PHP ASSIGNMENT

Q1: - Explain OOPS concept with an example.
Soln: -
oops refers to the object oriented programming . The main focus is on objetcs rather then doing things. There are basically following concepts of oops :-

1. Class :- class refers to the blue print that consists of data members and member functions. Each class needs an object to acces the data members and member functions syntax for writting an class starts with an keyword class. A single class can have multiple objects.

```
syntax :- class class name
          {
               data memebers ...;
               member functions---;
           }
```

2. Objects :- Object refers to the instances of classes. As soon as an object is created it occupies a physical memory . In order to create an object we need to have a class associated with it. A object can be associated with only one class.

Syntax for creating objects :-

```
$objectname = new classname ();
```

3. Data Abstraction :- Data abtraction refers to abstract or get a required data from the bulk of data .

4. Data Encapsulation :- Data encapsulation means to encapsulate or wrap the data objects and member functions together in a single class

5. Inheritance :- Inheritance is the method to inherit or transfer the properties(data members or member functions) from one class to another the class that inherits the property of another class is called child class and the class whose properties are bieng inherited are called parent class.

6. Polymorphism :- poly means many and morph means form. Polymorphism is the characterstics of oops that allows a function or member function with same name but diiferent working or functionality

there are basically two main type of polymorphism

1. compile time polymorphism (static polymorphism) :- a complie time polymorpgism is said to be achieved by the means of function overloading , that means function with same but different arguments and functionality

2 Run time polymorphism :- Run time polymorphism is achieved by means of function overriding that means to function with same name , same argument and different defination or functionality

QUES 2 : - Explain Classes & Objects with an example.

Soln : - Class :- class refers to the blue print that consists of data members and member functions. Each class needs an object to acces the data members and member functions syntax for writting an class starts with an keyword class. A single class can have multiple objects.

syntax :- class class name
{
data memebers ...;
member functions---;
}

Objects :- Object refers to the instances of classes. As soon as an object is created it occupies a physical memory . In order to create an object we need to have a class associated with it. A object can be associated with only one class.

Syntax of creating objects :-
$objectname = new classname();

```
<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
  function set_color($color) {
    $this->color = $color;
  }
  function get_color() {
    return $this->color;
  }
}

$apple = new Fruit();
$apple->set_name('Apple');
$apple->set_color('Red');
echo "Name: " . $apple->get_name();
echo "<br>";
echo "Color: " .  $apple->get_color();
?>
```

QUES 3: - What is a Namespace?

Soln: - Namespaces are the way of encapsulating items so that same names can be reused without name conflicts.

- •It can be seen as an abstract concept in many places. It allows declaringthe same functions/classes/interfaces/constant functions in the separate namespace without getting the fatal error.
- •A namespace is a hierarchically labeled code block holding a regular PHP code.
- •A namespace can contain valid PHP code.
- •Namespace affects following types of code: classes (including abstracts and traits), interfaces, functions, and constants.
- •Namespaces are declared using the namespace keyword.

Ques4 :- Explain constructor and destructor with an example.

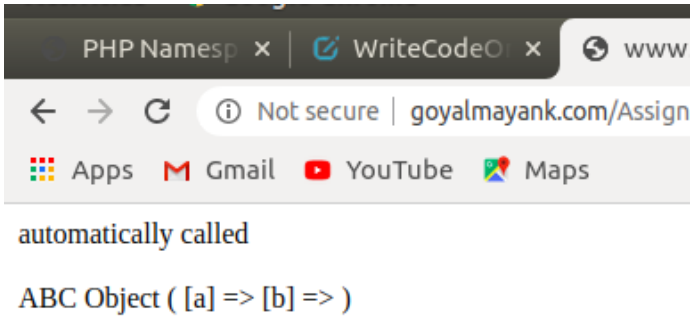
Sol:-

constructors refers to the special functions that plays an very important role in the oops concept. These are the special functions that have same name as the name of the class. There are two types of constructors in php. If we not initalize or declare an constructor then the compiler itself initates the default constructor .

These constructors are initiated as when the object of the class is bieng created.

The constructors that starts with __constructors are also known as magic functions .

