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

YEAR: 2017 -2019(LATERAL ENTRY)

16CAP501 **PHP5/ MYSQL** **4H- 4C**

Instruction Hours / week: L: 4 T: 0 P: 0 C : 4 Marks: Internal: **40** External: **60 Total: 100**

End Semester Exam: 3Hours

Scope: PHP course is designed to learn how to do client-side programming, which will run on either a LAMP or a Windows web server.

Objective: To help students to

- Get hands-on experience in scripting, debugging, testing.
- Establish a working environment for PHP web page development
- Use variables, constants, and environment variables in a PHP program
- Learn to create dynamic interactive pages with PHP.
- Learn to manipulate files with PHP.
- Understand how MySQL works.
- Learn to use SQL to output reports with MySQL

UNIT I

Creating a Simple PHP Programs: How PHP Code Works- How Online PHP Programs Run Web; Communications: Internet Protocols and HTTP: TCP/IP- The HTTP Protocol; Using Variable in PHP Issues concerning Creating Variables-Defined Constants; Operators and Expressions: PHP Operators – PHP Expressions- Operators Types-Arrays

HTML Primer: The HTML Document type definition- The Form and Input Elements; Accessing PHP and HTTP Data: Predefined Variables- Variables in HTTP Request and Response- Super Global Arrays; Links; Query Strings; HTML(Web) Forms; HTML Form Elements-HTML Form Fields(Controls) and PHP; The Concept Of State: State Maintenance-Native Sessions in PHP.

Designing PHP Program Logic: Problem Statement- Writing Pseudo Code- Boolean Logic; Conditional Or Branching Statements: if statements- Switch statements- Loops and Arrays: Loops- Arrays.

UNIT II

Testing and Debugging: Values that break your code- Basic error types; Debugging PHP Script: Understanding PHP error Massages- Syntax Errors- Logic Errors-Runtime Errors; Debugging and Handling Errors in PHP5: Preventing the display of private information-Roll your-Own Debugging tools; Form Validation: Using the Exit statement- string validation and regular expressions- validating data entry- using reg exps to check file path parameters; Handling Errors: Gracefully- Configuring PHP for error handling- Try/Catch-New in PHP5.

Development planning: Formal software Development processes – optimizing your code- Using Coding standard; Writing user-defined functions in PHP: The Structure of Functions- Switching Functions – How Values Get Inside functions; Scope of variables: Global and Local Variables-Creating Static Function Variable-Nesting-Recursion-The Include and Require Statements-Things to be careful about with include and require.

UNIT III

Files and Directories: Files and Directory Handling- Working with Files- Opening and Closing files-Getting Information about a file-reading and writing to files-Reading and writing characters in files-Reading Entire files-Random Access to file data-Getting Information on Files-Ownership and permissions; Working with files you own: Splitting the Name and path from a file-copying ,renaming and deleting files; Working with Directories: other

Directory Functions –Traversing a directory hierarchy-creating a directory navigator-Building a Text Editor-Uploading Files. Classes- Objects: Creating class- Adding a Method- Adding a Property- Protecting Access to Member Variables- Using _get and _set- Initializing objects- Destroying Objects- Inheritance- Overriding Methods-Interfaces- Encapsulation

UNIT IV

The SQL Framework- Managing databases-Creating & Managing tables- Managing indexes; Inserting & Updating data in a MYSQL database-Deleting & Retrieving data from a MySQL database; SELECT statement-Optional clauses of a SELECT statement; Creating MySQL Expressions-using operators in expressions-Comparing and Converting Data; Managing different types of data: String functions-Numeric function- Date/Time functions-Summarizing date-Summary functions. Performing System Operations: Encryption functions- System related Functions- Query and Insert Functions; Accessing data from Multiple tables: Creating joins in your SQL statement-Creating subqueries in your SQL statements; Creating Unions that join SELECT statements. Exporting, Copying and importing data; Managing transactions: Introducing transactions- Performing a transaction- Setting the auto commit mode and transaction isolation level- Locking Nontransactional tables

UNIT V

Connecting to MySQL from a PHP application- Inserting and updating records in table- Deleting and retrieving data from table- Creating a user Registration Script. Structure of an E-Mail Message-sending E-mail with PHP- Working with Raster Images- Manipulating Raster Images- Using Text in Images

SUGGESTED READINGS

1. Dave W.Mercer, Allan Kent, Steven D.Nowicki, Davd Mercer, Dan Squie, Wankyu Choi.(2009), Beginning PHP5. Wiley India (P) Ltd, New Delhi
2. Luke welling, Laura Thomson (2010), PHP and MySQL Web Development, 4th Edition, Pearson Education.
3. Julie Meloni (2012), Sams Teach Yourself PHP, MySQL and Apache All in One, 5th Edition, Pearson Education India.
4. Paul Dubois (2006), MySQL, 1st Edition, Tech Media, New Delhi.
5. Tim Converse & Joyce Park with Clark Morgan (2006), PHP5 & MySQL Bible, 1st Edition, John Wily, India.
6. Baron Schwartz, Peter Zaitsev, Vadim Tkachenko (2012), High Performance MySQL: Optimization, Backups, 3rd Edition, O'REILLY.

WEB SITES

1. www.php.net/
2. en.wikipedia.org/wiki/PHP
3. www.w3schools.com/PHP/DEfaULT.asP

Question Paper Pattern:

CIA	**Max.Marks : 50**
Part A	Objective type questions : 20 x 1 = 20 Marks
Part B	Answer all the questions Either/Or : 3 x 10 = 30 Marks

ESE	**Max.Marks : 60**
Part A	Objective type questions : 20 x 1 = 20 Marks
Part B	Answer all the questions Either/Or : 5 x 6 = 30 Marks
Part C	Answer all the questions Compulsory : 1 x 10 = 10 Marks

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act, 1956)
Coimbatore-21

DEPARTMENT OF CS, CA & IT

PHP5/ MYSQL 16CAP501

Lesson Plan

S.NO	Duration in hours	TOPICS TO BE COVERED	SUPPORT MATERIALS T1
1	1hr	**Unit I** Creating a simple php programs:- How php code works,web communications, Internet protocols to http:-Tcp /ip,http protocols	31-37,W1
2	1hr	Variables in PHP , Issues concerning creating variables, defined constants	41-45,W3 R1:46-62
3	1hr	Operators & expressions:- PHO Operators, expressions, operators types, Arrays.	R4:67-76
4	1hr	**HTML primer:-** HTML Document type definition- the form and to input elements	63-67
5	1hr	**Acessing php and http data:-** Predefined variable ,variable in http request & response.	67-70
6	1hr	Super global arrays ,links.	71-72
7	1hr	Query string HTML forms & HTML form elements. HTML form fields (controls) and PHP	73-79
8	1hr	The Concept of state : State Maintenance. Native sessions in php Designing php program logic.	108-125
9	1hr	Conditional &branching statements if stat-switch stat-loop to Arrays.	127-161
10	1hr	Recapitulation to Discussion of Important Questions.	

TEXT BOOK

T1: Dave W.Mercer, Allan Kent, Steven D.Nowicki, Davd Mercer, Dan Squie, Wankyu Choi.2009. Beginning PHP5. Wiley India (P) Ltd, New Delhi

R1: Julie Meloni . 2012. Sams Teach Yourself PHP, MySQL and Apache All in One, 5th Edition, Pearson Education India.

Unit-II			
S.NO	Duration in hours	TOPICS TO BE COVERED	SUPPORT MATERIALS T1
1	1hr	**Testing &debugging:-** Values that break your code-Basic error types, Debugging PHP Script: understanding PHP error messages-syntax errors.	187-190
2	1hr	Logic Error, Runtime Error, Debugging to Handling Errors in PHP5.	191-194
3	1hr	Preventing to the display of private information Roll your own Debugging tools.	195-198
4	1hr	Form validation: using the exit statement string validation and regular expression. validating date entry-using ref to check file path parameters	199-218
5	1hr	Handling Errors: Gracefully; configuring PHP for error handling-try/catch-New in PHP5	220-225,W3
6	1 hr	Development planning: Formal software Development processes. Optimizing your code-using coding standard.	226-229, 231-234
7	1hr	Writing user-defined fun in PHP :the structure of fun's-switching funs –how values get inside functions	273 -279,W3
8	1hr	Scope of variables: Global to Local variables , creating static functions , Variables , Nesting , Recursion	108 – 161 R4:83-105
9	1hr	The include to require stat. Things to be careful about with include to require	R4:83-105
10	1hr	Recapitulation to Discussion of Important Questions.	

TEXT BOOK

T1:Dave W.Mercer, Allan Kent, Steven D.Nowicki, Davd Mercer, Dan Squie, Wankyu Choi.2009. Beginning PHP5. Wiley India (P) Ltd, New Delhi

R4: Tim Converse & Joyce Park with Clark Morgan . 2006. PHP5 & MySQL Bible, 1st Edition, John Wily, India.

Unit III			
S.NO	Duration in hours	TOPICS TO BE COVERED	SUPPORT MATERIALS T1
1	1hr	**Files &directories:-** Files to Directory handling – working with files.	260-264,W1
2	1hr	Opening to closing files , getting information about a file- reading to writing files .	R2:439- 447
3	1hr	Reading to writing characters in files Reading Entire files. Randam access to file data, Getting information on files	269 - 278,W3
4	1hr	Ownership to Permissions : Working with files your own . splitting the name of path from a file – copy,renaming to deleting files .	282 - 288
5	1hr	Working with Directories : other directories file , traversing a directory hierarchy – creating a directory navigator .	290 – 295 R2:377 - 386
6	1hr	Building a Text Editor – uploading files .	300 – 307
7	1hr	Classes – Objects : Creating class – Adding the method – adding a property. Protecting access to member variables.	461 – 476 R2:365 – 377
8	1hr	Using _ get to _set initializing objects _ destroying objects, inheritance overriding.	150 -156
9	1hr	Overriding methods – interfaces – encapsulation.	477 – 490
10	1hr	Recapitulation to Discussion of Important Questions.	

TEXT BOOK

T1: Dave W.Mercer, Allan Kent, Steven D.Nowicki, Davd Mercer, Dan Squie, Wankyu Choi.2009. Beginning PHP5. Wiley India (P) Ltd, New Delhi

R2: Julie Meloni . 2012. Sams Teach Yourself PHP, MySQL and Apache All in One, 5th Edition, Pearson Education India.

Unit IV			
S.NO	**Duration in hours**	**TOPICS TO BE COVERED**	**SUPPORT MATERIALS T1**
1	1hr	The SQL framework – managing databases creating to mapping tables – managing indexes Inserting to Updating data in MySql db.	375 -379 R5:9 – 27
2	1hr	Deleting, retrieving data from MySql db , SELECT stat , optional classes of a SELECT statement.	385 – 388
3	1hr	Creating MySql expressions – using Operators in Expressions –String functions , numeric functions	R5:269
4	1hr	Comparing to converting data managing different type of data. Date / Time, summarizing date – summary functions	R5: 270-73
5	1hr	Performing system operations : Encryption functions – system related functions- Query to Insert functions	358-368
6	1hr	Accessing data from multiple tables:- Creating joins in your sql statement. Creating joins to subqueries in sql statement.	347 – 355 W3
7	1hr	Creating unions that join select statement Exporting,coping and importing data,managing transactions:Introducing transactions performing a transaction.	356 – 361 R5:421-449
8	1hr	Setting the auto commit mode and transaction isolation level locking non transactional tables.	364-365
9	1hr	Recapitulation to discussion of Important Questions.	

TEXT BOOK

T1: Dave W.Mercer, Allan Kent, Steven D.Nowicki, Davd Mercer, Dan Squie, Wankyu Choi.2009. Beginning PHP5. Wiley India (P) Ltd, New Delhi

R5: Baron Schwartz, Peter Zaitsev, Vadim Tkachenko. 2012 High Performance MySQL: Optimization, Backups, 3rd Edition, O'REILLY.

Unit V			
S.NO	**Duration in hours**	**TOPICS TO BE COVERED**	**SUPPORT MATERIALS T1**
1	1hr	Connecting to mysql from o php application	367-368
2	1hr	Inserting to updating records in tables. Deleting and retriving data from the table.	417-431, R3: 312-318
3	1hr	Creating user registration script Structure of an email message	435-453,R3
4	1hr	Sending email with php. Working with raster images.	568-583,W3 585-593,J1
5	1hr	Manipulating raster images. Using text in images.	598-604,J1 607-612
6	1hr	Recapituletion to discussion of important questions.	
7	1hr	Discussion on preriors year ESE Question papers.	
8	1hr	Discussion on preriors year ESE Question papers.	
9	1hr	Discussion on preriors year ESE Question papers.	

TEXT BOOK

1. Dave W.Mercer, Allan Kent, Steven D.Nowicki, Davd Mercer, Dan Squie, Wankyu Choi.2009. Beginning PHP5. Wiley India (P) Ltd, New Delhi

REFERENCE BOOKS

1. Luke welling, Laura Thomson, 2010. PHP and MySQL Web Development, 4th Edition, Pearson Education.
2. Julie Meloni . 2012. Sams Teach Yourself PHP, MySQL and Apache All in One, 5th Edition, Pearson Education India.
3. Paul Dubois. 2006. MySQL, 1st Edition, Tech Media, New Delhi.
4. Tim Converse & Joyce Park with Clark Morgan . 2006. PHP5 & MySQL Bible, 1st Edition, John Wily, India.
5. Baron Schwartz, Peter Zaitsev, Vadim Tkachenko. 2012 High Performance MySQL: Optimization, Backups, 3rd Edition, O'REILLY.

WEB SITES

1. www.php.net/
2. en.wikipedia.org/wiki/PHP
3. www.w3schools.com/PHP/default.asp

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

COIMBATORE - 641 021

PHP5/MYSQL [16CAP501]

UNIT I

PHP Introduction

PHP is a server-side scripting language.

What is PHP?

- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

What is MySQL?

- MySQL is a database server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use

PHP + MySQL

- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Where to Start?

To get access to a web server with PHP support, you can:

- Install Apache (or IIS) on your own server, install PHP, and MySQL
- Or find a web hosting plan with PHP and MySQL support

PHP Installation

What do you Need?

If your server supports PHP you don't need to do anything.

Just create some .php files in your web directory, and the server will parse them for you. Because it is free, most web hosts offer PHP support.

However, if your server does not support PHP, you must install PHP.

Here is a link to a good tutorial from PHP.net on how to install PHP5: http://www.php.net/manual/en/install.php

PHP Syntax

Basic PHP Syntax

A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document.

On servers with shorthand support enabled you can start a scripting block with <? and end with ?>.

For maximum compatibility, we recommend that you use the standard form (<?php) rather than the shorthand form.

```
<?php
?>
```

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

Below, we have an example of a simple PHP script which sends the text "Hello World" to the browser:

```
<html>
<body>

<?php
echo "Hello World";
?>

</body>
</html>
```

Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.

There are two basic statements to output text with PHP: **echo** and **print**. In the example above we have used the echo statement to output the text "Hello World".

Note: The file must have a .php extension. If the file has a .html extension, the PHP code will not be executed.

Comments in PHP

In PHP, we use // to make a single-line comment or /* and */ to make a large comment block.

```
<html>
<body>

<?php
//This is a comment
```

```

/*
This is
a comment
block
*/
?>

</body>
</html>
```

What is PHP?

- PHP == ‘Hypertext Preprocessor’
- Open-source, server-side scripting language
- Used to generate dynamic web-pages
- PHP scripts reside between reserved PHP tags
- This allows the programmer to embed PHP scripts within HTML pages
- Interpreted language, scripts are parsed at run-time rather than compiled beforehand
- Executed on the server-side
- Source-code not visible by client
  - ❑ ‘View Source’ in browsers does not display the PHP code
- Various built-in functions allow for fast development
- Compatible with many popular databases

What does PHP code look like?

- Structurally similar to C/C++
- Supports procedural and object-oriented paradigm (to some degree)
- All PHP statements end with a semi-colon
- Each PHP script must be enclosed in the reserved PHP tag

```
<?php
 …
?>
```

PHP Variables

As with algebra, PHP variables are used to hold values or expressions.

A variable can have a short name, like x, or a more descriptive name, like carName.

Rules for PHP variable names:

- Variables in PHP starts with a $ sign, followed by the name of the variable
- The variable name must begin with a letter or the underscore character
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- A variable name should not contain spaces
- Variable names are case sensitive (y and Y are two different variables)

Creating (Declaring) PHP Variables

PHP has no command for declaring a variable.

A variable is created the moment you first assign a value to it:

```
$myCar="Volvo";
```

After the execution of the statement above, the variable myCar will hold the value Volvo.

Tip: If you want to create a variable without assigning it a value, then you assign it the value of null.

Let's create a variable containing a string, and a variable containing a number:

```
<?php
$txt="Hello World!";
$x=16;
?>
```

PHP Variable Scope

The scope of a variable is the portion of the script in which the variable can be referenced.

PHP has four different variable scopes:

- local
- global
- static
- parameter

Local Scope

A variable declared within a PHP function is local and can only be accessed within that function. (the variable has local scope):

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

function myTest()
{
echo $a; // local scope
}

myTest();
?>
```

The script above will not produce any output because the echo statement refers to the local scope variable $a, which has not been assigned a value within this scope.

You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

Local variables are deleted as soon as the function is completed.

Global Scope

Global scope refers to any variable that is defined outside of any function.

Global variables can be accessed from any part of the script that is not inside a function.

To access a global variable from within a function, use the global keyword:

```
<?php
$a = 5;
$b = 10;
```

```
function myTest()
{
global $a, $b;
$b = $a + $b;
}

myTest();
echo $b;
?>
```

The script above will output 15.

PHP also stores all global variables in an array called $GLOBALS[index]. Its index is 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 as this:

```
<?php
$a = 5;
$b = 10;

function myTest()
{
$GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
}

myTest();
echo $b;
?>
```

Static Scope

When a function is completed, all of its variables are normally deleted. However, sometimes you want a local variable to not be deleted.

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

```
static $rememberMe;
```

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.

Note: The variable is still local to the function.

Parameters

A parameter is a local variable whose value is passed to the function by the calling code.

Parameters are declared in a parameter list as part of the function declaration:

```
function myTest($para1,$para2,...)
{
// function code
}
```

Variables in PHP:

- PHP variables must begin with a “$” sign
- Case-sensitive ($Foo != $foo != $fOo)
- Global and locally-scoped variables
  - ❑ Global variables can be used anywhere
  - ❑ Local variables restricted to a function or class
- Certain variable names reserved by PHP
  - ❑ Form variables ($_POST, $_GET)
  - ❑ Server variables ($_SERVER)
  - ❑ Etc.

Variable usage:

```
<?php

$foo = 25;            // Numerical variable
$bar = “Hello”;       // String variable

$foo = ($foo * 7);    // Multiplies foo by 7
$bar = ($bar * 7);    // Invalid expression

?>
```

Echo:

- The PHP command ‘**echo**’ is used to output the parameters passed to it
    - The typical usage for this is to send data to the client’s web-browser
- Syntax
    - void **echo** (string arg***1*** [, string arg***n***...])
    - In practice, arguments are not passed in parentheses since **echo** is a language construct rather than an actual function

Echo example:

```
<?php

$foo = 25;              // Numerical variable
$bar = “Hello”;         // String variable

echo $bar;              // Outputs Hello

echo $foo,$bar;         // Outputs 25Hello

echo “5x5=”,$foo;       // Outputs 5x5=25

echo “5x5=$foo”;        // Outputs 5x5=25
echo ‘5x5=$foo’;        // Outputs 5x5=$foo
```

- **Notice** how echo ‘5x5=$foo’ outputs $foo rather than replacing it with 25
- Strings in single quotes (‘ ’) are not interpreted or evaluated by PHP
- This is true for both variables and character escape-sequences (such as “\n” or “\\”)

Arithmetic Operations:

```
<?php

        $a=15;

        $b=30;

        $total=$a+$b;

        Print $total;
```

Print "<p><h1>$total</h1>";

// total is 45

?>

- $a - $b // subtraction
- $a * $b// multiplication
- $a / $b // division
- $a += 5 // $a = $a+5 Also works for *= and /=

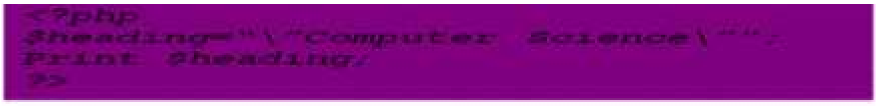
Concatenation:

- Use a period to join strings into one.
- ***<?php***
- ***$string1="Hello";***
- ***$string2="PHP";***
- ***$string3=$string1 . " " . $string2;***
- ***Print $string3;***
- ***?>***

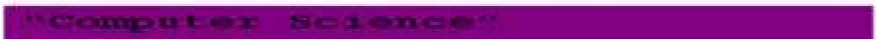
Hello PHP

Escaping the Character:

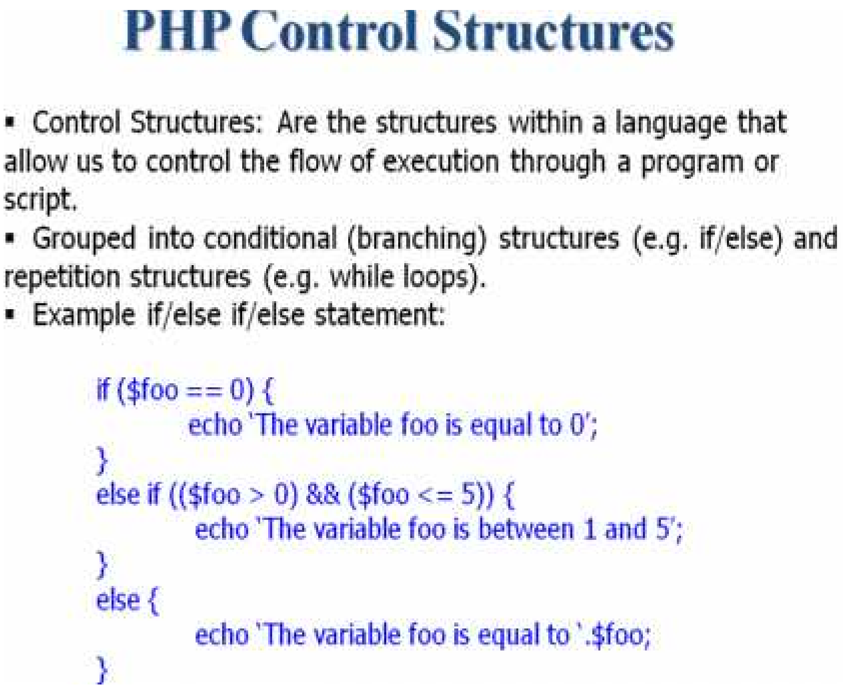
- If the string has a set of double quotation marks that must remain visible, use the \ [backslash] before the quotation marks to ignore and display them.



