek
    [1] => sfor
    [2] => Geek
    [3] => s
)
```

STRING RELATED LIBRARY FUNCTIONS

1. PHP strtolower() function

The strtolower() function returns string in lowercase letter.

Syntax

```
string strtolower ( string $string )
```

Example

```
<?php
$str="My name is KHAN";
$str=strtolower($str);
echo $str;
?>
```

Output:

```
my name is khan
```

2. PHP strtoupper() function

The strtoupper() function returns string in uppercase letter.

Syntax

```
string strtoupper ( string $string )
```

Example

```
<?php
$str="My name is KHAN";
$str=strtoupper($str);
echo $str;
?>
```

Output:

```
MY NAME IS KHAN
```

3. PHP ucfirst() function

The ucfirst() function returns string converting first character into uppercase. It doesn't change the case of other characters.

Syntax

string ucfirst (string $str)

Example

```
<?php
$str="my name is KHAN";
$str=ucfirst($str);
echo $str;
?>
```

Output:

```
My name is KHAN
```

4.PHP lcfirst() function

The lcfirst() function returns string converting first character into lowercase. It doesn't change the case of other characters.

Syntax

string lcfirst (string $str)

Example

```
<?php
$str="MY name IS KHAN";
$str=lcfirst($str);
echo $str;
?>
```

Output:

```
mY name IS KHAN
```

5.PHP ucwords() function

The ucwords() function returns string converting first character of each word into uppercase.

Syntax

string ucwords (string $str)

Example

```
<?php
$str="my name is Sonoo jaiswal";
$str=ucwords($str);
echo $str;
?>
```

Output:

```
My Name Is Sonoo Jaiswal
```

6. PHP strrev() function

The strrev() function returns reversed string.

Syntax

string strrev (string $string)

Example

```
<?php
$str="my name is Sonoo jaiswal";
$str=strrev($str);
echo $str;
?>
```

Output:

```
lawsiaj oonoS si eman ym
```

7. PHP strlen() function

The strlen() function returns length of the string.

Syntax

int strlen (string $string)

Example

```
<?php
$str="my name is Sonoo jaiswal";
$str=strlen($str);
echo $str;
?>
```

Output:

```
24
```

REGULAR EXPRESSIONS

- Regular expressions are powerful pattern matching algorithm that can be performed in a single expression.
- Regular expressions use arithmetic operators such as (+,-,^) to create complex expressions.
- Regular expressions help you accomplish tasks such as validating email addresses, IP address etc.

Use of regular expressions

- Regular expressions simplify identifying patterns in string data by calling a single function. This saves us coding time.
- When validating user input such as email address, domain names, telephone numbers, IP addresses,
- Highlighting keywords in search results
- When creating a custom HTML template. Regular expressions can be used to identify the template tags and replace them with actual data.

Regular expressions in PHP

PHP has built in functions that allow us to work with regular functions. Let's now look at the commonly used regular expression functions in PHP.

1. preg_match – this function is used to perform a pattern match on a string. It returns true if a match is found and false if a match is not found.
2. preg_split – this function is used to perform a pattern match on a string and then split the results into a numeric array
3. preg_replace – this function is used to perform a pattern match on a string and then replace the match with the specified text.

Below is the syntax for a regular expression function such as preg_match,preg_split or preg_replace.

```
<?php

function_name('/pattern/',subject);

?>
```

HERE,

- "function_name(...)" is either preg_match, preg_split or preg_replace.
- "/.../" The forward slashes denote the beginning and end of our regular expression
- "'/pattern/'" is the pattern that we need to matched
- "subject" is the text string to be matched against

PHP Preg_match

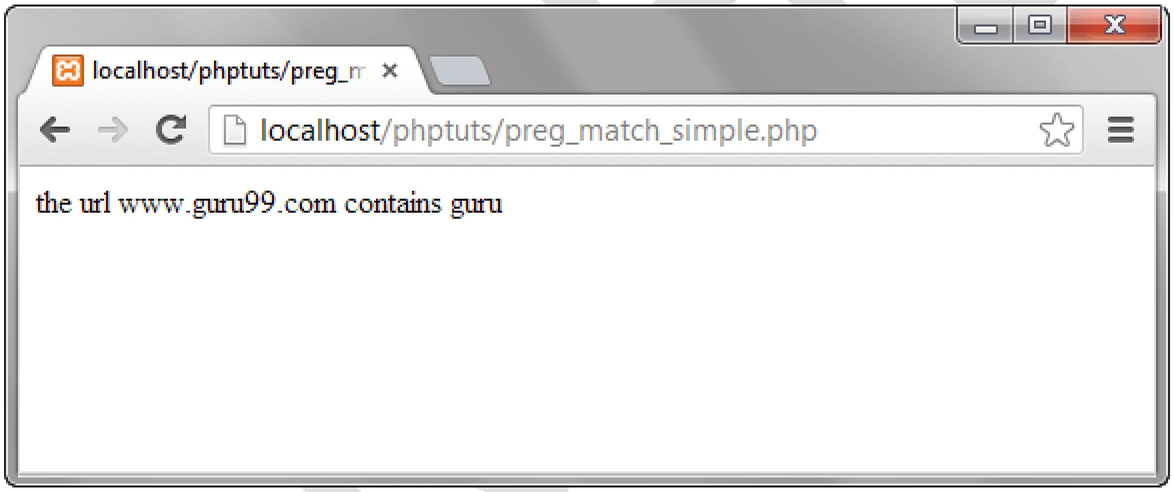
The first example uses the preg_match function to perform a simple pattern match for the word guru in a given URL.

The code below shows the implementation for the above example.

```
<?php

$my_url = "www.guru99.com";

if (preg_match("/guru/", $my_url))
```

```
{
        echo "the url $my_url contains guru";
}
else
{
        echo "the url $my_url does not contain guru";
}
?>
```

Output



```
the url www.guru99.com contains guru
```

PHP Preg_split

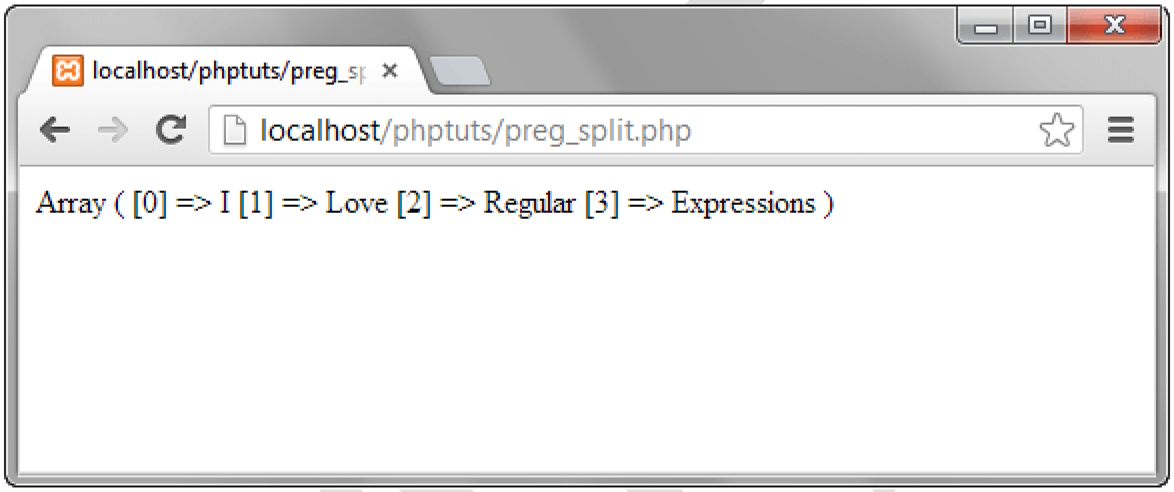
We will take a string phrase and explode it into an array; the pattern to be matched is a single spa

The text string to be used in this example is "I Love Regular Expressions".

The code below illustrates the implementation of the above example.

```
<?php
```

```
$my_text="I Love Regular Expressions";

$my_array  = preg_split("/ /", $my_text);

print_r($my_array );

?>
```



```
Array ( [0] => I [1] => Love [2] => Regular [3] => Expressions )
```

PHP Preg_replace

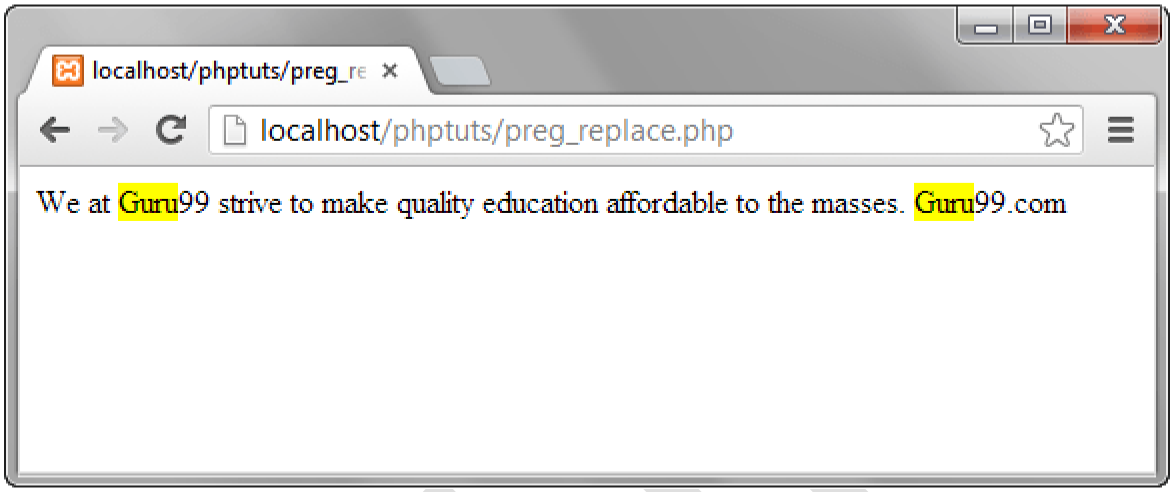
The preg_replace function that performs a pattern match and then replaces the pattern with something else.

The code below searches for the word guru in a string.

It replaces the word guru with the word guru surrounded by css code that highlights the background colour.

```
<?php

$text = "We at Guru99 strive to make quality education affordable to the masses. Guru99.com";

$text = preg_replace("/Guru/", '<span style="background:yellow">Guru</span>', $text);

echo $text;
```

```
?>
```



```
We at Guru99 strive to make quality education affordable to the masses. Guru99.com
```

Meta characters

The above examples used very basic patterns; metacharacters simply allow us to perform more complex pattern matches such as test the validity of an email address. Let's now look at the commonly used meta characters.

	Description	Example
.	Matches any single character except a new line	/./ matches anything that has a single character
^	Matches the beginning of or string / excludes characters	/^PH/ matches any string that starts with PH
$	Matches pattern at the end of the string	/com$/ matches guru99.com,yahoo.com Etc.
*	Matches any zero (0) or more characters	/com*/ matches computer, communication etc.
+	Requires preceding character(s) appear at least once	/yah+oo/ matches yahoo
\	Used to escape meta characters	/yahoo+\.com/ treats the dot as a literal value
[...]	Character class	/[abc]/ matches abc
a-z	Matches lower case letters	/a-z/ matches cool, happy etc.
A-Z	Matches upper case letters	/A-Z/ matches WHAT, HOW, WHY etc.
0-9	Matches any number between 0 and 9	/0-4/ matches 0,1,2,3,4

The above list only gives the most commonly used metacharacters in regular expressions.

Example that checks the validity of an email address.

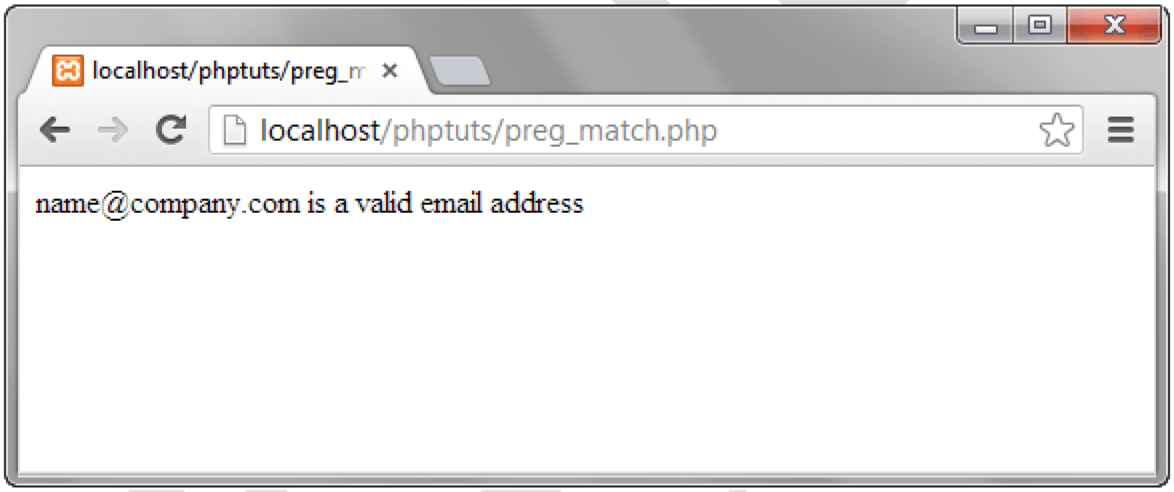
```
<?php

$my_email = "name@company.com";

if (preg_match("/^[a-zA-Z0-9._-]+@[a-zA-Z0-9-]+\.[a-zA-Z.]{2,5}$/", $my_email)) {

echo "$my_email is a valid email address";

}

else

{

  echo "$my_email is NOT a valid email address";

}
```

```
?>
```

Explaining the pattern "[/^[a-zA-Z0-9._-]+@[a-zA-Z0-9-]+\.[a-zA-Z.]{2,5}$/]"

HERE,

- "/…/" starts and ends the regular expression
- "^[a-zA-Z0-9._-]" matches any lower or upper case letters, numbers between 0 and 9 and dots, underscores or dashes.
- "+@[a-zA-Z0-9-]" matches the @ symbol followed by lower or upper case letters, numbers between 0 and 9 or dashes.
- "+\.[a-zA-Z.]{2,5}$/" escapes the dot using the backslash then matches any lower or upper case letters with a character length between 2 and 5 at the end of the string.