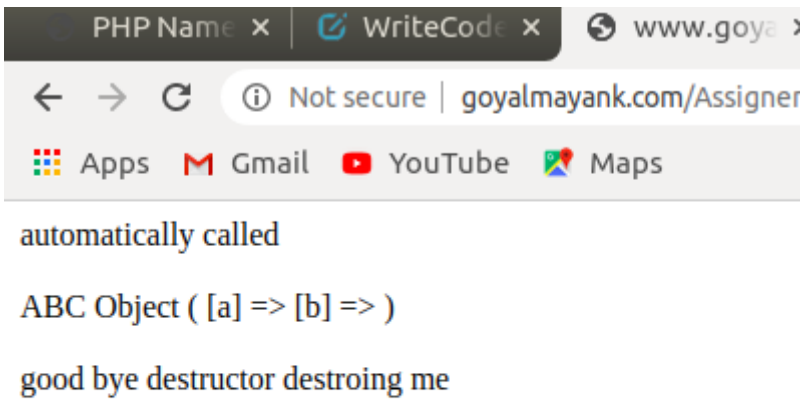
```
<?php
ini_set('display_errors',1);
class ABC {
        public $a;
        public $b;
        public function __construct()
        {
          echo "automatically called<br><br>";
        }
        public function ABC()
        {
        echo "quot;called Second<br><br>";
        }
        public function a()
        {
        echo "A
function";
        }
}
$obj = new ABC()
print_r($obj);
?>
```



Destructor in PHP

destructors are also a special fucntions used in php. The task of desturctors is just reverse of the function of constructors the main task of the destructors is to destruct the object values . The destructor is handled by garbage collectors in php it basically destroys the objects created automatically in case they are not required

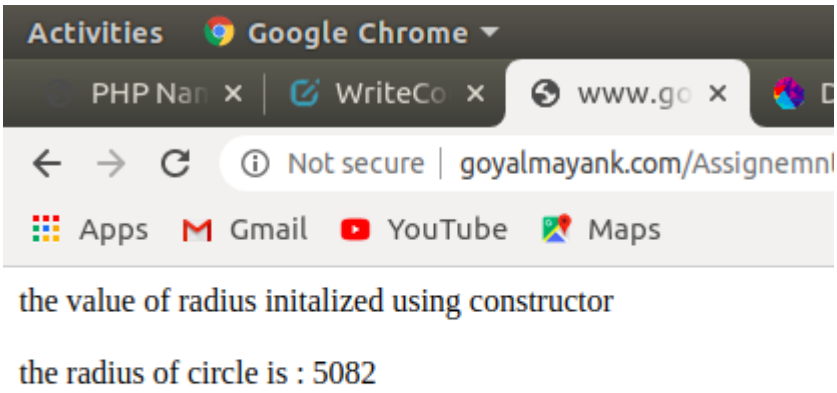
```
<?php
ini_set('display_errors',1);
class ABC {
          public $a;
          public $b;
          public function __construct()
          {
            echo "automatically called<br><br>";
          }
          public function ABC()
          {
          echo "quot;called Second<br><br>";
          }
          public function a()
         {
          echo "A function";
          }
         public function __destruct()
          {
             echo"<br><br>good bye destructor destroing me "; }
}
$obj = new ABC();
print_r($obj);
?>
```



Ques 5 Write a program that initialize class property using constructor.

Soln :-

```
<?php
ini_set('display_errors',1);
class circle
 {
  public $radius;

  function circle()
   {
    $radius =11;
    $area=3*14 *$radius *$radius;
    echo" the value of radius initalized using constructor <br> <br>";
    echo "the radius of circle is   :  ".$area;
   }


}
$obj1 = new circle();

?>
```

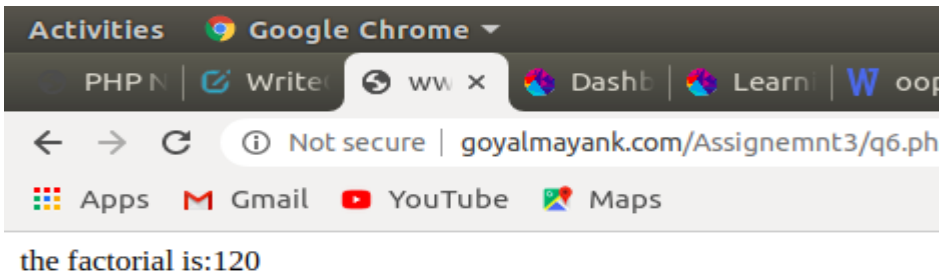


Ques 6:- Write a PHP program to return Factorial value of number in a Object oriented way. Factorial logic should be in separate function.

