1)Upgrade PHP version to PHP 7.4

```
eram@eram:~$ php -version
PHP 7.2.24-0ubuntu0.18.04.2 (cli) (built: Jan 13 2020 18:39:59) ( NTS )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend Technologies
    with Zend OPcache v7.2.24-0ubuntu0.18.04.2, Copyright (c) 1999-2018, by Zend Technologies
```

```
eram@eram:~$ sudo apt -y install php7.4
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libapache2-mod-php7.4 libpcre2-8-0 php-common php7.4-cli php7.4-common php7.4-json php7.4-opcache php7.4-readline
Suggested packages:
```

```
eram@eram:~$ php -version
PHP 7.4.3 (cli) (built: Feb 21 2020 17:50:20) ( NTS )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
    with Zend OPcache v7.4.3, Copyright (c), by Zend Technologies
```

2) What are the advantages of spread operator over array_merge in PHP?

1. Spread operator has a better performance than array_merge. It's because spread operator is a language structure while array_merge is just a function call. Also the compile time optimization functions very well for constant arrays when using Spread Operator.

2. array_merge only supports arrays, while spread operator supports arrays as well as objects implementing Traversable.

3)Write the output of these;

```
$arr1 = [1, 2, 3];
$arr2 = [...$arr1];
$arr3 = [0, ...$arr1];
$arr4 = array(...$arr1, ...$arr2, 111);
```

what will be the output of $arr2, $arr3, $arr4.

Result:

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
)
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
)
```

```
)
Array
(
    [0] => 0
    [1] => 1
    [2] => 2
    [3] => 3
)
```

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
    [3] => 1
    [4] => 2
    [5] => 3
    [6] => 111
)
```

4)Write a program to print an array in which output is returned by function using the spread operator.

fri4.php > ...

```php
<?php
    function getEven()
    {
        $numbers = [0,1,2,3,4,5,6,7,8,9,10];
        $even = [];
        foreach($numbers as $n)
        {
            if(($n%2)==0)
            {
                $even=[...$even,$n];
            }
        }
        return $even;
    }
    print_r(getEven());
?>
```

```
Array
(
    [0] => 0
    [1] => 2
    [2] => 4
    [3] => 6
    [4] => 8
    [5] => 10
)

```

5) What is dependency Injection?

Soln:- Dependency Injection:-

Dependency Injection in simple way is the design pattern that helps avoiding hard copied dependencies for some piece of codes or hardwares .

fri5.php > ...

```php
<?php
class Programmer {
    private $skills;
    public function __construct($skills){
        $this->skills = $skills;
    }
    public function showSkills(){
        return $this->skills;
    }
}
$skills1 = array("PHP", "JQUERY", "HTML", "CSS");
$eram = new Programmer($skills1);
print_r($eram->showSkills());
//echo "<br>";

$skills2 = array("Java","C","C++","Spring Boot");
$sadaf = new Programmer($skills2);
print_r($sadaf->showSkills());
//echo "<br>";
?>
```

```
Array
(
    [0] => PHP
    [1] => JQUERY
    [2] => HTML
    [3] => CSS
)
Array
(
    [0] => Java
    [1] => C
    [2] => C++
    [3] => Spring Boot
)
```

6)Write an example of a factory class where we pass 4 different car models and it returns price and builds year of the car.

fri6.php > ...

```php
<?php
    // CarFactory Class
    class CarFactory {
        public static function make($model=null)
        {
            echo $model.' in Progress...<br>';
            switch(strtolower($model))
            {
                case 'suv':
                    return new SUVCar();
                case 'sedan':
                    return new SEDANCar();
                case 'convertible':
                    return new CONVERTIBLECar();
                case 'pickup':
                    return new PICKUPCar();
                default:
                    return '404 : Car Not Found';
                    break;
            }
        }
    }

    // CAR Interface
    interface Car {
        function getModel();
        function getPrice();
        function getBuildYear();
    }
```

```php
// Concrete Car Classes
// SUV Car
class SUVCar implements Car
{
    private $price;
    private $buildYear;
    private $model;
    public function __construct()
    {
        $this->model = 'SUV';
        $this->price='20 lac Rupees';
        $this->buildYear='2019';
    }
    public function getModel()
    {
        return $this->model;
    }
    public function getPrice()
    {
        return $this->price;
    }
    public function getBuildYear()
    {
        return $this->buildYear;
    }
}
```

```php
// SEDAN Car
class SEDANCar implements Car
{
    private $price;
    private $buildYear;
    private $model;
    public function __construct()
    {
        $this->model = 'SEDAN';
        $this->price='10 lac Rupees';
        $this->buildYear='2020';
    }
    public function getModel()
    {
        return $this->model;
    }
    public function getPrice()
    {
        return $this->price;
    }
    public function getBuildYear()
    {
        return $this->buildYear;
    }
}
```

```php
// CONVERTIBLE Car
class CONVERTIBLECar implements Car
{
    private $price;
    private $buildYear;
    public function __construct()
    {
        $this->model = 'CONVERTIBLE';
        $this->price='80 lac Rupees';
        $this->buildYear='2018';
    }
    public function getModel()
    {
        return $this->model;
    }
    public function getPrice()
    {
        return $this->price;
    }
    public function getBuildYear()
    {
        return $this->buildYear;
    }
}
```