```
name@company.com is a valid email address
```

POSSIBLE QUESTIONS

Part B (2 Marks)

1. How to create a String?
2. What is the use of regular expression?
3. Mention advantage of regular expression.
4. List out the types of functions in regular expression.
5. Define String.

Part C (6 Marks)

1. Explain in detail about Searching & Replacing String-Formatting.
2. Describe about joining and splitting String in PHP
3. List out the String Library functions and explain it with example.
4. Depict on use and advantage of regular expression over inbuilt function.
5. Discuss on preg_match(), preg_replace(), preg_split() functions in regular expression.

KARPAGAM ACADEMY OF HIGHER EDUCATION

DEPARTMENT OF COMPUTER TECHNOLOGY

Subject Name PHP PROGRAMMING

Class: III B.SC CT

SUBCODE: 16CTU601A **Sem:VI**

S.N	Question	Choice1	Choice2	Choice3	Choice4	Answer Key
1	___ is the most universally implemented database language.	SQL	C	C++	JAVA	SQL
2	The first commercial RDBMS to hit the market.	Oracle9i	Oracle8i	Oracle	sql	Oracle
3	ANSI released the first published SQL standard in the year ___	1984	1986	1990	1991	1986
4	ISO stands for _______	Indian Organization for Standardization.	International Standardized	International Organization for	Indian Standard Organization.	International Organization for Standardization.
5	____ is a set of features that a vendor could implement in a RDBMS.	Stored procedure	Persistent module	Persistent stored module	Package	Package
6	_____ is a collection of SQL statements	Package	Persistent stored	Stored module	Persistent module	Stored module
7	PSM stands for	Persistent stored	Program stored	Procedure stored	Package stored	Persistent stored module
8	____ is a grid like structure	Data base	Table	Table index	Index	Table
9	____ is a list of values taken from a specified column.	Data base	Index	Table	Table index	Index
10	The CREATE TABLE statement is a type of	DDL statement	DML statement	DCL statement	DBA statement	DDL statement
11	____ acts as central point of administration for the tables in the data base.	Data	Data base	Queries	statements	Data base
12	____ specifies which default collation to use.	COLLATE clause	Character set	DEFAULT clause	VALUE clause.	COLLATE clause
13	A ____ is a named sorting order used for a specified character set.	Collation	Character set	Default	Value.	Collation
14	The statement is one of the most complex SQL statements in MYSQL.	MODIFY TABLE	CREATE TABLE	DELETE TABLE	UPDATE TABLE	CREATE TABLE
15	The statement for deleting a data base from your system is ______	DELETE data base	DROP date base	REMOVE data base	ADD data base	DROP date base
16	Decimal data base is referred to as ______	Exact	Numeric	Approximate	Alphabet	Exact

17	The binary data type takes ______	One argument	Two argument	Do not take any	More than one	Do not take any argument
18	The ____ provides a handy way to record each transaction that occurs in a particular table	DATE	DATE TIME	TIMESTAMP	TIME	TIMESTAMP
19	DATE data type ranges between	1000-01-01 00:00:00 through	1000-01-01 through 9999	1901 to 2155	1900 to 2786	1000-01-01 through 9999
20	MYSQL allows to assign default values through the use of a _____	DEFAULT clause	Character set	Collate clause	Value clause	DEFAULT clause
21	The default values for TIMESTAMP columns in the table are ____ values in place of the date	Zero	One	Two	Default	Zero
22	For numeric columns that are not configured with the AUTO-INCREMENT option, the default value is ____	One	Two	Three	Zero	Three
23	For columns configured with the ___ data type the default value is the first Value specified in the column definition.					
24	The table that contains the foreign key, is referred to as ____	Parent table	Child table	Referencing table	Key table	Child table
25	The referenced table is referred to as _____	Parent table	Child table	Referencing table	Key table	Parent table
26	An ____ is a device that MYSQL user to speed up searches and reduce the time it takes to execute complex quarries	Table	Regular index	Index	Search index	Index
27	An index types that permits duplicate values and null values in the Columns on which the index is defined.	Regular	index	Full-text	Half-text	Regular
28	____ constraint is used to create a unique	primary key	foreign key	unique	Candidate key.	unique
29	An ___ statement is the most common method used to directly insert data in a table.	add	insert	update	delete	insert
30	An ___ is a type of formula that helps define the value to insert in a column.	Expression	Insert	Add	Append	Expression
31	A __ is an object that carries out a predefined	Variable	Calling function	function	Entity	function
32	___ function can be used in SQL statements to return value that is equalant to the current date	date( )	time( )	now( )	date( ) and time( )	now( )

33	The ___ is useful for any characters that could be interpreted when executing a statement that contain a string value.	Backslash	Space bar	Tab	Under score	Backslash
34	The ___ clause includes a value for each column, entered in the order in which the columns appear in the table definition.	value	Default	Collate	select	value
35	____ statement deletes the old row and adds the new row.	REPLACE	DELETE	DROP	REMOVE	REPLACE
36	____ clause includes one or more conditions that define the extent of the delete operation.	ORDER BY	LIMIT	WHERE	DELETE	WHERE
37	____ clause sorts rows according to the column or columns specified in the clause.	ORDER BY	LIMIT	WHERE	SORT	ORDER BY
38	The clause limits the number of rows to be deleted to the number specified in the clause.	ORDER BY	LIMIT	WHERE	DELETE	LIMIT
39	The ___ statement removes all rows from a	DELETE	TRUNCATE	REMOVE	DROP	TRUNCATE
40	A ____ is a set of one or more SQL statements that perform a set of related action	Transaction	Trigger	Queries	view	Transaction
41	The SELECT clause includes SELECT keyword and ____	+	*	;	/	;
42	An ____ function is type of function that summarizes data such as Count ( ) function.	Aggregate ( )	Sum ( )	Cal( )	add()	Aggregate ( )
43	____ operates perform calculations on the arguments within an expression.	Arithmetic	comparison	logical	bitwise	Arithmetic
44	_____ operators compare the arguments in an expression test whether a condition is true, false, or null.	Arithmetic	comparison	sort	bitwise	comparison
45	___ operators verify the validity of one or more expressions to test whether they return a condition of true,false,or null.	comparison	Arithmetic	logical	bitwise	logical
46	___ operators manipulate the bit values associated with numerical values	bitwise	comparison	logical	Arithmetic	bitwise

47	_____ operator specify the collation and case sensitivity of searches and sorting operations.	Arithmetic	comparison	logical	sort	sort
48	If either argument is null or if both arguments are null the condition is Considered.	true	false	null	zero	null
49	____ operator converts a string to a binary string so that comparing and sorting data is	Special	COLLATE	LIKE	BINARY	BINARY
50	____ operator specifies that a particular collation be used to compare and sort string	BINARY	COLLATE	LIKE	Special.	COLLATE
51	The ___ function identifies character set used for specified string.	Strstr ( )	COLLATION ( )	CONCATE ( )	CHARACTER SET ( )	CHARACTER SET ( )
52	The INSTR( ) takes ____ argument	One	Two	Three	Zero.	One
53	The ___ and ___ operators allow you to create an expression that compares a column to any of the values returned by a subquerry.	max and min	Any and some	sum and diff	add and min	Any and some
54	____ file is text file that contains one or more rows of exported data in a delimited format.	Infile	Dump file	Outfile	deletefile	Outfile
55	A ____ format is one in which the values and rows are separated and enclosed by specific types of characters.	Delimited	Curly braces	Function caps	Caps lock	Delimited
56	___ file is a text file that contains only one row that is not delimited.	Infile	Dump file	Outfile	Unique file	Dump file
57	____ statement at MYSQL command prompt is used to import delimited values directly from	MYSQL command	LOAD DATA	Insert.	delete	LOAD DATA
58	_____ command at the MYSQL command prompt is used to run SQL statements and MYSQL commands that are saved in a text file.	LOAD DATA	Add	MYSQL	SOURCE	SOURCE
59	The ___ statement is used to terminate a transaction and to save all changes made by the transaction to the data base.	COMMIT	save point	roll back	time()	COMMIT
60	The ____ function converts a data that is retrieved as a string value.	strstr( )	time( )	strtotime ( )	strtime()	strtotime ( )

UNIT-V

Array:

Anatomy of an Array ,Creating index based and Associative array, Accessing array-Looping with Index based array, with associative array using each() and foreach() -Some useful Library function

PHP Array

Arrays in PHP is a type of data structure that allows us to store multiple elements of similar data type under a single variable thereby saving us the effort of creating a different variable for every data. The arrays are helpful to create a list of elements of similar types, which can be accessed using their index or key.

An array is created using an array () function in PHP.

There are basically three types of arrays in PHP:

1. Indexed or Numeric Arrays: An array with a numeric index where values are stored linearly.
2. Associative Arrays: An array with a string index where instead of linear storage, each value can be assigned a specific key.
3. Multidimensional Arrays: An array which contains single or multiple array within it and can be accessed via multiple indices.

Numeric & Indexed Arrays

- These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.
- Numeric arrays use number as access keys.
- An access key is a reference to a memory slot in an array variable.
- The access key is used whenever we want to read or assign a new value an array element.

Below is the syntax for creating numeric array in php.

```
<?php
```

```
$variable_name[n] = value;

?>
```

Or

```
<?php

$variable_name = array(n => value, ...);

?>
```

HERE,

- "$variable_name..." is the name of the variable
- "[n]" is the access index number of the element
- "value" is the value assigned to the array element.

Example

```
<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>
```

This will produce the following result −

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
Value is one
Value is two
Value is three
Value is four
Value is five
```

Associative Arrays

- In an associative array, the keys assigned to values can be arbitrary and user defined strings.
- Associative array differ from numeric array in the sense that associative arrays use descriptive names for id keys.

Syntax for associative arrays:

array(key=>value,key=>value,key=>value,etc.);

Parameter	Description
key	Specifies the key (numeric or string)
value	Specifies the value

For Example

To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.

```
<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>
```

This will produce the following result −

```
Salary of mohammad is 2000

Salary of qadir is 1000

Salary of zara is 500

Salary of mohammad is high

Salary of qadir is medium

Salary of zara is low
```

Multidimensional Arrays

Multi-dimensional arrays are such arrays which stores an another array at each index instead of single element. In other words, we can define multi-dimensional arrays as array of arrays. As the name suggests, every element in this array can be an array and they can also hold other sub-arrays within. Arrays or sub-arrays in multidimensional arrays can be accessed using multiple dimensions.

Example Program

```
<?php
```

```
  // Defining a multidimensional array
$favorites = array(
   array(
      "name" => "Dave Punk",
      "mob" => "5689741523",
      "email" => "davepunk@gmail.com",
   ),
   array(
      "name" => "Monty Smith",
      "mob" => "2584369721",
      "email" => "montysmith@gmail.com",
   ),
   array(
      "name" => "John Flinch",
      "mob" => "9875147536",
      "email" => "johnflinch@gmail.com",
   )
);

// Accessing elements
echo "Dave Punk email-id is: " . $favorites[0]["email"], "\n";
echo "John Flinch mobile number is: " . $favorites[1]["mob"];


?>
```

Output

```
Dave Punk email-id is: davepunk@gmail.com

John Flinch mobile number is: 2584369721
```

LOOPING WITH INDEX BASED ARRAY

There are two ways to create indexed arrays.

- First way to use array() function without any index, index are assigned automatically starting from 0.
- Second way to manually assign index and create the array.

PHP count() function is used to get the length of an array. We can use for loop to loop through all the values of an indexed array.