Soln: -

```
<?php
class fact
{
  protected $value;
  public function __construct($val)
   {
    if(!is_int($val))
     {
       throw new InvalidArgumentException("not a number");
```

```
      }
     $this->value=$val;
    }
     public function factorial()
     {
        $factorial =1;
        for($i =1;$i<=$this->value;$i++)
         {
           $factorial =$factorial * $i;
         }
        return $factorial;
    }
}

$obj1 = new fact(5);
echo "the factorial is:".$obj1->factorial();
```

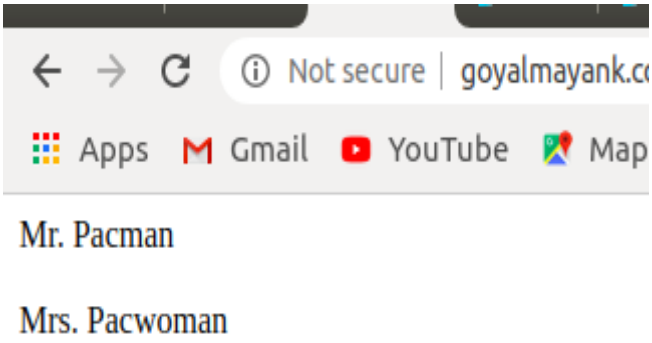


Ques 7 : - Write a class that extends Abstract class.

Abstract class :- An abstract class is always initiated or declared with help of an keyword Abstract. An abstract class contains at least one abstract function . An abstract function is the function that has been only declared but not defined.
If any chlid class inherits the abstract class then all the abstract methods of the abstract class are required to be defined

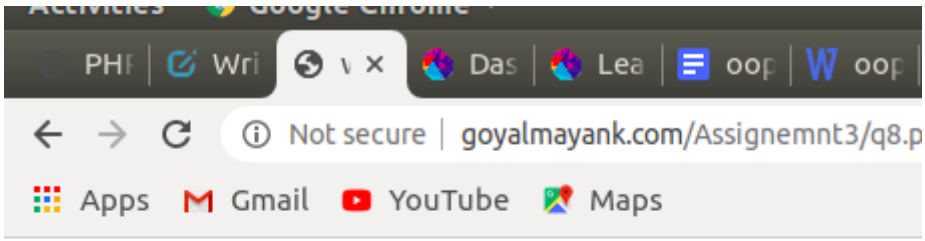
```
<?php
abstract class AbstractClass
{
      abstract protected function prefixName($name);

}

class ConcreteClass extends AbstractClass
{


    public function prefixName($name, $separator = ".") {
       if ($name == "Pacman") {
```

```
            $prefix = "Mr";
        } elseif ($name == "Pacwoman") {
            $prefix = "Mrs";
        } else {
            $prefix = "";
        }
        return "{$prefix}{$separator} {$name}";
    }
}

$class = new ConcreteClass;
echo $class->prefixName("Pacman"), "\n";
echo"<br> <br>";
echo $class->prefixName("Pacwoman"), "\n";
?>
```



Que 8 :- Implement Inheritance in PHP program to display the Message when Parent class method is called and when the Child method is called.

```
<?php
   ini_set('display_errors',1);
 class Parent1
  {
    public function method1()
     {
      echo "<br> <br> method of parent class called";
     }
  }

  class Child1 extends Parent1
   {
     public function method2()
      {
        echo "<br><br> method of child class called ";
       }
    }

  $obj1 = new Child1();
  $obj1->method1();
  $obj1->method2();
?>
```