```
<?php
$heading="\"Computer Science\"";
Print $heading;
?>
```



```
"Computer Science"
```



PHP Control Structures

- Control Structures: Are the structures within a language that allow us to control the flow of execution through a program or script.
- Grouped into conditional (branching) structures (e.g. if/else) and repetition structures (e.g. while loops).
- Example if/else if/else statement:

```
if ($foo == 0) {
        echo 'The variable foo is equal to 0';
}
else if (($foo > 0) && ($foo <= 5)) {
        echo 'The variable foo is between 1 and 5';
}
else {
        echo 'The variable foo is equal to '.$foo;
}
```

PHP If...Else Statements:

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions.

You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if a condition is true and another code if the condition is false
- **if...elseif....else statement** - use this statement to select one of several blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

The if Statement

Use the if statement to execute some code only if a specified condition is true.

Syntax

if (*condition*) *code to be executed if condition is true;*

The following example will output "Have a nice weekend!" if the current day is Friday:

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri") echo "Have a nice weekend!";
?>

</body>
</html>
```

Notice that there is no ..else.. in this syntax. The code is executed **only if the specified condition is true**.

The if...else Statement

Use the if....else statement to execute some code if a condition is true and another code if a condition is false.

Syntax

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

Example

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":

```
<html>
<body>
```

```
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
else
  echo "Have a nice day!";
?>

</body>
</html>
```

If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces:

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  {
  echo "Hello!<br />";
  echo "Have a nice weekend!";
  echo "See you on Monday!";
  }
?>

</body>
</html>
```

The if...elseif....else Statement

Use the if....elseif...else statement to select one of several blocks of code to be executed.

Syntax

if (*condition*)
  code to be executed if condition is true;

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

Example

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>

</body>
</html>
```

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;
default:
  code to be executed if n is different from both label1 and label2;
}
```

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 lines in a script we can use loops to perform a task like this.

In PHP, we have the following looping statements:

while - loops through a block of code while a specified condition is true

do...while - loops through a block of code once, and then repeats the loop as long as a specified condition is true

for - loops through a block of code a specified number of times

foreach - loops through a block of code for each element in an array

---

The while Loop

The while loop executes a block of code while a condition is true.

Syntax

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

Example

The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>

<?php
$i=1;
```

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

</body>
</html>
```

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

The do...while Statement

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

Syntax

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

Example

The example below defines a loop that starts with i=1. It will then increment i with 1, and write some output. Then the condition is checked, and the loop will continue to run as long as i is less than, or equal to 5:

```
<html>
<body>
```

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

</body>
</html>
```

The for Loop

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

Syntax

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

Parameters:

init: Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop)

condition: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.

increment: Mostly used to increment a counter (but can be any code to be executed at the end of the loop)

Note: Each of the parameters above can be empty, or have multiple expressions (separated by commas).

Example

The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>

<?php
for ($i=1; $i<=5; $i++)
  {
  echo "The number is " . $i . "<br />";
  }
?>

</body>
</html>
```

What is an Array?

A variable is a storage area holding a number or text. The problem is, a variable will hold only one value.

An array is a special variable, which can store multiple values in one single variable.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1="Saab";
$cars2="Volvo";
$cars3="BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The best solution here is to use an array!

An array can hold all your variable values under a single name. And you can access the values by referring to the array name.

Each element in the array has its own index so that it can be easily accessed.

In PHP, there are three kind of arrays:

Numeric array - An array with a numeric index

Associative array - An array where each ID key is associated with a value

Multidimensional array - An array containing one or more arrays

---

Numeric Arrays

A numeric array stores each array element with a numeric index.

There are two methods to create a numeric array.

1. In the following example the index are automatically assigned (the index starts at 0):

```
$cars=array("Saab","Volvo","BMW","Toyota");
```

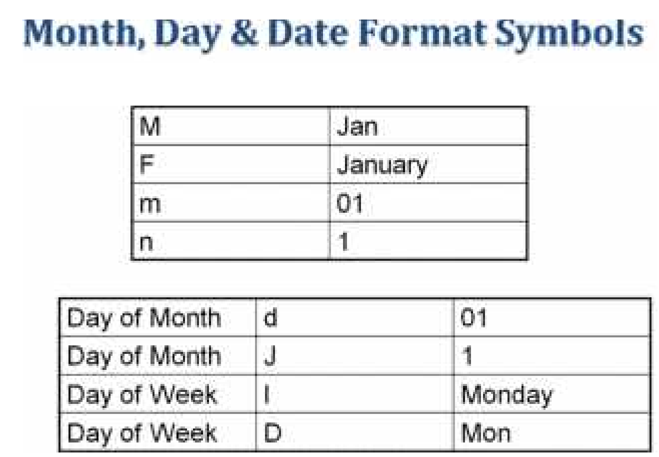
2. In the following example we assign the index manually:

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

Example

In the following example you access the variable values by referring to the array name and index:

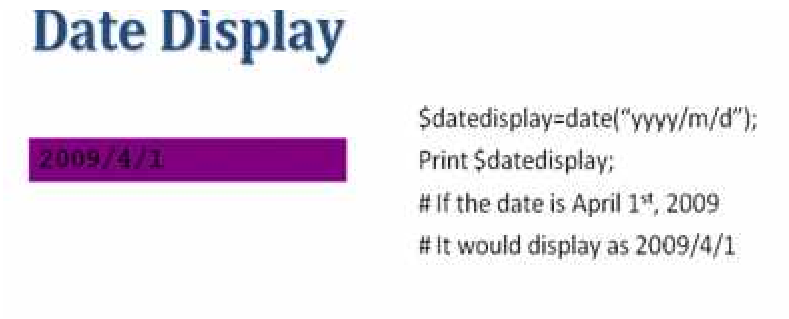
```
<?php
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
?>
```



Month, Day & Date Format Symbols

M	Jan
F	January
m	01
n	1

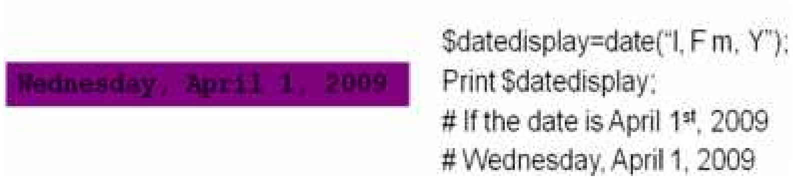
Day of Month	d	01
Day of Month	J	1
Day of Week	l	Monday
Day of Week	D	Mon



Date Display

2009/4/1

```
$datedisplay=date("yyyy/m/d");
Print $datedisplay;
# If the date is April 1st, 2009
# It would display as 2009/4/1
```



Wednesday, April 1, 2009

```
$datedisplay=date("l, F m, Y");
Print $datedisplay;
# If the date is April 1st, 2009
# Wednesday, April 1, 2009
```

Functions:

- Functions MUST be defined before then can be called
- Function headers are of the format
  - ❑ Note that no return type is specified
- Unlike variables, function names are not case sensitive (foo(…) == Foo(…) == FoO(…))

Functions example:

<?php

```
    // This is a function

function foo($arg_1, $arg_2)

 {
  $arg_2 = $arg_1 * $arg_2;

  return $arg_2;
}


$result_1 = foo(12, 3);                                  // Store the function

echo $result_1;                                          // Outputs 36

echo foo(12, 3);                                         // Outputs 36

?>
```

Include Files:

```
Include “opendb.php”;

Include “closedb.php”;
```

This inserts files; the code in files will be inserted into current code. This will provide useful and protective means once you connect to a database, as well as for other repeated functions.

```
Include (“footer.php”);
```

The file footer.php might look like:

```
<hr SIZE=11 NOSHADE WIDTH=“100%”>

<i>Copyright © 2008-2010 KSU </i></font><br>

<i>ALL RIGHTS RESERVED</i></font><br>

<i>URL: http://www.kent.edu</i></font><br>
```

PHP – Forms:

- Access to the HTTP POST and GET data is simple in PHP
- The global variables $_POST[] and $_GET[] contain the request data

```
  <?php
```

```
   if ($_POST["submit"])
      echo "<h2>You clicked Submit!</h2>";
   else if ($_POST["cancel"])
      echo "<h2>You clicked Cancel!</h2>";
?>
<form action="form.php" method="post">
   <input type="submit" name="submit" value="Submit">
   <input type="submit" name="cancel" value="Cancel">
</form>
```

http://www.cs.kent.edu/~nruan/form.php

First PHP script:

- Save as sample.php:

```
      <!– sample.php -->
      <html><body>
      <strong>Hello World!</strong><br />
    <?php
     echo “<h2>Hello, World</h2>”; ?>
   <?php
               $myvar = "Hello World";
               echo $myvar;
          ?>
      </body></html>
```

HTML Primer:

HTML Document type definition:

- Defines the structure of an XML document
- Only the elements defined in a DTD can be used in an XML document
- can be internal or external
- A DTD defines the structure of a “valid” XML document
- Processing overhead is incurred when validating XML with a DTD
- There are four types of declarations:
  - Element type declarations
    - http://www.w3.org/TR/REC-xml#elemdecls
  - Attribute List Declarations
    - http://www.w3.org/TR/RECxml-attdecls
  - Entity declarations
    - http://www.w3.org/TR/REC-xml#sec-entity-decl
  - Notation declarations
    - http://www.w3.org/TR.REC-xml#Notations

HTML FORMS

n So far we have used the client-server model to make requests for documents and have documents served or returned.

n The Common Gateway Interface (CGI) allows a variation of this

– in the sense that a CGI request is understood to be a request to execute an application rather than simply return a document.

n Of course in returning information to the client (browser) a (virtual) document must be used.

n Forms provide a means of submitting information from the client to the server. A form consists of one or more:

n text input boxes

- clickable radio buttons
- multiple-choice check boxes
- pull-down menus
- clickable images
- text and images (maybe instructions on form use);

The <form> tag

- All of the form elements within a <form> tag comprise a single form.
- The browser sends all of the values of these elements - blank, default, or user modified - when the user *submits* the form to the server.
- Browsers flow forms into the containing elements as if they were small embedded images. So layout elements such as <p> and <br> need to be used.

Form Attributes

- action - gives the URL of the application that is to receive and process the forms data; most of these are kept in *cgi-bin* or *cgi-win*; *Common Gateway Interface binaries*;
    - Example:<form action="http://141.132.64.152/cgi-win/testgen.exe" ...</form>
- enctype - the browser encodes the form's data before it is passed to the server. The server may then decode the parameters or pass them still encoded to the application;
    - standard encoding format is Internet Media Type named "application/x-www-form-urlencoded"; only other type is "multipart/form-data
  - Example of what is sent to the server when a form with just 2 fields (name & address) is submitted: name=T+Rex&address=101+Jurassic+Park+Drive%0D%0APostCode%0D%0A3350
  - note that space becomes + and %0D%0A is the line break;

) method - sets the HTTP method that the browser uses to send the form's data to the server for processing; Either POST or GET; Example: <form method=GET action="http://141.132.64.152/cgi-win/testgen.exe" ...</form>

The method attribute in more detail

n **Post**

- The browser sends the data in two steps:
    - contacts the form processing server specified in the action attribute;
    - sends the data to the server in a separate transmission;
- On the server side POST-style applications are expected to read the parameters from a standard location once they begin execution

n **GET**

- contacts the form-processing server and sends the form data in a single transmission step:
- the browser appends the data to the form's action URL, separated by the ? character.

A FORM example

```
Name:
        <input type=text name=name size=32 maxlength=80>
<p> Sex:
        <input type=radio name=sex value="M"> Male
        <input type=radio name=sex value="F"> Female
<p> Date of Birth:
        <input type=text name=year size=4 maxlength=4> Year
        <input type=text name=month size=2 maxlength=2> Month
        <input type=text name=day size=2 maxlength=2> Day
```

<p> <input type=submit> </form>

FORM Input Elements

n The <INPUT> Tag

– This is used to define text fields, multiple choice lists, clickable images and submit buttons. Only the TYPE and NAME attributes are required.

n The <input> tag attributes

- align -
- checked -
- maxlength - maximum number of characters accepted by the browser

Text Fields

- **Conventional text field -**
    - size attribute dictates the width of the text box;
    - maxlength dictates the maximum number of characters that the user can see and type;
    - if maxlength exceeds size then text scrolls back and forth within the text entry box;
    - server side application must trap errors;
    - Masked text field - type=password; obscured onscreen but is transmitted unencrypted;
    - File selection field - lets the user select and send a file; type pathname directly or use the Browse button;

Checkboxes

n **type=checkbox**

– The name and value attributes are required.

– If the item is selected it will contribute a value when the form is submitted;

Checkbox form code

```
<form>
What units are you studying this semester?
  <p>
    <input type=checkbox name=unit                    value="CP747"> CP747
  <br>
    <input type=checkbox name=unit              value="CP725"> CP725
</form>
```

PHP Session Variables

When you are working with an application, you open it, do some changes and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are and what you do because the HTTP address doesn't maintain state.

A PHP session solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping items, etc). However, session information is temporary and will be deleted after the user has left the website. If you need a permanent storage you may want to store the data in a database.

Sessions work by creating a unique id (UID) for each visitor and store variables based on this UID. The UID is either stored in a cookie or is propagated in the URL.

Starting a PHP Session

Before you can store user information in your PHP session, you must first start up the session.

Note: The session_start() function must appear BEFORE the <html> tag:

```
<?php session_start(); ?>

<html>
<body>

</body>
</html>
```

The code above will register the user's session with the server, allow you to start saving user information, and assign a UID for that user's session.

Storing a Session Variable

The correct way to store and retrieve session variables is to use the PHP $_SESSION variable:

```
<?php
session_start();
// store session data
$_SESSION['views']=1;
?>

<html>
<body>

<?php
//retrieve session data
echo "Pageviews=". $_SESSION['views'];
?></body></html>
```

Output:

```
Pageviews=1
```

In the example below, we create a simple page-views counter. The isset() function checks if the "views" variable has already been set. If "views" has been set, we can increment our counter. If "views" doesn't exist, we create a "views" variable, and set it to 1:

```
<?php
session_start();

if(isset($_SESSION['views']))
$_SESSION['views']=$_SESSION['views']+1;
else
$_SESSION['views']=1;
echo "Views=". $_SESSION['views'];
?>
```

Destroying a Session

If you wish to delete some session data, you can use the unset() or the session_destroy() function.

The unset() function is used to free the specified session variable:

```
<?php
unset($_SESSION['views']);
?>
```

You can also completely destroy the session by calling the session_destroy() function:

```
<?php
session_destroy();
?>
```

UNIT I

POSSIBLE QUESTIONS

Part - B (Each Question carries 6 Marks)

1. Write a session application to count the number of times a user has accessed a webpage.
2. Explain branching statements with example.
3. Explain the use of operators in PHP.
4. Explain HTML form elements with example.
5. Explain various functions used to sort an array with example.
6. What is a session? Explain native sessions with example.
7. Develop a php application for uploading online resume
8. What are cookies? Explain how to set and retrieve cookies.
9. Design an online job application form
10. Write a program to demonstrate working with numbers

Part - C (Each Question carries 10 Marks)

1. Illustrate loops and arrays in PHP with an example.
2. Design a program to count number of times the web page has been visited by the user.
3. Develop a php program for online shopping application

KARPAGAM ACADEMY OF HIGHER EDUCATION

DEPARTMENT OF COMPUTER APPLICATIONS

Subject Name PHP5/MYSQL

Class: III MCA

SUBCODE: 16CAP501

Sem:V

S.No	Unit	Question	Choice1	Choice2	Choice3	Choice4	Answer Key
1	1	PHP stands for ____________________	Program Holding Process	Hypertext Preprocessor	Personal HypertextProgramming	program process	Hypertext Preprocessor
2	1	The part that interprets and executes PHP code	zend engine	search engine	web browser	interpreter	zend engine
3	1	PHP is used for building____________________websites	static	dynamic	none	a&b	dynamic
4	1	PHP program are run on_________________	web browser	interpreter	web server	compiler	web server
5	1	PHP is devised by_____________	Tim Berners- Lee	Rasmus Lerdorf	Robert Caillau	Richard Fairly	Rasmus Lerdorf
6	1	PHP programs run via__________	Web browser	interpreter	web server	none of these	Web browser
7	1	HTML stands for________________	Home Tool Management Logic	HyperText Managing Language	HyperText Markup Language	none of these	HyperText Markup Language
8	1	PHP was devised in the year	1996	1990	1986	1994	1994
9	1	HTML is used for building________________websites	dynamic	static	both	none	static
10	1	CLI stands for_________	Command Line Interpreter	Common Line Integrator	Common Line Interface	Command Line Interface	Command Line Interface
11	1	CLI was introduced in____________	PHP4	PHP2	PHP5	none of these	PHP4

12	1	GTK stands for________________	General Tool Kit	Gnome Track Kit	Gnome Tool Kit	General Tab Kit	Gnome Tool Kit
13	1	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	1	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	1	ODBC stands for_______________	Open Database Connectivity	Object Declaration and Basic Commands	Open Database Control	none of these	Open Database Connectivity
16	1	_____________ is a programming language designed to work with HTML	c	c++	PHP	COBOL	PHP
17	1	______________ is the language or format for communications from browser to web	XML	HTTP	HTML	JAVA	HTTP
18	1	___________provides programmatic functionality with in web pages.	Java script	XML	JSP	none of these	Java script
19	1	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	1	HTTP stands for___________	Hyper Tool Transfer Protocol	Hypertext Transmission Protocol	Home Tool Transfer Protocol	Hypertext Transfer Protocol	Hypertext Transfer Protocol

21	1	___________is a method by which results are stored temporarily	Caching	packing	compiling	none of these	Caching
22	1	RPM stands for ___________	Redhat Protocol Manager	Removable Program Memory	Redhat Package Manager	none of these	Redhat Package Manager
23	1	___________function is used to send a string value to the browser.	Echo	cookie	cout	none of these	Echo
24	1	IIS stands for _______________	Information Integrated Service	Internet Information Standard	Internet Information Server	Internal Information Service	Internet Information Server
25	1	__________ are special characters that indicates where the data starts and stop.	Modifiers	Limiters	Identifiers	Delimiters	Delimiters
26	1	In PHP code statements end with _________	Colon	Semicolon	Comma	Star operator	Semicolon
27	1	___________ are enclosed in curly braces.	Functions	Data	Code blocks	none of these	Code blocks
28	1	________file is parsed when PHP is first loaded and executed.	php.ini	php.exe	php.awt	php.init	php.ini
29	1	_________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	1	SAPI stands for ___________	Server And Programmers Interaction	Standard Analysis Program Interface	Server Application Programming Interface	none of these	Server Application Programming Interface
31	1	PWS stands for___________	Personal Web Service	Programed Window Standard	Personal Web Server	Powered Window Service	Personal Web Server

32	1	___________ means the processing engine reads the individual commands and checks for syntax errors.	Parsing	Caching	Storing	Checking	Parsing
33	1	__________ function is used to determine whether a search string exists with in a searched string.	Strstr( )	Chr( )	Strlen( )	Strpos( )	Strpos( )
34	1	________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	1	__________function returns a string character value corresponding to the decimal ASCII value entered as the argument.	Chr()	Strstr( )	Strlen()	Strpos( )	Chr()
36	1	_________ function finds the length of a string.	Strstr( )	Chr( )	Strlen()	none of these	Strlen()
37	1	DSO stands for __________	Digital Service Operator	Dynamic Shared Object	Domain Service Operator	Dynamic Standard Object	Dynamic Shared Object
38	1	___________can be used between string values to join them together.	Concatenation operator	Star operator	Colon	Plus operator	Concatenation operator
39	1	SGML stands for ____________	Standard Geometric Manipulation Logic	Static Generalized Making logic	Standard Generalized Markup Language	none of these	Standard Generalized Markup Language
40	1	PHP automatically makes a few variables called _________	Predefined variables	Defined variables	Library variables	User defined Variables	Predefined variables

41	1	Predefined variables are also called ___________	Global variables	Super variables	Local variables	Superglobal variables	Superglobal variables
42	1	_________ function prints out information about variables	echo	print_r( )	cout	none of these	print_r( )
43	1	XML stands for _____	External Markup language	Extended Markup Logic	Extended Markup Language	none of these	Extended Markup Language
44	1	_____________ is a series of statements in plain language that are logically	Pseudo code	function code	method code	standard code	Pseudo code
45	1	___________ is very important to control for structures.	Control logic	Control structures	Boolean logic	none of these	Boolean logic
46	1	___________ devised Boolean Algebra.	Tim Berners Lee	Robert Boole	Richard Fairly	George Boole	George Boole
47	1	__________ function is used to view the value of the pointer.	set( )	Current( )	seek( )	tell( )	Current( )
48	1	__________indicates the element that is currently being used by the script.	object	function	Pointer	none of these	Pointer
49	1	___________function used to find out index value.	Key( )	index( )	value( )	indexkey( )	Key( )
50	1	__________ is the another name for index .	Coordinate	key	id	none of these	key
51	1	_________ 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	1	_________ 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	1	_____function takes the content of the array and sort them in alphabetic order.	Sort( )	list( )	each( )	assort( )	Sort( )

54	1	________________takes arrays created with a string index and sort them according to their contents.	sort( )	each( )	asort( )	list( )	asort( )
55	1	_________ returns the appending element in reverse alphabetical order.	sort( ) and rsort( )	sort( ) and list( )	arsort( ) and list( )	rsort( ) and arsort( )	rsort( ) and arsort( )
56	1	_________ sorts the contents of an associated array according to index.	ksort( )	rsort( )	sort( )	list( )	ksort( )
57	1	__________function sorts multiple arrays or multidimensional arrays.	sort( )	array_multisort ( )	list( )	ksort( )	array_multisort( )
58	1	________ are controls that typically display several items in a list.	Text box	List box	Text area	none of these	List box
59	1	________ 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	1	________ attribute controls the way that the information is sends to the server.	Method attribute	Action attribute	Shift attribute	Function attribute	Method attribute

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

COIMBATORE - 641 021

PHP5/MYSQL [16CAP501]

UNIT II

Debugging and testing Tutorial

What about Debugging and testing?

Debugging and testing is done when a program has been coded by using

1. Compiler, generator or assembler
2. Interpreter

Compiler, generator or assembler:

- If a compiler, generator or assembler is used, the entire program is processed as a unit.
- Checks for syntax error and produce a list of detected syntax error.
- If syntax errors are detected then will go no further.
- If there is no syntax error then check the generated output. If output is incorrect then programmer checks for logical error.
- When the errors have been corrected the program is resubmitted to the Compiler, generator or assembler and the process of correcting the errors are repeated.

Interpreter:

- Each instruction is checked as it is entered.
- Instructions contain syntax error are corrected and reentered.
- Instructions without errors are executed immediately.

- Resulting output is examined for logical error.
- Logical errors must be corrected and interpreted and executed again.
- This process continues until all bugs have been removed from program.
- No object program is produced.

PHP Error Handling

When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks.

This tutorial contains some of the most common error checking methods in PHP.

We will show different error handling methods:

- Simple "die()" statements
- Custom errors and error triggers
- Error reporting

Basic Error Handling: Using the die() function

The first example shows a simple script that opens a text file:

```
<?php
$file=fopen("welcome.txt","r");
?>
```

If the file does not exist you might get an error like this:

```
Warning: fopen(welcome.txt) [function.fopen]: failed to open stream:
No such file or directory in C:\webfolder\test.php on line 2
```

To avoid that the user gets an error message like the one above, we test if the file exist before we try to access it:

```
<?php
if(!file_exists("welcome.txt"))
  {
  die("File not found");
  }
```

```
else
  {
  $file=fopen("welcome.txt","r");
  }
?>
```

Now if the file does not exist you get an error like this:

File not found

The code above is more efficient than the earlier code, because it uses a simple error handling mechanism to stop the script after the error.

However, simply stopping the script is not always the right way to go. Let's take a look at alternative PHP functions for handling errors.

Creating a Custom Error Handler

Creating a custom error handler is quite simple. We simply create a special function that can be called when an error occurs in PHP.

This function must be able to handle a minimum of two parameters (error level and error message) but can accept up to five parameters (optionally: file, line-number, and the error context):

Syntax

```
error_function(error_level,error_message,
error_file,error_line,error_context)
```

Parameter	Description
error_level	Required. Specifies the error report level for the user-defined error. Must be a value number. See table below for possible error report levels
error_message	Required. Specifies the error message for the user-defined error
error_file	Optional. Specifies the filename in which the error occurred
error_line	Optional. Specifies the line number in which the error occurred
error_context	Optional. Specifies an array containing every variable, and their values, in use when the error occurred

Error Report levels

These error report levels are the different types of error the user-defined error handler can be used for:

Value	Constant	Description
2	E_WARNING	Non-fatal run-time errors. Execution of the script is not halted
8	E_NOTICE	Run-time notices. The script found something that might be an error, but could also happen when running a script normally
256	E_USER_ERROR	Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function trigger_error()
512	E_USER_WARNING	Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function trigger_error()
1024	E_USER_NOTICE	User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function trigger_error()
4096	E_RECOVERABLE_ERROR	Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle (see also set_error_handler())
8191	E_ALL	All errors and warnings, except level E_STRICT (E_STRICT will be part of E_ALL as of PHP 6.0)

Now lets create a function to handle errors:

```
function customError($errno, $errstr)
  {
  echo "<b>Error:</b> [$errno] $errstr<br />";
  echo "Ending Script";
  die();
  }
```

The code above is a simple error handling function. When it is triggered, it gets the error level and an error message. It then outputs the error level and message and terminates the script.

Now that we have created an error handling function we need to decide when it should be triggered.

Set Error Handler

The default error handler for PHP is the built in error handler. We are going to make the function above the default error handler for the duration of the script.

It is possible to change the error handler to apply for only some errors, that way the script can handle different errors in different ways. However, in this example we are going to use our custom error handler for all errors:

```
set_error_handler("customError");
```

Since we want our custom function to handle all errors, the set_error_handler() only needed one parameter, a second parameter could be added to specify an error level.

Example

Testing the error handler by trying to output variable that does not exist:

```
<?php
//error handler function
function customError($errno, $errstr)
  {
  echo "<b>Error:</b> [$errno] $errstr";
  }

//set error handler
set_error_handler("customError");

//trigger error
echo($test);
?>
```

The output of the code above should be something like this:

Error: [8] Undefined variable: test

Trigger an Error

In a script where users can input data it is useful to trigger errors when an illegal input occurs. In PHP, this is done by the trigger_error() function.

Example

In this example an error occurs if the "test" variable is bigger than "1":

```
<?php
$test=2;
if ($test>1)
{
```

```
trigger_error("Value must be 1 or below");
}
?>
```

The output of the code above should be something like this:

```
Notice: Value must be 1 or below
in C:\webfolder\test.php on line 6
```

An error can be triggered anywhere you wish in a script, and by adding a second parameter, you can specify what error level is triggered.

Possible error types:

- E_USER_ERROR - Fatal user-generated run-time error. Errors that can not be recovered from. Execution of the script is halted
- E_USER_WARNING - Non-fatal user-generated run-time warning. Execution of the script is not halted
- E_USER_NOTICE - Default. User-generated run-time notice. The script found something that might be an error, but could also happen when running a script normally

Example

In this example an E_USER_WARNING occurs if the "test" variable is bigger than "1". If an E_USER_WARNING occurs we will use our custom error handler and end the script:

```
<?php
//error handler function
function customError($errno, $errstr)
  {
  echo "<b>Error:</b> [$errno] $errstr<br />";
  echo "Ending Script";
  die();
  }

//set error handler
set_error_handler("customError",E_USER_WARNING);

//trigger error
$test=2;
if ($test>1)
  {
  trigger_error("Value must be 1 or below",E_USER_WARNING);
  }
```

```
?>
```

The output of the code above should be something like this:

```
Error: [512] Value must be 1 or below
Ending Script
```

Now that we have learned to create our own errors and how to trigger them, lets take a look at error logging.

Error Logging

By default, PHP sends an error log to the servers logging system or a file, depending on how the error_log configuration is set in the php.ini file. By using the error_log() function you can send error logs to a specified file or a remote destination.

Sending errors messages to yourself by e-mail can be a good way of getting notified of specific errors.

Send an Error Message by E-Mail

In the example below we will send an e-mail with an error message and end the script, if a specific error occurs:

```
<?php
//error handler function
function customError($errno, $errstr)
  {
  echo "<b>Error:</b> [$errno] $errstr<br />";
  echo "Webmaster has been notified";
  error_log("Error: [$errno] $errstr",1,
  "someone@example.com","From: webmaster@example.com");
  }

//set error handler
set_error_handler("customError",E_USER_WARNING);

//trigger error
$test=2;
if ($test>1)
  {
  trigger_error("Value must be 1 or below",E_USER_WARNING);
  }
?>
```

The output of the code above should be something like this:

Error: [512] Value must be 1 or below
Webmaster has been notified

And the mail received from the code above looks like this:

Error: [512] Value must be 1 or below

This should not be used with all errors. Regular errors should be logged on the server using the default PHP logging system.

What is an Exception

With PHP 5 came a new object oriented way of dealing with errors.

Exception handling is used to change the normal flow of the code execution if a specified error (exceptional) condition occurs. This condition is called an exception.

This is what normally happens when an exception is triggered:

- The current code state is saved
- The code execution will switch to a predefined (custom) exception handler function
- Depending on the situation, the handler may then resume the execution from the saved code state, terminate the script execution or continue the script from a different location in the code

We will show different error handling methods:

- Basic use of Exceptions
- Creating a custom exception handler
- Multiple exceptions
- Re-throwing an exception
- Setting a top level exception handler

Basic Use of Exceptions

When an exception is thrown, the code following it will not be executed, and PHP will try to find the matching "catch" block.

If an exception is not caught, a fatal error will be issued with an "Uncaught Exception" message.

Lets try to throw an exception without catching it:

```
<?php
//create function with an exception
function checkNum($number)
  {
  if($number>1)
    {
    throw new Exception("Value must be 1 or below");
    }
  return true;
  }

//trigger exception
checkNum(2);
?>
```

The code above will get an error like this:

```
Fatal error: Uncaught exception 'Exception'
with message 'Value must be 1 or below' in C:\webfolder\test.php:6
Stack trace: #0 C:\webfolder\test.php(12):
checkNum(28) #1 {main} thrown in C:\webfolder\test.php on line 6
```

Try, throw and catch

To avoid the error from the example above, we need to create the proper code to handle an exception.

Proper exception code should include:

1. Try - A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However if the exception triggers, an exception is "thrown"
2. Throw - This is how you trigger an exception. Each "throw" must have at least one "catch"
3. Catch - A "catch" block retrieves an exception and creates an object containing the exception information

Lets try to trigger an exception with valid code:

```
<?php
//create function with an exception
function checkNum($number)
  {
  if($number>1)
    {
```

```
    throw new Exception("Value must be 1 or below");
    }
  return true;
  }

//trigger exception in a "try" block
try
  {
  checkNum(2);
  //If the exception is thrown, this text will not be shown
  echo 'If you see this, the number is 1 or below';
  }

//catch exception
catch(Exception $e)
  {
  echo 'Message: ' .$e->getMessage();
  }
?>
```

PHP Functions

In this chapter we will show you how to create your own functions.

To keep the script from being executed when the page loads, you can put it into a function.

A function will be executed by a call to the function.

You may call a function from anywhere within a page.

Create a PHP Function

A function will be executed by a call to the function.

Syntax

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

PHP function guidelines:

- Give the function a name that reflects what the function does

- The function name can start with a letter or underscore (not a number)

Example

A simple function that writes my name when it is called:

```
<html>
<body>

<?php
function writeName()
{
echo "Kai Jim Refsnes";
}

echo "My name is ";
writeName();
?>

</body>
</html>
```

Output:

```
My name is Kai Jim Refsnes
```

PHP Functions - Adding parameters

To add more functionality to a function, we can add parameters. A parameter is just like a variable.

Parameters are specified after the function name, inside the parentheses.

Example 1

The following example will write different first names, but equal last name:

```
<html>
<body>
```

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

echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Hege");
echo "My brother's name is ";
writeName("Stale");
?>

</body>
</html>
```

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes.
My brother's name is Stale Refsnes.
```

Example 2

The following function has two parameters:

```
<html>
<body>

<?php
function writeName($fname,$punctuation)
{
echo $fname . " Refsnes" . $punctuation . "<br />";
}
echo "My name is ";
writeName("Kai Jim",".");
```

```
echo "My sister's name is ";
writeName("Hege","!");
echo "My brother's name is ";
writeName("Ståle","?");
?>
</body></html>
```

Output: My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes!
My brother's name is Ståle Refsnes?

PHP Functions - Return values

To let a function return a value, use the return statement.

Example

```
<html>
<body>
<?php
function add($x,$y)
{
$total=$x+$y;
return $total;
}
echo "1 + 16 = " . add(1,16);
?>
</body></html>
```

Output:

```
1 + 16 = 17
```

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;
default:
  code to be executed if n is different from both label1 and label2;
}
```

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

```
<html>
<body>
<?php
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
}
?></body></html>
```

UNIT II

POSSIBLE QUESTIONS

Part - B (Each Question carries 6 Marks)

1. Explain in detail about numeric function in PHP with example.
2. Write a detail description about Errors in PHP.
3. Explain try/catch functions
4. Write a factorial program using recursion.
5. Define using Regexps to Check file path parameters?
6. Explain strlen( ),strstr( ),substr( )?
7. Briefly explain using coding standards (PEAR).
8. Explain about include and require statements
9. Generate a php script to check whether a given number is prime or not
10. Write a factorial program using recursion

Part - C (Each Question carries 10 Marks)

1. Analysis Testing and Debugging with examples
2. Design a program to explain form validation with sample program
3. Explain working with functions in PHP

KARPAGAM ACADEMY OF HIGHER EDUCATION

DEPARTMENT OF COMPUTER APPLICATIONS

Subject Name PHP5/MYSQL

Class: III MCA

SUBCODE: 16CAP501

Sem:V

S.No	Uni	Question	Choice1	Choice2	Choice3	Choice4	Answer Key
UNIT II		**UNIT II**					
1	2	_____ is the first step of debugging.	compiling	testing	analyzing	processing	testing
2	2	_____ occurs when your program produces an incorrect response or answer.	logic error	syntax error	fatal error	semantic error	logic error
3	2	____ sets the display of error messages to the screen on or off.	logic errors	view_errors	display_errors	Compling	display_errors
4	2	Runtime errors that stop your program from completing execution are termed ____	fatal	local	runtime error	bugs	fatal
5	2	_____ is used to set values to variables	=	= =	>=	<=	=
6	2	_____ is used for comparing values.	=	>=	= =		= =
7	2	____ function adds slashes wherever it finds string characters.	stripslashes()	addslashes()	putslashes()	string_slashes()	addslashes()
8	2	____occur when you leave out a semicolon.	semantic error	logic error	syntax error	notice error	syntax error
9	2	Syntax error is also known as	parse error	report error	notice error	logic error	parse error
10	2	Square bracket surrounding a pattern of character is called a ____	string class	word class	pattern class	character class	character class

11	2	URL stands for____	user resource locator	universal resource loader	uniform resource locator	uniform resource location	uniform resource locator
12	2	ISO stands for____	International standard organization	Indian standard organization	International service organization	Indian standard oganization	International standard organization
13	2	RAD stands for _____	Random Access Devices	Rapid Application Design	Rapid Application Device	Random Identifier	Rapid Application Design
14	2	_____ is used to comment a single line of code.	/	/*	**	//	//
15	2	____ is used to comment a block of code.	/* and */	// and //	* and *	//* and *//	/* and */
16	2	All data submitted in a browser to your web server is formatted as ____	numbers	character	strings	boolean	strings
17	2	Trouble shooting works for both ____ & ____ errors.	syntax & runtime	runtime & logic	syntax & logic	semantic & syntax	syntax & logic
18	2	____ gradually eliminates potential causes of problems until you have found the right	fatal	qualifier	boundaries	trouble shooting	trouble shooting
19	2	Serious errors cause PHP to simply quit processing, displaying a	Notice error	fatal error	warning error	flash	fatal error
20	2	Errors that are not quite serious may cause ____ message to be displayed.	flash	list error	blinking	warning error	warning error
21	2	The most recent error messages is available by enabling _____	recent_error	find_error	track_error	built_error	track_error
22	2	Notice error include _____ and user generated notices.	labels	runtime notices	error report	flash	runtime notices
23	2	PHP5 has _____ function to handle errors.	error()	track_error	try/catch	throw()	try/catch

24	2	In PHP _____ displays the value on the screen.	print	echo	disp()	cout	echo
25	2	____ statement ends all processing.	Exit	End	Break	Stop	Exit
26	2	____ function changes HTML tags into special characters.	special char()	convert()	HTMLspecial chars()	char_set()	HTMLspecial chars()
27	2	____functions are quantum leap more powerful when it comes to manipulating data.	Manip()	Regular expression	Compound expression	Exchg()	Regular expression
28	2	The _____ function is used to look for a string within a string.	substr()	find()	search()	strstr()	strstr()
29	2	____ function is used to separate out data values in a string.	extract()	exploded()	remove()	find()	exploded()
30	2	____ are like mini programming language for creating very powerful patterns.	regexps	minexp	exps	cmpexp	regexps
31	2	PHP’s regular expression functions that allow Perl notation are called ____ functions.	PREXP	PCRE	PHPEXP	EXGP	PCRE
32	2	____ is used to store successfully matched expressions.	exp()	egep()	match()	ereg()	ereg()
33	2	The symbols that can be used to indicate the location on the string	locators	matcher	anchors	provider	anchors
34	2	____ anchor appears at the beginning of the pattern anchoring a	^	&	@	$	^
35	2	____ anchor appears at the end of the pattern anchoring a match to the end of the string.	#	$	?	%	$
36	2	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	2	____ operator in regular expression is same as bitwise “or” operator.	either-or	||	OR	AND	either-or
38	2	____ are used to set limits and ranges on the quantity of characters to be matched.	Delimiter	Match()	Quantifier	Boundaries	Quantifier
39	2	____ 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	2	____meta character shows a word character.	\c	\wc	\w	\m	\w
41	2	_____ meta character shows a non-digit.	\D	\d	\nd	/d	\D
42	2	_____ meta character shows any character.	*	/	.	^	.
43	2	\s is used to indicate that it is a _____	string	white space	shift	control	white space
44	2	The meta character \d means that it is a _____	digit	date	non-dight	double	digit
45	2	_____ function is used to find out index value.	val()	index()	key()	ereg()	key()
46	2	The error_log function can take up to _____ arguments.	three	five	one	four	four
47	2	The function transfers any argument values into new variables called ______	parameters	qualifier	meta character	meta data	parameters
48	2	_____ 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	2	Multiple parameters are separated by _____	semi colon	colon	slashes	commas	commas

50	2	Calling a function from within itself is known as _____	recursion	looping	branching	nesting	recursion
51	2	The process of creating and calling functions within functions is known as _____	recursion	nesting	branching	looping	nesting
52	2	_____ is used to bring external files into the current scripts and run them.	import	accept	extern	include	include
53	2	A failure of require results in a _____	fatal error	logic error	warning	syntax error	fatal error
54	2	A failure of include results in a _____	syntax error	fatal error	flash error	warning	warning
55	2	_____ variables are created outside a function and remain alive until the script ends.	local	global	static	char	global
56	2	_____ are created inside a function.	fatal	global	local	logical	local
57	2	The name of the global variable is preceded by _____	underscore	backslash	pound	dot	underscore
58	2	If we leave out an argument, the function will automatically assume a _____ for numeric argument	three	zero	two	one	zero
59	2	_____ function is used to determine whether the form has been submitted.	submit()	is_set()	isset()	return()	isset()
60	2	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

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

COIMBATORE - 641 021

PHP5/MYSQL [16CAP501]

UNIT III

PHP File Handling

Opening a File

The fopen() function is used to open files in PHP.

The first parameter of this function contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened:

```
<html>
<body>

<?php
$file=fopen("welcome.txt","r");
?>

</body>
</html>
```

The file may be opened in one of the following modes:

Modes	Description
r	Read only. Starts at the beginning of the file
r+	Read/Write. Starts at the beginning of the file
w	Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist
w+	Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist

a	Append. Opens and writes to the end of the file or creates a new file if it doesn't exist
a+	Read/Append. Preserves file content by writing to the end of the file
x	Write only. Creates a new file. Returns FALSE and an error if file already exists
x+	Read/Write. Creates a new file. Returns FALSE and an error if file already exists

Example

The following example generates a message if the fopen() function is unable to open the specified file:

```
<html>
<body>

<?php
$file=fopen("welcome.txt","r") or exit("Unable to open file!");
?>

</body>
</html>
```

Closing a File

The fclose() function is used to close an open file:

```
<?php
$file = fopen("test.txt","r");

//some code to be executed

fclose($file);
?>
```

Check End-of-file

The feof() function checks if the "end-of-file" (EOF) has been reached.

The feof() function is useful for looping through data of unknown length.

Note: You cannot read from files opened in w, a, and x mode!

```
if (feof($file)) echo "End of file";
```

Reading a File Line by Line

The fgets() function is used to read a single line from a file.

Note: After a call to this function the file pointer has moved to the next line.

Example

The example below reads a file line by line, until the end of file is reached:

```
<?php
$file = fopen("welcome.txt", "r") or exit("Unable to open file!");
//Output a line of the file until the end is reached
while(!feof($file))
  {
  echo fgets($file). "<br />";
  }
fclose($file);
?>
```

Reading a File Character by Character

The fgetc() function is used to read a single character from a file.

Note: After a call to this function the file pointer moves to the next character.

Example

The example below reads a file character by character, until the end of file is reached:

```
<?php
$file=fopen("welcome.txt","r") or exit("Unable to open file!");
while (!feof($file))
  {
```

```
  echo fgetc($file);
  }
fclose($file);
?>
```

Create an Upload-File Form

To allow users to upload files from a form can be very useful.

Look at the following HTML form for uploading files:

```
<html>
<body>

<form action="upload_file.php" method="post"
enctype="multipart/form-data">
<label for="file">Filename:</label>
<input type="file" name="file" id="file" />
<br />
<input type="submit" name="submit" value="Submit" />
</form>

</body>
</html>
```

Notice the following about the HTML form above:

- The enctype attribute of the <form> tag specifies which content-type to use when submitting the form. "multipart/form-data" is used when a form requires binary data, like the contents of a file, to be uploaded
- The type="file" attribute of the <input> tag specifies that the input should be processed as a file. For example, when viewed in a browser, there will be a browse-button next to the input field

Note: Allowing users to upload files is a big security risk. Only permit trusted users to perform file uploads.

Create The Upload Script

The "upload_file.php" file contains the code for uploading a file:

```
<?php
```

```
if ($_FILES["file"]["error"] > 0)
  {
  echo "Error: " . $_FILES["file"]["error"] . "<br />";
  }
else
  {
  echo "Upload: " . $_FILES["file"]["name"] . "<br />";
  echo "Type: " . $_FILES["file"]["type"] . "<br />";
  echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
  echo "Stored in: " . $_FILES["file"]["tmp_name"];
  }
?>
```

By using the global PHP $_FILES array you can upload files from a client computer to the remote server.

The first parameter is the form's input name and the second index can be either "name", "type", "size", "tmp_name" or "error". Like this:

- $_FILES["file"]["name"] - the name of the uploaded file
- $_FILES["file"]["type"] - the type of the uploaded file
- $_FILES["file"]["size"] - the size in bytes of the uploaded file
- $_FILES["file"]["tmp_name"] - the name of the temporary copy of the file stored on the server
- $_FILES["file"]["error"] - the error code resulting from the file upload

This is a very simple way of uploading files. For security reasons, you should add restrictions on what the user is allowed to upload.

Restrictions on Upload

In this script we add some restrictions to the file upload. The user may only upload .gif or .jpeg files and the file size must be under 20 kb:

```
<?php
if ((($_FILES["file"]["type"] == "image/gif")
|| ($_FILES["file"]["type"] == "image/jpeg")
|| ($_FILES["file"]["type"] == "image/pjpeg"))
&& ($_FILES["file"]["size"] < 20000))
  {
  if ($_FILES["file"]["error"] > 0)
    {
    echo "Error: " . $_FILES["file"]["error"] . "<br />";
```

```
   }
  else
   {
   echo "Upload: " . $_FILES["file"]["name"] . "<br />";
   echo "Type: " . $_FILES["file"]["type"] . "<br />";
   echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
   echo "Stored in: " . $_FILES["file"]["tmp_name"];
   }
  }
else
  {
  echo "Invalid file";
  }
?>
```

Note: For IE to recognize jpg files the type must be pjpeg, for FireFox it must be jpeg.

Saving the Uploaded File

The examples above create a temporary copy of the uploaded files in the PHP temp folder on the server.

The temporary copied files disappears when the script ends. To store the uploaded file we need to copy it to a different location:

```
<?php
if ((($_FILES["file"]["type"] == "image/gif")
|| ($_FILES["file"]["type"] == "image/jpeg")
|| ($_FILES["file"]["type"] == "image/pjpeg"))
&& ($_FILES["file"]["size"] < 20000))
  {
  if ($_FILES["file"]["error"] > 0)
   {
   echo "Return Code: " . $_FILES["file"]["error"] . "<br />";
   }
  else
   {
   echo "Upload: " . $_FILES["file"]["name"] . "<br />";
   echo "Type: " . $_FILES["file"]["type"] . "<br />";
   echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
   echo "Temp file: " . $_FILES["file"]["tmp_name"] . "<br />";

   if (file_exists("upload/" . $_FILES["file"]["name"]))
```

```
      {
      echo $_FILES["file"]["name"] . " already exists. ";
      }
    else
      {
      move_uploaded_file($_FILES["file"]["tmp_name"],
      "upload/" . $_FILES["file"]["name"]);
      echo "Stored in: " . "upload/" . $_FILES["file"]["name"];
      }
    }
  }
else
  {
  echo "Invalid file";
  }
?>
```

The script above checks if the file already exists, if it does not, it copies the file to the specified folder.

