VPMP POLYTECHNIC, GANDHINAGAR
DEPARTMENT OF COMPUTER ENGINEERING

IMP QUESTIONS - Solution

SEMESTER:4th **SUBJECT:IWD**

UNIT 1

1) Differentiate ECHO and PRINT statements of PHP.

Echo	Print
It displays **one or more** strings	It **displays** a string.
Syntax: echo (string)	**Syntax**: print (string)
It **does not return** any value.	It **always return** a value.
It is **faster** as compared to print statement.	It is **slower** as compared to echo statement.
Ex. **echo "hello world";**	Ex. **print("Hello World");**
echo is not a function but it is language construct.	print() is a function

2) Explain PHP file structure with example.

PHP structure:

1. PHP is a scripting language, so it has merge with any web development language like html.
2. PHP file always saves with .php extension.
3. Can be modify with any editor like notepad, vi, wordpad, dreamweaver.
4. PHP script must contain between <?php and?>.
5. To exucute successfule PHP script you must follow syntax rules given below.

Syntax:

1. PHP always starts and ends with tags.

```
<?php     //This tag is used as PHP starting tag.
?>                  //This tag is used as PHP ending tag.
```

2. PHP lines end with the semicolon.

```
<?php
        echo "hello";
?>
```

3. Commenting with // or /*and*/.

4. String must contained between either single quotation mark or double quotation mark.

Example:

```
<?php
        echo "hello"; //valid
```

```
        echo 'hello '; //valid
        echo hello; //Not valid
?>
```

5. Every variable start with $ sign in a php script.

Example:

```
<?php
        no = 10;  //Not valid
        $no = 10; //valid
?>
```

6. Name of variable can not start with digit and it can not contain white spaces.

Example:

```
<?php
        $1no = 10;  //Not valid
        $no = 10; //valid
        $no  1 = 10; //Not valid
?>
```

7. PHP is case sensitive.

Example:

```
<?php
        NO = 10;
        $no = 20;
?>
```

3) What is variable? List the variable naming rules in PHP.

Variables in PHP:

- ✓ Variable is a name which is used to store temporary value.
- ✓ It can be use anytime through program.
- ✓ It can hold alphabetic, numeric or string value.
  - **Rules for define the variable is given below:**
    a. Always declare variable name with "$" (dollar) symbol.
    b. Variable name contains letters, digits and underscore symbol.
    c. Declaration of variable name should not contain any white space.
    d. It cannot start with digits.
    e. Uppercase and Lowercase letters are distinct
    f. Examples given below:

$_abc (Valid)	$abc (Valid)	$1abc (Invalid)	$abc2 (Valid)
$@abc (Invalid)	$ab@c (Invalid)	$.abc (Invalid)	$ab.c (Invalid)

4) Explain different methods to implement constant in PHP.

- Value of constant cannot be changed during whole program

- The PHP constants are define with function "define()" .
- Retrieve value of it by use of function "constant()".
- Constant are generally defined with capital letters, so it can easily find within the code.
- Value of constant is defined before use the constant.

<?php define("PI",3.14); echo PI; echo " ".constant("PI"); ?>	Output 3.14 3.14

Predefined constant:

- Constants defined at the time of development is known as predefined constants.
- They are also known as built in constants
- Ex. PHP_VERSION, E_ERROR etc.

Constant	Purpose
PHP_VERSION	Represent current version of PHP
PHP_MAJOR_VERSION	Represent major version of PHP
PHP_MINOR_VERSION	Represent minor version of PHP
E_ERROR	Used to represent various types of error
E_COMPILE_ERROR	
E_ALL	

Magical constant:

- They are called magical constant because they change depending upon context in which they are used.
- Always in capital letters, start and end with double underscore
- Ex. __LINE__ ,__DIR__ etc.
- Used for debugging purpose
- Other examples given below:

Constant	Purpose
__LINE__	Represent current line no in file
__FILE__	Represent name of file along with full path
__DIR__	Represent directory of file
__FUNCTION_	Represent function name
__CLASS__	Represent class name
__METHOD__	Represent method name
__NAMESPACE__	Represent current namespace

5) Explain Bitwise and logical operator in PHP.

Logical Operators.

Operator	Name	Example	Result
and	AND	Condition1 and condition2	Return TRUE only if both the conditions are true.
or	OR	Condition1 or condition2	Return TRUE when any one condition is true.

xor	X-or	Condition1 xor condition2	Return TRUE either any condition is true, but return FALSE if both conditions are true.
!	NOT	! condition	TRUE if condition is not true.
&&	AND	Condition1 && condition2	Return TRUE only if both the conditions are true.
||	OR	Condition1 || condition2	Return TRUE when any one condition is true.