Example Program

```
<?php

$colors = array("Red","Green","Blue");

$colors1[0] = "Red";

$colors1[1] = "Green";

$length = count($colors);

echo "colors array length=" . $length; // prints "colors array length=3"

echo "<br>";

echo "colors1 array length=" . count($colors1); // prints "colors1 array length=2"
```

```
//looping an indexed array

for($i=0; $i<$length; $i++){

        echo $colors[$i];

        echo "<br>";

}

?>
```

Associative Array Using Each ()

each — Return the current key and value pair from an array and advance the array cursor

Syntax

array each (array &$array)

Return the current key and value pair from an array and advance the array cursor.

After each() has executed, the array cursor will be left on the next element of the array, or past the last element if it hits the end of the array.

Example

```
<?php

$foo = array("bob", "fred", "jussi", "jouni", "egon", "marliese");

$bar = each($foo);

print_r($bar);

?>
```

Output

```
Array
```

```
(
   [1] => bob
   [value] => bob
   [0] => 0
   [key] => 0
)
```

Example

```
<?php
$foo = array("Robert" => "Bob", "Seppo" => "Sepi");
$bar = each($foo);
print_r($bar);
?>
```

Output

```
Array
(
   [1] => Bob
   [value] => Bob
   [0] => Robert
   [key] => Robert
)
```

Associative array using foreach()

Associative arrays uses named keys for values and we can create them in similar way like indexed arrays. foreach is used to loop through an associative array.

Syntax

```
foreach ($array as $value) {
    code to be executed;
}
```

Example

```
<?php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
    echo "$value <br>";
}
?>
```

Output

```
red
green
blue
yellow
```

Example

```
<?php
$colors = array("0"=>"Red","1"=>"Green","2"=>"Blue");
echo "0th element of array is " . $colors["0"];
echo "<br>";

//looping

foreach ($colors as $key=>$value){

        echo "Key=".$key." value=".$value;

        echo "<br>";

}

?>
```

Output of above PHP script is:

```
0th element of array is Red

Key=0 value=Red

Key=1 value=Green

Key=2 value=Blue
```

SOME USEFUL LIBRARY FUNCTION

<u>Function</u>	<u>Explanation</u>	<u>Example</u>
sizeof($arr)	This function returns the number of elements in an array. Use this function to find out how many elements an array contains; this information is most commonly used to initialize a loop counter when processing the	Code: $data = array("red", "green", "blue"); echo "Array has " . sizeof($data) . " elements"; ?> <br>Output: Array has 3 elements

	array.	
array_values($arr)	This function accepts a PHP array and returns a new array containing only its values (not its keys). Its counterpart is the array_keys() function. Use this function to retrieve all the values from an associative array.	Code: $data = array("hero" => "Holmes", "villain" => "Moriarty"); print_r(array_values($data)); ?> Output: Array ([0] => Holmes [1] => Moriarty)
array_keys($arr)	This function accepts a PHP array and returns a new array containing only its keys (not its values). Its counterpart is the array_values() function. Use this function to retrieve all the keys from an associative array.	Code: $data = array("hero" => "Holmes", "villain" => "Moriarty"); print_r(array_keys($data)); ?> Output: Array ([0] => hero [1] => villain)

array_pop($arr)	This function removes an element from the end of an array.	Code:
		$data = array("Donald", "Jim", "Tom");
		array_pop($data);
		print_r($data);
		?>
		
		Output:
		Array
		(
		[0] => Donald
		[1] => Jim
		)

array_push($arr,	This function adds an	Code:

$val)	element to the end of an array.	$data = array("Donald", "Jim", "Tom");
		array_push($data, "Harry");
		print_r($data);
		?>
		
		Output:
		Array
		(
		[0] => Donald
		[1] => Jim
		[2] => Tom
		[3] => Harry
		)

array_shift($arr)	This function removes an element from the beginning of an array.	Code:
		$data = array("Donald", "Jim", "Tom");
		array_shift($data);
		print_r($data);
		?>
		
		Output:
		Array
		(
		[0] => Jim
		[1] => Tom
		)

array_unshift($arr, $val)	This function adds an element to the beginning of an array.	Code:
		$data = array("Donald", "Jim", "Tom");
		array_unshift($data, "Sarah");
		print_r($data);
		?>
		
		Output:
		Array
		(
		[0] => Sarah
		[1] => Donald
		[2] => Jim
		[3] => Tom
		)

each($arr)	This function is most often used to iteratively traverse an array. Each time each() is called, it returns the current key-value pair and moves the array cursor forward one element. This makes it most suitable for use in a loop.	Code:
		$data = array("hero" => "Holmes", "villain" => "Moriarty");
		while (list($key, $value) = each($data)) {
		echo "$key: $value \n";
		}
		?>
		
		Output:
		hero: Holmes
		villain: Moriarty

sort($arr)	This function sorts the elements of an array in ascending order. String values will be arranged in ascending alphabetical order.
	
	Note: Other sorting functions include asort(), arsort(), ksort(), krsort() and rsort().
		Code:
		$data = array("g", "t", "a", "s");
		sort($data);
		print_r($data);
		?>
		
		Output:
		Array
		(
		[0] => a
		[1] => g
		[2] => s
		[3] => t
		)

array_flip($arr)	The function exchanges the keys and values of a PHP associative array. Use this function if you have a tabular (rows and columns) structure in an array, and you want to interchange the rows and columns.	Code: $data = array("a" => "apple", "b" => "ball"); print_r(array_flip($data)); ?> Output: Array ([apple] => a [ball] => b)
array_reverse($arr)	The function reverses the order of elements in an array. Use this function to re-	Code: $data = array(10, 20, 25, 60); print_r(array_reverse($data)); ?>

	order a sorted list of values in reverse for easier processing—for example, when you're trying to begin with the minimum or maximum of a set of ordered values.	Output: Array ([0] => 60 [1] => 25 [2] => 20 [3] => 10)
array_merge($arr)	This function merges two or more arrays to create a single composite array. Key collisions are resolved in favor of the latest entry. Use this function when you need to combine data from two or more arrays into a single structure—for example, records from two different SQL queries.	Code: $data1 = array("cat", "goat"); $data2 = array("dog", "cow"); print_r(array_merge($data1, $data2)); ?> Output: Array ([0] => cat [1] => goat [2] => dog [3] => cow)
array_rand($arr)	This function selects one or more random elements from an array. Use this function when you need to randomly select from a collection of discrete values—for example, picking a random color from a list.	Code: $data = array("white", "black", "red"); echo "Today's color is " . $data[array_rand($data)]; ?> Output: Today's color is red
array_search($search, $arr)	This function searches the values in an array for a match to the search term, and returns the	Code: $data = array("blue" => "#0000cc", "black" => "#000000", "green" => "#00ff00"); echo "Found " .

	corresponding key if found. If more than one match exists, the key of the first matching value is returned. Use this function to scan a set of index-value pairs for matches, and return the matching index.	array_search("#0000cc", $data); ?> Output: Found blue
array_slice($arr, $offset, $length)	This function is useful to extract a subset of the elements of an array, as another array. Extracting begins from array offset $offset and continues until the array slice is $length elements long. Use this function to break a larger array into smaller ones—for example, when segmenting an array by size ("chunking") or type of data.	Code: $data = array("vanilla", "strawberry", "mango", "peaches"); print_r(array_slice($data, 1, 2)); ?> Output: Array ([0] => strawberry [1] => mango)
array_unique($data)	This function strips an array of duplicate values. Use this function when you need to remove non-unique elements from an array—for example, when creating an array to hold values for a table's primary key.	Code: $data = array(1,1,4,6,7,4); print_r(array_unique($data)); ?> Output: Array ([0] => 1 [3] => 6 [4] => 7 [5] => 4)

array_walk($arr, $func)	This function "walks" through an array, applying a user-defined function to every element. It returns the changed array.
	
	Use this function if you need to perform custom processing on every element of an array—for example, reducing a number series by 10%.
		Code:
		function reduceBy10(&$val, $key)
		{
		$val -= $val * 0.1;
		}
		
		$data = array(10,20,30,40);
		array_walk($data, 'reduceBy10');
		print_r($data);
		?>
		
		Output:
		Array
		(
		[0] => 9
		[1] => 18
		[2] => 27
		[3] => 36
		)

POSSIBLE QUESTIONS

Part B (2 Marks)

1. Define Array.
2. What is difference between each() and foreach()?
3. What to access the array?
4. List out the library function of array.
5. How to create indexed based Array

Part C (6 Marks)

1. Describe about Associative array in PHP.
2. How to use looping with Index based array and explain it.
3. Explain in detail about associative array using each() and foreach().
4. Discuss on Library function in Array.
5. Write in detail about anatomy of an Array.

KARPAGAM ACADEMY OF HIGHER EDUCATION
DEPARTMENT OF COMPUTER TECHNOLOGY

Subject Name PHP PROGRAMMING

Class: III B.SC CT

SUBCODE: 16CTU601A **Sem:VI**

S.No	Question	Choice1	Choice2	Choice3	Choice4	Answer Key
1	The function used to create a connection to a MYSQL server.	MY SQL__connection()	MYSQL__server()	MY SQL__create()	MYSQL_link()	MY
2	_____________ function link to the MYSQL server which is closed when the script is terminated.	MYSQL__open()	MYSQL__link()	MYSQL__close()	MYSQL_post()	MYSQL__close()
3	_____________PHP's equivalent to MYSQL's SHOWDATABASES command.	MYSQL__show()	MYSQL__list__DBs	MYSQL__show __DBs	MYSQL__POST()	MYSQL__list__DBs
4	_____________ function returns the error number.	MYSQL__error()	SET__error()	POST__error()	GET__error()	MYSQL__error()
5	_____________ function returns the error text from the previous MYSQL operation.	POST__prev()	POST__prev(error)	GET__(error)	MYSQL__error()	MYSQL__error()
6	_____________ provides a couple of functions that catch server errors and determine what when wrong.	MYSQL__catch()	catch(error)	PHP	couple__catch__error()	PHP
7	______________ function call to set the level of error	SET(error)	call__SET(error)	list__SET()	Error__reporting()	Error__reporting()
8	By specifying _____________ we can ensure that no PHP error or warning messages be invoked.	Error__reporting(0)	PHP__message()	PHP__no error()	warning error()	Error__reporting(0)
9	______________ works only with character and varchar type	UNARY	BINARY	TERNARY	null	BINARY
10	Fields cannot take a null value.	NULL	NULL VALUE	NOT NULL	binary value	NOT NULL
11	Fields that are declared NULL will store a specified default value whenever a _____________ value is given.	NOT NULL	NULL	ZERO	ONE	NULL
12	Fields that are declared NULL will store a specified default value whenever a _____________ value is given.	NULL	NULL VALUE	default__value	DEFAULT default__value.	DEFAULT default__value.
13	_____________ only works unique,integer type fields.	AUTO__DECREMENT	INTEGER		INT__FIELD.	
14	There can be only one _____________ field per table.	AUTO__DECREMENT	INTEGER	INT__FIELD	AUTO__INCREM	AUTO__INCREMEN
15	_____________ are synonymous keywords that specify fields to be used as indexes.	LITERALS	IDENTIFIERS	KEY	TOKENS.	KEY
16	_____________ used as an index of unique values.	Primary key	Secondary Key	Foreign Key	none.	Primary key
17	To use a fields as an primary keys it must be declared as	NULL	NULL VALUE	NOT NULL	ZERO.	NOT NULL
18	The value of each entry in a unique field must be _____________ among all other entire in the same field.	Unique	Varying	Same	differ	Unique
19	____________ is a requirement for auto increment fields.	Varying	AUTO__INCREMENT	INT__FIELDS	Uniqueness.	Uniqueness.
20	A TIME STAMP type field can always takes a ___________	ZERO	NOT NULL	Null	none.	Null
21	The __________ servers as the key to define a relationship between the two tables.	Table ID	User ID	Server ID	Admin ID.	User ID
22	The __________ has the same user ID fields as the user table.	User Table	User ID	Server ID	Admin ID.	User Table
23	_________ takes a query string as its first argument and makes the query on the currently selected database.	Query__String	MYSQL__query()	MSQL__dbs	none.	MYSQL__query()