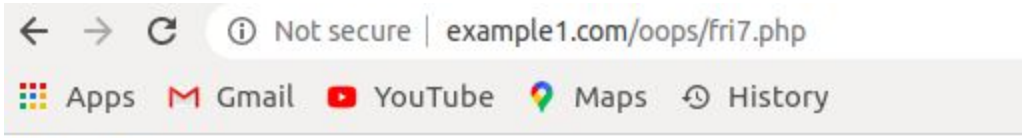
```php
// PICKUP Car
class PICKUPCar implements Car
{
    private $price;
    private $buildYear;
    public function __construct()
    {
        $this->model = 'PICKUP';
        $this->price='18 lac Rupees';
        $this->buildYear='2019';
    }
    public function getModel()
    {
        return $this->model;
    }
    public function getPrice()
    {
        return $this->price;
    }
    public function getBuildYear()
    {
        return $this->buildYear;
    }
}
```

```php
    }
}
// Program Start
$suvCar = CarFactory::make('sedan');
if($suvCar->getPrice()!=null)
{
    echo '<br>Car Model : '.$suvCar->getModel();
    echo '<br>Price : '.$suvCar->getPrice();
    echo '<br>Build Year : '.$suvCar->getBuildYear();
}
?>
```

7)Give an example of singleton class.

```php
class PrimeMinister
{
    private static $currentPM;
    private $name;
    private final function __construct($name)
    {
        $this->name=$name;
    }
    public static function getPM($name)
    {
        if (!isset(self::$currentPM)) {
            self::$currentPM = new PrimeMinister($name);
        }
        return self::$currentPM;
    }
    public function getName()
    {
        return $this->name;
    }
}

$pm1 = PrimeMinister::getPM('Modi');
$pm2 = PrimeMinister::getPM('AAP');
echo $pm1->getName();
echo $pm2->getName();
```



ModiModi

8)What are the benefits of following design patterns?

Design patterns are commonly defined as time-tested solutions to recurring design problems. The term refers to both the description of a solution that you can read, and an instance of that solution as used to solve a particular problemDesign patterns have two major benefits:-

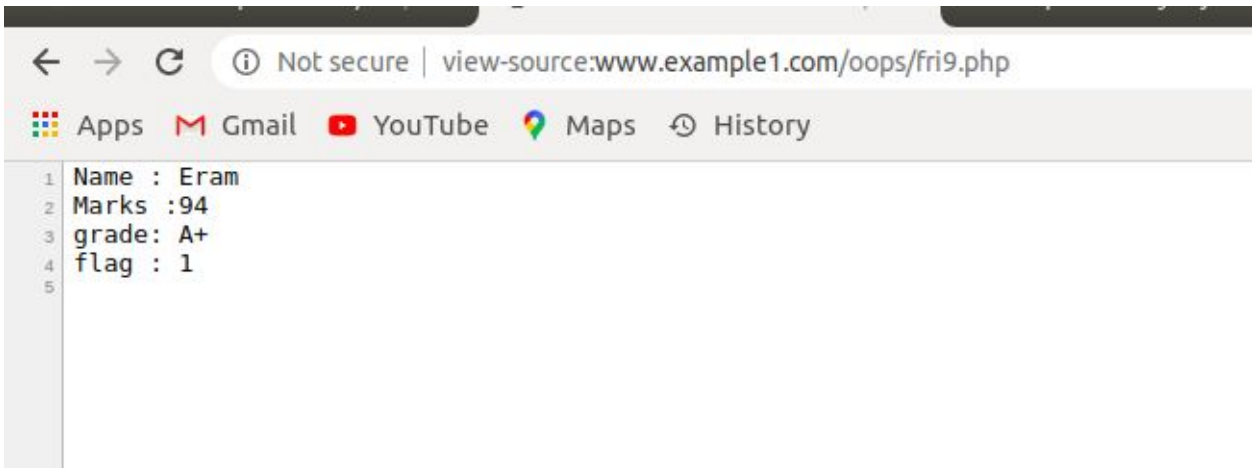
1. they provide you with a way to solve issues related to software development using a proven solution. The solution facilitates the development of highly cohesive modules with minimal coupling. They isolate the variability that may exist in the system requirements, making the overall system easier to understand and maintain.

2.Design patterns make communication between designers more efficient. Software professionals can immediately picture the high-level design in their heads when they refer the name of the pattern used to solve a particular issue when discussing system design.

9)Define a class with type properties.

fri9.php > ...

```php
<?php
class Student
{
public String $name;
var int $marks;
public String $grade;
public bool $flag;
public function __Construct(String $name ,int $marks, String $grade ,bool $flag)
{
$this->name=$name;
$this->marks=$marks;
$this->grade=$grade;
$this->flag=$flag;
}
}
$obj=new student("Eram",94,"A+",true);echo"Name : "."$obj->name \n";
echo "Marks :".$obj->marks."\n";
echo "grade: "."$obj->grade \n";
echo "flag : "."$obj->flag \n";
?>
```



10)Write a function using arrow function array_map.

fri10.php > ...

```php
<?php
    $a = [1, 2, 3, 4, 5];
    $b = array_map(function ($n){
        return ($n * $n * $n);
    }, $a);
    print_r($b);
    echo '<br>With Arrow Function : <br>';
    $a = [1, 2, 3, 4, 5];
    $b = array_map(fn($n)=>($n * $n), $a);
    print_r($b);
?>
```

```
Array
(
    [0] => 1
    [1] => 8
    [2] => 27
    [3] => 64
    [4] => 125
)
<br>With Arrow Function : <br>Array
(
    [0] => 1
    [1] => 4
    [2] => 9
    [3] => 16
    [4] => 25
)
```