Example: <?php

```
$a=10;
$b=15;
If($a<$b)
    echo "a is greater";
else
    echo "b is greater"; ?>
```

Output: b is greater

Bitwise Operators:

Operator	Name	Example	Result	Operation
&	AND	$x = $x & $y	$x = 1;	Logical AND Operation
|	OR	$x = $x | $y	$x = 7;	Logical OR Operation
^	Ex - OR	$x = $x ^ $y	$x = 6;	Logical Exclusive OR Operation
~	NOT	$x = ~$x	$x = -4;	Logical NOT Operation
<<	Shift Left	$x = $x << $y	$x = 96;	Shift Left Operation (Multiply by 2). So here 3*2 doing 5 times. So, result = 96.
>>	Shift Right	$x = 8, $y = 2; $x = $x >> $y	$x = 2;	Shift Right Operation (Divide by 2). So here 8/2 doing 2 times. So, result = 2.

Example:

5=101 , 3=011

101

011

---

001 = 1 [AND]

111 = 7 [OR]

6) Explain arithmetic and comparison operator in PHP.

Arithmetic Operators:

It contains basic arithmetic operation like addition, subtraction, multiplication, division.

Operator	Name	Example	Result
-	1. Negation	1. -$x	1. Opposite of $x.

	2. Subtraction	2. $x - $y	2. Difference of $x and $y.
+	Addition	$x + $y	Summation of $x and $y.
*	Multiplication	$x * $y	Product of $x and $y.
/	Division	$x / $y	Division of $x and $y.
%	Modulus	$x % $y	Reminder of $x divided by $y.

comparison operator: This type of operators useful for make a condition, which are used in conditional or looping structures.

Operator	**Name**	**Example**	**Result**
==	Is equal to	$x == $y	True, Only when value of $x and $y are same.
===	Is identical	$x === $y	True, Only when value and type of $x and $y are same.
!=	Is Not equal to	$x != $y	True, Only when value of $x and $y are different.
<>	Is Not equal to	$x <> $y	True, Only when value of $x and $y are different.
!==	Is not identical	$x !== $y	True, Only when value and type of $x and $y are Different.
<	Is less than	$x < $y	True, Only if value of $x is strictly less than $y.
>	Is greater than	$x > $y	True, Only if value of $x is strictly greater than $y.
<=	Is less than or Equal to	$x <= $y	True, Only if value of $x is less than or equal to $y.
>=	Is less than or Equal to	$x >= $y	True, Only if value of $x is greater than or equal to $y.

7) List out various PHP conditional structures and explain any one with example.

Conditional Structures:

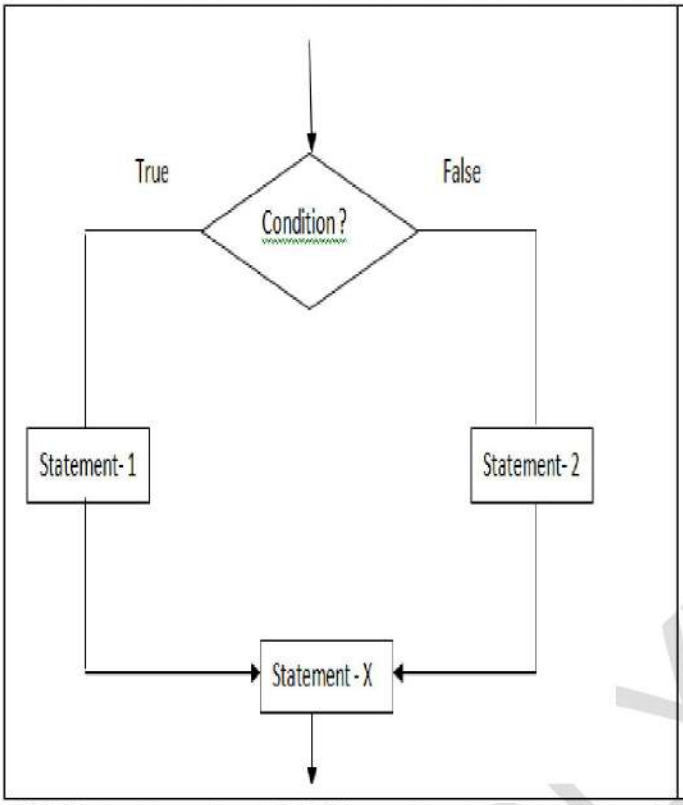
- ✓ Simple if
- ✓ If---else
- ✓ Nested if---else
- ✓ Else if ladder
- ✓ Switch case

if-else statement

Syntax:

```
if(condition)
{
        Statement 1;
}
else
{
        Statement  2;
 }
```

- ✓ first the condition is checked.
- ✓ if condition is true the statement -1 is executed.
- ✓ if condition is false the statement - 2 is executed.



Example:

```
<?php
 $marks=45;
 if( $marks < 35 )
 {
    echo "Student is fail";
 }
 else
 {
    echo "Student is pass";
 }
?>
```

Output:
Student is pass

8) List out various PHP loop structures and explain while and for each loop with example.

loop structures:

- ✓ while
- ✓ do—while
- ✓ for
- ✓ foreach

while loop: While loop is very simple type of loop in PHP. The while loop executes a block of code as long as the specified condition is true.