24	If the given query is successfully executed ____________ returns a non—zero value pointing to the required resulting set,or	Query__String()	MSQL__dbs()	MYSQL__query()	none.	MYSQL__query()
25	_________ and _________ are the pair of functions specially designed for creating and dropping.	MYSQL__open__db() and MYSQL__close__db()	MYSQL__query() and MYSQL__drop()	MYSQL__create__	MYSQL__create_ _db() and	MYSQL__create__db( ) and
26	Subsequent calls to create the user and access _________ lock	MYSQL__query()	Query__String()	MSQL__dbs()	MYSQL__locktabl	MYSQL__query()
27	__________ returns a single row of data from a result set	MYSQL__drop__db()	MYSQL__fetch__row()	MYSQL__query()	MYSQL__row()	
28	Variable _________ stores the returned result identifier.	$ result	$identifiers	$ literals	$ tokens.	$ result
29	The two functions of PHP that used to fetch data___________.	MYSQL__create__db() and MYSQL__delete__db	MYSQL__create__db() and	MYSQL__fetch__array() and	MYSQL__create_ _db() and	MYSQL__fetch__array() and
30	_________ returns a simple associative array from a result set.	$ result	MYSQL__fetch__object()		MYSQL__fetch__	MYSQL__fetch__arra
31	_________ returns a single object from a result set.	MYSQL__create__db()	MYSQL__fetch__array()	MYSQL__fetch__o	none.	MYSQL__fetch__obje
32	_________ returns the value of the specified field in a specified	MYSQL__field()	MYSQL__result()	MYSQL__row()	MYSQL__query()	MYSQL__result()
33	_________ takes the usual result indicator, plus a row number and field name as its arguments.	MYSQL__result()	MYSQL__field()	MYSQL__row()	MYSQL__affected__row().	MYSQL__result()
34	_________ is used to jump straight to a specific row of data.	MYSQL__row()	$fetched__row()		MYSQL__row__o	
35	The _________ function returns the current time of the system on which the server is running.	SHOW()	NOW()	current__time()	SERVER(time).	NOW()
36	We can retrive the current date and time separetely using __________ and ___________ functions.	CurrDate() and CurrTime()	Date() and Time()	CURR(Date) and CURR(Time)	Time	CurrDate() and CurrTime()
37	A __________ clause is used to selectively retrieve rows of data according to specified condition	DISTICTION	FROM	ORDER BY	WHERE.	WHERE.
38	_________ clause always comes at the end of the query.	ORDER BY	LIMIT	WHERE	SORT.	LIMIT
39	The _________ operator enables us to use values that may or may	ZERO	NOT NULL	NULL__safe	all the above.	NULL__safe
40	An __________ clause sorts retrieved values in ascending order	ORDER BY	LIMIT	WHERE	SORT.	ORDER BY
41	To sort the values in descending order, use the __________	LIMIT	WHERE	SORT.	DESC.	DESC.
42	Values in __________ type field are case sensitive.	NUMERALS	INTEGERS	TEXT	FLOAT	TEXT
43	_________ supports the largest number in the given field	LARGE()	open__new()	MAX(n)	LARGE(n)	MAX()
44	_________ supports the number of rows returned.	SUM	MAX(row)	MIN(row)	COUNT().	COUNT().
45	We can retrieve both the min and max values or integer fields by using the _________ and _________ function.	MIN() and MAX()	MAX(large) and MIN(small)	MAX(int) and MIN(int)	all the above.	MIN() and MAX()
46	___________ counts every row in the result set.	COUNT()	ROW()	CONT(*)	COUNT(row).	CONT(*)
47	__________ function starts an HTML page and defines the java	HTML__start()	HTML__header()	BODY	HEAD	HTML__header()
48	__________ function is used to call to open a new window when displaying a users record.	Open(new)	Open__new()	Open__Window(new)	Open__Window().	Open__Window().
49	__________ function ends an HTML page.	HTML__footer()	HTML__end()	BODY	HTML.	HTML__footer()
50	__________ function reports errors using the java script alert()	alert()	MYSQL__ERROR()	The	mysql__error().	The error__message()
51	__________ script makes use of the include file	db__connect()	cmn__file()	db.inc()	User viewer.PHP.	User viewer.PHP.
52	__________ function displays a list of listed users along with navigation link and a record viewer link that calls the	Link__record()	List__record()	cur__page()	records__per__page()	List__record()
53	The _________ function displays detailed information about given user and his/her access log data.	View_record()	open__window()	List__record()	close_record	View_record()

54	The MYSQL__num__rows() returns ________ when no record is found in the access__log table.	num(rows)	Zero	access__log()	two	Zero
55	To get formatted date string _________ is used	get(date)	SubStr(date)	PHPs SubStr()	none	PHPs SubStr()
56	If we want to throw away an existing record from a table __________ command is used.	REMOVE	DROP	THROW	DELETE.	DELETE.
57	_________ takes three arguments and return a result pointer that refers to a list of all fields in the specified table of specified	MYSQL__list__fields	access__log()	MYSQL__field__flag()	MYSQL__fetch__object()	MYSQL__list__fields
58	The ____________ variable holds a pointer to a list of fields in	$identifiers	$ literals	$ tokens	$result.	$result.
59	__________ function takes a result at pointer and a field index and returns a string containing the attribute for the given field.	MYSQL__list__fields	access__log()	MYSQL__field__flag()	MYSQL__fetch__object()	MYSQL__field__flag()
60	__________ function queries the MYSQL server about whether the type is already in use, and returns one of its.	access__log	In__use()	Fetch__log()	Fetch_close()	In__use()

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021

Computer Technology/Information Technology
Sixth Semester
FIRST INTERNAL EXAMINATION - December 2018

PHP PROGRAMMING

Class & Section: III B.Sc CT/ B.Sc (IT A&B) **Duration: 2 hours**
Date & Session: 17.12.18 (FN) **Maximum marks: 50 marks**
Sub.Code: 16CTU601A/16ITU601A

PART- A (20 * 1= 20 Marks)
Answer ALL the Questions

1. PHP stands for ___________________.
   (a) Php Hypertext Processor (b) Php Hypertext Preprocessor
   (c) Php Hypermarkup Preprocessor (d) Php Hypermarkup Processor

2. PHP is used for building___________________websites.
   (a)Static (b) Dynamic (c) None (d) A&B

3. PHP program are run on________________.
   (a)Web browser (b) Interpreter (c) Web server (d)Compiler

4. PHP is devised by_____________.
   (a)Tim Berners- Lee (b)Rasmus Lerdorf (c)Robert Caillau (d)Richard Fairly

5. ________ function prints out information about variables.
   (a)echo (b)print_r( ) (c)cout (d)none of these

6. PHP files have a default file extension of_______
   (a) .html (b) .xml (c) .php (d) .ph

7. A PHP script should start with ___ and end with ___
   (a) < php > (b) < ? php ?> (c) <? ?> (d) <?php ?>

8. Which function displays the information about PHP?
   (a)info() (b)sysinfo() (c)phpinfo() (d)php_info()

9. PHP is _____________scripting language.
   (a)Client-side (b)Middle Side (c)Server-side (d)Out-side

10. Which of the below symbols is a newline character?
    (a) \r (b) \n (c) /n (d) /r

11. Which of the following statements prints in PHP?
(a)Out (b)Write (c)Echo (d)Display

12. In PHP, each statement must be end with ______
(a).(dot) (b); (semicolon) (c) / (slash) (d) : (colon)

13. How to define a variable in PHP?
(a)$variable_name = value (b)$variable_name = value;
(c)$variable_name == value; (d)$variable_name as value;

14. Which of the following is not the scope of Variable in PHP?
(a)Local (b) Global (c) Static (d) Extern

15. PHP is _______ typed language.
(a)Loosely (b)Server (c)User (d) system

16. _______ tells the server which page to go to once the user has click the submit button on the form.

(a)Function attribute (b)Shift attribute (c)Action attribute (d)none of these

17. _______ attribute controls the way that the information is sends to the server.
(a)Method attribute (b)Action attribute (c)Shift attribute (d)Function attribute

18. Which of the following is not PHP Loops?
(a)while (b)do while (c)for (d) do for

19. What will be the result of combining a string with another data type in PHP?
(a)int (b) float (c)string (d)double

20. How many data types are there in PHP?
(a)2 (b)3 (c)4 (d)5

PART B (3 * 2 = 6 Marks)
Answer ALL the Questions

21. What are the tools required for PHP Programming?
22. Write the syntax of PHP?
23. Difference between GET and POST Methods.

PART C (3 * 8 = 24 Marks)
Answer ALL the Questions

24. a) Discuss about PHP inventions and versions.
(OR)

b) How to declare a variable in PHP? What are the rules to be followed to declare the variable?

25. a)Write a PHP Program to sort an array?

(OR)

b) What is Operator? Explain the operators in PHP with examples.

26. a) How to handle HTML form with PHP? Explain with example.

(OR)

b) How to declare the multi value fields in PHP?

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021

Computer Technology/Information Technology
Sixth Semester
FIRST INTERNAL EXAMINATION - December 2018

PHP PROGRAMMING
Class & Section: III B.Sc CT/ B.Sc (IT A&B) **Duration: 2 hours**
Date & Session : .12.18 (AN) **Maximum marks: 50 marks**
Subj.Code: 16CTU601A/16ITU601A

ANSWER KEY

PART- A (20 * 1= 20 Marks)
Answer ALL the Questions

1. PHP stands for ___________________.
   (a) Php Hypertext Processor **(b) Php Hypertext Preprocessor**
   (c) Php Hypermarkup Preprocessor (d) Php Hypermarkup Processor

2. PHP is used for building___________________websites.
   (a)Static **(b) Dynamic** (c) None (d) A&B

3. PHP program are run on________________.
   (a)Web browser (b) Interpreter (c) Web server (d)Compiler

4. PHP is devised by_____________.
   (a)Tim Berners- Lee **(b)Rasmus Lerdorf** (c)Robert Caillau (d)Richard Fairly

5. ________ function prints out information about variables.
   (a)echo (b)print_r( ) (c)cout (d)none of these

6. PHP files have a default file extension of_______
   (a) .html (b) .xml **(c) .php** **(**d) .ph

7. A PHP script should start with ___ and end with ___
   (a) < php > (b) < ? php ?> (c) <? ?> **(d) <?php ?>**

8. Which function displays the information about PHP?
   (a)info() (b)sysinfo() **(c)phpinfo()** (d)php_info()

9. PHP is _____________scripting language.
   (a)Client-side (b)Middle Side **(c)Server-side** (d)Out-side

10. Which of the below symbols is a newline character?
(a) \r **(b) \n** (c) /n (d) /r

11. Which of the following statements prints in PHP?
(a)Out (b)Write **(c)Echo** (d)Display

12. In PHP, each statement must be end with ______
(a).(dot) **(b); (semicolon)** (c) / (slash) (d) : (colon)

13. How to define a variable in PHP?
(a)$variable_name = value **(b)$variable_name = value;**
(c)$variable_name == value; (d)$variable_name as value;

14. Which of the following is not the scope of Variable in PHP?
(a)Local (b) Global (c) Static **(d) Extern**

15. PHP is _______ typed language.
(a)Loosely (b)Server (c)User (d) system

16. _______ tells the server which page to go to once the user has click the submit button on the form.
(a)Function attribute (b)Shift attribute **(c)Action attribute** (d)none of these

17. _______ attribute controls the way that the information is sends to the server.
(a)Method attribute (b)Action attribute (c)Shift attribute (d)Function attribute

18. Which of the following is not PHP Loops?
(a)while (b)do while (c)for **(d) do for**

19. What will be the result of combining a string with another data type in PHP?
(a)int (b) float **(c)string** (d)double

20. How many data types are there in PHP?
(a)2 **(b)3** (c)4 (d)5

PART B (3 * 2 = 6 Marks)
Answer ALL the Questions

21. What are the tools required for PHP Programming?
The three types of tools required for PHP Programming:

1. Tools for writing code: There are two main types of tools you can use: text editors and integrated development environments.
2. A file transfer program: It supports FTP, SFTP, and SCP. Easy to use, and free.

3. A local Web server: Install Web server software, XAMPP comes with a control panel you can use to start and stop Apache. Run it, and start Apache.

22. Write the syntax of PHP?

A PHP script can be placed anywhere in the document.

Canonical PHP Tags:

The script starts with **<?php** and ends with **?>** . These tags are also called 'Canonical PHP tags'.

A PHP script starts with <?php and ends with ?>:

Syntax

```
<?php
// PHP code goes here
?>
```

The default file extension for PHP files is ".php".

23. Difference between GET and POST Methods.

GET vs POST Methods

POST	GET
Values not visible in the URL	Values visible in the URL
Has not limitation of the length of the values since they are submitted via the body of HTTP	Has limitation on the length of the values usually 255 characters. This is because the values are displayed in the URL. Note the upper limit of the characters is dependent on the browser.
Has lower performance compared to Php_GET method due to time spent encapsulation the Php_POST values in the HTTP body	Has high performance compared to POST method dues to the simple nature of appending the values in the URL.
Supports many different data types such as string, numeric, binary etc.	Supports only string data types because the values are displayed in the URL
Results cannot be book marked	Results can be book marked due to the visibility of the values in the URL

PART C (3 * 8 = 24 Marks)

Answer ALL the Questions

24. a) Discuss about PHP inventions and versions.

The first version of what came to be known as PHP was created in 1995 by a man named Rasmus Lerdof. Rasmus, now an engineer at Yahoo!, needed something to make it easier to create content on his web site, something that would work well with HTML, yet give him power and flexibility beyond what HTML could offer him. Essentially, what he needed was an easy way to write scripts that would run on his web server both to create content, and handle data being passed back to the server from the web browser. Using the Perl language, he created some technology that gave him what he needed and decided to call this technology "Personal Home Page/Forms Interpreter". The technology provided a convenient way to process web forms and create content.