PHP Directory Functions

PHP Directory Functions

PHP: indicates the earliest version of PHP that supports the function.

Function	Description	PHP
chdir()	Changes the current directory	3
chroot()	Changes the root directory of the current process	4
dir()	Opens a directory handle and returns an object	3
closedir()	Closes a directory handle	3
getcwd()	Returns the current directory	4
opendir()	Opens a directory handle	3

readdir()	Returns an entry from a directory handle	3
rewinddir()	Resets a directory handle	3
scandir()	Lists files and directories inside a specified path	5

PHP Directory Constants

PHP: indicates the earliest version of PHP that supports the constant.

Constant	**Description**	**PHP**
DIRECTORY_SEPARATOR		3
PATH_SEPARATOR		4

File copy: Description

bool **copy** (string *$source* , string *$dest* [, resource *$context*])

Makes a copy of the file *source* to *dest*.

If you wish to move a file, use the rename() function.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Rename — Renames a file or directory

Description

bool **rename** (string *$oldname* , string *$newname* [, resource *$context*])

Attempts to rename *oldname* to *newname*.

Example #1 Example with rename()

```
<?php
rename("/tmp/tmp_file.txt", "/home/user/login/docs/my_file.txt");
?>
```

PHP - File Delete

In PHP you delete files by calling the *unlink* function.

PHP - File Unlink

When you view the contents of a directory you can see all the files that exist in that directory because the operating system or application that you are using displays a list of filenames. You can think of these filenames as links that join the files to the directory you are currently viewing.

If you unlink a file, you are effectively causing the system to forget about it or delete it!

Before you can delete (unlink) a file, you must first be sure that it is not open in your program. Use the *fclose* function to close down an open file.

PHP - Unlink Function

Remember from the PHP File Create lesson that we created a file named *testFile.txt*.

PHP Code:

```
$myFile = "testFile.txt";
$fh = fopen($myFile, 'w') or die("can't open file");
fclose($fh);
```

Now to delete *testFile.txt* we simply run a PHP script that is located in the same directory. Unlink just needs to know the name of the file to start working its destructive magic.PHP Code:

```
$myFile = "testFile.txt";
unlink($myFile);
```

The *testFile.txt* should now be removed.

UNIT III

POSSIBLE QUESTIONS

Part - B (Each Question carries 6 Marks)

1. Write a program for overriding methods.
2. Explain in detail about working with files and directories.
3. Explain about working with directories?
4. Illustrate building a text editor.
5. Illustrate the ownership & permissions with different commands.
6. Explain the following in detail with example:

   i) fgetc( ) ii) feoff( ) iii) fgets( ) iv) fseek( ) v) ftell( )

7. Write a detail description about directory functions with example.
8. Write a detailed note on traversing a directory hierarchy.
9. How will you create a directory navigator? Explain.
10. Explain inheritance with suitable sample program

Part - C (Each Question carries 10 Marks)

1. Develop an application to demonstrate E-ticketing
2. Design a program to uploading multiple files
3. Explain locking non transactional tables

KARPAGAM ACADEMY OF HIGHER EDUCATION

DEPARTMENT OF COMPUTER APPLICATIONS

Subject Name PHP5/MYSQL

Class: III MCA

SUBCODE: 16CAP501

Sem:V

S.N	Uni	Question	Choice1	Choice2	Choice3	Choice4	Answer Key
UNIT-							
1	3	___ is an ordered sequence of bytes stored on hard disk , floppy disk	Memory	file	record	disk	file
2	3	___________is a special type of file that holds the names of other files and directories.	file	memory	directory	disk	directory
3	3	A file handle is simply an _______ value that is used to identify the file.	integer	double	floating point	string	integer
4	3	________ function , which used to read data from a file and takes a	file( )	fopen( )	a+	r+	file( )
5	3	________ is used to open a file.	file()	fopen( )	a+	r+	fopen( )
6	3	___________ value is used to open file for reading & writing.	file()	fopen( )	a+	r+	r+
7	3	a+ value is used to open file for ________ &_________.	reading & appending	writing & reading	appending & deleting	writing&appending	reading & appending
8	3	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	3	____ function returns an indexed array that contain file statistic and information within each spot in the array.	fread( )	stat( )	fget c( )	sort( )	stat( )

10	3	___________ function can be used to extract a character string from a file.	fread( )	stat( )	fget c( )	sort( )	fread( )
11	3	_____________ function can be used to read from file one character at a time.	fread( )	stat( )	fget c( )	sort( )	fget c( )
12	3	___________ function must be given a valid file handle a numerical value higher than the length of each line.	fgetcsv( )	fread( )	stat( )	fwrite( )	fgetcsv( )
13	3	fputs( ) function is simply an alias for _____________.	fgetcsv( )	fread( )	stat( )	fwrite( )	fwrite( )
14	3	_____________ takes only one argument.	file( ) function	fwrite( ) function	fread( ) function	file mtime( )	file( ) function
15	3	____________ 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	3	____________ returns the time at which the file was last changed as a UNIX timestamp.	Fpassthru( )	filectime( )	filemtime( )	fwrite( ) function	filectime( )
17	3	________ returns the time at which the file was last modified as a UNIX timestamp.	Fpassthru( )	filectime( )	filemtime( )	fread( ) function	filemtime( )
18	3	__________returns the user ID of the owner of the specified file.	fileowners( )	filegroup( )	file type( )	fwrite( ) function	fileowners( )
19	3	_________returns the group ID of the owner of the specified file.	fileowners( )	filegroup( )	file type( )	fwrite( ) function	filegroup( )
20	3	___________ returns the type of the specified file.	fileowners( )	filegroup( )	file type( )	fwrite( ) function	file type( )
21	3	__________ is designed specifically to work with directories.	is_dir( )	is_file()	dir	file	is_dir( )

22	3	_______ returns true if the given file name refers to a regular file.	is_dir( )	is_file( )	dir	file	is_file( )
23	3	PHP also enables you to _______ ,________&______ files.	copy ,rename ,delete	copy ,paste, delete	cut ,copy, paste	edit, copy, paste	copy ,rename ,delete
24	3	_________ function is used to rename a file.	rename( )	rewind dir( )	rmdir( )	chdir( )	rename( )
25	3	________ function takes single string argument referring to a name of a file you want to delete.	unlink( )	sort( )	fread( )	delete()	unlink( )
26	3	Many versions of PHP for windows donot support the _______ function.	sort( )	unlink( )	fread( )	delete()	unlink( )
27	3	________ is called to arrange the array entries in ascending order, and display are accept the current & parent directories.	unlink( )	chdir( )	sort	dir( )	sort
28	3	_______ 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	3	_______ function call changes PHP’s current directory to the given directory.	mkdir( )	chdir( )	rewinddir( )	rename( )	chdir( )
30	3	_________ function removes a given directory.	rmdir( )	chdir( )	rewinddir( )	rename( )	rmdir( )
31	3	The ___________ function creates a directory as specified in its first argument.	rename( )	rmdir( )	chdir( )	mkdir( )	mkdir( )

32	3	The __________ function returns the directory part of a given filename.	dirname( )	rmdir( )	chdir( )	mkdir( )	dirname( )
33	3	___________ & __________ are the two properties provided by the dir object.	read( ) & write( )	handle & path	path & time	path&date	handle & path
34	3	_____________ 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	3	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	3	_________ function print out the contents of a given file in a new window.	display( )	show()	print	project	display( )
37	3	__________ 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	3	By creating a special function called ______ we can perform any activities required to instantiate the object.	_construct( )	_set()	_this()	_any()	_construct( )
39	3	The constructor opens a connection to the database and stores the resource handle in _________.	$this->-hdb	_construct( )	_set( )	_this()	$this->-hdb
40	3	In ________ we can trap the destruction of the object and take action.	PHP	PHP4	PHP5	PHP6	PHP5

41	3	The ___________ method is changed to keep track of any properties that might have been modified in the variable.	_construct( )	_set( )	_get()	_this()	_set( )
42	3	_____________ is the capability of an application to do different things based on the particular object.	polymorphism	inheritance	encapsulation	object	polymorphism
43	3	The ability to hide the details of implementation is known as ____________.	polymorphism	inheritence	encapsulation	object	encapsulation
44	3	______________ is an important concept in object_oriented programming.	class	inheritence	encapsulation	object	encapsulation
45	3	Static methods are invoked with the _________ operator.	?	::	?::	$	::
46	3	The __________ function is also very useful when working with timestamps.	gettime( )	getdata( )	getdate( )	getdisplay()	getdate( )
47	3	______________ method returns the user's decision.	confirm	decision	return( )	display	confirm
48	3	Opened a file or directory to be used using_____________ or_________.	open( ) ,dir( )	fopen( ) ,dir( )	fopen( ) ,opendir	fclose(),dir()	fopen( ) ,opendir
49	3	___________, which are the "blueprints" for an object and are the actual code that defines the properties and methods.	function	methods	classes	objects	classes
50	3	_________,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	3	The _____________ function enables to print the contents of a file without even having to call fopen( ).	readfile( )	closefile( )	writefile( )	openfile( )	readfile( )
52	3	_________ will move the file position indicator associated with fp to a position determined by offset.	ftell( )	flength( )	fseek	fsetpos( )	fseek
53	3	The beginning of the file + offset is _________.	SEEK_SET	SEEK_CUR	SEEK_END	SEEK_OPEN	SEEK_SET
54	3	The __________ function returns the next entry listed in the open directory.	closedir( )	dir( )	dir_list	readdir( )	readdir( )
55	3	__________ occurs when the next entry is a subdirectory.	recursion	function	method	array	recursion
56	3	The ________ function is invoked automatically when you instantiate a new object of class property object .	_set	_get	_construct	_thik()	_construct
57	3	The word ________ literally means to place in a capsule, or outer container.	inheritance	encapsulation	polymorphism	abstraction	encapsulation
58	3	Classes in PHP can now have destructors,through the ___________ method.	_destruct( )	_destroy	_delete	_set	_destruct( )
59	3	Which one of the following is not an OOP benefit?	Code reusability	easy to use	modularity	equal	none of these
60	3	The absolute path to the home directory of the user is _____________.	dir	shell	gid	gecos	dir

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

COIMBATORE - 641 021

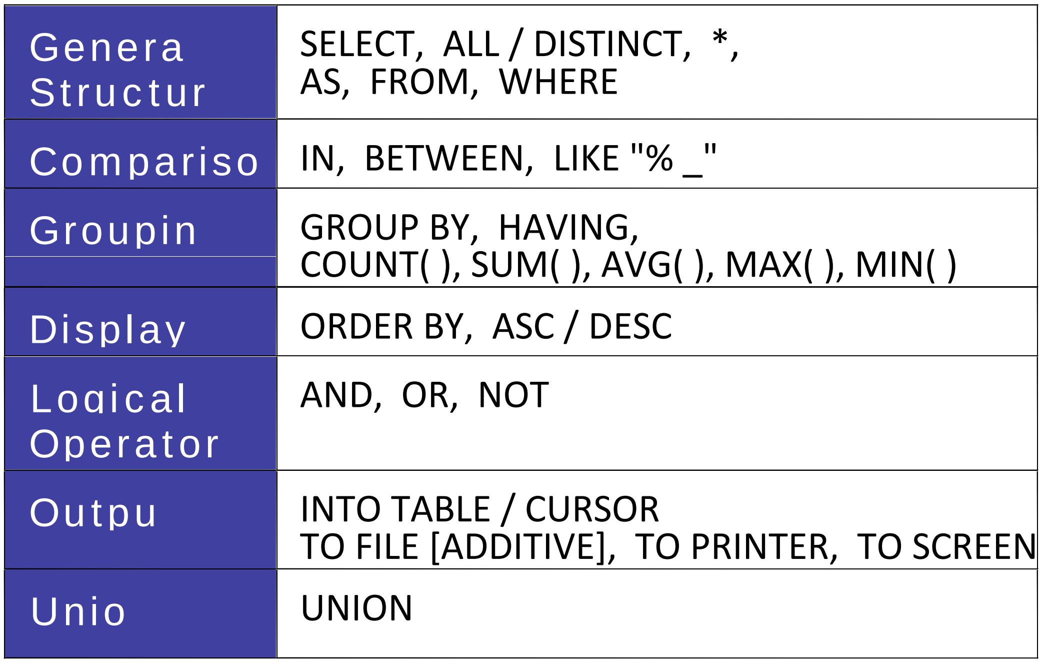
PHP5/MYSQL [16CAP501]

UNIT IV

SQL Framework:

- When a user wants to get some information from a database file, he can issue a ***query***.
- A query is a user–request to retrieve data or information with a certain condition.
- SQL is a query language that allows user to specify the conditions. (instead of algorithms)

Basic structure of an SQL query



Genera Structur	SELECT, ALL / DISTINCT, *, AS, FROM, WHERE
Compariso	IN, BETWEEN, LIKE "% _"

Groupin	GROUP BY, HAVING,
	COUNT( ), SUM( ), AVG( ), MAX( ), MIN( )

Display	ORDER BY, ASC / DESC
Logical Operator	AND, OR, NOT
Outpu	INTO TABLE / CURSOR TO FILE [ADDITIVE], TO PRINTER, TO SCREEN
Unio	UNION

SELECT ...... FROM ...... WHERE ......

SELECT [ALL / DISTINCT] *expr1* [AS *col1*], *expr2* [AS *col2*] ;

FROM *tablename* WHERE *condition*

SELECT [ALL / DISTINCT] *expr1* [AS *col1*], *expr2* [AS *col2*] ;

FROM *tablename* WHERE *condition*

- DISTINCT will eliminate duplication in the output while ALL will keep all duplicated rows.
- *condition* can be :
  - (1) an inequality, or
  - (2) a string comparison
  - using logical operators AND, OR, NOT.

List all the student records.

SELECT * FROM student

SELECT name, hcode, class FROM student ;

WHERE class="1A"

Functions:

# days :DATE( ) – dob

# years :(DATE( ) – dob) / 365

1 d.p.: ROUND(__ , 1)

SELECT name, ROUND((DATE( )-dob)/365,1) AS age ;

FROM student WHERE class="1B" AND sex="F"

Comparison:

expr IN (*value1*, *value2*, *value3*)

expr BETWEEN *value1* AND *value2*

expr LIKE "%_"

List the students who were born on Wednesday or Saturdays.

SELECT name, class, CDOW(dob) AS bdate ; FROM student ;

WHERE DOW(dob) IN (4,7)

List the 1A students whose Math test score is between 80 and 90 (incl.)

SELECT name, mtest FROM student ;

WHERE class="1A" AND ;

mtest BETWEEN 80 AND 90

List the students whose names start with "T".

SELECT name, class FROM student ;

WHERE name LIKE "T%"

String Operations:

SQL includes a string-matching operator for comparisons on character strings. Patterns are described using two special characters:

percent (%). The % character matches any substring.

underscore (_). The _ character matches any character.

Find the names of all customers whose street includes the substring “Main”.

```
        select customer-name
    from customer
    where customer-street like ‘%Main%’
```

Match the name “Main%”

```
            like ‘Main\%’ escape  ‘\’
```

SQL supports a variety of string operations such as

concatenation (using “||”)

converting from upper to lower case (and vice versa)

finding string length, extracting substrings, etc.

n List in alphabetic order the names of all customers having a loan in Perryridge branch

```
        select distinct customer-name
from    borrower, loan
where borrower loan-number - loan.loan-number and
        branch-name = 'Perryridge'
order by customer-name
```

n We may specify desc for descending order or asc for ascending order, for each attribute; ascending order is the default.

H E.g. order by *customer-name* desc

Aggregate Functions

n These functions operate on the multiset of values of a column of a relation, and return a value

avg: average value
min: minimum value
max: maximum value
sum: sum of values
count: number of values

n Find the average account balance at the Perryridge branch.

```
        select avg (balance)
from account
where branch-name = 'Perryridge'
```

n Find the number of tuples in the *customer* relation.

```
        select count (*)
from customer
```

n Find the number of depositors in the bank.

```
        select count (distinct customer-name)
from depositor
```

Nested Subqueries

n SQL provides a mechanism for the nesting of subqueries.

n A subquery is a select-from-where expression that is nested within another query.

n A common use of subqueries is to perform tests for set membership, set comparisons, and set cardinality.

Example Query

n Find the names of all branches that have greater assets than all branches located in Brooklyn.

```
        select branch-name
from branch
where assets > all
        (select assets
        from branch
        where branch-city = ‘Brooklyn’)
```

Modification of the Database – Insertion

n Add a new tuple to *account*

```
        insert into account
        values (‘A-9732’, ‘Perryridge’,1200)
```

or equivalently

```
insert into account (branch-name, balance, account-number)
    values (‘Perryridge’, 1200, ‘A-9732’)
```

n Add a new tuple to *account* with *balance* set to null

```
        insert into account
        values (‘A-777’,‘Perryridge’, null)
```

n Provide as a gift for all loan customers of the Perryridge branch, a $200 savings account. Let the loan number serve as the account number for the new savings account

```
      insert into account
    select loan-number, branch-name,  200
    from loan
    where branch-name = ‘Perryridge’
   insert into depositor
    select customer-name, loan-number
    from loan, borrower
```

```
where branch-name = ‘Perryridge’
        and loan.account-number = borrower.account-number
```

n The select from where statement is fully evaluated before any of its results are inserted into the relation (otherwise queries like

```
insert into table1 select * from table1
```

would cause problems

Modification of the Database – Updates

n Increase all accounts with balances over $10,000 by 6%, all other accounts receive 5%.

H Write two update statements:

```
        update account
set balance = balance ∗ 1.06
where balance > 10000



        update account
set balance = balance ∗ 1.05
where balance ≤ 10000
```

H The order is important

H Can be done better using the case statement (next slide)

Case Statement for Conditional Updates

n Same query as before: Increase all accounts with balances over $10,000 by 6%, all other accounts receive 5%.

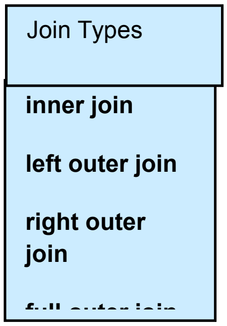
```
    update account
set balance =  case
                   when balance <= 10000 then balance *1.05
                   else   balance * 1.06
                 end
```

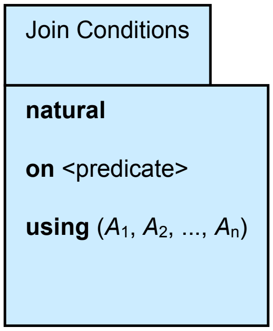
Joined Relations

n Join operations take two relations and return as a result another relation.

n These additional operations are typically used as subquery expressions in the from clause n Join condition – defines which tuples in the two relations match, and what attributes are present in the result of the join.

n Join type – defines how tuples in each relation that do not match any tuple in the other relation (based on the join condition) are treated.





Joined Relations – Datasets for Examples

Relation *loan*

Find all customers who have either an account or a loan (but not

both) at the bank.

select *customer-name*
from (*depositor* natural full outer join *borrower*)
where *account-number* is *null* or *loan-number* is *null*

Date/Time Types in SQL (Cont.):

n date. Dates, containing a (4 digit) year, month and date

H E.g. date ‘2001-7-27’

n time. Time of day, in hours, minutes and seconds.

H E.g. time ’09:00:30’ time ’09:00:30.75’

n timestamp: date plus time of day

H E.g. timestamp ‘2001-7-27 09:00:30.75’

n Interval: period of time

H E.g. Interval ‘1’ day

H Subtracting a date/time/timestamp value from another gives an interval value

H Interval values can be added to date/time/timestamp values

n Can extract values of individual fields from date/time/timestamp

H E.g. extract (year from r.starttime)

n Can cast string types to date/time/timestamp

H E.g. cast <string-valued-expression> as date

Transactions

Transaction can complete in one of four
ways:
- COMMIT ends transaction successfully,making changes permanent.
- ROLLBACK aborts transaction, backing outany changes made by transaction.
- For programmatic SQL, successful programtermination ends final transaction successfully, even if COMMIT has not beenexecuted.
- For programmatic SQL, abnormal programend aborts transaction.

New transaction starts with nexttransaction-initiating statement.
• SQL transactions cannot be nested.
• SET TRANSACTION configures
SET TRANSACTION
[READ ONLY | READ WRITE] |
[ISOLATION LEVEL READ UNCOMMITTED |
READ COMMITTED|REPEATABLE READ
|SERIALIZABLE]

Access Control – Authorization, Identifiers and Ownership

Authorization identifier is normal SQL identifier used to establish identity of a user. Usually has an associated password.

• Used to determine which objects user may reference and what operations may be

performed on those objects.

• Each object created in SQL has an owner, as defined in AUTHORIZATION clause of

schema to which object belongs.

Privileges:

Actions user permitted to carry out on given base table or view:

SELECT Retrieve data froma table.

INSERT Insert new rows into a table.

UPDATE Modify rows of data in a table.

DELETE Delete rows of data froma table.

REFERENCES Reference columns of named table in integrity constraints.

USAGE Use domains, collations, character sets, and translations.

GRANT:

GRANT {PrivilegeList | ALL PRIVILEGES}

ON ObjectName

TO {AuthorizationIdList | PUBLIC}

[WITH GRANT OPTION]

• PrivilegeList consists of one or more of above privileges separated by commas.

• ALL PRIVILEGES grants all privileges to a user.

PUBLIC allows access to be granted to all present and future authorized users.

• ObjectName can be a base table, view, domain, character set, collation or translation.

• WITH GRANT OPTION allows privileges to be passed on.

Give Manager full privileges to Staff table.

GRANT ALL PRIVILEGES

ON Staff

TO Manager WITH GRANT OPTION;

Give users Personnel and Director SELECT and UPDATE on column salary of Staff.

GRANT SELECT, UPDATE (salary)

ON Staff

TO Personnel, Director;

REVOKE:

REVOKE takes away privileges granted with GRANT.

REVOKE [GRANT OPTION FOR]

{PrivilegeList | ALL PRIVILEGES}

ON ObjectName

FROM {AuthorizationIdList | PUBLIC}

[RESTRICT | CASCADE]

- ALL PRIVILEGES refers to all privileges granted to a user by user revoking privileges.