Syntax for while loop:

```
<?php
while(condition)
{
    // statement 1;
    // statement 2;
}
?>
```

Example:

```
<?php
$x = 1;

while($x <= 5)
{
 echo $x."<br>";
 $x++;
}
?>
```

Output:

1
2
3
4
5

foreach loop: It is used in PHP to loop all elements of an array.

- ✓ If you want to perform some task repeatedly for each elements of an array, Foreach looping structure is used.

✓ In for each loop, how many times the loop will execute depends on number of elements in the array.
✓ For example, if an array contains 10 elements, then loop will execute 10 times.

Syntax:	<?php
	$no = array("10", "20", "30", "40");
	
	foreach ($no as $n) {
	echo $n;
	}
	?>
	Output:
	10 20 30 40

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

9) Explain break and continue statement with example

Break: The **break** statement terminates the whole iteration of a loop

✓ The break statement can also be used to jump out of a loop.
✓ As the break statement encountered inside a loop, it immediately terminated the loop statement and transferred the control to the next statement, followed by a loop to resume the execution.

Example of break:	Output:

<?php
for ($x = 1; $x <=10; $x++)
{
if ($x == 4)
break;
 	
echo $x ."
";
}
?>
	1
	2
	3

Continue: The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

Example of Continue:	Output:

<?php
for ($x = 1; $x <= 10; $x++)
{
if ($x == 4)
continue;
echo $x ." ";
}
?>
	1 2 3 5 6 7 8 9 10

Unit 2

10) Define Array. Explain different types of array in PHP.

Array:

✓ An array stores multiple values in one single variable.

- ✓ An array is a special variable, which can hold more than one value at a time.

Types of Array: In PHP, there are three types of arrays:

- ✓ **Indexed arrays/Numeric Array** - Arrays with a numeric index
- ✓ **Associative arrays** - Arrays with named keys
- ✓ **Multidimensional arrays** - Arrays containing one or more arrays

Creating arrays: There are two ways for creating array:

- ✓ Create array Using array() function
- ✓ Create array using array identifier.

Indexed arrays/Numeric: Array Arrays with a numeric index

Create array : Using array() function

- ✓ In PHP, the array() function is used to create an array.
- ✓ Syntax: $ArrayName = array(value1,value2….valueN);

Example:

```
<?php
  $no=array(10, 20,30);
  print_r($no);
?>
```

Output:

```
Array([0] =>10 [1] =>20 [2]=>30)
```

Create array : Using Array Identifier

```
<?php
      $no[ ] = 10;
      $no[ ] = 20;
      $no[ ] = 30;
      print_r($no);
?>
```

Output:

```
Array([0] =>10 [1] =>20 [2]=>30)
```

Associative array

- ✓ In associative array, each elements having key associated with it.
- ✓ It can be either numeric or string.
- ✓ Syntax:
  $ArrayName = array(Key1=>Value1, Key2=>Value2, …, KeyN=>ValueN)
- ✓ In PHP, if we don't specify key value for each element then it will create a numeric array.

```
<?php
```

```
$name = array(19=> "Arpan", 20=>"Charmi");

print_r($name);

?>
```

Output:

Array ([19]=>Arpan [20]=>Charmi)

Multidimensional array

- ✓ An array contain both the row and column makes a two dimensional array.
- ✓ array contain another array makes a multi-dimensional array.

```
<?php

$a = array(array(1, 1, 1),array(1, 1, 1),array(1, 1, 1));

?>
```

11) List out string Declaration types in PHP and explain heredoc and nowdoc for string declaration.

Declaring Strings: There are three ways.

- ✓ Single Quoted
- ✓ Double Quoted
- ✓ Doc Syntax
  - Nowdoc - Behaves like Single Quoted
  - Heredoc - Behaves like Double Quoted

Heredoc:

Syntax:

```
<?php

$str = <<<IDENTIFIER
place a string here
it can span multiple lines
and include single quote ' and double quotes "
IDENTIFIER;
```

How it works:

- ✓ First, start with the <<< operator, an identifier, and a new line:
- ✓ Second, specify the string, which can span multiple lines and includes single quotes (') or double quotes (").
- ✓ Third, close the string with the same identifier.

Using Heredoc:

```
<?php

$a = 'Mr. Patel';
$b= 'Mr. Joshi';
```

```
$text = <<<TEXT
$a said:"Hello".
$b said:"How are you?";
TEXT;

echo $text;
```

Output:
Mr. Patel said: "Hello". Mr. Joshi said:"How are you?"

Nowdoc:

- ✓ A nowdoc string is similar to a heredoc string except that it **doesn't expand the variables.**
- ✓ The nowdoc's syntax is similar to the heredoc's syntax except that the identifier which follows the <<< operator needs to be enclosed in single quotes.