The name "Personal Home Page/Forms Interpreter" was later shortened to PHP/FI and eventually renamed to represent "PHP: Hypertext Preprocessor". The name is said to be recursive because the full name also includes the acronym "PHP" - an odd geeky joke that is common in technology circles when people have trouble naming things. GNU is another recursive name that represents "GNU's Not Unix".

PHP/FI version 1.0 was never really used outside of Rasmus' own web site. With the introduction of PHP/FI 2.0 this began to change. When PHP 3 was released in 1997, adoption of PHP exploded beyond all belief.

PHP 3 Hits the Big Time

By the time 1997 arrived the number of web sites on the internet was growing exponentially and most of these web sites were being implemented using the Apache web server. It was around this time that Andy Gutmans and Zeev Suraski launched the PHP 3 project, a project designed to take PHP to the next level. One of the key achievements of the PHP 3 project was to implement PHP as a robust Apache Module.

PHP 3 was implemented using a modular approach that made it easy for others to extend functionality, and also introduced the first elements of object-orientation that would continue to evolve through subsequent releases.

The combination of PHP 3 and Apache quickly lead to the widespread adoption of PHP, and it is commonly estimated that, at its peak adoption level, PHP3 was used to power over 10% of all web sites on the internet.

PHP 4 - Optimization, Scalability and More

With PHP 4 Andi Gutmans and Zeev Suraski once again re-architected PHP from the ground up. PHP 4 was built upon a piece of technology called the Zend Engine. The move to the Zend Engine brought about a number of key improvements in PHP:

- Support for other web servers (Microsoft's Internet Information Server (IIS) being of particular significance).
- Improved memory handling to avoid memory leaks (one of the most difficult types of problems to isolate in a program).
- Improved efficiency and performance to support large scale, complex, mission critical enterprise application development using PHP.

In addition PHP 4 also built on the earlier Object Oriented Programming features of PHP 3 with the introduction of classes.

PHP 5 - Object Orientation, Error Handling and XML

The main, though far from only, feature of PHP 5 is the improved support for Object Oriented Programming (OOP). In addition, PHP 5 introduced some features common in other languages such as Java like try/catch error and exception handling.

PHP 5 also introduced new extensions aimed at easing the storage and manipulation of data. Significant new features include SimpleXML for handling XML documents, and SQLite, an embedded basic and easy to use database interface.

PHP 6 : In this version, ICU (International Components for Unicode) library wasembedded into the program. But due to several other reasons this version was abandoned and was not launched in the market.

PHP 7 : This is the newest version of the php programming language having several features which were not present in the previous versions such as 64-bit integer support, return and scalar type declarations etc. It is powered by Zend Engine 3.

(OR)

b) How to declare a variable in PHP? What are the rules to be followed to declare the variable?

A variable in PHP is a name of memory location that holds data. A variable is a temporary storage that is used to store data temporarily.

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)

Syntax of declaring a variable in PHP is given below:

```
1. $variablename=value;
```

PHP VARIABLE: DECLARING STRING, INTEGER AND FLOAT

```
1. <?php
2. $str="hello string";
3. $x=200;
4. $y=44.6;
5. echo "string is: $str <br/>";
6. echo "integer is: $x <br/>";
7. echo "float is: $y <br/>";
8. ?>
```

Output:

```
string is: hello string
integer is: 200
float is: 44.6
```

25. a) Write a PHP Program to displays the reverse of provided string.

```
<html>
<head>
  <title></title>

  </head>
  <body>
  <form method='POST'>
  <h2>Please input your name:</h2>
 <input type="text" name="name">
 <input type="submit" value="Submit Name">
 </form>
<?php
//Retrieve name from query string and store to a local variable
$str1 = $_POST['name'];
function reverse($str1) {
 $n =strlen($str1);
if($n ==1)
   {
return $str1;
   }
else
   {
   $n--;
return reverse(substr($str1,1, $n)) .substr($str1, 0, 1);
   }
}
print_r(reverse($str1));
//echo "<h3> Hello $str1 </h3>";
?>
</body>
</html>
```

(OR)

b) What is Operator? Explain the operators in PHP with examples.

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators

PHP Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y

*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power (Introduced in PHP 5.6)

Example Program

```
<?php
  // variable 1
$x = 29;
  // variable 2
$y = 4;
  // some arithmetic operations on
// these two variables
echo ($x + $y), "\n";
echo($x - $y), "\n";
echo($x * $y), "\n";
echo($x / $y), "\n";
echo($x % $y), "\n";

?>
```

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the valu of the assignment expression on the right.

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication

x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

Example Program

```
<?php
// simple assign operator
$y = 75;
echo $y, "\n";
 // add then assign operator
$y = 100;
$y += 200;
echo $y, "\n";
// subtract then assign operator
$y = 70;
$y -= 10;
echo $y, "\n";
// multiply then assign operator
$y = 30;
$y *= 20;
echo $y, "\n";
  // Divide then assign(quotient) operator
$y = 100;
$y /= 5;
echo $y, "\n";
  // Divide then assign(remainder) operator
$y = 50;
$y %= 5;
echo $y;
?>
```

PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y

===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y

Example Program

```
<?php
$a = 80;
$b = 50;
$c = "80";
  // Here var_dump function has been used to
// display structured information. We will learn
// about this function in complete details in further
// articles.
var_dump($a == $c) + "\n";
var_dump($a != $b) + "\n";
var_dump($a <> $b) + "\n";
var_dump($a === $c) + "\n";
var_dump($a !== $c) + "\n";
var_dump($a < $b) + "\n";
var_dump($a > $b) + "\n";
var_dump($a <= $b) + "\n";
var_dump($a >= $b);

  ?>
```

PHP Increment / Decrement Operators

The PHP increment operators are used to increment a variable's value.

The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

Example Program

```
<?php

$x = 2;
echo ++$x, " First increments then prints \n";
echo $x, "\n";

$x = 2;
```

```
echo $x++, " First prints then increments \n";
echo $x, "\n";

$x = 2;
echo --$x, " First decrements then prints \n";
echo $x, "\n";

$x = 2;
echo $x--, " First prints then decrements \n";
echo $x;

?>
```

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

Example Program

```
<?php
  $x = 50;
$y = 30;
  if ($x == 50 and $y == 30)
    echo "and Success \n";
  if ($x == 50 or $y == 20)
    echo "or Success \n";
if ($x == 50 xor $y == 20)
    echo "xor Success \n";
  if ($x == 50 && $y == 30)
    echo "&& Success \n";
  if ($x == 50 || $y == 20)
    echo "|| Success \n";
  if (!$z)
    echo "! Success \n";
  ?>
```

PHP String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

Example Program

```
<?php

$x = "Geeks";
$y = "for";
$z = "Geeks!!!";
echo $x . $y . $z, "\n";
  $x .= $y . $z;
echo $x;
?>
```

PHP Array Operators

The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

Example Program

```
<?php
 $x = array("k" => "Car", "l" => "Bike");
$y = array("a" => "Train", "b" => "Plane");
 var_dump($x + $y);
var_dump($x == $y) + "\n";
var_dump($x != $y) + "\n";
var_dump($x <> $y) + "\n";
var_dump($x === $y) + "\n";
var_dump($x !== $y) + "\n";
?>
```

Conditional or Ternary Operators

These operators are used to compare two values and take either of the result simultaneously, depending on whether the outcome is TRUE or FALSE. These are also used as shorthand notation for if...else statement that we will read in the article on decision making.

Syntax:

$var = (condition)? value1 : value2;

Here, condition will either evaluate to true or false. If the condition evaluates to True, then value1 will be assigned to the variable $var otherwise value2 will be assigned to it.

Operator	Name	Operation
?:	Ternary	If condition is true ? then $x : or else $y. This means that if condition is true then left result of the colon is accepted otherwise the result on right.

Example Program

```
<?php
 $x = -12;
 echo ($x > 0) ? 'The number is positive' : 'The number is negative';
 ?>
```

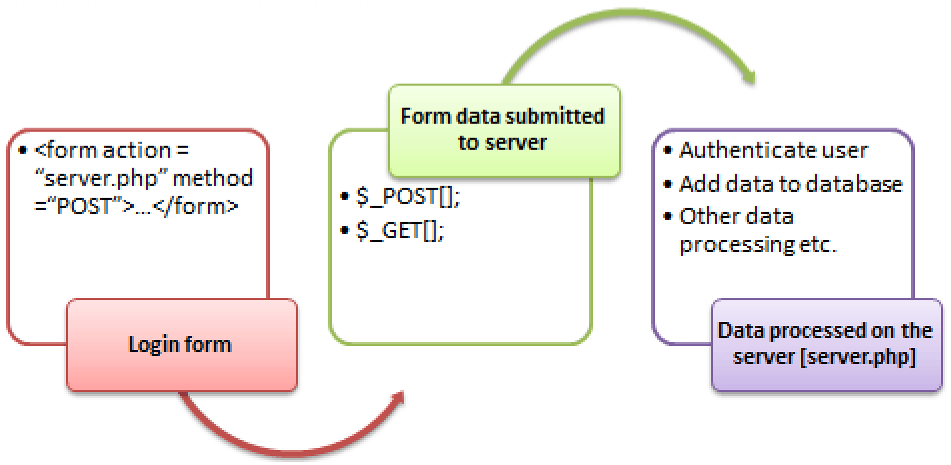
26. a) How to handle HTML form with PHP? Explain with example.

What is Form?

When you login into a website or into your mail box, you are interacting with a form.

Forms are used to get input from the user and submit it to the web server for processing.

The diagram below illustrates the form handling process.


• <form action = "server.php" method ="POST">...</form>
Login form
Form data submitted to server
• $_POST[];
• $_GET[];
• Authenticate user
• Add data to database
• Other data processing etc.
Data processed on the server [server.php]

A form is an HTML tag that contains graphical user interface items such as input box, check boxes radio buttons etc.

The form is defined using the <form>…</form> tags and GUI items are defined using form elements such as input.

When and why we are using forms?

- Forms come in handy when developing flexible and dynamic applications that accept user input.
- Forms can be used to edit already existing data from the database

Create a form

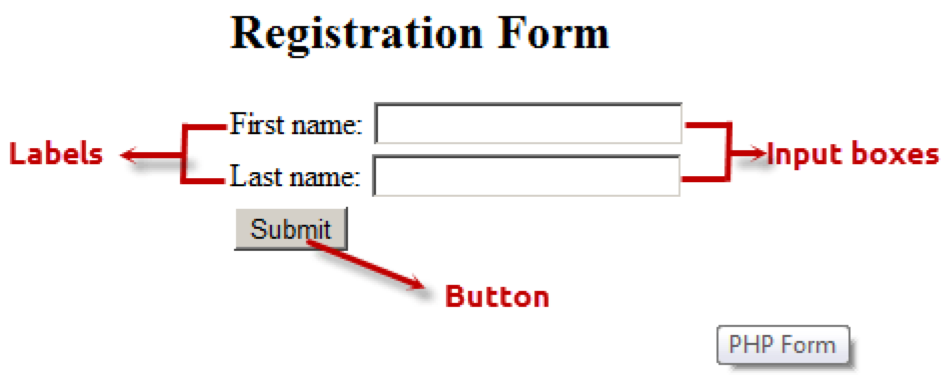
We will use HTML tags to create a form. Below is the minimal list of things you need to create a form.

- Opening and closing form tags <form>…</form>
- Form submission type POST or GET
- Submission URL that will process the submitted data
- Input fields such as input boxes, text areas, buttons,checkboxes etc.

The code below creates a simple registration form

```
<html>
<head>
        <title>Registration Form</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>

  <h2>Registration Form</h2>

  <form action="registration_form.php" method="POST"> First name:

    <input type="text" name="firstname"> <br> Last name:

    <input type="text" name="lastname">

    <input type="hidden" name="form_submitted" value="1" />

    <input type="submit" value="Submit">

  </form>
</body>
</html>
```

Viewing the above code in a web browser displays the following form.



HERE,

- <form...>...</form> are the opening and closing form tags
- action="registration_form.php" method="POST"> specifies the destination URL and the submission type.
- First/Last name: are labels for the input boxes
- <input type="text"...> are input box tags
- <br> is the new line tag
- <input type="hidden" name="form_submitted" value="1"/> is a hidden value that is used to check whether the form has been submitted or not
- <input type="submit" value="Submit"> is the button that when clicked submits the form to the server for processing

Submitting the form data to the server

The action attribute of the form specifies the submission URL that processes the data. The method attribute specifies the submission type.

PHP POST method

- This is the built in PHP super global array variable that is used to get values submitted via HTTP POST method.
- The array variable can be accessed from any script in the program; it has a global scope.
- This method is ideal when you do not want to display the form post values in the URL.
- A good example of using post method is when submitting login details to the server.