Example:
Revoke privilege SELECT on Branch table fromall users.
REVOKE SELECT
ON Branch
FROMPUBLIC;
Revoke all privileges given to Director on Staff table.
REVOKE ALL PRIVILEGES
ON Staff
FROMDirector;

UNIT IV

POSSIBLE QUESTIONS

Part - B (Each Question carries 6 Marks)

1. Explain briefly about full joins, outer joins, and natural joins.
2. Explain briefly about Copying Data in to a table
3. Explain briefly about setting the Transaction Isolation Level
4. Explain briefly about exporting and importing data in to a table.
5. Explain briefly about optional clauses of a SELECT statement in MySQL.
6. Explain the following: i) Numeric functions ii) Date/Time functions
7. Explain string function with sample program
8. Explain briefly about Performing a Basic Transaction
9. Explain commit, auto commit, revoke and grant commands
10. Write notes on creating subqueries in SQL

Part - C (Each Question carries 10 Marks)

1. How to create indexes in table? Explain briefly about its types
2. Develop an application for online advertisement for car showroom management.
3. Explain locking nontransactional tables

KARPAGAM ACADEMY OF HIGHER EDUCATION

DEPARTMENT OF COMPUTER APPLICATIONS

Subject Name PHP5/MYSQL

Class: III MCA

SUBCODE: 16CAP501

Sem:V

S.N	Uni	Question	Choice1	Choice2	Choice3	Choice4	Answer Key
		Unit IV					
1	4	___ is the most universally implemented database language.	SQL	C	C++	JAVA	SQL
2	4	The first commercial RDBMS to hit the market.	Oracle9i	Oracle8i	Oracle	sql	Oracle
3	4	ANSI released the first published SQL standard in the year ___	1984	1986	1990	1991	1986
4	4	ISO stands for _______	Indian Organization for Standardization.	International Standardized Organization.	International Organization for Standardization.	Indian Standard Organization.	International Organization for Standardization.
5	4	____ is a set of features that a vendor could implement in a RDBMS.	Stored procedure	Persistent module	Persistent stored module	Package	Package
6	4	_____ is a collection of SQL statements	Package	Persistent stored module	Stored module	Persistent module	Stored module
7	4	PSM stands for	Persistent stored module	Program stored module	Procedure stored module	Package stored module	Persistent stored module
8	4	____ is a grid like structure	Data base	Table	Table index	Index	Table
9	4	____ is a list of values taken from a specified column.	Data base	Index	Table	Table index	Index

10	4	The CREATE TABLE statement is a type of _____	DDL statement	DML statement	DCL statement	DBA statement	DDL statement
11	4	____ acts as central point of administration for the tables in the data base.	Data	Data base	Queries	statements	Data base
12	4	____ specifies which default collation to use.	COLLATE clause	Character set	DEFAULT clause	VALUE clause.	COLLATE clause
13	4	A ____ is a named sorting order used for a specified character set.	Collation	Character set	Default	Value.	Collation
14	4	The statement is one of the most complex SQL statements in MYSQL.	MODIFY TABLE	CREATE TABLE	DELETE TABLE	UPDATE TABLE	CREATE TABLE
15	4	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	4	Decimal data base is referred to as ______	Exact	Numeric	Approximate	Alphabet	Exact
17	4	The binary data type takes ______	One argument	Two argument	Do not take any argument	More than one argument	Do not take any argument
18	4	The ____ provides a handy way to record each transaction that occurs in a particular table	DATE	DATE TIME	TIMESTAMP	TIME	TIMESTAMP
19	4	DATE data type ranges between	1000-01-01 00:00:00 through 9999	1000-01-01 through 9999	1901 to 2155	1900 to 2786	1000-01-01 through 9999
20	4	MYSQL allows to assign default values through the use of a _____	DEFAULT clause	Character set	Collate clause	Value clause	DEFAULT clause
21	4	The default values for TIMESTAMP columns in the table are ____ values in place of the date and time	Zero	One	Two	Default	Zero
22	4	For numeric columns that are not configured with the AUTO-INCREMENT option, the default value is ____	One	Two	Three	Zero	Three

23	4	For columns configured with the ___ data type the default value is the first Value specified in the column definition.					
24	4	The table that contains the foreign key, is referred to as ____	Parent table	Child table	Referencing table	Key table	Child table
25	4	The referenced table is referred to as _____	Parent table	Child table	Referencing table	Key table	Parent table
26	4	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	4	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	4	____ constraint is used to create a unique index.	primary key	foreign key	unique	Candidate key.	unique
29	4	An ___ statement is the most common method used to directly insert data in a table.	add	insert	update	delete	insert
30	4	An ___ is a type of formula that helps define the value to insert in a column.	Expression	Insert	Add	Append	Expression
31	4	A __ is an object that carries out a predefined task.	Variable	Calling function	function	Entity	function
32	4	___ function can be used in SQL statements to return value that is equalant to the current date and time.	date( )	time( )	now( )	date( ) and time( )	now( )
33	4	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	4	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	4	____ statement deletes the old row and adds the new row.	REPLACE	DELETE	DROP	REMOVE	REPLACE
36	4	____ clause includes one or more conditions that define the extent of the delete operation.	ORDER BY	LIMIT	WHERE	DELETE	WHERE
37	4	____ clause sorts rows according to the column or columns specified in the clause.	ORDER BY	LIMIT	WHERE	SORT	ORDER BY
38	4	The clause limits the number of rows to be deleted to the number specified in the clause.	ORDER BY	LIMIT	WHERE	DELETE	LIMIT
39	4	The ___ statement removes all rows from a table.	DELETE	TRUNCATE	REMOVE	DROP	TRUNCATE
40	4	A ____ is a set of one or more SQL statements that perform a set of related action	Transaction	Trigger	Queries	view	Transaction
41	4	The SELECT clause includes SELECT keyword and ____	+	*	;	/	;
42	4	An ____ function is type of function that summarizes data such as Count ( ) function.	Aggregate ( )	Sum ( )	Cal( )	add()	Aggregate ( )
43	4	____ operates perform calculations on the arguments within an expression.	Arithmetic	comparison	logical	bitwise	Arithmetic
44	4	_____ operators compare the arguments in an expression test whether a condition is true, false, or null.	Arithmetic	comparison	sort	bitwise	comparison
45	4	___ 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	4	___ operators manipulate the bit values associated with numerical values	bitwise	comparison	logical	Arithmetic	bitwise

47	4	_____ operator specify the collation and case sensitivity of searches and sorting operations.	Arithmetic	comparison	logical	sort	sort
48	4	If either argument is null or if both arguments are null the condition is Considered.	true	false	null	zero	null
49	4	____ operator converts a string to a binary string so that comparing and sorting data is case-sensitive.	Special	COLLATE	LIKE	BINARY	BINARY
50	4	____ operator specifies that a particular collation be used to compare and sort string data.	BINARY	COLLATE	LIKE	Special.	COLLATE
51	4	The ___ function identifies character set used for specified string.	Strstr ( )	COLLATION ( )	CONCATE ( )	CHARACTER SET ( )	CHARACTER SET ( )
52	4	The INSTR( ) takes ____ argument	One	Two	Three	Zero.	One
53	4	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	4	____ file is text file that contains one or more rows of exported data in a delimited format.	Infile	Dump file	Outfile	deletefile	Outfile
55	4	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	4	___ file is a text file that contains only one row that is not delimited.	Infile	Dump file	Outfile	Unique file	Dump file
57	4	____ statement at MYSQL command prompt is used to import delimited values directly from a text file.	MYSQL command	LOAD DATA	Insert.	delete	LOAD DATA

58	4	_____ 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	4	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	4	The ____ function converts a data that is retrieved as a string value.	strstr( )	time( )	strtotime ( )	strtime()	strtotime ( )

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

COIMBATORE - 641 021

PHP5/MYSQL [16CAP501]

UNIT V

PHP MySQL Introduction

What is MySQL?

MySQL is a database.

The data in MySQL is stored in database objects called tables.

A table is a collection of related data entries and it consists of columns and rows.

Databases are useful when storing information categorically. A company may have a database with the following tables: "Employees", "Products", "Customers" and "Orders".

Database Tables

A database most often contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"). Tables contain records (rows) with data.

Below is an example of a table called "Persons":

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

The table above contains three records (one for each person) and four columns (LastName, FirstName, Address, and City).

Queries

A query is a question or a request.

With MySQL, we can query a database for specific information and have a recordset returned.

Look at the following query:

SELECT LastName FROM Persons

The query above selects all the data in the "LastName" column from the "Persons" table, and will return a recordset like this:

LastName
Hansen
Svendson
Pettersen

PHP MySQL Connect to a Database:

Create a Connection to a MySQL Database

Before you can access data in a database, you must create a connection to the database.

In PHP, this is done with the mysql_connect() function.

Syntax

mysql_connect(servername,username,password);

Parameter	Description
servername	Optional. Specifies the server to connect to. Default value is "localhost:3306"
username	Optional. Specifies the username to log in with. Default value is the name of

	the user that owns the server process
password	Optional. Specifies the password to log in with. Default is ""

Example

In the following example we store the connection in a variable ($con) for later use in the script. The "die" part will be executed if the connection fails:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

// some code
?>
```

Closing a Connection

The connection will be closed automatically when the script ends. To close the connection before, use the mysql_close() function:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

// some code

mysql_close($con);
?>
```

Create a Database

The CREATE DATABASE statement is used to create a database in MySQL.

Syntax

```
CREATE DATABASE database_name
```

To learn more about SQL, please visit our SQL tutorial.

To get PHP to execute the statement above we must use the mysql_query() function. This function is used to send a query or command to a MySQL connection.

Example

The following example creates a database called "my_db":

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

if (mysql_query("CREATE DATABASE my_db",$con))
  {
  echo "Database created";
  }
else
  {
  echo "Error creating database: " . mysql_error();
  }

mysql_close($con);
?>
```

Create a Table

The CREATE TABLE statement is used to create a table in MySQL.

Syntax

```
CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
column_name3 data_type,
....
)
```

We must add the CREATE TABLE statement to the mysql_query() function to execute the command.

Example

The following example creates a table named "Persons", with three columns. The column names will be "FirstName", "LastName" and "Age":

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

// Create database
if (mysql_query("CREATE DATABASE my_db",$con))
  {
  echo "Database created";
  }
else
  {
  echo "Error creating database: " . mysql_error();
  }

// Create table
mysql_select_db("my_db", $con);
$sql = "CREATE TABLE Persons
(
FirstName varchar(15),
LastName varchar(15),
Age int
)";
```

```
// Execute query
mysql_query($sql,$con);

mysql_close($con);
?>
```

Primary Keys and Auto Increment Fields

Each table should have a primary key field.

A primary key is used to uniquely identify the rows in a table. Each primary key value must be unique within the table. Furthermore, the primary key field cannot be null because the database engine requires a value to locate the record.

The following example sets the personID field as the primary key field. The primary key field is often an ID number, and is often used with the AUTO_INCREMENT setting. AUTO_INCREMENT automatically increases the value of the field by 1 each time a new record is added. To ensure that the primary key field cannot be null, we must add the NOT NULL setting to the field.

Example

```
$sql = "CREATE TABLE Persons
(
personID int NOT NULL AUTO_INCREMENT,
PRIMARY KEY(personID),
FirstName varchar(15),
LastName varchar(15),
Age int
)";

mysql_query($sql,$con);
```

PHP MySQL Insert Into

Insert Data Into a Database Table

The INSERT INTO statement is used to add new records to a database table.

Syntax

It is possible to write the INSERT INTO statement in two forms.

The first form doesn't specify the column names where the data will be inserted, only their values:

```
INSERT INTO table_name
VALUES (value1, value2, value3,...)
```

The second form specifies both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

To get PHP to execute the statements above we must use the mysql_query() function. This function is used to send a query or command to a MySQL connection.

Example

In the previous chapter we created a table named "Persons", with three columns; "Firstname", "Lastname" and "Age". We will use the same table in this example. The following example adds two new records to the "Persons" table:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

mysql_query("INSERT INTO Persons (FirstName, LastName, Age)
VALUES ('Peter', 'Griffin', '35')");

mysql_query("INSERT INTO Persons (FirstName, LastName, Age)
VALUES ('Glenn', 'Quagmire', '33')");

mysql_close($con);
?>
```

Insert Data From a Form Into a Database

Now we will create an HTML form that can be used to add new records to the "Persons" table.

Here is the HTML form:

```
<html>
<body>

<form action="insert.php" method="post">
Firstname: <input type="text" name="firstname" />
Lastname: <input type="text" name="lastname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>

</body>
</html>
```

When a user clicks the submit button in the HTML form in the example above, the form data is sent to "insert.php".

The "insert.php" file connects to a database, and retrieves the values from the form with the PHP $_POST variables.

Then, the mysql_query() function executes the INSERT INTO statement, and a new record will be added to the "Persons" table.

Here is the "insert.php" page:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

$sql="INSERT INTO Persons (FirstName, LastName, Age)
```

```
VALUES
('$_POST[firstname]','$_POST[lastname]','$_POST[age]')";

if (!mysql_query($sql,$con))
  {
  die('Error: ' . mysql_error());
  }
echo "1 record added";

mysql_close($con)
?>
```

Select Data From a Database Table

The SELECT statement is used to select data from a database.

Syntax

```
SELECT column_name(s)
FROM table_name
```

Example 1

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

$result = mysql_query("SELECT * FROM Persons");

while($row = mysql_fetch_array($result))
  {
  echo $row['FirstName'] . " " . $row['LastName'];
  echo "<br />";
  }

mysql_close($con);
?>
```

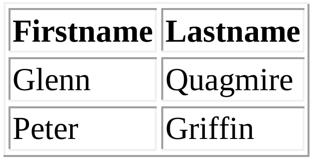
Example 2

Display the Result in an HTML Table

The following example selects the same data as the example above, but will display the data in an HTML table:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

$result = mysql_query("SELECT * FROM Persons");

echo "<table border='1'>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>";

while($row = mysql_fetch_array($result))
  {
  echo "<tr>";
  echo "<td>" . $row['FirstName'] . "</td>";
  echo "<td>" . $row['LastName'] . "</td>";
  echo "</tr>";
  }
echo "</table>";

mysql_close($con);
?>
```

The output of the code above will be:



Firstname	Lastname
Glenn	Quagmire
Peter	Griffin

Working with raster images:

graphics: vector objects and bitmap images.

Bitmap versus vector graphics - Graphic images that have been processed by a computer can usually be divided into two distinct categories. Such images are either bitmap files or vector graphics. If you work in prepress, you need a good comprehension on the advantages and disadvantages of both types of data. These pages tries to explain these differences.

Bitmap Vs. Vector Graphics 1 - Graphic images that have been processed by a computer can usually be divided into two distinct categories. Such images are either bitmap files or vector graphics. If you work in prepress, you need a good comprehension on the advantages and disadvantages of both types of data. These pages tries to explain these differences.

Bitmap Vs. Vector - To show how bitmap-based and vector-based images differ. The main difference is in the way computers analyze their content.

Bitmap Vs. Vector Graphics 2 - This page contains more information on vector graphics. A third page discusses the conversion from bitmap to vector data and back.

Bitmap vs. Vector Graphics - When dealing with graphic images, there are two types, either bitmap, or more correctly raster, and vector. Vector images refer to images that are made up of lines that are described mathematically. Raster images are made up of a grid of dots, or pixels, each pixel containing color information.

Bitmap Vs. Vector - This webpage explains the differences between Bitmap-based image files and Vector-based image files.

Bitmapped Vs. Vector Graphics - Find out the differences between these 2 formats.

Bitmap And Vector Graphics - There are two main types of graphics that should be understood before moving into Fireworks MX. Bitmap and Vector graphics are different in the way they are created, and treated within a graphics editing program. When you save a file for the web in a graphics editing program, however, the file is saved as a bitmap file regardless of what method was used to create the file.

Bitmapped vs. Vector-based and Bitmapped Images - There are two basic storage structures for grahics: bitmapped and vector-based. All images display on a computer screen as a grid of pixels of various colors. The image files that contain these images store that image data in one of two fundamentally different ways:

Captain Bitmap Vs. Vectorman - Art, no matter how you produce it, requires tools. As we approach the real millennium, the favorite tool for many artists is the computer. While hardware makes it possible to create digital graphics, software enables the artist to harness the computer's energy and create illustrations, photographs, and drawings.

Choosing a Graphics Format - A graphics file format is a specification for storing and organizing data in a file. MATLAB supports many different graphics file formats. Some are built into MATLAB and others are Ghostscript formats. File formats also differ in color support, graphics style (bitmap or vector), and bit depth.

Dell Tips : Raster vs Vector Graphics - Adobe® Illustrator® 10 is a vector-based graphics program, meaning that any graphic that you create entirely within Illustrator will be a vector graphic. Vector graphics are mathematically-defined lines and curves. Because vector images are defined mathematically, you can infinitely resize your graphic and it will remain crisp and clear at any resolution and at any magnification. Illustrator 10 also has new features, making it more compatible with raster-based programs like Adobe Photoshop® 7.0.

Design Concepts: Bitmapped vs. Vector Graphics - Bitmapped images in your logo design can look horrible when resized beyond the size that they are originally created in

Devnet and Bitmap Images Versus Vector Graphics - Bitmap images improve performance, because they are rendered already. The caveat is that they will often result in larger file sizes. For complex backgrounds, there's no way around using bitmap images, because animation on top of a complex vector background would be very slow. However, on a small device screen, a lot of that detail will be lost anyway.

Flash Animations and Interaction - Vector Graphics Versus Bitmap Graphics - Bitmapped graphics can be imported from various sources, ranging from digital photos to user-created graphics developed on drawing applications.

Lesson 5.1: Graphics Basics : Bitmap vs. Vector-Based Graphics - There are two main categories of graphic images: bitmap and vector.

Macromedia Fireworks Vector vs Raster (Bitmap) - Macromedia Fireworks is a primarily a vector based program. It is different from other graphics editing programs such as Adobe Photoshop, which is bitmap based. This article lists differences between bitmap and vector file types.

Raster vs. Vector - On a computer monitor, images are nothing more than variously colored pixels. Certain kinds of image-file formats record images literally in terms of the pixels to display. These are raster images, and you can edit them only by altering the pixels directly with a bitmap editor. Photoshop and Paint Shop Pro are two of the most popular bitmap editors.

Raster Images versus Vector Images - All electronic art images are divided into one of two core types, raster images (also know as 'bitmaps') and vector images. In a nutshell raster images are composed of connected dots and vectors are images composed of connected lines.

Raster Images vs. Vector Graphics - Computer graphics can be created as either raster or vector images. Raster graphics are bitmaps. A bitmap is a grid of individual pixels that collectively compose an image. Raster graphics render images as a collection of countless tiny squares. Each square, or pixel, is coded in a specific hue or shade. Individually, these pixels are worthless. Together, they’re worth a thousand words.

The Low-Down on Art Programs - A primer for working with vector- and raster-based graphics programs.

Two Kinds of Computer Graphics - There are two kinds of computer graphics - raster (composed of pixels) and vector (composed of paths). Raster images are more commonly called bitmap images. A bitmap image uses a grid of individual pixels where each pixel can be a different color or shade. Bitmaps are composed of pixels.

Vector & Bitmap Images: Which Is Better? - A graphic designer will come into contact with two kinds of computer image files. They may look the same but upon closer inspection, one finds that they are quite different in many ways.

Vector Vs. Bitmap - One of the main issues that concern Web Designers is the need to keep the file size of an image small which in turn relates to bandwidth. Most images on the Web are bitmaps, such as jpegs and gifs, this means that information about the image must be stored pixel by pixel.

Vector versus Bitmap & Fireworks - Vector graphics create digital images through a sequence of commands or mathematical statements that place points, lines, and shapes in a given two-dimensional or three-dimensional space.

Vector Vs. Bitmap - Programs like MS Paint and PhotoShop are both bitmap applications, treating the images you work with as a fixed-size resource made up of a fixed number of dots, or pixels. Once a line or curve or piece of text has been 'committed' to the canvas you can not go back and change it without undoing and starting over again.

Vector and Bitmap Images - Two Types of 2D Graphics - It's almost impossible to discuss graphics software without first establishing an understanding of the differences between the two major 2D graphic types: bitmap and vector images. This is an important lesson and often a tough one to grasp. If you work with graphics at all, it's bound to come up, so it's an important concept to understand. Let's start by talking about the more common type: bitmap images.

Vector Vs. Bitmap Graphics - Graphics fall into to two main categories: Vector graphics and Bitmap graphics ... The difference between these two types is what they're made up of. Vector graphics are made up of lines and curves. Bitmap graphics are made up of little squares called pixels.

Vector vs. Bitmap Graphics - an Introductory Guide for Clients and Designers - One of the most common misunderstandings among both clients, and unfortunately many designers, is the difference between Vector and Bitmap (also known as Raster) graphics.

Vector and Raster Images - There are two types of computer graphics - vector and raster. Vector images define curves and lines by mathematical formulas, enabling you to scale the image larger or smaller without taking a hit on image quality.

Vector vs. Raster Images - Word processors and spreadsheet or presentation applications, although suitable for creating files for office or Internet use, are not recommended for creating digital art for print. Microsoft Office applications are included in this group. In some cases, however, such files may be converted so as to enable use.

What are bitmap and vector graphics, and how are they different? - A bitmap (also called "raster") graphic is created from rows of different colored pixels that together form an image. In their simplest form, bitmaps have only two colors, with each pixel being either black or white. With increasing complexity, an image can include more colors; photograph-quality images may have millions

image create

imagecreate — Create a new palette based image

Description

resource **imagecreate** (int *$width* , int *$height*)

imagecreate() returns an image identifier representing a blank image of specified size.

We recommend the use of imagecreatetruecolor().

Example #1 Creating a new GD image stream and outputting an image.

```
<?php
header("Content-Type: image/png");
$im = @imagecreate(110, 20)
    or die("Cannot Initialize new GD image stream");
$background_color = imagecolorallocate($im, 0, 0, 0);
$text_color = imagecolorallocate($im, 233, 14, 91);
imagestring($im, 1, 5, 5,  "A Simple Text String", $text_color);
imagepng($im);
imagedestroy($im);
?>
```

Imagedestroy

Description

bool **imagedestroy** (resource *$image*)

imagedestroy() frees any memory associated with image *image*.

Example #1 Using imagedestroy() example

```
<?php
// create a 100 x 100 image
$im = imagecreatetruecolor(100, 100);

// alter or save the image

// frees image from memory
imagedestroy($im);
?>
```

Elements of array returned by gd_info()

Attribute	Meaning
GD Version	string value describing the installed *libgd* version.
Freetype Support	boolean value. **TRUE** if Freetype Support is installed.
Freetype Linkage	string value describing the way in which Freetype was linked. Expected values are: 'with freetype', 'with TTF library', and 'with unknown library'. This element will only be defined if *Freetype Support* evaluated to **TRUE**.
T1Lib Support	boolean value. **TRUE** if *T1Lib* support is included.
GIF Read Support	boolean value. **TRUE** if support for ***reading*** *GIF* images is included.
GIF Create Support	boolean value. **TRUE** if support for ***creating*** *GIF* images is included.
JPEG Support	boolean value. **TRUE** if *JPEG* support is included.
PNG Support	boolean value. **TRUE** if *PNG* support is included.
WBMP Support	boolean value. **TRUE** if *WBMP* support is included.
XBM Support	boolean value. **TRUE** if *XBM* support is included.