Syntax:

```
<?php
$str= <<<'IDENTIFIER'
place a string here
it can span multiple lines
and include single quote ' and double quotes "
IDENTIFIER;
```

```
<?php

$a = 'Mr. Patel';
$b= 'Mr. Joshi';

$text = <<<'TEXT'
$a said:"Hello".
$b said:"How are you?";
TEXT;

echo $text;
```

Output:
$a said:"Hello". $b said:"How are you?";

12) Explain user defined function with default argument.

- ✓ In PHP, You can define function having default argument.
- ✓ If you don't pass the value for that argument then it will consider default value for that argument.
- ✓ If you pass explicit value for that argument then it will overwrite the default argument value.
- ✓ Default arguments must be written from right to left direction.
- ✓ Syntax:

```
funtionfunctionName( $argument1, $argument2=Value)
{
        Function Body
```

```
}
```

Example	**Output:**

<?php
function total($a, $b = 50)
{
echo "Total = " . $a + $b ."
";
}
 	
total(350);
total(10,65);
?>
	400
	75

13) Explain any two string functions with example.

strlen()

- ✓ This function accepts a string as an argument and returns the number of character of a string.
- ✓ Syntax : strlen(string)
- ✓ Example

```
<?php
    Echo strlen("hello");
?>
```

Output

5

strrev()

- ✓ This function accepts a string as an argument and reverse that string.
- ✓ Syntax : strrev(string);
- ✓ Example

```
<?php
    echo strrev("HELLO");
?>
```

Output

OLLEH

strtolower()

- ✓ This function accepts a string as an argument and coverts all the characters of the string into lowercase letters.
- ✓ Syntax : strtolower(string);
- ✓ Example

```
<?php
    echo strtolower("HELLO");
?>
```

Output :helllo

strtoupper()

- ✓ This function accepts a string as an argument and coverts all the characters of the string into uppercase letters.
- ✓ Syntax : strtoupper(string);

✓ Example

```
<?php
        echo strtoupper("hello");
?>
```

Output

HELLO

14) Explain settype() and gettype() functions with example.

gettype: Get the type of a variable.

Syntax: gettype($var)

- ✓ Return the type of PHP variable var.
- ✓ Possible values for the returned string are: boolean, integer, double, string, array, object, resource, NULL, unknown type.
- ✓ Example:

```
<?php
$a=10;
echo gettype($a);
?>
```

Output:

integer

settype: Set the type of a variable.

Syntax: settype($var, string type)

- ✓ Possible values for the returned string are: boolean, integer, double, string, array, object, resource, NULL, unknown type.
- ✓ Example:

```
<?php
$a=10.5;
settype($a,"integer");
echo $a;
?>
```

Output: 10

15) Explain checkdate() and mktime() functions with example.

mktime(): The mktime() function returns the Unix timestamp for a date.

It contains the number of seconds between the Unix Epoch (January 1 1970 00:00:00 GMT) and the time specified.

Syntax : mktime(hour,minute,second,month,day,year)	Example <?php $d=mktime(11, 14, 54, 8, 12, 2014); echo "Created date is " . date("Y-m-d h:i:sa", $d); ?> Output Created date is 2014-08-12 11:14:54am

checkdate() : The checkdate() function is used to validate a date.

Syntax :	Example	Output

checkdate(month,day,year);	<?php
	var_dump(checkdate(12,31,-400));
	echo "
	";
	var_dump(checkdate(2,29,2003));
	echo "
	";
	var_dump(checkdate(2,29,2004));
	?>
		bool(false)
		bool(false)
		bool(true)

16) Explain any two MATH functions with example.

pow(): The pow() function returns x raised to the power of y.

Syntax :	Example	Output
pow(x,y); x specifies the base to use. y specifies the exponent.	<?php echo(pow(2,4) . " "); echo(pow(-2,4) . " "); ?>	16 16

sqrt(): The sqrt() function returns the square root of a number.

Syntax : sqrt(*number*);	Example	Output
	<?php echo(sqrt(1) . " "); echo(sqrt(9) . " "); ?>	1 3

17) Explain GLOBAL and STATIC variable in PHP with example.

Global variable

- ✓ A variable that is declared outside all the function are known as global variable.
- ✓ But in PHP, to access the global variable inside the function we need to declare them using *global* keyword.
- ✓ Syntax:

global $variableName;

Example:	**Output**

<?php
$a=5;
$b=10;
$c;
function sum()
{
global $a,$b,$c;
$c=$a+$b;
}
sum();
echo "Sum is:->".$c;
?>
	Sum is:->15

Static variable

- ✓ The static variable is the variable, whose value is constant through the program.