It has the following syntax.

```
<?php
 $_POST['variable_name'];
?>
```

HERE,

- “$_POST[…]” is the PHP array
- “'variable_name'” is the URL variable name.

PHP GET method

- This is the built in PHP super global array variable that is used to get values submitted via HTTP GET method.
- The array variable can be accessed from any script in the program; it has a global scope.
- This method displays the form values in the URL.
- It’s ideal for search engine forms as it allows the users to book mark the results.

It has the following syntax.

```
<?php
$_GET['variable_name'];
?>
```

HERE,

- “$_GET[…]” is the PHP array
- “'variable_name'” is the URL variable name.

(OR)

b) How to declare the multi value fields in PHP?

Working with check boxes, radio buttons

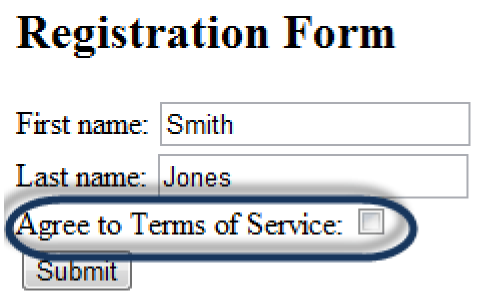
If the user does not select a check box or radio button, no value is submitted, if the user selects a check box or radio button, the value one (1) or true is submitted.

We will modify the registration form code and include a check button that allows the user to agree to the terms of service.

```
<html>
<head>
		<title>Registration Form</title>
		<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>

   <?php if (isset($_POST['form_submitted'])): ?>

      <?php if (!isset($_POST['agree'])): ?>
```

```
        <p>You have not accepted our terms of service</p>

        <?php else: ?>

           <h2>Thank You <?php echo $_POST['firstname']; ?></h2>

           <p>You have been registered as
              <?php echo $_POST['firstname'] . ' ' . $_POST['lastname']; ?>
           </p>

           <p> Go <a href="/registration_form2.php">back</a> to the form</p>

        <?php endif; ?>

        <?php else: ?>

                 <h2>Registration Form</h2>

                 <form action="registration_form2.php" method="POST">

                    First name:
                    <input type="text" name="firstname">

                    <br> Last name:
                    <input type="text" name="lastname">

                    <br> Agree to Terms of Service:
                    <input type="checkbox" name="agree">
                    <br>

                    <input type="hidden" name="form_submitted" value="1" />

                    <input type="submit" value="Submit">

                 </form>
           <?php endif; ?>
</body>
</html>
```

View the above form in a browser



Registration Form

First name: Smith

Last name: Jones

Agree to Terms of Service: ☐

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021

Computer Technology/Information Technology
Sixth Semester
SECOND INTERNAL EXAMINATION - February 2019

PHP PROGRAMMING

Class & Section: III B.Sc CT/ B.Sc (IT A&B) **Duration: 2 hours**
Date & Session: 04.02.19 (FN) **Maximum marks: 50 marks**
Sub.Code: 16CTU601A/16ITU601A

PART- A (20 * 1= 20 Marks)
Answer ALL the Questions

1. PHP___ statement is executed if condition is true
   a) if-else-if-else b) for each c) if d)while
2. PHP ____statement breaks the execution of current loop.
   a) goto b) break c) continue d) while
3. When the keyword _____executed inside a loop the control automatically passes to the beginning of loop
   a) goto b)break c)switch d) continue
4. ____ is the operator of unconditional transition.
   a) goto b) break c) continue d) switch
5. ___ loop should be used if number of iteration is known.
   a) for b) for each c) while d) do while
6. The PHP do-while loop is guaranteed to run at least ____times.
   a)once b)twice c)thrice d) five
7. The PHP break statement breaks the execution of____ loop only.
   a) inner b) outer c) whole program d) none
8. Most complicated looping structure is
   a)for b) do c)for…while d) switch
9. Which one of the following PHP functions can be used to build a function that accepts any number of arguments
   a)func_get_argv() b)func_get_argc() c)get_argv() d)get_argc()
10. Which one of the following PHP functions can be used to find files?
    a)glob() b)file() c)fold() d)get_file()
11. The filesize() function returns the file size in ___.
    a)bits b)bytes c)kilobytes d) gigabytes
12. Which one of the following PHP function is used to determine a file's last access time?
    a)fileltime() b)filectime() c)fileatime() d)filetime()
13. Which one of the following function is capable of reading a file into an array?
    a)file() b)arrfile() c)arr_file() d)file_arr()

14. The function func_num_args() returns
    a)the number of arguments passed to the function
    b)the total length of the arguments
    c)the number of lines in the program
    d)the number of variables used in the program
15. All these func_num_args ( ), func_get_arg ( ), func_get_args ( ), functions are introduced in
    a)PHP1 b)PHP2 c)PHP3 d)PHP4
16. PHP function arguments are modified in function definition and
    a)In a function call b)In execution time c)In deceleration time d)None of them
17. ____ scope refers to any variable that is defined outside of any function
    a)Local b)Global c)Static d) parameters
18. A parameter is a______variable whose value is passed to the function by the calling code.

    a)Local b)Global c)Static d) parameters
19. A static variable is again a variable with______scope
    a)Global b)Static c) parameters d) local
20. The _____ represents reference of the variable.
    a)$ b)& c)* d)@

PART B (3 * 2 = 6 Marks)
Answer ALL the Questions

21. What is the difference between break and exit?
22. Define Function.
23. How to declare the function?

PART C (3 * 8 = 24 Marks)
Answer ALL the Questions

24. a) Explain in detail about conditional statements in PHP with example..
(Or)
b) Discuss on different types of looping statements in PHP.

25. a) Explain in detail about PHP Function with arguments with example program
(Or)
b) Elucidate on Scope of Function in Global and Local.

26. a) Discuss about Default Arguments in Function with example.
(Or)
b) Describe on Function argument with call by value, call by reference.

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021

Computer Technology/Information Technology
Sixth Semester
SECOND INTERNAL EXAMINATION - February 2019

PHP PROGRAMMING

Class & Section: III B.Sc CT/ B.Sc (IT A&B) **Duration: 2 hours**
Date & Session: 4.02.19 (FN) **Maximum marks: 50 marks**
Sub.Code: 16CTU601A/16ITU601A

PART- A (20 * 1= 20 Marks)
Answer ALL the Questions

1. PHP___ statement is executed if condition is true
   a) if-else-if-else b) for each **c) if** d) while
2. PHP ____statement breaks the execution of current loop.
   a) goto **b) break** c) continue d) while
3. When the keyword _____executed inside a loop the control automatically passes to the beginning of loop
   a) goto b)break c)switch **d) continue**
4. ____ is the operator of unconditional transition.
   a) goto b) break c) continue d) switch
5. ___ loop should be used if number of iteration is known.
   a) for b) for each c) while d) do while
6. The PHP do-while loop is guaranteed to run at least ____times.
   a)once b)twice c)thrice d) five
7. The PHP break statement breaks the execution of____ loop only.
   a) inner b) outer c) whole program d) none
8. Most complicated looping structure is
   a)for b) do c)for…while d) switch
9. Which one of the following PHP functions can be used to build a function that accepts any number of arguments
   a)func_get_argv() **b)func_get_argc()** c)get_argv() d)get_argc()
10. Which one of the following PHP functions can be used to find files?
    a)glob() b)file() c)fold() d)get_file()
11. The filesize() function returns the file size in ___.
    a)bits **b)bytes** c)kilobytes d) gigabytes
12. Which one of the following PHP function is used to determine a file's last access time?
    a)fileltime() b)filectime() **c)fileatime()** d)filetime()
13. Which one of the following function is capable of reading a file into an array?
    a)file() b)arrfile() c)arr_file() d)file_arr()

14. The function func_num_args() returns
    a)the number of arguments passed to the function
    b)the total length of the arguments
    c)the number of lines in the program
    d)the number of variables used in the program
15. All these func_num_args ( ), func_get_arg ( ), func_get_args ( ), functions are introduced in
    a)PHP1 b)PHP2 c)PHP3 **d)PHP4**
16. PHP function arguments are modified in function definition and
    a)In a function call b)In execution time c)In deceleration time d)None of them
17. ____ scope refers to any variable that is defined outside of any function
    a)Local **b)Global** c)Static d) parameters
18. A parameter is a______variable whose value is passed to the function by the calling code.

    a)Local b)Global c)Static d) parameters
19. A static variable is again a variable with______scope
    a)Global b)Static c) parameters **d) local**
20. The _____ represents reference of the variable.
    a)$ **b)&** c)* d)@

PART B (3 * 2 = 6 Marks)
Answer ALL the Questions

21. What is the difference between break and exit?
    The major **difference between break** and **exit**() is that **break** is a keyword, which causes an immediate **exit** from the switch or loop (for , while or do), while **exit**() is a standard library function, which terminates program execution when it is called. ...**exit**() terminates program execution when it is called.

22. Define Function.

Functions are basically named scripts that can be called upon from any other script to perform a specific task. Values (known as arguments) can be passed into a function so that they can be used in the function script, and functions can, in turn, return results to the location from which they were called.

23. How to declare the function?
    Declaration and Calling of a Function-in PHP
    A user-defined function declaration starts with the word function:
    Syntax
```
function functionName() {
   code to be executed;
}
```
    The declaration of a user-defined function start with the word function, followed by the name of the function you want to create followed by parentheses i.e. () and finally place your function's code between curly brackets {}.
    Note: A function name can start with a letter or underscore (not a number).
    Example
```
<?php
```

```
function writeMsg() {
   echo "Hello world!";
}
writeMsg(); // call the function
?>
```

PART C (3 * 8 = 24 Marks)
Answer ALL the Questions

24. a) Explain in detail about conditional statements in PHP with example.

PHP Conditional Statements

Conditional statements are used to perform different actions based on different conditions.

In PHP we have the following conditional statements:

1. if statement - executes some code if one condition is true
2. if...else statement - executes some code if a condition is true and another code if that condition is false
3. if...elseif....else statement - executes different codes for more than two conditions
4. switch statement - selects one of many blocks of code to be executed

PHP - The if Statement

The if statement executes some code if one condition is true.

Syntax

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

The example below will output "Have a good day!" if the current time (HOUR) is less than 20:

Example

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
}
?>
```

PHP - The if...else Statement

The if....else statement executes some code if a condition is true and another code if that condition is false.

Syntax

```
if (condition) {
    code to be executed if condition is true;
} else {
    code to be executed if condition is false;
}
```

The example below will output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

Example

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

PHP - The if...elseif....else Statement

The if....elseif...else statement executes different codes for more than two conditions.

Syntax

```
if (condition) {
    code to be executed if this condition is true;
} elseif (condition) {
    code to be executed if this condition is true;
} else {
    code to be executed if all conditions are false;
}
```

The example below will output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

Example

```
<?php
$t = date("H");

if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

PHP - The switch Statement

The switch statement is used to perform different actions based on different conditions.

The PHP switch Statement

Use the switch statement to select one of many blocks of code to be executed.

Syntax

```
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;
}
```

This is how it works: First we have a single expression n (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use break to prevent the code from running into the next case automatically. The default statement is used if no match is found.

Example

```
<?php
$favcolor = "red";

switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

(Or)

b) Discuss on different types of looping statements in PHP.

PHP Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal code-lines in a script, we can use loops to perform a task like this.

In PHP, we have the following looping statements:

1. while - loops through a block of code as long as the specified condition is true
2. do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true
3. for - loops through a block of code a specified number of times
4. foreach - loops through a block of code for each element in an array

The PHP while Loop

The while loop executes a block of code as long as the specified condition is true.

Syntax

```
while (condition is true) {
    code to be executed;
}
```

The example below first sets a variable $x to 1 ($x = 1). Then, the while loop will continue to run as long as $x is less than, or equal to 5 ($x <= 5). $x will increase by 1 each time the loop runs ($x++):

Example

```
<?php
$x = 1;

while($x <= 5) {
    echo "The number is: $x <br>";
    $x++;
}
?>
```

do...while Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax

```
do {
    code to be executed;
} while (condition is true);
```

The example below first sets a variable $x to 1 ($x = 1). Then, the do while loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

Example

```
<?php
$x = 1;
```

```
do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

do while loop the condition is tested AFTER executing the statements within the loop. This means that the do while loop would execute its statements at least once, even if the condition is false the first time.