Example #1 Using gd_info()

```
<?php
var_dump(gd_info());
?>
```

The above example will output something similar to:

```
array(9) {
  ["GD Version"]=>
  string(24) "bundled (2.0 compatible)"
  ["FreeType Support"]=>
  bool(false)
  ["T1Lib Support"]=>
  bool(false)
  ["GIF Read Support"]=>
  bool(true)
  ["GIF Create Support"]=>
  bool(false)
  ["JPEG Support"]=>
  bool(false)
  ["PNG Support"]=>
  bool(true)
  ["WBMP Support"]=>
  bool(true)
  ["XBM Support"]=>
  bool(false)
}
```

Using Text in Images:

Imageft text

imagefttext — Write text to the image using fonts using FreeType 2

Description

array **imagefttext** (resource *$image* , float *$size* , float *$angle* , int *$x* , int *$y* , int *$color* , string *$fontfile* , string *$text* [, array *$extrainfo*])

Parameters

image

An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

size

The font size to use in points.

angle

The angle in degrees, with 0 degrees being left-to-right reading text. Higher values represent a counter-clockwise rotation. For example, a value of 90 would result in bottom-to-top reading text.

x

The coordinates given by *x* and *y* will define the basepoint of the first character (roughly the lower-left corner of the character). This is different from the imagestring(), where *x* and *y* define the upper-left corner of the first character. For example, "top left" is 0, 0.

y

The y-ordinate. This sets the position of the fonts baseline, not the very bottom of the character.

color

The index of the desired color for the text, see imagecolorexact().

fontfile

The path to the TrueType font you wish to use.

Depending on which version of the GD library PHP is using, ***when fontfile does not begin with a leading / then .ttf will be appended*** to the filename and the library will attempt to search for that filename along a library-defined font path.

When using versions of the GD library lower than 2.0.18, a *space* character, rather than a semicolon, was used as the 'path separator' for different font files. Unintentional use of this feature will result in the warning message: *Warning: Could not find/open font*. For these affected versions, the only solution is moving the font to a path which does not contain spaces.

In many cases where a font resides in the same directory as the script using it the following trick will alleviate any include problems.

```
<?php
// Set the enviroment variable for GD
putenv('GDFONTPATH=' . realpath('.'));

// Name the font to be used (note the lack of the .ttf extension)
$font = 'SomeFont';
?>
```

Text

Text to be inserted into image.

Extrainfo

Possible array indexes for *extrainfo*

Key	Type	Meaning

Possible array indexes for *extrainfo*

Key	Type	Meaning
linespacing	float	Defines drawing linespacing

PHP EMAIL

PHP allows you to send e-mails directly from a script.

The PHP mail() Function

The PHP mail() function is used to send emails from inside a script.

Syntax

```
mail(to,subject,message,headers,parameters)
```

Parameter	Description
to	Required. Specifies the receiver/receivers of the email
subject	Required. Specifies the subject of the email. Note: This parameter cannot contain any newline characters
message	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters
headers	Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n)
parameters	Optional. Specifies an additional parameter to the sendmail program

Note: For the mail functions to be available, PHP requires an installed and working email system. The program to be used is defined by the configuration settings in the php.ini file. Read more in our PHP Mail reference.

PHP Simple E-Mail

The simplest way to send an email with PHP is to send a text email.

In the example below we first declare the variables ($to, $subject, $message, $from, $headers), then we use the variables in the mail() function to send an e-mail:

```
<?php
$to = "someone@example.com";
$subject = "Test mail";
$message = "Hello! This is a simple email message.";
$from = "someonelse@example.com";
$headers = "From:" . $from;
mail($to,$subject,$message,$headers);
echo "Mail Sent.";
?>
```

PHP Mail Form

With PHP, you can create a feedback-form on your website. The example below sends a text message to a specified e-mail address:

```
<html>
<body>

<?php
if (isset($_REQUEST['email']))
//if "email" is filled out, send email
  {
  //send email
  $email = $_REQUEST['email'] ;
  $subject = $_REQUEST['subject'] ;
  $message = $_REQUEST['message'] ;
  mail("someone@example.com", $subject,
  $message, "From:" . $email);
  echo "Thank you for using our mail form";
  }
else
//if "email" is not filled out, display the form
  {
```

```
  echo "<form method='post' action='mailform.php'>
  Email: <input name='email' type='text'><br>
  Subject: <input name='subject' type='text'><br>
  Message:<br>
  <textarea name='message' rows='15' cols='40'>
  </textarea><br>
  <input type='submit'>
  </form>";
  }
?>

</body>
</html>
```

This is how the example above works:

- First, check if the email input field is filled out
- If it is not set (like when the page is first visited); output the HTML form
- If it is set (after the form is filled out); send the email from the form
- When submit is pressed after the form is filled out, the page reloads, sees that the email input is set, and sends the email

Note: This is the simplest way to send e-mail, but it is not secure. In the next chapter of this tutorial you can read more about vulnerabilities in e-mail scripts, and how to validate user input to make it more secure.

PHP E-mail Injections

First, look at the PHP code from the previous chapter:

```
<html>
<body>

<?php
if (isset($_REQUEST['email']))
//if "email" is filled out, send email
  {
  //send email
  $email = $_REQUEST['email'] ;
  $subject = $_REQUEST['subject'] ;
  $message = $_REQUEST['message'] ;
  mail("someone@example.com", "Subject: $subject",
  $message, "From: $email" );
  echo "Thank you for using our mail form";
  }
else
//if "email" is not filled out, display the form
```

```
  {
  echo "<form method='post' action='mailform.php'>
  Email: <input name='email' type='text'><br>
  Subject: <input name='subject' type='text'><br>
  Message:<br>
  <textarea name='message' rows='15' cols='40'>
  </textarea><br>
  <input type='submit'>
  </form>";
  }
?>

</body>
</html>
```

The problem with the code above is that unauthorized users can insert data into the mail headers via the input form.

What happens if the user adds the following text to the email input field in the form?

```
someone@example.com%0ACc:person2@example.com
%0ABcc:person3@example.com,person3@example.com,
anotherperson4@example.com,person5@example.com
%0ABTo:person6@example.com
```

The mail() function puts the text above into the mail headers as usual, and now the header has an extra Cc:, Bcc:, and To: field. When the user clicks the submit button, the e-mail will be sent to all of the addresses above!

PHP Stopping E-mail Injections

The best way to stop e-mail injections is to validate the input.

The code below is the same as in the previous chapter, but now we have added an input validator that checks the email field in the form:

```
<html>
<body>
<?php
function spamcheck($field)
  {
  //filter_var() sanitizes the e-mail
  //address using FILTER_SANITIZE_EMAIL
  $field=filter_var($field, FILTER_SANITIZE_EMAIL);
  //filter_var() validates the e-mail
  //address using FILTER_VALIDATE_EMAIL
  if(filter_var($field, FILTER_VALIDATE_EMAIL))
    {
    return TRUE;
    }
  else
    {
    return FALSE;
    }
  }

if (isset($_REQUEST['email']))
  {//if "email" is filled out, proceed

  //check if the email address is invalid
  $mailcheck = spamcheck($_REQUEST['email']);
  if ($mailcheck==FALSE)
    {
    echo "Invalid input";
    }
  else
    {//send email
    $email = $_REQUEST['email'] ;
    $subject = $_REQUEST['subject'] ;
    $message = $_REQUEST['message'] ;
    mail("someone@example.com", "Subject: $subject",
    $message, "From: $email" );
    echo "Thank you for using our mail form";
    }
  }
else
  {//if "email" is not filled out, display the form
  echo "<form method='post' action='mailform.php'>
  Email: <input name='email' type='text'><br>
  Subject: <input name='subject' type='text'><br>
  Message:<br>
  <textarea name='message' rows='15' cols='40'>
  </textarea><br>
  <input type='submit'>
  </form>";
  }
?>

</body>
</html>
```

In the code above we use PHP filters to validate input:

- The FILTER_SANITIZE_EMAIL filter removes all illegal e-mail characters from a string
- The FILTER_VALIDATE_EMAIL filter validates value as an e-mail address

PHP Error Handling

When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks.

This tutorial contains some of the most common error checking methods in PHP.

We will show different error handling methods:

- Simple "die()" statements
- Custom errors and error triggers
- Error reporting

Basic Error Handling: Using the die() function

The first example shows a simple script that opens a text file:

```
<?php
$file=fopen("welcome.txt","r");
?>
```

If the file does not exist you might get an error like this:

```
Warning: fopen(welcome.txt) [function.fopen]: failed to open stream:
No such file or directory in C:\webfolder\test.php on line 2
```

To prevent the user from getting an error message like the one above, we test whether the file exist before we try to access it:

```
<?php
if(!file_exists("welcome.txt"))
  {
  die("File not found");
  }
else
  {
  $file=fopen("welcome.txt","r");
  }
?>
```

Now if the file does not exist you get an error like this:

```
File not found
```

The code above is more efficient than the earlier code, because it uses a simple error handling mechanism to stop the script after the error.

However, simply stopping the script is not always the right way to go. Let's take a look at alternative PHP functions for handling errors.

Creating a Custom Error Handler

Creating a custom error handler is quite simple. We simply create a special function that can be called when an error occurs in PHP.

This function must be able to handle a minimum of two parameters (error level and error message) but can accept up to five parameters (optionally: file, line-number, and the error context):

Syntax

```
error_function(error_level,error_message,
error_file,error_line,error_context)
```

Parameter	Description
error_level	Required. Specifies the error report level for the user-defined error. Must be a value number. See table below for possible error report levels
error_message	Required. Specifies the error message for the user-defined error
error_file	Optional. Specifies the filename in which the error occurred
error_line	Optional. Specifies the line number in which the error occurred
error_context	Optional. Specifies an array containing every variable, and their values, in use when the error occurred

Error Report levels

These error report levels are the different types of error the user-defined error handler can be used for:

Value	Constant	Description
2	E_WARNING	Non-fatal run-time errors. Execution of the script is not halted
8	E_NOTICE	Run-time notices. The script found something that might be an error, but could also happen when running a script normally
256	E_USER_ERROR	Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function trigger_error()
512	E_USER_WARNING	Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function trigger_error()
1024	E_USER_NOTICE	User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function trigger_error()
4096	E_RECOVERABLE_ERROR	Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle (see also set_error_handler())
8191	E_ALL	All errors and warnings (E_STRICT became a part of E_ALL in PHP 5.4)

Now lets create a function to handle errors:

```
function customError($errno, $errstr)
  {
  echo "<b>Error:</b> [$errno] $errstr<br>";
  echo "Ending Script";
  die();
  }
```

The code above is a simple error handling function. When it is triggered, it gets the error level and an error message. It then outputs the error level and message and terminates the script.

Now that we have created an error handling function we need to decide when it should be triggered.

Set Error Handler

The default error handler for PHP is the built in error handler. We are going to make the function above the default error handler for the duration of the script.

It is possible to change the error handler to apply for only some errors, that way the script can handle different errors in different ways. However, in this example we are going to use our custom error handler for all errors:

```
set_error_handler("customError");
```

Since we want our custom function to handle all errors, the set_error_handler() only needed one parameter, a second parameter could be added to specify an error level.

Example

Testing the error handler by trying to output variable that does not exist:

```
<?php
//error handler function
function customError($errno, $errstr)
  {
  echo "<b>Error:</b> [$errno] $errstr";
  }

//set error handler
set_error_handler("customError");

//trigger error
echo($test);
?>
```

The output of the code above should be something like this:

```
Error: [8] Undefined variable: test
```

Trigger an Error

In a script where users can input data it is useful to trigger errors when an illegal input occurs. In PHP, this is done by the trigger_error() function.

Example

In this example an error occurs if the "test" variable is bigger than "1":

```
<?php
$test=2;
if ($test>1)
{
trigger_error("Value must be 1 or below");
}
?>
```

The output of the code above should be something like this:

```
Notice: Value must be 1 or below
in C:\webfolder\test.php on line 6
```

An error can be triggered anywhere you wish in a script, and by adding a second parameter, you can specify what error level is triggered.

Possible error types:

- E_USER_ERROR - Fatal user-generated run-time error. Errors that can not be recovered from. Execution of the script is halted
- E_USER_WARNING - Non-fatal user-generated run-time warning. Execution of the script is not halted
- E_USER_NOTICE - Default. User-generated run-time notice. The script found something that might be an error, but could also happen when running a script normally

Example

In this example an E_USER_WARNING occurs if the "test" variable is bigger than "1". If an E_USER_WARNING occurs we will use our custom error handler and end the script:

```
<?php
//error handler function
function customError($errno, $errstr)
  {
  echo "<b>Error:</b> [$errno] $errstr<br>";
  echo "Ending Script";
  die();
  }

//set error handler
set_error_handler("customError",E_USER_WARNING);
```

```
//trigger error
$test=2;
if ($test>1)
  {
  trigger_error("Value must be 1 or below",E_USER_WARNING);
  }
?>
```

The output of the code above should be something like this:

```
Error: [512] Value must be 1 or below
Ending Script
```

Now that we have learned to create our own errors and how to trigger them, lets take a look at error logging.

Error Logging

By default, PHP sends an error log to the server's logging system or a file, depending on how the error_log configuration is set in the php.ini file. By using the error_log() function you can send error logs to a specified file or a remote destination.

Sending error messages to yourself by e-mail can be a good way of getting notified of specific errors.

Send an Error Message by E-Mail

In the example below we will send an e-mail with an error message and end the script, if a specific error occurs:

```
<?php
//error handler function
function customError($errno, $errstr)
  {
  echo "<b>Error:</b> [$errno] $errstr<br>";
  echo "Webmaster has been notified";
  error_log("Error: [$errno] $errstr",1,
  "someone@example.com","From: webmaster@example.com");
  }

//set error handler
set_error_handler("customError",E_USER_WARNING);

//trigger error
$test=2;
```

```
if ($test>1)
  {
  trigger_error("Value must be 1 or below",E_USER_WARNING);
  }
?>
```

The output of the code above should be something like this:

```
Error: [512] Value must be 1 or below
Webmaster has been notified
```

And the mail received from the code above looks like this:

```
Error: [512] Value must be 1 or below
```

This should not be used with all errors. Regular errors should be logged on the server using the default PHP logging system.

PHP Exception Handling

Exceptions are used to change the normal flow of a script if a specified error occurs.

What is an Exception

With PHP 5 came a new object oriented way of dealing with errors.

Exception handling is used to change the normal flow of the code execution if a specified error (exceptional) condition occurs. This condition is called an exception.

This is what normally happens when an exception is triggered:

- The current code state is saved
- The code execution will switch to a predefined (custom) exception handler function
- Depending on the situation, the handler may then resume the execution from the saved code state, terminate the script execution or continue the script from a different location in the code

We will show different error handling methods:

- Basic use of Exceptions
- Creating a custom exception handler
- Multiple exceptions
- Re-throwing an exception
- Setting a top level exception handler

Note: Exceptions should only be used with error conditions, and should not be used to jump to another place in the code at a specified point.

Basic Use of Exceptions

When an exception is thrown, the code following it will not be executed, and PHP will try to find the matching "catch" block.

If an exception is not caught, a fatal error will be issued with an "Uncaught Exception" message.

Lets try to throw an exception without catching it:

```
<?php
//create function with an exception
function checkNum($number)
  {
  if($number>1)
    {
    throw new Exception("Value must be 1 or below");
    }
  return true;
  }

//trigger exception
checkNum(2);
?>
```

The code above will get an error like this:

```
Fatal error: Uncaught exception 'Exception'
with message 'Value must be 1 or below' in C:\webfolder\test.php:6
Stack trace: #0 C:\webfolder\test.php(12):
checkNum(28) #1 {main} thrown in C:\webfolder\test.php on line 6
```

Try, throw and catch

To avoid the error from the example above, we need to create the proper code to handle an exception.

Proper exception code should include:

1. Try - A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However if the exception triggers, an exception is "thrown"
2. Throw - This is how you trigger an exception. Each "throw" must have at least one "catch"

3. Catch - A "catch" block retrieves an exception and creates an object containing the exception information

Lets try to trigger an exception with valid code:

```
<?php
//create function with an exception
function checkNum($number)
  {
  if($number>1)
    {
    throw new Exception("Value must be 1 or below");
    }
  return true;
  }

//trigger exception in a "try" block
try
  {
  checkNum(2);
  //If the exception is thrown, this text will not be shown
  echo 'If you see this, the number is 1 or below';
  }

//catch exception
catch(Exception $e)
  {
  echo 'Message: ' .$e->getMessage();
  }
?>
```

The code above will get an error like this:

```
Message: Value must be 1 or below
```

Example explained:

The code above throws an exception and catches it:

1. The checkNum() function is created. It checks if a number is greater than 1. If it is, an exception is thrown
2. The checkNum() function is called in a "try" block
3. The exception within the checkNum() function is thrown
4. The "catch" block retrives the exception and creates an object ($e) containing the exception information
5. The error message from the exception is echoed by calling $e->getMessage() from the exception object

However, one way to get around the "every throw must have a catch" rule is to set a top level exception handler to handle errors that slip through.

Creating a Custom Exception Class

Creating a custom exception handler is quite simple. We simply create a special class with functions that can be called when an exception occurs in PHP. The class must be an extension of the exception class.

The custom exception class inherits the properties from PHP's exception class and you can add custom functions to it.

Lets create an exception class:

```
<?php
class customException extends Exception
  {
  public function errorMessage()
    {
    //error message
    $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
    .': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
    return $errorMsg;
    }
  }

$email = "someone@example...com";

try
  {
  //check if
  if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE)
    {
    //throw exception if email is not valid
    throw new customException($email);
    }
  }

catch (customException $e)
  {
  //display custom message
  echo $e->errorMessage();
  }
?>
```

The new class is a copy of the old exception class with an addition of the errorMessage() function. Since it is a copy of the old class, and it inherits the properties and methods from the old class, we can use the exception class methods like getLine() and getFile() and getMessage().

Example explained:

The code above throws an exception and catches it with a custom exception class:

1. The customException() class is created as an extension of the old exception class. This way it inherits all methods and properties from the old exception class
2. The errorMessage() function is created. This function returns an error message if an e-mail address is invalid
3. The $email variable is set to a string that is not a valid e-mail address
4. The "try" block is executed and an exception is thrown since the e-mail address is invalid
5. The "catch" block catches the exception and displays the error message

---

Multiple Exceptions

It is possible for a script to use multiple exceptions to check for multiple conditions.

It is possible to use several if..else blocks, a switch, or nest multiple exceptions. These exceptions can use different exception classes and return different error messages:

```
<?php
class customException extends Exception
{
public function errorMessage()
{
//error message
$errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
.': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
return $errorMsg;
}
}

$email = "someone@example.com";

try
  {
  //check if
  if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE)
    {
    //throw exception if email is not valid
    throw new customException($email);
    }
  //check for "example" in mail address
  if(strpos($email, "example") !== FALSE)
    {
    throw new Exception("$email is an example e-mail");
    }
  }

catch (customException $e)
  {
  echo $e->errorMessage();
  }

catch(Exception $e)
  {
  echo $e->getMessage();
  }
?>
```

Example explained:

The code above tests two conditions and throws an exception if any of the conditions are not met:

1. The customException() class is created as an extension of the old exception class. This way it inherits all methods and properties from the old exception class
2. The errorMessage() function is created. This function returns an error message if an e-mail address is invalid
3. The $email variable is set to a string that is a valid e-mail address, but contains the string "example"
4. The "try" block is executed and an exception is not thrown on the first condition
5. The second condition triggers an exception since the e-mail contains the string "example"
6. The "catch" block catches the exception and displays the correct error message

If the exception thrown were of the class customException and there were no customException catch, only the base exception catch, the exception would be handled there.

Re-throwing Exceptions

Sometimes, when an exception is thrown, you may wish to handle it differently than the standard way. It is possible to throw an exception a second time within a "catch" block.

A script should hide system errors from users. System errors may be important for the coder, but is of no interest to the user. To make things easier for the user you can re-throw the exception with a user friendly message:

```
<?php
class customException extends Exception
```

```
  {
  public function errorMessage()
    {
    //error message
    $errorMsg = $this->getMessage().' is not a valid E-Mail address.';
    return $errorMsg;
    }
  }

$email = "someone@example.com";

try
  {
  try
    {
    //check for "example" in mail address
    if(strpos($email, "example") !== FALSE)
      {
      //throw exception if email is not valid
      throw new Exception($email);
      }
    }
  catch(Exception $e)
    {
    //re-throw exception
    throw new customException($email);
    }
  }

catch (customException $e)
  {
  //display custom message
  echo $e->errorMessage();
  }
?>
```

Example explained:

The code above tests if the email-address contains the string "example" in it, if it does, the exception is re-thrown:

1. The customException() class is created as an extension of the old exception class. This way it inherits all methods and properties from the old exception class
2. The errorMessage() function is created. This function returns an error message if an e-mail address is invalid
3. The $email variable is set to a string that is a valid e-mail address, but contains the string "example"
4. The "try" block contains another "try" block to make it possible to re-throw the exception

5. The exception is triggered since the e-mail contains the string "example"
6. The "catch" block catches the exception and re-throws a "customException"
7. The "customException" is caught and displays an error message

If the exception is not caught in its current "try" block, it will search for a catch block on "higher levels".

Set a Top Level Exception Handler

The set_exception_handler() function sets a user-defined function to handle all uncaught exceptions.

```
<?php
function myException($exception)
{
echo "<b>Exception:</b> " , $exception->getMessage();
}

set_exception_handler('myException');

throw new Exception('Uncaught Exception occurred');
?>
```

The output of the code above should be something like this:

```
Exception: Uncaught Exception occurred
```

In the code above there was no "catch" block. Instead, the top level exception handler triggered. This function should be used to catch uncaught exceptions.

Rules for exceptions

- Code may be surrounded in a try block, to help catch potential exceptions
- Each try block or "throw" must have at least one corresponding catch block
- Multiple catch blocks can be used to catch different classes of exceptions
- Exceptions can be thrown (or re-thrown) in a catch block within a try block

A simple rule: If you throw something, you have to catch it.

PHP Filter

PHP filters are used to validate and filter data coming from insecure sources, like user input.

What is a PHP Filter?

A PHP filter is used to validate and filter data coming from insecure sources.

To test, validate and filter user input or custom data is an important part of any web application.

The PHP filter extension is designed to make data filtering easier and quicker.

Why use a Filter?