- ✓ It contains the last value which assign to the static variable.
- ✓ Syntax: static $variablename = value;

```
<?php
    function increment()
    {
    Static $a=10;
    $a++;
    echo $a;
}
increment();
increment();
increment();
?>
Output: 11 12 13
```

18) Explain Passing arguments by value and Passing arguments by reference

Passing Arguments by value

- ✓ When we pass arguments to the function, the values of the passed arguments are copied into argument variables declared inside the argument list of function definition.
- ✓ So, called function works with copies of argument instead of original passed arguments.
- ✓ So any changes made to these variables in the body of the function are local to that function and are not reflected outside it.

```
<?php
        function Swap($a,$b)
        {
        $c=$a;
        $a=$b;
        $b=$c;
        }
    $a=5;
    $b=10;
    echo "Before swapping<br>";
    echo "a=".$a."<br>";
    echo "b=".$b."<br>";
    Swap($a,$b);
    echo "After swapping<br>";
    echo "a=".$a."<br>";
    echo "b=".$b."<br>";
?>
```

Output:

```
Before swapping
a=5
b=10
After swapping
a=5
b=10
```

Passing Arguments by Reference

- ✓ When we want to work with original variable that are passed as an argument , Call by Reference is used.
- ✓ When arguments are passed by reference the function works with original variable instead of copies of that variable.
- ✓ In order to pass arguments by reference each argument in the function definition should be preceded by an (&) sign.

```
<?php
        function Swap(&$a,&$b)
        {
```

Output:

```
        $c=$a;
        $a=$b;
        $b=$c;
        }
    $a=5;
    $b=10;
    echo "Before swapping<br>";
    echo "a=".$a."<br>";
    echo "b=".$b."<br>";
    Swap($a,$b);
    echo "After swapping<br>";
    echo "a=".$a."<br>";
    echo "b=".$b."<br>";
?>
```

```
Before swapping
a=5
b=10
After swapping
a=10
b=5
```

Unit 3

19) How to create class and object in PHP. Explain with example.

Class:

- ✓ A class is defined by using the `class` keyword, followed by the name of the class and a pair of curly braces ({}). All its properties and methods go inside the braces:
  Syntax:

```
<?php
class Classname {
  // code goes here...
}
?>
```

Object:

- ✓ Objects of a class are created using the `new` keyword.
- ✓ We can create multiple objects from a class. Each object has all the properties and methods defined in the class, but they will have different property values.

```
<?php
class Add {
  protected $a;
  protected $b;

  function set_no($a,$b) {
    $this->a = $a;
    $this->b = $b;
  }
  function sum() {
    return $this->a + $this->b;
  }
}

$no = new Add();
$no->set_no(10,20);
```

```
30
```

echo $no->sum(); echo " "; ?>	

20) Explain constructor and destructor with example.

Constructors:

✓ constructor is a method defined inside a class is called automatically at the time of creation of object.

✓ Purpose of a constructor method is to initialize the object.

✓ A constructor is a public method which is named as __construct

Syntax:

```
function __construct()
 {
 // initialize the object and its properties by assigning values
 }
```

✓ Destructors are called when an object destructs. Usually, it is when the script ends.

✓ A destructor is a public method which is named as __ destruct

Syntax:

```
function __destruct()
{
// destroying the object or clean up resources here
}
```

Example:	Output:

<?php
class MyClass {
public function __construct() {
echo "Object created"."
";
}
 	
public function __destruct() {
echo "Object destroyed";
}
}
 	
$obj = new MyClass();
unset($obj);
?>
	Object created
	Object destroyed

21) Explain Inheritance with example.

✓ **It is the process the by which object of one class derived (acquired) the properties of object of another class.**

- ✓ In inheritance, the old class is called as the base class and the new class is called as the derived class or subclass.
- ✓ PHP supports three types of inheritance based on their functionality.
  - ✓ Single Inheritance: There is only one base class and one sub/derived class in a single inheritance
  - ✓ Hierarchical Inheritance: multiple derived classes are inherited from the base class.
  - ✓ Multilevel Inheritance: one base class is inherited by a derived class, then that derived class is inherited by other derived classes, and so on.

Syntax for Inheriting a Class: class ChildClass extends ParentClass

where, ChildClass is a derived class (also called extended class and subclass) that extends ParentClass (also called superclass and base class).

```
<?php

class ParentClass
{
 function a()
 {
  echo "A member function of the base class.<br>";
 }
}


class ChildClass extends ParentClass
{
 function b()
 {
  echo "A member function of the child class.<br>";
 }
}

$Obj = new ChildClass();
$Obj->a();
$Obj->b();
?>
```

Output:

member function of the base class.
A member function of the child class.

22) Explain function overloading and function overriding.

- ✓ **Overloading** means to use the same thing for different purpose.
- ✓ It contains the same function name and that function performs different tasks according to the number of arguments.

- ✓ For example, find the area of certain shapes where the radius is given then it should return the area of a circle if height and width are given then it should give the area of rectangle and others.