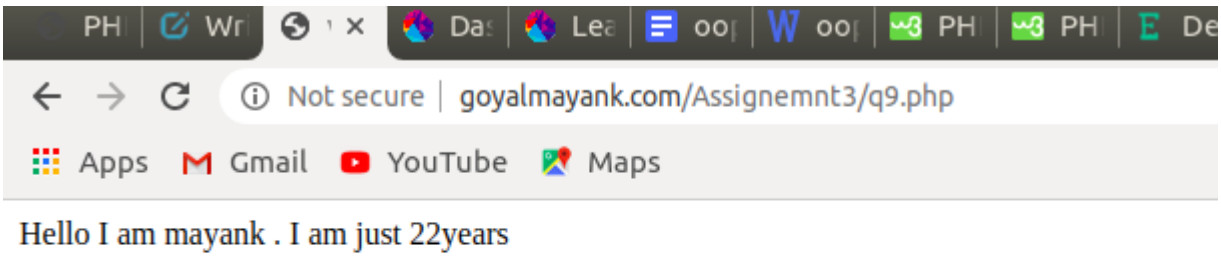
method of parent class called

method of child class called

Ques 9 : What is method chaining explain with an example.

Soln :- method chaining as the name suggests is the method of chaining or sequencing all the methods in a single instructions .

```
<?php
 class Person
  {
    private $name ="";
    private $age ="";

    public function setname($name="")
     {
       $this->name=$name;
       return $this;
     }
     public function setage($age="22")
       {
         $this->age=$age;
         return $this;
       }
     public function display()
      {
        echo " Hello I am ".$this->name." . I am just ".$this->age ."years";
      }
  }

  $obj = new Person();
  $obj->setname("mayank")->setage("22")->display();
?>
```



Ques 10 : What is **$this** keyword.

Soln:- $this :- $this keyword always point towards the current object in action . $this refers to the current object instance . We use -> to point towards the current objects member functions and data members

```
<?php
class Car {


    public $comp;
    public $color = 'beige';
    public $hasSunRoof = true;


    public function hello()
    {
      return "Beep I am a <i>" . $this -> comp .
          "</i>, and I am <i>" . $this -> color;
    }
}
$bmw = new Car();
$mercedes = new Car();

$bmw -> color = 'blue';
$bmw -> comp = "BMW";
$mercedes -> comp = "Mercedes Benz";


echo $bmw -> hello();
?>
```

QUES 11:- Write program to explain Multi Level Inheritance.

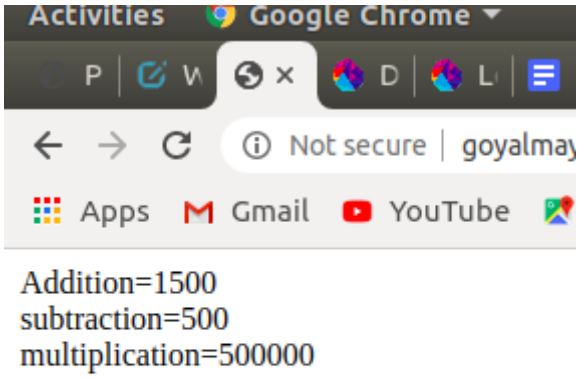
Soln:- multi level inheritance means to inherit a class from a class that is itself inherited from some other class.
In other words we can say that it creates a ledder of inherited classes where a class is inherited from some other class and is parent of another class

class A ----- parent of ------> class B -------- parent of ------>class C

example

```
<?php

class BaseClass

{

function add()

{

$x=1000;

$y=500;

$add=$x+$y;

echo "Addition=".$add."<br/>";

}

}


class chld extends BaseClass

{

function sub()

{

$x=1000;

$y=500;

$sub=$x-$y;

echo "subtraction=".$sub."<br/>";

 }

}
```

```
class Nestedchld extends chld
{
function mult()
{
$x=1000;
$y=500;
$mult=$x*$y;
echo "multiplication=".$mult."<br/>";
}
}

class show extends Nestedchld
{
function __construct()
{
parent::add();
parent::sub();
parent::mult();
}
}
$obj= new show();
?>
```



Q12 :- Develop prototype for flight-booking service. Here, we don't own any flights. we are just a service provider mediator.
Define two methods :
a. checkFlightsAvailability() - to check available flights from two vendors like "Air India", "IndiGO".
b. bookFlights() - to book a seat from respective vendor.