Almost all web applications depend on external input. Usually this comes from a user or another application (like a web service). By using filters you can be sure your application gets the correct input type.

You should always filter all external data!

Input filtering is one of the most important application security issues.

What is external data?

- Input data from a form
- Cookies
- Web services data
- Server variables
- Database query results

Functions and Filters

To filter a variable, use one of the following filter functions:

- filter_var() - Filters a single variable with a specified filter
- filter_var_array() - Filter several variables with the same or different filters
- filter_input - Get one input variable and filter it
- filter_input_array - Get several input variables and filter them with the same or different filters

In the example below, we validate an integer using the filter_var() function:

```
<?php
$int = 123;

if(!filter_var($int, FILTER_VALIDATE_INT))
  {
  echo("Integer is not valid");
  }
else
  {
  echo("Integer is valid");
  }
?>
```

The code above uses the "FILTER_VALIDATE_INT" filter to filter the variable. Since the integer is valid, the output of the code above will be: "Integer is valid".

If we try with a variable that is not an integer (like "123abc"), the output will be: "Integer is not valid".

For a complete list of functions and filters, visit our PHP Filter Reference.

Validating and Sanitizing

There are two kinds of filters:

Validating filters:

- Are used to validate user input
- Strict format rules (like URL or E-Mail validating)
- Returns the expected type on success or FALSE on failure

Sanitizing filters:

- Are used to allow or disallow specified characters in a string
- No data format rules
- Always return the string

Options and Flags

Options and flags are used to add additional filtering options to the specified filters.

Different filters have different options and flags.

In the example below, we validate an integer using the filter_var() and the "min_range" and "max_range" options:

```
<?php
$var=300;

$int_options = array(
"options"=>array
  (
  "min_range"=>0,
  "max_range"=>256
  )
);

if(!filter_var($var, FILTER_VALIDATE_INT, $int_options))
  {
  echo("Integer is not valid");
  }
else
  {
  echo("Integer is valid");
  }
?>
```

Like the code above, options must be put in an associative array with the name "options". If a flag is used it does not need to be in an array.

Since the integer is "300" it is not in the specified range, and the output of the code above will be: "Integer is not valid".

For a complete list of functions and filters, visit our PHP Filter Reference. Check each filter to see what options and flags are available.

Validate Input

Let's try validating input from a form.

The first thing we need to do is to confirm that the input data we are looking for exists.

Then we filter the input data using the filter_input() function.

In the example below, the input variable "email" is sent to the PHP page:

```
<?php
if(!filter_has_var(INPUT_GET, "email"))
  {
  echo("Input type does not exist");
  }
else
  {
  if (!filter_input(INPUT_GET, "email", FILTER_VALIDATE_EMAIL))
    {
    echo "E-Mail is not valid";
    }
  else
    {
    echo "E-Mail is valid";
    }
  }
?>
```

Example Explained

The example above has an input (email) sent to it using the "GET" method:

1. Check if an "email" input variable of the "GET" type exist
2. If the input variable exists, check if it is a valid e-mail address

---

Sanitize Input

Let’s try cleaning up an URL sent from a form.

First we confirm that the input data we are looking for exists.

Then we sanitize the input data using the filter_input() function.

In the example below, the input variable "url" is sent to the PHP page:

```
<?php
if(!filter_has_var(INPUT_POST, "url"))
  {
  echo("Input type does not exist");
  }
else
  {
  $url = filter_input(INPUT_POST,
  "url", FILTER_SANITIZE_URL);
  }
?>
```

Example Explained

The example above has an input (url) sent to it using the "POST" method:

1. Check if the "url" input of the "POST" type exists
2. If the input variable exists, sanitize (take away invalid characters) and store it in the $url variable

If the input variable is a string like this "http://www.W3ååSchøøools.com/", the $url variable after the sanitizing will look like this:

```
http://www.W3Schools.com/
```

Filter Multiple Inputs

A form almost always consist of more than one input field. To avoid calling the filter_var or filter_input functions over and over, we can use the filter_var_array or the filter_input_array functions.

In this example we use the filter_input_array() function to filter three GET variables. The received GET variables is a name, an age and an e-mail address:

```
<?php
$filters = array
  (
  "name" => array
    (
    "filter"=>FILTER_SANITIZE_STRING
    ),
  "age" => array
    (
    "filter"=>FILTER_VALIDATE_INT,
    "options"=>array
      (
      "min_range"=>1,
      "max_range"=>120
      )
    ),
  "email"=> FILTER_VALIDATE_EMAIL
  );

$result = filter_input_array(INPUT_GET, $filters);

if (!$result["age"])
  {
```

```
  echo("Age must be a number between 1 and 120.<br>");
  }
elseif(!$result["email"])
  {
  echo("E-Mail is not valid.<br>");
  }
else
  {
  echo("User input is valid");
  }
?>
```

Example Explained

The example above has three inputs (name, age and email) sent to it using the "GET" method:

1. Set an array containing the name of input variables and the filters used on the specified input variables
2. Call the filter_input_array() function with the GET input variables and the array we just set
3. Check the "age" and "email" variables in the $result variable for invalid inputs. (If any of the input variables are invalid, that input variable will be FALSE after the filter_input_array() function)

The second parameter of the filter_input_array() function can be an array or a single filter ID.

If the parameter is a single filter ID all values in the input array are filtered by the specified filter.

If the parameter is an array it must follow these rules:

- Must be an associative array containing an input variable as an array key (like the "age" input variable)
- The array value must be a filter ID or an array specifying the filter, flags and options

Using Filter Callback

It is possible to call a user defined function and use it as a filter using the FILTER_CALLBACK filter. This way, we have full control of the data filtering.

You can create your own user defined function or use an existing PHP function

The function you wish to use to filter is specified the same way as an option is specified. In an associative array with the name "options"

In the example below, we use a user created function to convert all "_" to whitespaces:

```
<?php
function convertSpace($string)
{
```

```
return str_replace("_", " ", $string);
}
$string = "Peter_is_a_great_guy!";

echo filter_var($string, FILTER_CALLBACK,
array("options"=>"convertSpace"));
?>
```

The result from the code above should look like this:

```
Peter is a great guy!
```

Example Explained

The example above converts all "_" to whitespaces:

1. Create a function to replace "_" to whitespaces
2. Call the filter_var() function with the FILTER_CALLBACK filter and an array containing our function

UNIT V

POSSIBLE QUESTIONS

Part - B (Each Question carries 6 Marks)

1. Write short notes on Using True Type Fonts Using Text Images.
2. How to adding standard text using text images? Explain.
3. Explain briefly about creating a user manager.
4. Describe in detail about Structure of an Email message
5. How will you send an Email with php?
6. Write detailed notes on Using True Type Fonts Using Text Images.
7. Design an application to insert and update records in tables using mysql.
8. Explain manipulating with Raster Images
9. Explain briefly about basic drawing functions
10. Explain the steps in connecting to mysql from a php applications

Part - C (Each Question carries 10 Marks)

1. Design an online loan application form using php
2. Design a program to creating a new GD image stream and outputting an image.
3. Design an application to insert and update records in tables using mysql.
4. Explain briefly about Creating a User Registration Script

KARPAGAM ACADEMY OF HIGHER EDUCATION

DEPARTMENT OF COMPUTER APPLICATIONS

Subject Name PHP5/MYSQL

Class: III MCA

SUBCODE: 16CAP501

Sem:V

S.N	Uni	Question	Choice1	Choice2	Choice3	Choice4	Answer Key
unit	V	**UNIT V**					
1	5	The function used to create a connection to a MYSQL server.	MY SQL__connection()	MYSQL__server()	MY SQL__create()	MYSQL_link()	MY SQL__connection()
2	5	______________ 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	5	______________PHP's equivalent to MYSQL's SHOWDATABASES command.	MYSQL__show()	MYSQL__list__DBs	MYSQL__show__DBs	MYSQL__POST()	MYSQL__list__DBs
4	5	______________ function returns the error number.	MYSQL__error()	SET__error()	POST__error()	GET__error()	MYSQL__error()
5	5	______________ function returns the error text from the previous MYSQL operation.	POST__prev()	POST__prev(error)	GET__(error)	MYSQL__error()	MYSQL__error()
6	5	______________ provides a couple of functions that catch server errors and determine what when wrong.	MYSQL__catch()	catch(error)	PHP	couple__catch__error()	PHP
7	5	_______________ function call to set the level of error reporting.	SET(error)	call__SET(error)	list__SET()	Error__reporting()	Error__reporting()

8	5	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	5	_______________ works only with character and varchar type fields.	UNARY	BINARY	TERNARY	null	BINARY
10	5	Fields cannot take a null value.	NULL	NULL VALUE	NOT NULL	binary value	NOT NULL
11	5	Fields that are declared NULL will store a specified default value whenever a ______________ value is given.	NOT NULL	NULL	ZERO	ONE	NULL
12	5	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	5	______________ only works unique,integer type fields.	AUTO__DECREMENT	INTEGER	AUTO__INCREMENT	INT__FIELD.	AUTO__INCREMENT
14	5	There can be only one ______________ field per table.	AUTO__DECREMENT	INTEGER	INT__FIELD	AUTO__INCREMENT.	AUTO__INCREMENT .
15	5	______________ are synonymous keywords that specify fields to be used as indexes.	LITERALS	IDENTIFIERS	KEY	TOKENS.	KEY
16	5	______________ used as an index of unique values.	Primary key	Secondary Key	Foreign Key	none.	Primary key
17	5	To use a fields as an primary keys it must be declared as _________________.	NULL	NULL VALUE	NOT NULL	ZERO.	NOT NULL

18	5	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	5	_____________ is a requirement for auto increment fields.	Varying	AUTO__INCREMENT	INT__FIELDS	Uniqueness.	Uniqueness.
20	5	A TIME STAMP type field can always takes a ____________ value.	ZERO	NOT NULL	Null	none.	Null
21	5	The ___________ servers as the key to define a relationship between the two tables.	Table ID	User ID	Server ID	Admin ID.	User ID
22	5	The ___________ has the same user ID fields as the user table.	User Table	User ID	Server ID	Admin ID.	User Table
23	5	__________ 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	5	If the given query is successfully executed _____________ returns a non—zero value pointing to the required resulting set,or falls upon the error.	Query__String()	MSQL__dbs()	MYSQL__query()	none.	MYSQL__query()
25	5	__________ 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__db() and MYSQL__delete__db()	MYSQL__create__db() and MYSQL__drop__db()	MYSQL__create__db() and MYSQL__drop__db()
26	5	Subsequent calls to create the user and access __________ lock tables.	MYSQL__query()	Query__String()	MSQL__dbs()	MYSQL__locktables( ).	MYSQL__query()

27	5	__________ returns a single row of data from a result set	MYSQL__drop__db()	MYSQL__fetch__row()	MYSQL__query()	MYSQL__row()	MYSQL__fetch__row()
28	5	Variable _________ stores the returned result identifier.	$ result	$identifiers	$ literals	$ tokens.	$ result
29	5	The two functions of PHP that used to fetch data___________.	MYSQL__create__db() and MYSQL__delete__db	MYSQL__create__db() and MYSQL__drop__db()	MYSQL__fetch__array() and MYSQL__fetch__object()	MYSQL__create__db() and MYSQL__drop__db().	MYSQL__fetch__array() and MYSQL__fetch__object()
30	5	_________ returns a simple associative array from a result set.	$ result	MYSQL__fetch__object()	MYSQL__create__db()	MYSQL__fetch__array().	MYSQL__fetch__array().
31	5	_________ returns a single object from a result set.	MYSQL__create__db()	MYSQL__fetch__array()	MYSQL__fetch__object()	none.	MYSQL__fetch__object()
32	5	_________ returns the value of the specified field in a specified row.	MYSQL__field()	MYSQL__result()	MYSQL__row()	MYSQL__query()	MYSQL__result()
33	5	_________ 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	5	_________ is used to jump straight to a specific row of data.	MYSQL__row()	$fetched__row()	MYSQL__data__seek()	MYSQL__row__object()	MYSQL__data__seek()
35	5	The _________ function returns the current time of the system on which the server is running.	SHOW()	NOW()	current__time()	SERVER(time).	NOW()
36	5	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	5	A __________ clause is used to selectively retrieve rows of data according to specified condition	DISTICTION	FROM	ORDER BY	WHERE.	WHERE.
38	5	_________ clause always comes at the end of the query.	ORDER BY	LIMIT	WHERE	SORT.	LIMIT
39	5	The _________ operator enables us to use values that may or may not be null.	ZERO	NOT NULL	NULL__safe comparision	all the above.	NULL__safe comparision
40	5	An __________ clause sorts retrieved values in ascending order by default.	ORDER BY	LIMIT	WHERE	SORT.	ORDER BY
41	5	To sort the values in descending order, use the __________ keyword.	LIMIT	WHERE	SORT.	DESC.	DESC.
42	5	Values in __________ type field are case sensitive.	NUMERALS	INTEGERS	TEXT	FLOAT	TEXT
43	5	_________ supports the largest number in the given field	LARGE()	open__new()	MAX(n)	LARGE(n)	MAX()
44	5	_________ supports the number of rows returned.	SUM	MAX(row)	MIN(row)	COUNT().	COUNT().
45	5	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	5	___________ counts every row in the result set.	COUNT()	ROW()	CONT(*)	COUNT(row).	CONT(*)
47	5	__________ function starts an HTML page and defines the java script function.	HTML__start()	HTML__header()	BODY	HEAD	HTML__header()
48	5	__________ 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	5	__________ function ends an HTML page.	HTML__footer()	HTML__end()	BODY	HTML.	HTML__footer()
50	5	__________ function reports errors using the java script alert() method.	alert()	MYSQL__ERROR()	The error__message ()	mysql__error( ).	The error__message()
51	5	__________ script makes use of the include file common__db.inc.	db__connect()	cmn__file()	db.inc()	User viewer.PHP.	User viewer.PHP.
52	5	__________ function displays a list of listed users along with navigation link and a record viewer link that calls the view__record().	Link__record()	List__record()	cur__page()	records__per__page()	List__record()
53	5	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	5	The MYSQL__num__rows() returns ________ when no record is found in the access__log table.	num(rows)	Zero	access__log()	two	Zero
55	5	To get formatted date string _________ is used	get(date)	SubStr(date)	PHPs SubStr()	none	PHPs SubStr()
56	5	If we want to throw away an existing record from a table __________ command is used.	REMOVE	DROP	THROW	DELETE.	DELETE.
57	5	_________ takes three arguments and return a result pointer that refers to a list of all fields in the specified table of specified database.	MYSQL__list__fields	access__log()	MYSQL__field__flag()	MYSQL__fetch__object()	MYSQL__list__fields

58	5	The _____________ variable holds a pointer to a list of fields in the user table.	$identifiers	$ literals	$ tokens	$result.	$result.
59	5	___________ 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__fet ch__object()	MYSQL__field__flag()
60	5	___________ 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()

Reg. No.....................................

[13CAP502]

KARPAGAM UNIVERSITY

Karpagam Academy of Higher Education
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021
(For the candidates admitted from 2013 onwards)

MCA DEGREE EXAMINATION, NOVEMBER 2015

Fifth Semester

COMPUTER APPLICATIONS

PHP5/MySQL

Time: 3 hours Maximum : 60 marks

PART – A (20 x 1 = 20 Marks) (30 Minutes)
(Question Nos. 1 to 20 Online Examinations)

PART B (5 x 8 = 40 Marks) (2 ½ Hours)
Answer ALL the Questions

21. (a) Design an online job application form

(Or)

(b) Write a program to demonstrate working with numbers

22. (a) Explain string validation and regular expression

(Or)

(b) Explain scope of variables with illustrations

23. (a) Explain working with files

(Or)

(b) Explain inheritance

24. (a) How to create indexes in table? Explain briefly about its types

(Or)

(b) Explain use of mysql_query() function with sample program

25. (a) Explain manipulating with Raster Images

(Or)

(b) Design an application to insert and update records in tables using mysql.

----

nature of Dean

Reg. No.................................

[12CAP502]

KARPAGAM UNIVERSITY

(Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021
(For the candidates admitted from 2012 onwards)

MCA DEGREE EXAMINATION, NOVEMBER 2014

Fifth Semester

COMPUTER APPLICATIONS

PHP5/MySQL

Time: 3 hours Maximum : 100 marks

PART – A (15 x 2 = 30 Marks)

Answer ALL the Questions

1. Define Variables. April'14
2. What is Operator?
3. Define Array.
4. What is testing?
5. What is Runtime Errors? April'14
6. What is handling error?
7. Define file.
8. How to Create a Class?
9. Define Encapsulation.
10. List out the types of Inheritance.
11. What is String function?
12. Define Encryption.
13. What is Query?
14. Define Raster Images.
15. What is MYSQL and PHP application?

PART B (5 X 14= 70 Marks)

Answer ALL the Questions

16. a) Discuss about Branching statement.

Or

b) Write a short note on Recursion.

Nov'12

17. a) Describe about Overriding Method. Apr'14

Or

b) Explain Numeric Function with example.

18. a) Write a short about using Text in Image. '14

Or

b) Explain Switch Statement with example.

19. a) Describe the Handling Error.

Or

b) Explain Interface with example.

20. **Compulsory : -**

How to Deleting and Retrieving data from a MYSQL database.

----

tinuous Internal Assessm

Reg. No.................................

[11CAP502]

KARPAGAM UNIVERSITY

(Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021
(For the candidates admitted from 2011 onwards)

MCA DEGREE EXAMINATION, APRIL 2014

Fifth Semester

COMPUTER APPLICATIONS

PHP5 / MYSQL

Time: 3 hours Maximum : 100 marks

PART – A (15 x 2 = 30 Marks)

Answer ALL the Questions

1. Explain the use of variables in PHP Nov'14
2. Describe various data types in PHP. Nov'13
3. Define testing and debugging?
4. Define runtime errors and logical errors. Nov'14
5. Explain about working with directories?
6. How to initialize an object explain with an example?
7. Explain briefly about String functions and its types
8. Explain briefly about Copying Data in to a table
9. Explain about Connecting to a MYSQL database
10. Write short notes on Using True Type Fonts Using Text Images
11. What do you mean by recursive resolution? Nov'13
12. Write a brief note on inverse domain.
13. What is meant by remote login?
14. What is the role of SNMP?
15. Describe FTP.

PART B (5 X 14= 70 Marks)

Answer ALL the Questions

16. (a) What is a session? Explain native sessions with example. Nov'12

Or

(b) Develop a php application for uploading online resume

17. (a) Explain form validation with sample program

Or

(b) Generate a php script to check whether a given number is prime or not Nov'12

18. (a) Write a program for overriding methods? Nov'14

Or

(b) Explain working with directories Nov'13

19. (a) Explain briefly about full joins, outer joins, natural joins Nov'13

Or

(b) Explain locking nontransactional tables

20. (a) Brief about the steps in connecting to mysql from a php applications. Nov'12

Or

(b) Explain briefly about basic drawing functions Nov'12

Reg. No.................................

[11CAP502]

KARPAGAM UNIVERSITY

(Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021
(For the candidates admitted from 2011 onwards)

MCA DEGREE EXAMINATION, NOVEMBER 2013

Fifth Semester

COMPUTER APPLICATIONS

PHP5 / MYSQL

Time: 3 hours Maximum : 100 marks

PART – A (15 x 2 = 30 Marks)

Answer ALL the Questions

1. Define settype function in PHP.
2. Define arrays in PHP with an example.
3. What is the use of HTTP protocol?
4. Define Debugging. April'14
5. What do you mean by error handling?
6. Define Recursion. April'14
7. Define the term object.
8. What do you mean by overriding?
9. List files permissions.
10. List string types in MY-SQL.
11. Give an example for LIKE operator in SELECT statement.
12. List trimming and padding functions in MY-SQL.
13. Write function to connect PHP and MY-SQL.
14. Write a function for new image creation.
15. Define E-mail.

PART B (5 X 14= 70 Marks)

Answer ALL the Questions

16. a. Discuss PHP operators with suitable examples. Nov'12

Or

b. How PHP code works? Discuss

17. a. How error messages are identified and handled? Discuss

Or

b. How user defined functions are written in PHP.

18. a. Discuss directory functions. April'14

Or

b. Discuss the feature of inheritance with suitable examples.

19. a. Discuss DML statements in MY-SQL with suitable examples.

Or

b. Explain various joins in SQL with examples. April'14 / Nov'12

20. a. How data insertion is possible in PHP with MY-SQL.

Or

b. Explain functions in PHP for to design fill areas.

Karpagam Academy Of Higher Education
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 64 021
MCA Degree Examination
(For the candidates admitted from 2016 onwards)
Fifth Semester
First Internal Exam Aug 2018
PHP5 / MYSQL [16CAP501]
Answer Key

Part - A (20 X 1 = 20 Marks)
(Answer all the Questions)

1) PHP stands for __________________
a) Program Holding Process b) **Hypertext Preprocessor**
c) Personal Hypertext Programming d) Hypertext Processor

2) The part that interprets and executes PHP code
a) **zend engine** b) search engine c) web browser d) interpreter

3) PHP is used for building __________________websites
a) static b) **dynamic** c) speed d) personal

4) PHP program are run on________________
a) **web browser** b) interpreter c) web server d) compiler

5) PHP is devised by____________
a) Tim Berners- Lee b) **Rasmus Lerdorf**
c) Robert Caillau d) Richard Fairly

6) PHP programs run via_________
a) Web browser b) Interpreter c) **Web server** d) Web client

7) HTML stands for_______________
a) Home Tool Management Logic b) Hyper Text Managing Language
c) **Hyper Text Markup Language** d) Hyper Text Mainframe Language

8) PHP was devised in the year
a) 1996 b) 1990 c) 1986 d)**1994**

9) CLI stands for________
a) Command Line Interpreter b) Common Line Integrator
c) Common Line Interface d) **Command Line Interface**

10) MIME stands for_________
a) Multipurpose Internal Mail Extension b) Multiple Interface Multiple Extension
c) **Multiple Interface and Mail Extension** d)Multipurpose Internet Mail Extension

11) RPM stands for __________
a) Redhat Protocol Manager b) Removable Program Memory
c) Redhat Package Manager d) **Removable Protocol Manager**

12) __________function is used to send a string value to the browser.
a) **Echo** b) cookie c) cout d) session

13) IIS stands for ______________
a) Information Integrated Service b) Internet Information Standard
c) **Internet Information Server** d) Internal Information Service

14) _________ are special characters that indicates where the data starts and stop.
a) Modifiers b) Limiters c) Identifiers d) **Delimiters**

15) In PHP code statements end with ________
a) Colon b) **Semicolon** c) Comma d) Star operator

16) ___________ are enclosed in curly braces.

a) Functions b) Data c) **Code blocks** d) Structures

17) _______file is parsed when PHP is first loaded and executed.

a) **php.ini** b) php.exe c) php.awt d) php.init

18) ________are programmatic capabilities that add to or enhance PHP's built-in capabilities.

a) PHP standard b) PHP extension c) PHP loader d) **PHP compiler**

19) _____ is the first step of debugging.

a) compiling b) **testing** c) analyzing d) processing

20) _____ occurs when your program produces an incorrect response or answer.

a) **logic error** b) syntax error

c) fatal error d)semantic error

Part - B (3 X 2 =6 Marks)

21. List Out Unique Features of PHP

- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Performance Scripts written in PHP execute faster than those written in other scripting languages, with numerous independent benchmarks putting the language ahead of competing alternatives like JSP, ASP.NET, and Perl. The PHP 5.0 engine was completely redesigned with an optimized memory manager to improve performance, and is noticeably faster than previous versions. In addition, third-party accelerators are available to further improve performance and response time.

