**END TERM EXAMINATION**

FIFTH SEMESTER (BCA) DECEMBER 2015-JANUARY 2016

**UNIT-IV**

Q9. (a) Explain Errors in PHP with its types with suitable example.

An error is a type of mistake. We can say an error is a condition of having incorrect or false knowledge or an error is defined as an unexpected, invalid program state from which it is possible to recover.

Types of Error

Basically, there are four types of errors in PHP, which are as follows:

* Parse error (Syntax error)
* Fatal error
* Warning error
* Notice error

**1. Parse Errors (Syntax errors)**

The Parse error occurs if there is a syntax mistake in the script; the output is parse errors. A parse error stops the execution of the script. There are many reasons for the occurrence of parse errors in PHP. The common reasons are as follows:

* Unclosed quotes
* Missing or Extra parentheses
* Unclosed braces
* Missing semicolon

Example: -

<?php

echo "cat";

echo "Dog"

echo "Lion";

?>

Output: - In the above code we missed semicolon in the second line. When happens there will be a parse error which stops execution of the script.

**2. Fatal errors**

Fatal errors are caused when PHP understands that you've written, however what you're asking it to do can't be done. Fatal errors stop the execution of the script. If you trying to access the undefined functions, then the output is fatal error.

Example: -

<?php

Function fun1() {

echo "ABC"; }

Fun2()

echo "Fatal error !";

?>

Output: - In the above code we defined a function fun1() but we call another function fan2() i.e., fun2() is not defined. So, a fatal error will be produced that stops execution of the script.

**3. Warning Errors**

Warning errors will not stop execution of the script. The main reason for warning

errors are to include a missing file or using the incorrect number of parameters in a function.

Example: -

<?php

echo "Warning error";

Include('welcome.php');

?>

Output: - In the above we include a welcome.php, however the welcome.php file does not exist in the directory so there will be a warning error produced but that does not stop the execution of the script i.e., you will see a message warning error.

**4. Notice errors**

Notice error is the same error as a warning error i.e., in the notice error execution of the script does not stop. Notice that the error occurs when you try to access the undefined variable, and then produce a notice error.

Example: -

<?php

$a= "Hello";

echo "Notice error";

echo $b;

?>

Output: - In the above code we defined a variable which named $a. but we call another variable $b which is not defined so there will be a notice error produced but execution of the script does not stop, you will get a message notice error.

(b) Explain any two functions in detail: -

1. mysql\_affected\_rows()
2. mysql\_select\_db()
3. mysql\_query()

mysql\_affected\_rows(): - It gets the number of affected rows by the last INSERT, UPDATE, REPLACE or DELETE query associated with link\_identifier.

Syntax: - mysql\_affected\_rows(resource $link\_identifier = NULL): int

Parameters: - link\_identifier - The MySQL connection. If the link identifier is not specified, the last link opened by mysql\_connect() is assumed. If no such link is found, it will try to create one as if mysql\_connect() had been called with no arguments. If no connection is found or es Return Values blished, an E\_WARNING level error is generated.

Return Values: - Returns the number of affected rows on success, and -1 if the last query failed.

If the last query was a DELETE query with no WHERE clause, all of the records will have been deleted from the table but this function will return zero with MySQL versions prior to 4.1.2.

When using UPDATE, MySQL will not update columns where the new value is the same as the old value. This creates the possibility that mysql\_affected\_rows() may not actually equal the number of rows matched, only the number of rows that were literally affected by the query.

The REPLACE statement first deletes the record with the same primary key and then inserts the new record. This function returns the number of deleted records plus the number of inserted records.

In the case of "INSERT ... ON DUPLICATE KEY UPDATE" queries, the return value will be 1 if an insert was performed, or 2 for an update of an existing row.

Example: -

<?php

$link = mysql\_connect('localhost', 'mysql\_user', 'mysql\_password');

if (!$link) {

die('Could not connect: ' . mysql\_error());

}

mysql\_select\_db('mydb');

/\* this should return the correct numbers of deleted records \*/

mysql\_query('DELETE FROM mytable WHERE id < 10');

printf("Records deleted: %d\n", mysql\_affected\_rows());

/\* with a where clause that is never true, it should return 0 \*/

mysql\_query('DELETE FROM mytable WHERE 0');

printf("Records deleted: %d\n", mysql\_affected\_rows());

?>

mysql\_select\_db: - It sets the current active database on the server that's associated with the specified link identifier. Every subsequent call to mysql\_query() will be made on the active database.

Syntax: - mysql\_select\_db(string $database\_name, resource $link\_identifier = NULL): bool

Parameters: - database\_name - The name of the database that is to be selected.

Return Values: - Returns true on success or false on failure.

Example: -

<?php

$link = mysql\_connect('localhost', 'mysql\_user', 'mysql\_password');

if (!$link) {

die('Not connected : ' . mysql\_error());

}

// make foo the current db

$db\_selected = mysql\_select\_db('foo', $link);

if (!$db\_selected) {

die ('Can\'t use foo : ' . mysql\_error());

}

?>

mysql\_query(): - It sends a unique query (multiple queries are not supported) to the currently active database on the server that's associated with the specified link\_identifier.

Syntax: - mysql\_query(string $query, resource $link\_identifier = NULL): mixed

Parameters: - query - An SQL query, the query string should not end with a semicolon. Data inside the query should be properly escaped.

Return Values: - For SELECT, SHOW, DESCRIBE, EXPLAIN and other statements returning resultset, mysql\_query() returns a resource on success, or false on error.

For other type of SQL statements, INSERT, UPDATE, DELETE, DROP, etc, mysql\_query() returns true on success or false on error.

The returned result resource should be passed to mysql\_fetch\_array(), and other functions for dealing with result tables, to access the returned data.

Use mysql\_num\_rows() to find out how many rows were returned for a SELECT statement or mysql\_affected\_rows() to find out how many rows were affected by a DELETE, INSERT, REPLACE, or UPDATE statement.

mysql\_query() will also fail and return false if the user does not have permission to access the table(s) referenced by the query.

Example: -

<?php

$firstname = 'fred';

$lastname = 'fox';

$query = sprintf("SELECT firstname, lastname, address, age FROM friends WHERE firstname='%s' AND lastname='%s'", mysql\_real\_escape\_string($firstname), mysql\_real\_escape\_string($lastname));

$result = mysql\_query($query);

if (!$result) { $message = 'Invalid query: ' . mysql\_error() . "\n"; $message .= 'Whole query: ' . $query; die($message); }

while ($row = mysql\_fetch\_assoc($result)) { echo $row['firstname'];

echo $row['lastname'];

echo $row['address'];

echo $row['age'];}

mysql\_free\_result($result);

?>