The example below sets the $x variable to 6, then it runs the loop, and then the condition is checked:

Example

```
<?php
$x = 6;
do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

The PHP for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (init counter; test counter; increment counter) {
    code to be executed;
}
```

Parameters:

- init counter: Initialize the loop counter value
- test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- increment counter: Increases the loop counter value
- The example below displays the numbers from 0 to 10:

Example

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

PHP foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

```
foreach ($array as $value) {
    code to be executed;
}
```

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

The following example demonstrates a loop that will output the values of the given array ($colors):

Example

```
<?php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
    echo "$value <br>";
}
?>
```

25. a) Explain in detail about PHP Function with arguments with example program

PHP Function Arguments

Information can be passed to functions through arguments. An argument is just like a variable.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

Example
```
<?php
function familyName($fname) {
   echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

The following example has a function with two arguments ($fname and $year):
```
<?php
function familyName($fname, $year) {
   echo "$fname Refsnes. Born in $year <br>";
}
familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```
PHP Default Argument Value
The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

Example
```
<?php
function setHeight($minheight = 50) {
   echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

Function argument with call by value
Call by value means passing the value directly to a function. The called function uses the value in a local variable; any changes to it do not affect the source variable.
Example

This is a "global", or automatic global, variable. This simply means that it is available in all scopes throughout a script. There is no need to do global $x; to access it within functions or methods.

```
<?php
//Call by value program
function abc($x)
{
$x=$x+10;
return($x);
}
$a=20;
echo abc($a)."<br>";
echo ($a);
?>
```

Note: Call by value: in the call by value method, the called function creates a new set of variables and copies the values of arguments into them.

Function argument with call by reference

Call by reference means passing the address of a variable where the actual value is stored. The called function uses the value stored in the passed address; any changes to it do affect the source variable.

```
<?php
//call by reference program in php
function abc($x)
{
$x=$x-10;
return($x);
}
$a=50;
echo abc($a)."<br>";
echo ($a);
?>
```

Note: Call by reference: in the call by reference method, instead of passing a value to the function being called a reference/pointer to the original variable is passed.

(Or)

b) Elucidate on Scope of Function in Global and Local.

SCOPE OF FUNCTION GLOBAL AND LOCAL

1. Variables declared outside of functions and classes are global. Global variables are available else where in the script.
2. Function variables are self-contained and do not affect variables in the main script.
3. Variables from the main script are not implicitly made available inside functions.

Example

Take a look at this example:

```
<?PHP
function foo() {
  $bar = "java2s.com";
}
$bar = "PHP";
foo();
print $bar;
?>
```

The code above generates the following result.

```
PHP
```

- Execution of the script starts at the $bar = "PHP" line, and then calls the foo() function.
- foo() sets $bar to java2s.com, then returns control to the main script where $bar is printed out.
- Function foo() is called, and, having no knowledge that a $bar variable exists in the global scope, creates a $bar variable in its local scope.
- Once the function ends, all local scopes are gone, leaving the original $bar variable intact.

PHP Global Variables

A global variable can be accessed anywhere in your script, whether inside or outside a function.

In PHP, all variables created outside a function are, in a sense, global in that they can be accessed by any other code in the script that's not inside a function.

To use such a variable inside a function, write the word global followed by the variable name inside the function ' s code block.

```
<?PHP
      $myGlobal = "Hello there!";

      function hello() {
        global $myGlobal;
        echo "$myGlobal\n";
      }
      hello(); // Displays "Hello there!"
?>
```

The code above generates the following result.

Hello there!

hello() function accesses the $myGlobal variable by declaring it to be global using the global statement. The function can then use the variable to display the greeting.

Example 1

Created a variable outside a function to use it as a global variable. Take a look at the following script:

```
<?PHP
        function setup() {
          global $myGlobal;
          $myGlobal = "Hello there!";
        }
        function hello() {
          global $myGlobal;
          echo "$myGlobal\n";
        }
        setup();
        hello(); // Displays "Hello there!"
?>
```

The code above generates the following result.

Hello there!

In this script, the setup() function is called first. It declares the $myGlobal variable as global, and gives it a value.

Then the hello() function is called. It too declares $myGlobal to be global, which means it can now access its value previously set by setup() and display it.

Example 2

The $GLOBALS array can access global variables within functions. All variables declared in the global scope are in the $GLOBALS array, which you can access anywhere in the script. Here is a demonstration:

```
<?PHP
function foo() {
   $GLOBALS['bar'] = "java2s.com";
}

$bar = "PHP";
foo();
print $bar;
?>
```

The code above generates the following result.

```
java2s.com
```

We can read variables in the same way:

```
$localbar = $GLOBALS['bar'];
```

PHP GLOBAL keyword allow a variable to be accessed locally.

```
function myfunc() {
   GLOBAL $foo, $bar, $baz;
   ++$baz;
}
```

The code above reads the global variables $foo, $bar, and $baz. The ++$baz line will increment $baz by 1, and this will be reflected in the global scope.

Note

We can also declare more than one global variable at once on the same line, just separate the variables using commas:

```
function myFunction() {
  global $oneGlobal, $anotherGlobal;
}
```

Be careful with global variables. If you modify the value of a global variable in many different places within your application, it can make it hard to debug your code.

26. a) Discuss about Default Arguments in Function with example.

PHP Function: Default Argument Value

We can specify a default argument value in function. While calling PHP function if you don't specify any argument, it will take the default argument. Let's see a simple example of using default argument value in PHP function.

File: functiondefaultarg.php

```
<?php
function sayHello($name="Sonoo"){
echo "Hello $name<br/>";
}
sayHello("Rajesh");
sayHello();//passing no value
sayHello("John");
?>
```

Output:

```
Hello Rajesh
Hello Sonoo
Hello John
```

(Or)

b) Describe on Function argument with call by value, call by reference.

Function argument with call by value

Call by value means passing the value directly to a function. The called function uses the value in a local variable; any changes to it **do not** affect the source variable.

Example

This is a "global", or automatic global, variable. This simply means that it is available in all scopes throughout a script. There is no need to do global $x; to access it within functions or methods.

```
<?php
//Call by value program
function abc($x)
{
$x=$x+10;
return($x);
}
$a=20;
```

```
echo abc($a)."<br>";
echo ($a);
?>
```

Note: Call by value: in the call by value method, the called function creates a new set of variables and copies the values of arguments into them.

Function argument with call by reference

Call by reference means passing the address of a variable where the actual value is stored. The called function uses the value stored in the passed address; any changes to it **do** affect the source variable.

```
<?php
//call by reference program in php
function abc($x)
{
$x=$x-10;
return($x);
}
$a=50;
echo abc($a)."<br>";
echo ($a);
?>
```

Note: Call by reference: in the call by reference method, instead of passing a value to the function being called a reference/pointer to the original variable is passed.

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021

Computer Technology
Sixth Semester
THIRD INTERNAL EXAMINATION – March 2018

PHP PROGRAMMING

Class & Section: III B.Sc CT	**Duration: 2 hours**
Date & Session: 11.3.19 (FN)	**Maximum marks: 50 marks**
Sub.Code: 16CTU601A	

PART- A (20 * 1= 20 Marks)
Answer ALL the Questions

1. PHP only supports a ____character set.
   a) 256 b) 128 c)64 d)32
2. PHP string can be as large as ____.
   a) "less than 100MB" b) 1KB c)2GB d)"less than 2GB"
3. _____the easiest way to specify string in PHP.
   a) single quoted b) double quoted c) heredoc d) newdoc
4. A placeholder is inserted after the____ sign in a sprint ().
   a) & b) % c) * d) $
5. A _____ is a sequence of letters, numbers, special characters and arithmetic values or combination of all.
   a) array b) string c) structures d) functions
6. The ______ parameter cannot be an empty string.
   a) separator b) function c) string d) format
7. The______ function returns string in lowercase letter.
   a) lower() b) strtolower() c)str_low() d)string()
8. The_____ function returns string converting first character into uppercase.
   a) upper() b) ucase() c) strtoupper() d) ucfirst()
9. Through _____ function you can display or extract a string from a particular position.
   a) substr() b) extract() c) sub() d) sub_ext()
10. ____is dedicated to remove white spaces and predefined characters from a both the sides of a string.
    a) remove() b) sub() c) white() d) Trim()
11. _____function enables to display of the number of words in any specific string.
    a) str_word() b) count() c)str_word_count() d)word()
12. ______ arrays are with named keys.
    a) associative b) indexed c) multiple d) declarative

13. In index array, the index always starts at ___.

a)0	b) 1	c)2	d)3

14. The ___ function is used to return the length of an array.

a) length()	b) term()	c) index()	d) count ()

15. An ____ is a special variable, which can hold more than one value at a time.

a) literal	b) array	c) keywords	d) index

16. You can access the array values by referring to an____.

a) counter	b) keywords	c) index number	d) variable

17. The ______function is used to create an array.

a) array()	b) arr()	c) new()	d) create ()

18. ______arrays are with a numeric index.

a) associative	b) indexed	c) multiple	d) declarative

19. We can define ____ arrays as array of arrays.

a) associative	b) indexed	c) declarative	d) Multidimensional

20. Array elements can be accessed using the ___ syntax.

a) index[key]	b) array[key]	c) element[key]	d) access[key]

PART B (3 * 2 = 6 Marks)
Answer ALL the Questions

21. How to create a String?
22. What is the use of regular expression?
23. Differentiate between each () and foreach ()?

PART C (3 * 8 = 24 Marks)
Answer ALL the Questions

24. a) Explain in detail about Searching & Replacing String-Formatting.
    (Or)
    b) Describe about joining and splitting String in PHP

25. a) Discuss on preg_match(), preg_replace(), preg_split() functions in regular expression.
    (Or)
    b) Describe about Associative array in PHP.
26. a) How to use looping with Index based array and explain it.
    (Or)
    b) Explain in detail about associative array using each() and foreach().

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021

Computer Technology
Sixth Semester
THIRD INTERNAL EXAMINATION - March2018

PHP PROGRAMMING

Class & Section: III B.Sc CT **Duration: 2 hours**
Date & Session: 11.3.19 (FN) **Maximum marks: 50 marks**
Sub.Code: 16CTU601A

PART- A (20 * 1= 20 Marks)
Answer ALL the Questions

1. PHP only supports a ____character set.
   a) 256 b) 128 c)64 d)32
2. PHP string can be as large as ____.
   a) "less than 100MB" b) 1KB **c)2GB** d)"less than 2GB"
3. _____the easiest way to specify string in PHP.
   a) single quoted b) double quoted c)heredoc d)newdoc
4. A placeholder is inserted after the____ sign in a sprint().
   a)& **b)%** c)* d)$
5. A _____ is a sequence of letters, numbers, special characters and arithmetic values or combination of all.
   a) array **b) string** c)structures d)functions
6. The ______ parameter cannot be an empty string.
   a)separator b)function c)string d)format
7. The______ function returns string in lowercase letter.
   a)lower() **b)strtolower()** c)str_low() d)string()
8. The_____ function returns string converting first character into uppercase.
   a)upper() b)ucase() c)strtoupper() **d)ucfirst()**
9. Through _____ function you can display or extract a string from a particular position.
   a)substr() b)extract() c)sub() d)sub_ext()
10. ____is dedicated to remove white spaces and predefined characters from a both the sides of a string.
    a)remove() b)sub() c)white() **d)Trim()**
11. _____function enables to display of the number of words in any specific string .
    a)str_word() b) count() **c)str_word_count()** d)word()
12. ______ arrays are with named keys.
    a)associative b) indexed c)multiple d)declarative

13. In index array , the index always starts at ___.
    a)0 b) 1 c)2 d)3
14. The ___ function is used to return the length of an array.
    a)length() b)term() c)index() **d)count()**
15. An ____ is a special variable, which can hold more than one value at a time.
    a) literal **b) array** c)keywords d)index
16. You can access the array values by referring to an____.
    a) counter b) keywords **c)index number** d)variable
17. The ______function is used to create an array.
    a) array() b) arr() c)new() d)create()
18. ______arrays are with a numeric index.
    a) associative **b) indexed** c)multiple d)declarative

19. We can define ____ arrays as array of arrays.
    a)associative b)indexed c)declarative **d)Multidimensional**
20. Array elements can be accessed using the ___ syntax.
    a)index[key] **b)array[key]** c)element[key] d)access[key]

PART B (3 * 2 = 6 Marks)
Answer ALL the Questions

21. How to create a String?
    The simplest way to create a string is to enclose the string literal (i.e. string characters) in single quotation marks ('), like this:

```
$my_string = 'Hello World';
```

22. What is the use of regular expression?
    Regular expressions simplify identifying patterns in string data by calling a single function. This saves us coding time. When validating user input such as email address, domain names, telephone numbers, IP addresses, Highlighting keywords in search results, When creating a custom HTML template.

    Regular expressions can be used to identify the template tags and replace them with actual data.

23. Differenciate between each() and foreach()?

The each() function returns the current element key and value, and moves the internal pointer forward.

The `foreach` loop works only on arrays, and is used to loop through each key/value pair in an array.

PART C (3 * 8 = 24 Marks)
Answer ALL the Questions

24. a)Explain in detail about Searching & Replacing String-Formatting.

Searching & Replacing String

PHP provides various string functions to access and manipulate strings.

Counting of the number of words in a String

Another function which enables display of the number of words in any specific string is str_word_count(). This function is also useful in validation of input fields.

Syntax

Str_word_count(string)

Example

```
<?php
echostr_word_count(“Welcome to Cloudways”);//will return the number of words in a string
?>
```

Output