22. **Write a simple program to Calculate Factorial of a given number**

```
<?php

$num = 4;
$factorial = 1;
for ($x=$num; $x>=1; $x--)
{
  $factorial = $factorial * $x;
}
echo "Factorial of $num is $factorial";?>
```

Output

```
Factorial of 4 is 24
```

23. **Write a program to read a file line by line**

The fgets() function returns a line from an open file.

The fgets() function stops returning on a new line, at the specified length, or at EOF, whichever comes first.

This function returns FALSE on failure.

Syntax

```
fgets(file,length)
```

EXAMPLE :

```
<?php
$file = fopen("test.txt","r");
echo fgets($file);
fclose($file);
?>
```

The output of the code above will be:

```
Hello, this is a test file.
```

PART-C

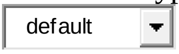
24. a) What are Cookies? Explain how to set and retrieve cookies

```
<?php

if($_POST[tsel])

{

setcookie("font[type]",$_POST[tsel], time()+3600);

}

if($_POST[ssel])

{

setcookie("font[size]",$_POST[ssel], time()+3600);

}
```

```
$type= array("arial","helvetica","sans-serif","courier");
$size= array("1","2","3","4","5","6","7");
echo "<html><head><title>Cookie Test</title></head><body><div align='center'>";
echo"<form method='POST'>";
echo"What font type would u like to use?";
echo"<select name='tsel'>";
echo"<option selected value=' '>default</option>";
foreach($type as $var)
{
echo"<option>$var</option>";
}
echo"</select><br><br>";
echo"What font size would u like to use?";
echo"<select name='ssel'>";
echo"<option selected value=' '>default</option>";
foreach($size as $var)
{
echo"<option>$var</option>";
}
echo"</select><br><br>";
echo"<input type='submit' value='Get Cookies'>";
echo"</form>";
echo"<b> Your Cookies say : </b><br>";
echo" <font ";
if($_COOKIE[font][type])
{
$cookie_font_type =$_COOKIE[font][type];
echo"face='$cookie_font_type'";
}
if($_COOKIE[font][size])
```

```
{
$cookie_font_size=$_COOKIE[font][size];
echo"size='$cookie_font_size'";
}
echo" >";
echo"\$font[type]= $cookie_font_type<br>";
echo"\$font[size]= $cookie_font_size<br>";
echo"</font><br>";
echo"<b>Ur Form Variables say :</b><br>";
echo" <font ";
if($_POST[tsel])
{
$post_type_sel =$_POST[tsel];
echo"face='$_post_type_sel'";
}
if($_POST[ssel])
{
$post_size_sel =$_POST[ssel];
echo"size='$_post_size_sel'";
}
echo" >";
echo"\$tsel=$post_type_sel<br>";
echo"\$ssel=$post_size_sel<br>";
echo"</font>";
echo"</div></body></html>";
```

OUTPUT:

"; echo"What font type would u like to use?"; echo""; foreach($type as $var) { echo""; } echo"
default

"; echo"What font size would u like to use?"; echo""; foreach($size as $var) { echo""; } echo"
default

```
"; echo"Get Cookies"; echo"
"; echo" Your Cookies say :
"; echo" "; echo"\$font[type]= $cookie_font_type
"; echo"\$font[size]= $cookie_font_size
"; echo"
"; echo"Ur Form Variables say :
"; echo" "; echo"\$tsel=$post_type_sel
"; echo"\$ssel=$post_size_sel
"; echo""; echo"
";
```

24.b) Write a program for Online Shopping Application

```
<html>

<head></head>

<body bgcolor=white>

<p align="center"><b><front size="7" face="Franklin Gothic Medium">

<img src="warning.gif' width "40" height "40"> GET IT
SHOPPING</font></b></p>

<p align="center">&nbsp;</p>

<p align="center"><b><font face="Franklin Gothic Medium" size="7">>Welcome to

</font><font face "Franklin Gothic Medium" size "6" color "#008000">

GET IT SHOPPING..</font></b></p>

<table border="0">

<tr><td width "50%">

<font color="#400000" size="5" face="Arial Black".

SHOP LATEST MODEL PORTABLES AND LAPTOPS</font><p>

   <a href="index.html">LAPTOPS</a><BR>

   <a href="lapindex.html">LAPTOPS</a></td>

     <td width="50%>

<img src="img_10602_samsung-f210-00_450*360.jpg" width="450" height="338">

</td> </tr> </table></body> </html>

Lapindex.html

<html>
```

```
<head></head>
<body bgcolor=white>
<FORM METHOD="POST" ACTION="select.php">
<p align "center"><b><font size "7" face "Franklin Gothic Medium">
<img border="0" src="warning.gif" width="40" height="40">GET IT
LAPTOPS</font></b></p>
<p align "left">&nbsp;</p>
<p align="left"><b><font face="Franklin Gothic Medium" size="4"
color="#FFFFFF">
LATEST MODELS LISTING: </font></b></p>
<p align="left"><b><font face="Franklin Gothic Medium" size="4"
color="FFFFF">&nbsp;</font></b></p>
<table border "0" cellpadding "0" cellspacing "0" style "border collapse: collapse"
bordcolor="#111111" width="100%" id="AutoNumber1"><tr>
<td width="50%">
<p align="center">
<img border="0" src="laptop16.jpg" width="40" height="40"><input type="submit"
name="rad3" value="GET IT LC166"</td>
<<td width="50%">MODEL NO: GET IT &nbsp; L CJ66<br>
14&quot; LCD DISPLAY<br>
INTEL DUAL CORE 3.0&nbsp;PROCESSOR<br>
1 GP DDRII RAM<BR>
160 GB HARDDISK<br>
IN BUILT MODEM<BR>
BLUE TOOTHENABLED<BR>
</td></tr>
</table>
<p>&nbsp;</p>
table birder="0" cellpadding="0"    cellspacing="0" style="border-collapse: collapse"
bordercolor "#111111" width "100%" id "AutoNumber2">
<tr><rd width="50%">
<p align="center">
```

```
<img border="0" src="imagesfd.jpg" width="240" height="180"></p>
<p align "center">
<img border="0" src="warning.gif" width="rad4" value="GET IT LCJ76"></td>
<td width "50%"> MODEL NO: GET IT LCJ76<br>
4&quot; LCD DISPLAY<br>
INTEL DUAL CORE&nbsp;
1 GP DDRII RAM<br>
160 GB HARDDISK<br>
IN BUILT MODEM<br>
BLUE TOOTH ENABLED<br>
</td></tr>
</table>
<p>&nbsp;</p>
table border="0" cellpadding="0" cellspacing="0" style="border-collapse:collapse" bordercolor "#111111" width "100%" id "AutoNumber2">
<tr><td width="50%">
<p align="center">
<img border="0" src="imagesfd.jpg" width="240" height="180"></p>
<p align "center">
<img border="0" src="warning.gif" width="40" height="40">
<input type="submit" name="red4" value="GET IT L CJ76"></td>
<td width "50%">MODEL NO: GET IT L CJ76<br> 4&quot; LCD DISPLAY<br>
INTEL DUAL CORE&nbsp; 4.0 PROCESSOR<br>
2 GB DDRII RAM<br>
160 GB HARDDISK<br>
IN BUILT MODEM<br>
BLUE TOOTH ENABLED<br>
WI-FI ENABLED<br>
</td></td></tr></table></form></body></html>
```

Select.php

```
<html>
```

```
<body bgcolor="white">
<font color="BLACK">
<p align "center"><b><font size "7" face "Franklin Gothic Medium">
<img border="0" src="warning.gif" width="40" height="40">GET IT SHOPPING</font></b></p>
<form method 'post' action 'final.php'><br><br>
<?php
echo "SELECTED MODEL:";
$model1 $_POST['rad1'];
echo"<B><U>$model1</B></U>";
$model2=$_POST['rad'];
echo "<B><U>$model1</B></U>";
$model2=$_POST['rad2'];
echo "<B><U>$model2</B></U>";
$model3=$_POST['rad3'];
echo "<B><U>$model3</B></U>";
$model4=$_POST['rad4'];
echo "<B><U>$model4</B></U><br><br>";
echo"<h3><center>PLEASE ENTER THE FOLLOWING DETAILS</center></h3>";
echo "Name: <input type='text name='cusname'><br><br>";
echo "Mobile Number:<input type='text' name='mobile'><br><br>";
echo "Address:<br><textarea name 'add'cols '20'rows '5'></textarea><br><br><br>";
echo "Credit Card no: <input type='password' name='pass1'><br><br>";
echo "Confirm Credit Card no: <input type='password' name='pass2'><br>";
?>
<br><center><inputtype="SUBMIT" value="CONFIRM ORDER"></center></form></body></html>
```

25. a)Explain Branching Statement with suitable program.

In PHP we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if a condition is true and another code if the condition is false
- **if...elseif....else statement** - use this statement to select one of several blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

The if Statement

Use the if statement to execute some code only if a specified condition is true.

Syntax

if (*condition*) *code to be executed if condition is true;*

The following example will output "Have a nice weekend!" if the current day is Friday:

```
<html><body>
<?php
$d=date("D");
if ($d=="Fri") echo "Have a nice weekend!";
?>
</body></html>
```

Notice that there is no ..else.. in this syntax. The code is executed **only if the specified condition is true**.

The if...else Statement

Use the if....else statement to execute some code if a condition is true and another code if a condition is false.

Syntax

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

Example

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":

```
<html><body>
<?php
```

```
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
else
  echo "Have a nice day!";
?>
</body></html>
```

If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces:

```
<html><body>
<?php
$d=date("D");
if ($d=="Fri")
  {
  echo "Hello!<br />";
  echo "Have a nice weekend!";
  echo "See you on Monday!";
  }
?>
</body></html>
```

The if...elseif....else Statement

Use the if....elseif...else statement to select one of several blocks of code to be executed.

Syntax

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

Example 1

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

```
<html><body>
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
```

```
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>
</body></html>
```

Example 2

```
<?php
// handle multiple possibilities
// define a different message for each day
$today = 'Tuesday';
if ($today == 'Monday') {
echo 'Monday\'s child is fair of face.';
} elseif ($today == 'Tuesday') {
echo 'Tuesday\'s child is full of grace.';
} elseif ($today == 'Wednesday') {
echo 'Wednesday\'s child is full of woe.';
} elseif ($today == 'Thursday') {
echo 'Thursday\'s child has far to go.';
} elseif ($today == 'Friday') {
echo 'Friday\'s child is loving and giving.';
} elseif ($today == 'Saturday') {
echo 'Saturday\'s child works hard for a living.';
} else {
echo 'No information available for that day';
}
?>
```

25.b)Explain Scope of Variables with illustration.

Variable scope is known as its boundary within which it can be visible or accessed from code. In other words, it is the context within which a variable is defined. There are only two scopes available in PHP namely local and global scopes.

Local variables (local scope)

Global variables (special global scope)

Static variables (local scope)

Function parameters (local scope)

When a variable is accessed outside its scope it will cause PHP error Undefined Variable.

Local Scope Variables

A local scope is a restricted boundary of a variable within which code block it is declared. That block can be a function, class or any conditional span. The variable within this limited local scope is known as the local variable of that specific code block.

The following code block shows a PHP function. We have declared a variable *$count* inside this function. This variable is said to be a local variable of this function and it is in the local scope of the function block.

```
<?php
function calculate_count() {
$count = 5;
//will print 5; the value of local variable
echo $count++;
}
?>
```

Local variables will be destroyed once the end of the code block is reached. Hence the same named variables can be declared within different local scopes.

2. Global Scope Variables

As its name, the global scope provides widespread access to the variable declared in this scope. Variables in global scope can be accessed from anywhere from outside a function or class independent of its boundary.

PHP global variables can be defined by using **global** keyword. If we want to use global variables inside a function, we have to prefix the global keyword with the variable. The following code shows a code block to learn how to use the global keyword with a PHP variable to declared it as a global variable.

```
<?php
$count = 0;
function calculate_count() {
    global $count;
    // will print 0 and increment global variable
    echo $count++ . "<br/>";
}
```

```
calculate_count();

echo $count;

?>
```

PHP has a predefined superglobal variable called **$GLOBALS**. It is an associative array with the name of the variable as key and value as the array element. We can use this array variable to add an array of PHP variables in a global scope.

Let us change the above example with the global keyword by using $GLOBALS superglobal to access the variable in global scope.

```
<?php

$count = 0;

function calculate_count() {

// will print 0 and increment global variable declared outside function

echo $GLOBALS["count"]++ . "<br/>";

}

calculate_count();

echo $count;

?>
```

3. Static Variables (local scope)

A static variable is again a variable with local scope. But the difference with the regular local variable is that it is not destroyed outside the scope boundary. A variable can be defined by using the ‘static’ keyword inside a function.

A static variable does not lose its value when the program execution goes past the scope boundary. But it can be accessed only within that boundary. Let me demonstrate it using the following example code,

```
<?php

function counter()

{

    static $count = 0;

    echo $count;

    $count++;

}

?>
```

The above counter function has the static variable ‘count’ declared within its local scope. When the function execution is complete, the static count variable still retains its value for further computation. Every time the counter function is called, the value of count is incremented. The count value is initialized only once on the first call.

4. Function Parameters (Local Scope)

Function parameters (arguments) are local variables defined within the local scope of the function on which it is used as the argument.

Scope and File Includes

The boundary of file includes does not demarcate the scope of variables. The scope of a variable is only governed by the function block and not based on the file include. A variable can be declared in a PHP file and used in another file by using ‘include’ or ‘require’ that file.

Function Inside Function or Class

Remember that the scope in PHP is governed by a function block. Any new function declared anywhere starts a new scope. If an anonymous function is defined inside another function, the anonymous function has its own local scope.

```
<?php

function foo() {

    $fruit = 'apple';

    $bar = function () {

        // $fruit cannot be accessed inside here

        $animal = 'lion';

    };

    // $animal cannot be accessed outside here

}

?>
```

26. a)Design a php application for uploading online resume.

<html>

```
<head><title>result</title></head>
<body bgcolor="456789">
<font color="white">
<center><b><h1>FRESHERSWORLD.COM</b></h1><br>
    <br><br><br>
</center>
<?php
echo"<b><i>";
echo"<h3>";
if(isset($_POST['posted']))
{
$fn=$_POST['firstname'];
$ln=$_POST['lastname'];
$ad=$_POST['address'];
$eid=$_POST['em'];
if($fn == "")
{
echo "Enter ur name <br>";
}
if($ad == "")
{
echo "Enter ur address <br>";
}
if($eid == "")
{
echo "Enter ur E-mail ID <br>";
}
if($fn != "" && $ad != "" && $eid !="")
{
echo"$fn$ln your resume is accepted<br>";
}
}
echo"</font>";
echo"</h3></b></i><br><br>";
echo"<font color='yellow'><h2>press Alt + Left Arrow </h2>";
echo"</font>";
?>
</body>
</html>
```

Apple showRoom

First Name:

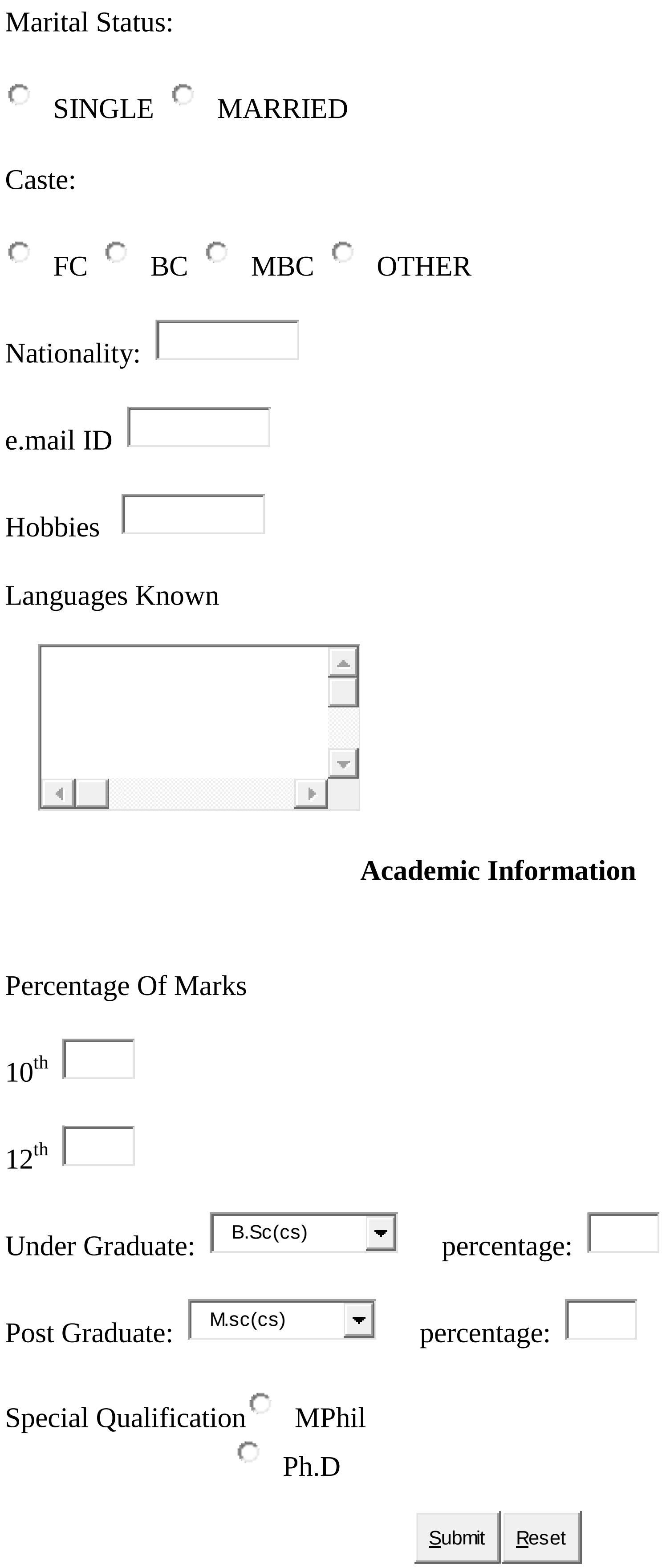
Last Name:

Date Of Birth:

Address:

Sex:

○ MALE ○ FEMALE



Marital Status:

○ SINGLE ○ MARRIED

Caste:

○ FC ○ BC ○ MBC ○ OTHER

Nationality:

e.mail ID

Hobbies

Languages Known

Academic Information

Percentage Of Marks

10th

12th

Under Graduate: B.Sc(cs) percentage:

Post Graduate: M.sc(cs) percentage:

Special Qualification ○ MPhil
○ Ph.D

26.b) Explain different types of Exception Handling

With PHP 5 came a new object oriented way of dealing with errors.

Exception handling is used to change the normal flow of the code execution if a specified error (exceptional) condition occurs. This condition is called an exception.

This is what normally happens when an exception is triggered:

- The current code state is saved
- The code execution will switch to a predefined (custom) exception handler function
- Depending on the situation, the handler may then resume the execution from the saved code state, terminate the script execution or continue the script from a different location in the code

Try, throw and catch

To avoid the error from the example above, we need to create the proper code to handle an exception.

Proper exception code should include:

1. Try - A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However if the exception triggers, an exception is "thrown"
2. Throw - This is how you trigger an exception. Each "throw" must have at least one "catch"
3. Catch - A "catch" block retrieves an exception and creates an object containing the exception information

Lets try to trigger an exception with valid code:

```
<?php
//create function with an exception
function checkNum($number)
  {
  if($number>1)
    {
    throw new Exception("Value must be 1 or below");
    }
  return true;
  }

//trigger exception in a "try" block
try
  {
  checkNum(2);
  //If the exception is thrown, this text will not be shown
  echo 'If you see this, the number is 1 or below';
  }

//catch exception
catch(Exception $e)
  {
  echo 'Message: ' .$e->getMessage();
  }
?>
```

The code above will get an error like this:

```
Message: Value must be 1 or below
```

Example explained:

The code above throws an exception and catches it:

1. The checkNum() function is created. It checks if a number is greater than 1. If it is, an exception is thrown
2. The checkNum() function is called in a "try" block
3. The exception within the checkNum() function is thrown
4. The "catch" block retrives the exception and creates an object ($e) containing the exception information
5. The error message from the exception is echoed by calling $e->getMessage() from the exception object

However, one way to get around the "every throw must have a catch" rule is to set a top level exception handler to handle errors that slip through.

Creating a Custom Exception Class

Creating a custom exception handler is quite simple. We simply create a special class with functions that can be called when an exception occurs in PHP. The class must be an extension of the exception class.

The custom exception class inherits the properties from PHP's exception class and you can add custom functions to it.

Lets create an exception class:

```
<?php
class customException extends Exception
  {
  public function errorMessage()
    {
    //error message
    $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
    .': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
    return $errorMsg;
    }
  }

$email = "someone@example...com";

try
  {
  //check if
  if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE)
    {
    //throw exception if email is not valid
    throw new customException($email);
    }
  }

catch (customException $e)
  {
  //display custom message
  echo $e->errorMessage();
  }
?>
```

The new class is a copy of the old exception class with an addition of the errorMessage() function. Since it is a copy of the old class, and it inherits the properties and methods from the old class, we can use the exception class methods like getLine() and getFile() and getMessage().

Example explained:

The code above throws an exception and catches it with a custom exception class:

1. The customException() class is created as an extension of the old exception class. This way it inherits all methods and properties from the old exception class
2. The errorMessage() function is created. This function returns an error message if an e-mail address is invalid
3. The $email variable is set to a string that is not a valid e-mail address
4. The "try" block is executed and an exception is thrown since the e-mail address is invalid
5. The "catch" block catches the exception and displays the error message

---

Multiple Exceptions

It is possible for a script to use multiple exceptions to check for multiple conditions.

It is possible to use several if..else blocks, a switch, or nest multiple exceptions. These exceptions can use different exception classes and return different error messages:

```
<?php
class customException extends Exception
{
public function errorMessage()
{
//error message
$errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
.': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
return $errorMsg;
}
}

$email = "someone@example.com";
```