Function Overloading

```
<?php
class Area
{

function __call($name, $arg)
{

if($name == 'area')
{
switch(count($arg))
{

case 0 : return 0 ;
case 1 : return 3.14 * $arg[0] ;
case 2 : return $arg[0] * $arg[1];
}

}

}
}
$c = new Area();
echo "Area of circle:".$c->area(5)."</br>";
$rect = new Area();
echo "Area of rectangle:".$rect->area(5,10);

?>


Area of circle:15.7
Area of rectangle:50
```

function overriding: The two methods with the same name and same parameter is called overriding.

- ✓ Both parent and child classes should have same function name with and number of arguments.
- ✓ It is used to replace parent method in child class. The purpose of overriding is to change the behavior of parent class method.

```
<?php
  class Base
 {
     function display()
  {
        echo "Base class function :display <br>";
```

```
  }

function demo()
{
      echo "Base class function:demo<br>";
  }
 }
  class Derived extends Base
 {
    function demo()
  {
      echo "Derived class function:demo<br>";
    }
  }
  $ob = new Base;
  $ob->demo();
  $ob->display();
  $ob1 = new Derived;
  $ob1->demo();
  $ob1->display();
?>

Output:
Base class function: demo
Base class function :display
Derived class function: demo
Base class function :display
```

23) Explain Interface with example.

- ✓ Interfaces are declared with the **interface** keyword.
- ✓ All methods declared in an interface must be public.
- ✓ To implement an interface, a class must use the **implements** keyword.
- ✓ A class that implements an interface must implement all of the interface's methods.

```
<?php
interface MyInterface
{

  public function method1();
  public function method2();

}

class MyClass implements MyInterface
{

  public function method1()
{
     echo "Method1 Called<br>" ;
  }
```

```
   public function method2()
{
      echo "Method2 Called<br>";
   }
}

$obj = new MyClass;
$obj->method1();
$obj->method2();

?>
```

Output:

```
Method1 Called
Method2 Called
```

24) Explain cloning of object.

- ✓ The clone keyword is used to create a copy of an object.
- ✓ When an object is cloned, PHP will perform a shallow copy of all of the object's properties.
- ✓ **Syntax:** $copy_object_name = clone $object_to_be_copied;
- ✓ The **clone** keyword creates a shallow copy.
- ✓ Change in value of property doesn't reflect in cloned object.

```
<?php

class Student {
   public $name;
   public $sem;
   public $spi;
}

$obj = new Student ();

$copy = clone $obj;

$obj->name = "abc";
$obj->sem = "3";
$obj->spi = "9.2";


$copy->name = "xyz";
$copy->sem = "3";
$copy->spi = "8.7";

echo $obj->name .$obj-> sem .$obj->spi."<br>" ;

echo $copy->name. $copy-> sem. $copy->spi."<br>";
```

```
?>

Output:

abc 3 9.2
xyz 3 8.7
```

25) Explain Abstract class with example.

- ✓ An abstract class is a class that contains at least one abstract method.
- ✓ An abstract method is a method that is declared, but not implemented in the code.
- ✓ An abstract class or method is defined with the `abstract` keyword.
- ✓ When inheriting from an abstract class, all methods marked abstract in the parent's class declaration must be defined by the child class.

```
<?php

abstract class Base
{
     abstract function display();
}
class Derived extends Base
{
          function display()
          {
          echo "Derived class";
          }
}

$b1 = new Derived;
$b1->display();
?>

Output:

Derived class
```

Unit 4

26) Explain cookie with example.

- ✓ Cookie: It is a small piece of information that is stored either on client computer's browser memory or in a file on the disk.
- ✓ A Cookie having a name and value.
- ✓ **setcookie () :**You can create a cookie using setcookie().

Syntax:

Setcookie('Name',Value,Expiretime)

Name: Name of the cookie.

Value : value associated with cookie.

Expiretime: It is time when cookie will expire

Example: create 2 cookies name and age and these coolies will expire after one hour.

Code: setcookie.php

```
<?php        setcookie("name","abc", time()+3600);
              setcookie("age", "25", time()+3600);
?>
```

Accessing cookies with php:

```
<?php echo $_COOKIE["name"] . "<br/>";
    echo $_COOKIE["age"] . "<br/>"; ?>
```

27) Explain session with example.

- ✓ Session: A session is a way to store information (in variables) to be used across multiple pages.
- ✓ To create or access session variable you need to start the session at the starting point of script using session_start().
- ✓ Creating a session:
  S_SESSION['Variable name']=Value;
  - ✓ Example: S_SESSION['UserName']="ABC";
  - ✓ Destroy a session:
  Session_destroy();

Example:
Code: session1.php

```
<?php
    session_start();
    $_SESSION['username']='vpmp';
    $_SESSION['password']='123';
?>
```

Code:session2.php

```
<?php
    session_start();
    echo "Welcome :" . $_SESSION['username']."<br>";
    echo "Your Password is:".$_SESSION['password'];
?>
```

28) Explain GET and POST with example.