```
3
```

Finding Text Within a String

Strpos() enables searching particular text within a string. It works simply by matching the specific text in a string. If found, then it returns the specific position. If not found at all, then it will return “False”. Strpos() is most commonly used in validating input fields like email.

Syntax

```
Strpos(string,text);
```

Example

```
<?php
echostrpos(“Welcome to Cloudways”,”Cloudways”);
?>
```

Output

```
11
```

Replacing text within a string

Str_replace() is a built-in function, basically used for replacing specific text within a string.

Syntax

```
Str_replace(string to be replaced,text,string)
```

Example

```
<?php
echostr_replace(“cloudways”, “the programming world”, “Welcome to cloudways”);
?>
```

Output

```
Welcome to the programming world
```

Repeating a String

PHP provides a built-in function for repeating a string a specific number of times.

Syntax

Str_repeat(string,repeat)

Example

```
<?php
echostr_repeat(“=”,13);
?>
```

Output

```
=============
```

Comparing Strings

You can compare two strings by using strcmp(). It returns output either greater than zero, less than zero or equal to zero. If string 1 is greater than string 2 then it returns greater than zero. If string 1 is less than string 2 then it returns less than zero. It returns zero, if the strings are equal.

Syntax

Strcmp(string1,string2)

Example

```
<?php
echostrcmp(“Cloudways”,”CLOUDWAYS”);
echo “<br>”;
echostrcmp(“cloudways”,”cloudways”);//Both the strings are equal
echo “<br>”;
echostrcmp(“Cloudways”,”Hosting”);
echo “<br>”;
echostrcmp(“a”,”b”);//compares alphabetically
```

```
echo "<br>";
echostrcmp("abb baa","abb baa caa");//compares both strings and returns the result in terms of number of characters.
?>
```

Output

```
1
0
-1
-1
-4
```

Displaying part of String

Through substr() function you can display or extract a string from a particular position.

Syntax

```
substr(string,start,length)
```

Example

```
<?php
echosubstr("Welcome to Cloudways",6)."<br>";
echosubstr("Welcome to Cloudways",0,10)."<br>";
?>
```

Output

```
e to Cloudways
Welcome to
s
```

Removing white spaces from a String

Trim() is dedicated to remove white spaces and predefined characters from a both the sides of a string.

Syntax

trim(string,charlist)

Example

<?php

$str = "Wordpess Hosting";

echo $str . "<br>";

echo trim("$str","Wording");

?>

Output

Wordpess Hosting

pess Host

(Or)

b)Describe about joining and splitting String in PHP

PHP `join()` Function

Join array elements with a string. The join() function returns a string from the elements of an array.

The join() function is an alias of the `implode()` function.

Note: The join() function accept its parameters in either order. However, for consistency with explode(), you should use the documented order of arguments.

Note: The separator parameter of join() is optional. However, it is recommended to always use two parameters for backwards compatibility.

Syntax

join(*separator,array*)

Parameter	Description
separator	Optional. Specifies what to put between the array elements. Default is "" (an empty string)
array	Required. The array to join to a string

```
<?php
$arr = array('Hello','World!','Beautiful','Day!');
echo join(" ",$arr);
?>
```

Separate the array elements with different characters:

```
<?php
$arr = array('Hello','World!','Beautiful','Day!');
echo join(" ",$arr)."<br>";
echo join("+",$arr)."<br>";
echo join("-",$arr)."<br>";
echo join("X",$arr);
?>
```

`explode()` Function

The explode() function breaks a string into an array.

Note: The "separator" parameter cannot be an empty string.

Note: This function is binary-safe.

Syntax

explode(*separator,string,limit*)

Parameter	Description
separator	Required. Specifies where to break the string
string	Required. The string to split
limit	Optional. Specifies the number of array elements to return. Possible values: Greater than 0 - Returns an array with a maximum of *limit*element(s) Less than 0 - Returns an array except for the last *-limit*elements() 0 - Returns an array with one element

Example: Break a string into an array:

```
<?php
$str = "Hello world. It's a beautiful day.";
print_r (explode(" ",$str));
?>
```

Example: Using the limit parameter to return a number of array elements:

```
<?php
$str = 'one,two,three,four';

// zero limit
print_r(explode(',',$str,0));

// positive limit
print_r(explode(',',$str,2));
```

```
// negative limit
print_r(explode(',',$str,-1));
?>
```

String Related Library functions

PHP strtolower() function

The strtolower() function returns string in lowercase letter.

Syntax

```
string strtolower ( string $string )
```

Example

```
<?php
$str="My name is KHAN";
$str=strtolower($str);
echo $str;
?>
```

Output:

```
my name is khan
```

PHP strtoupper() function

The strtoupper() function returns string in uppercase letter.

Syntax

```
string strtoupper ( string $string )
```

Example

```
<?php
$str="My name is KHAN";
$str=strtoupper($str);
echo $str;
?>
```

Output:

```
MY NAME IS KHAN
```

PHP ucfirst() function

The ucfirst() function returns string converting first character into uppercase. It doesn't change the case of other characters.

Syntax

```
string ucfirst ( string $str )
```

Example

```
<?php
$str="my name is KHAN";
$str=ucfirst($str);
echo $str;
?>
```

Output:

```
My name is KHAN
```

PHP lcfirst() function

The lcfirst() function returns string converting first character into lowercase. It doesn't change the case of other characters.

Syntax

string lcfirst (string $str)

Example

```
<?php
$str="MY name IS KHAN";
$str=lcfirst($str);
echo $str;
?>
```

Output:

```
mY name IS KHAN
```

PHP ucwords() function

The ucwords() function returns string converting first character of each word into uppercase.

Syntax

string ucwords (string $str)

Example

```
<?php
$str="my name is Sonoo jaiswal";
$str=ucwords($str);
echo $str;
?>
```

Output:

```
My Name Is SonooJaiswal
```

PHP strrev() function

The strrev() function returns reversed string.

Syntax

string strrev (string $string)

Example

```
<?php
$str="my name is Sonoo jaiswal";
$str=strrev($str);
echo $str;
?>
```

Output:

```
lawsiajoonoSsiemanym
```

PHP strlen() function

The strlen() function returns length of the string.

Syntax

int strlen (string $string)

Example

```
<?php
$str="my name is Sonoo jaiswal";
$str=strlen($str);
echo $str;
?>
```

Output:

```
24
```

25. a)Discuss on preg_match(), preg_replace(), preg_split() functions in regular expression.

preg_match – this function is used to perform a pattern match on a string. It returns true if a match is found and false if a match is not found.

preg_split – this function is used to perform a pattern match on a string and then split the results into a numeric array

preg_replace – this function is used to perform a pattern match on a string and then replace the match with the specified text.

Below is the syntax for a regular expression function such as preg_match,preg_split or

preg_replace.

```
<?php
function_name('/pattern/',subject);
?>
```

HERE,

"function_name(...)" is either preg_match, preg_split or preg_replace.

"/.../" The forward slashes denote the beginning and end of our regular expression

"'/pattern/'" is the pattern that we need to matched

"subject" is the text string to be matched against

Let’s now look at practical examples that implement the above regular expression functions in PHP.

PHP Preg_match

The first example uses the preg_match function to perform a simple pattern match for the word guru in a given URL.

The code below shows the implementation for the above example.

```
<?php
```

```
$my_url = "www.guru99.com";
if (preg_match("/guru/", $my_url))
{
        echo "the url $my_url contains guru";
}
else
{
        echo "the url $my_url does not contain guru";
}
?>
```

Let’s examine the part of the code responsible for our output "*preg_match('/guru/', $my_url)*" HERE,

- "preg_match(...)" is the PHP regular expression function
- "'/guru/'" is the regular expression pattern to be matched
- "$my_url" is the variable containing the text to be matched against.

PHP Preg_split

Let’s now look at another example that uses the preg_split function.

We will take a string phrase and explode it into an array; the pattern to be matched is a single space.

The text string to be used in this example is "I Love Regular Expressions".

The code below illustrates the implementation of the above example.

```
<?php

$my_text="I Love Regular Expressions";

$my_array  =preg_split("/ /", $my_text);

print_r($my_array );
```

```
?>
```

PHP Preg_replace

Let’s now look at the preg_replace function that performs a pattern match and then replaces the pattern with something else.

The code below searches for the word guru in a string.

It replaces the word guru with the word guru surrounded by css code that highlights the background colour.

```
<?php
$text = "We at Guru99 strive to make quality education affordable to the masses. Guru99.com";

$text = preg_replace("/Guru/", '<span style="background:yellow">Guru</span>', $text);

echo $text;

?>
```

(Or)

b)Describe about Associative array in PHP.

PHP Indexed Array

PHP indexed array is an array which is represented by an index number by default. All elements of array are represented by an index number which starts from 0.

PHP indexed array can store numbers, strings or any object. PHP indexed array is also known as numeric array.

Definition

There are two ways to define indexed array:

1st way:

```
$size=array("Big","Medium","Short");
```

2nd way:

```
$size[0]="Big";

$size[1]="Medium";

$size[2]="Short";
```

PHP Indexed Array Example

File: array1.php

```
<?php

$size=array("Big","Medium","Short");

echo "Size: $size[0], $size[1] and $size[2]";

?>
```

Output:

```
Size: Big, Medium and Short
```

File: array2.php

```
<?php

$size[0]="Big";

$size[1]="Medium";

$size[2]="Short";

echo "Size: $size[0], $size[1] and $size[2]";

?>
```

Output:

```
Size: Big, Medium and Short
```

Traversing PHP Indexed Array

We can easily traverse array in PHP using foreach loop. Let's see a simple example to traverse all the elements of PHP array.

File: array3.php

```
<?php
$size=array("Big","Medium","Short");
foreach( $size as $s )
{
  echo "Size is: $s<br />";
}
?>
```

Output:

```
Size is: Big
Size is: Medium
Size is: Short
```

Count Length of PHP Indexed Array

PHP provides count() function which returns length of an array.

```
<?php
$size=array("Big","Medium","Short");
echo count($size);
?>
```

Output:

3

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a `for` loop, like this:

Example

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

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

OUTPUT

```
Volvo
BMW
Toyota
```

26. a)How to use looping with Index based array and explain it.
    PHP Associative Array
    PHP allows you to associate name/label with each array elements in PHP using => symbol. Such way, you can easily remember the element because each element is represented by label than an incremented number.
    Definition
    There are two ways to define associative array:
    1st way:
    $salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
    2nd way:
    $salary["Sonoo"]="550000";
    $salary["Vimal"]="250000";
    $salary["Ratan"]="200000";
    Example
    File: arrayassociative1.php
    <?php
    $salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");

```
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "Vimal salary: ".$salary["Vimal"]."<br/>";
echo "Ratan salary: ".$salary["Ratan"]."<br/>";
?>
```
Output:
```
Sonoo salary: 550000
Vimal salary: 250000
Ratan salary: 200000
```

File: arrayassociative2.php
```
<?php
$salary["Sonoo"]="550000";
$salary["Vimal"]="250000";
$salary["Ratan"]="200000";
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "Vimal salary: ".$salary["Vimal"]."<br/>";
echo "Ratan salary: ".$salary["Ratan"]."<br/>";
?>
```
Output:
```
Sonoo salary: 550000
Vimal salary: 250000
Ratan salary: 200000
```
Traversing PHP Associative Array
By the help of PHP for each loop, we can easily traverse the elements of PHP associative array.
```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
foreach($salary as $k => $v) {
echo "Key: ".$k." Value: ".$v."<br/>";
}
?>
```
Output:
```
Key: Sonoo Value: 550000
Key: Vimal Value: 250000
Key: Ratan Value: 200000
```

Loop Through an Associative Array
To loop through and print all the values of an associative array, you could use a foreach loop, like this:
Example
```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
OUT PUT
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43
```

(Or)

b)Explain in detail about associative array using each() and foreach().

The PHP foreach Loop

The `foreach` loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

```
foreach ($array as $value) {
  code to be executed;
}
```

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

The following example demonstrates a loop that will output the values of the given array ($colors):

Example

```
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

OUTPUT

```
red
green
```

```
blue
yellow
```

PHP each() Function

Definition and Usage

The each() function returns the current element key and value, and moves the internal pointer forward.

This element key and value is returned in an array with four elements. Two elements (1 and Value) for the element value, and two elements (0 and Key) for the element key.

Related methods:

- current() - returns the value of the current element in an array
- end() - moves the internal pointer to, and outputs, the last element in the array
- next() - moves the internal pointer to, and outputs, the next element in the array
- prev() - moves the internal pointer to, and outputs, the previous element in the array
- reset() - moves the internal pointer to the first element of the array

Syntax

each(*array*)

Example

Return the current element key and value, and move the internal pointer forward:

```
<?php
$people = array("Peter", "Joe", "Glenn", "Cleveland");
print_r (each($people));
?>
```

OUTPUT

```
Array ( [1] => Peter [value] => Peter [0] => 0 [key] => 0 )
```