Soln :- class Flight

```
{

 protected    $vendor1 =”Air India” ;
 protected    $vendor2=”Indigo”;
  public function checkFlightsAvailability( $ total_ no_of_ flights  )
   {
       $count1 =0 ;
       $count2=0;
       for($i=1;$i<=$total_no_of_flights ;$i++)
        {
            if($vendor_name ==$vendor 1)
             { $count1++;}
            elseif($vendor_name== $vendor2)
              { $count2++;}
         }
       echo “ flight available for “.$vendor1.”are”.$count1;
       echo “ flight available for “.$vendor2.”are”.$count2;
  }
  public function bookFlights( $choice, $available_seats,$seats_to_be _booked);
   {
      $vendor_name =$choice;
       if($available_seats ==0)
       {
          echo” all seats are full”;
       }
       else
        {
```

```
                    if ($seats_to_be_booked > $available_seats)
                    {
                            echo " only ".$seats_to_be_booked -$available_seats. "seats are available";
                     }
                else
                     {
                            $available_seats-=$seats_to_be_booked;
                            echo" seats bookesd succesfully";
                       }
}
```

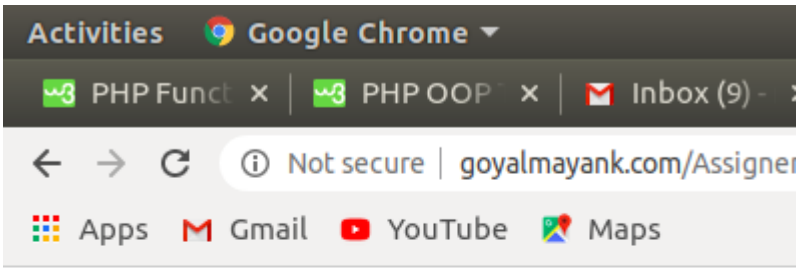
Ques 13:- Define three traits and use it in class and override it's function.

Soln :- traits are used to declare methods that can be used in multiple classes. Traits can have methods and abstract methods that can be used in multiple classes, and the methods can have any access modifier

creating traits and calling it

```
<?php
  trait first
   {
      public function show()
       {
          echo "first ----> show ";
        }
    }

   trait second
    {
       public function display()
        {
          echo "second ----> display";
         }
        public function show(
```

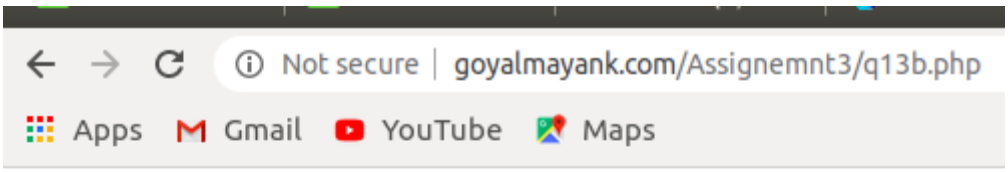
```
    }
   trait third
    {
      public function give()
        {
         echo " thid ----> give";
        }
     }
  class C1
   {
     use first,second,third;
    }
   $obj1 = new  C1();
   $obj1->display();
   $obj1->give();
   $obj1->show();
?>
```



second ----> display thid ----> givefirst ----> show

Overriding the functions of traits :-

```php
<?php
  trait first
   {
     public function show()
      {
         echo "<br><br>first ----> show ";
       }
    }

   trait second
    {
      public function display()
       {
        echo "<br><br>second ----> display";
       }

    }
   trait third
     {
       public function give()
         {
          echo "<br><br> thid ----> give";
         }
      }
  class C1
   {
     use first,second,third;
   }
  class C2
    {
       use first,second,third;
       public function show()
       {
         echo" <br><br>first <----> shoow <-----> overloaded";
       }
       public function display()
       {
          echo"<br><br> second <-----> display<---->overloaded";
        }
     }
    $obj1 = new  C1();
    $obj1->display();
    $obj1->give();
    $obj1->show();
    $obj2 = new  C2();
    $obj2->display();
    $obj2->give();
    $obj2->show();
?>
```



second ----> display

thid ----> give

first ----> show

second <-----> display<---->overloaded

thid ----> give

first <----> shoow <-----> overloaded