GET Method	POST Method
data is sent from one page to other in the URL	data is sent from one page to other within the body of the HTTP request.
The GET method, appends name/value pairs to the URL	POST method packages the name/value pair inside body of HTTP request
The length of URL is limited, so it works if there are few parameters	POST method has no size limitations on forms output.
It is insecure because parameters passed on the URL are visible in address field of browser.	It is secure because submitted data are passed through HTTP handler
It can't be used to send binary data, like images or word documents, to server.	It can be used to send ASCII as well as binary data.
Data send by GET method can be accessed using $_GET superglobal variable	data send by POST method can be accessed using $_POST superglobal variable

Example of GET Method:
querystring1.php

```
<html>
<body>
<form action=" querystring2.php" method="GET">
Enter your name: <input type="text" name="name" />
Enter your age: <input type="text" name="age" />
<input type="submit" value="OK" />
</form>
</body>
</html>
```

"querystring2.php"

```
<?php
$Name=$_GET["name"];
$Age=$_GET["age"];
echo "Name is". $Name . "<br>";
echo "Age is". $Age;
?>
```

Output:

Name is:abc

Age is:18

Example of POST Method:
querystring1.php

```
<html>
<body>
<form action=" querystring2.php" method="POST">
Enter your name: <input type="text" name="name" />
Enter your age: <input type="text" name="age" />
<input type="submit" value="OK" />
</form>
</body>
</html>
```

"querystring2.php"

```
<?php
$Name=$_POST["name"];
$Age=$_POST["age"];
echo "Name is". $Name . "<br>";
echo "Age is". $Age;
?>
```

Output:

Name is:abc

Age is:18

29) Explain following form elements: Text Box, Check Box and Radio button

Using Textbox: User can enter text or single line using textbox

Syntax:<input type="text" name="Textname" value="defaultvalue">
Example:

```
<form>
   Enter your Name:<input type="text" name="txtName">
</form>
```

Using Textarea: User can enter multiple line of text like address, feedback, comments etc.

Syntax: <textarea name="txtname" rows="rowsize" cols="columnsize">

</textarea>

Example:

```
<form>
   Enter Address:
        <textarea name="txtarea1" rows="5" cols="20">
        </textarea>
</form>
```

Using Checkbox:

- ✓ It allows you to represent list of options to the users from which user can select none, one or more than one options at a time.
- ✓ It displays as Small Square on web page

Syntax : <input type="checkbox" name="Name" value="Value" [checked]> Text </input>

Example:
<form> <input type="checkbox" name="vehicle" value="Bike"> I have a bike
 <input type="checkbox" name="vehicle" value="Car"> I have a car
</form>

☐ **I have a bike**
☐ **I have a car**

Using Radiobutton:

- ✓ It allows you to represent list of options to the users from which user can select only one options at a time.
- ✓ It displays as small circle on web page

Syntax : <input type="radio" name="Name" value="Value" [checked]> Text </input>

Example
<form> <input type="radio" name="gender" value="male">Male
 <input type="radio" name="gender" value="female">Female
</form>

○ **Male**
○ **Female**

30) Explain action and method attributes of the form.

ACTION: It is the URL of the CGI (Common Gateway Interface) program that is going to accept data from the form, Process it, then Send a response back to browser.
METHOD:GET(default) or POST specifies which HTTP method will be used to send the form's contents to web server.

Unit 5

31) Explain mysql_connect() and mysql_query() function.
mysqli_connect(): This function allows to connect with the MySQL server.
This Function returns a Boolean value TRUE or FALSE.

Syntax:
$varname = mysqli_connect("hostname", "username", "password")

Hostname/servername: Indicates the name of mysql server
Username: Indicates the name of user

Password: Indicates password of the user

Example:

```
<?php
        $con = mysqli_connect ("localhost", "root") ;
        if ($con)
        {       echo  "Connected with MySQL" ;
        }
        else
        {       echo "Can not connect with MySQL" ;
        }
?>
```

mysqli_query(): This function allows you to specify and execute MySQL command on MySQL server.

Syntax:

$varname = mysqli_query("query", connectionname)

query: MySQL command to be executed.

connectionname: Indicates the name of variable that is used at the time of establish connection with mysql.

Example:

```
<?php
        $con = mysqli_connect ("localhost", "root");
        $db = mysqli_selectdb ($con,"21CE3") ;
        $sql="create table Student (Eno integer, Name varchar (20))";
        $cmd=mysqli_query($con,$sql);

        if ($cmd)
        {               echo  "Table created" ;
        }
        else
        {               echo  "Error while creating " ;
        }
?>
```

32) Explain mysql_num_rows () and mysql_fetch_array() functions.

mysqli_num_rows(): This function allows to retrieve total numbers of records from table.

Syntax: mysqli_num_rows($queryresult);

Example:

```
<?php
        $con = mysqli_connect ("localhost" , "root") ;
        mysqli_select_db($con, "21CE3") ;

        $sql = "select * from Student" ;
        $cmd = mysqli_query ($con,$sql)
        $record = mysqli_num_rows ($cmd) ;
```

```
    echo $ records, "Records" ;
    mysqli_close ($con) ;
?>
```

mysqli_fetch_array(): This function allows to retrieve a record from the record set
The record that is returned by this function is in the form of either numeric array, associative array or both.
Syntax:

```
mysqli_fetch_array($variablename , ArrayType);
```

variablename: Indicates the record set that is returned from executing the MySQL command.
Arraytype: Indicates the type of array returned, it has values like

(1)MYSQL_ASSOC
(2)MYSQL_NUM
(3)MYSQL_BOTH

Example:

```
<?php
        $con = mysqli_connect ("localhost" , "root") ;
        mysqli_select_db($con, "21CE3") ;

        $sql = "select * from Student" ;
        $cmd = mysqli_query ($con,$sql)
        $Ans = mysqli_query_fetch_array($cmd,1) ;
        print_r ($Ans) ;

        mysqli_close ($con)
?>
```

Output: Array ([Eno] => 1 [Name] => abc)

33) Explain field modifier in MYSQL.

A. Null/Not null

- ✓ NULL means not compulsory field and NOTNULL means compulsory field.
- ✓ null tells MySQL that it is okay if nothing is stored in the field.
- ✓ not null tells MySQL to require something, *anything*, to be stored there
- ✓ EX: create table student (eno number NOT NULL, name varchar(20));

B. Unique

- ✓ When this parameter is turned on, MySQL makes sure that absolutely no duplicates exist for a particular field.
- ✓ Typically, this is used for only the primary key in your table, but it can be used with any field.

C. auto_increment

- ✓ You can apply a field to be auto incremented by simply adding the auto_increment command when setting up your table.
- ✓ You can also determine what the first number in the count will be, if you don't want it to be 1.

- ✓ EX: create table student (eno number AUTO_INCREMENT, name varchar(20));

D. DEFAULT

- ✓ It allows you to specify default value for field , If one don't enter any value then default is considered.
- ✓ EX: create table student (eno number NOT NULL, name varchar(20), dept varchar(20) DEFAULT 'computer');

34) Write PHP script to create table Employee with fields E_id, E_name, Salary .

```
<?php

    $con = mysqli_connect ("localhost", "root") ;

    mysqli_select_db("Company");

    $sql = "create table Employee(E_id number, E_name  varchar(20),Salary number)";

    mysqli_query($con,$sql);

    mysqli_close($con);

    echo "Table created successfully";

?>
```

35) Design HTML form and write PHP code to insert data in database table through HTML form.

Insert1.php

```
<html>
<body>
    <form action="insert2.php" method="POST">
    <table>
    <tr><td>Enter no:</td><td><input type="text" name="no"></td></tr>
    <tr><td>Enter name:</td><td><input type="text" name="name"></td></tr>
    <tr><td><input type="submit" name="submit" value="submit"></td></tr>
    </table>
    </form>
</body>
</html>
```

Insert2.php

```
<?php
$No=$_POST['no'];
$Name=$_POST['name'];
$con=mysqli_connect ("localhost","root");
mysqli_select_db($con,"21CE3");

$sql="insert into Student values($No,'$Name')";
mysqli_query($con,$sql);
mysqli_close($con);
echo "Record inserted";

?>
```

36) Write a PHP script to create and drop database.

Create Database:

```
<?php
        $con = mysqli_connect ("localhost",root") ;
       $sql = "create database 21CE3"
        mysqli_query ($con,$sql) ;
        echo "Database Created Successfully" ;
        mysqli_close();
?>
```

Drop database:

```
<?php

$con=mysqli_connect("localhost","root");

$sql="drop database 21CE3";

$cmd=mysqli_query($con,$sql);

if($cmd)

        echo "database droped";

else

        echo "error while droping database";

mysqli_close($con);

?>
```

37) Write PHP script to read information from ACCOUNT table (A_no, A_holder, A_balance) and display all these information in tabular format on the webpage.

```
<?php
```

```
$con=mysqli_connect("localhost","root");
mysqli_select_db($con,"Company");
$sql="select * from Account";
$cmd=mysqli_query($con,$sql);
$no=mysqli_num_rows($cmd);
if($no>0)
{
        echo "<table border=1>";
        echo "<tr><td>Account no</td>";
        echo "<td>Account Holder</td>";
        echo "<td>Account Balance</td></tr>";
        while($row=mysqli_fetch_row($cmd))
        {
        echo "<tr><td>$row[0]</td>";
        echo "<td>$row[1]</td>";
        echo "<td>$row[2]</td></tr>";

        }
}
else
{
        echo "No Record Found";
}
mysqli_close($con);
?>
```