Evaluation Parameters:

· PSR standards

· Security Vulnerabilities

· Database

· Logics

**Question 1:**

**Arrays** play an important role in programming. They are used to group similar data types. An array has a common name and each individual element is accessed by an index.

In this exercise we're working with an array of 10 integers, as follows: **[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]**. 0 is the first index and 9 is the last index of the array.

Write a function that receives two integers as parameters. **The function returns the sum of the elements in the array found between those two integers.**

For example, if we use 30 and 60 as parameters, the function should return 180.

A few additional requirements:

* The two integers passed to the function should be positive; if not, the function should return -1.
* Validate that the first integer is lower than the second integer. If not, the function should return 0.
* If the first integer is in the array, and the second is above 100, for example 90 and 120, then the function should return the sum of those integers that are within the array and in between the range given. In this case, that would be 190.
* If both integers are not found in the array, for example 110 and 120, the function should return 0.

<?php

function sum\_array\_elements($num1, $num2){

$array = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100];

$chk\_flag = validate\_numbers($array, $num1, $num2);

if($chk\_flag == 1){

$sum = 0;

$start\_key = array\_search($num1, $array);

if($num2 > $array[9])

{

$end\_key = 9;

}

else{

$end\_key = array\_search($num2, $array);

}

for($x=$start\_key; $x<=$end\_key; $x++){

$sum += $array[$x];

}

return $sum;

}

else{

return $chk\_flag;

}

}

function validate\_numbers($array, $num1, $num2)

{

$return\_value = 1;

if($num1 < 0 || $num2 < 0)

{

$return\_value = -1;

}

else if((!in\_array($num1, $array) && !in\_array($num2, $array)) || $num1 > $num2)

{

$return\_value = 0;

}

return $return\_value;

}

print sum\_array\_elements(60,120);

?>

**Question 2:**

The user interface contains two types of user input controls: *TextInput*, which accepts all texts and *NumericInput*, which accepts only digits.

Implement the class *TextInput* that contains:

* Public function *add($text)* - adds the given text to the current value.
* Public function *getValue()* - returns the current value (string).

Implement the class *NumericInput* that:

* Inherits from *TextInput.*
* Overrides the *add* method so that each non-numeric text is ignored.

<?php

class TextInput

{

protected $str = '';

public function add($text) {

$this->str .= $text;

}

public function getValue() {

return $this->str;

}

}

class NumericInput extends TextInput

{

public function add($text) {

if ( is\_numeric($text)) {

$this->str .= $text;

}

}

}

?>

**Question 3:**

The *SalaryTable* class records the salary of each employee in a Year. After each month, the salary records their salary with the *recordSalary*function.

The employee rank in the company is calculated using the following logic:

* The employee with the highest salary is ranked first (rank 1). The employee with the lowest salary is ranked last.
* If two employees are tied on salary, then the employee who has paid highest average salary is ranked higher.
* If two employees are tied on salary and highest average salary, then the employee who was first joined the company is ranked higher.

Please show examples for MySQL queries to insert & fetch the data from Database.

-- create

CREATE TABLE EMPLOYEE (

Id INTEGER PRIMARY KEY,

Emp\_Name varchar(255) NOT NULL,

Emp\_Add varchar(255) NOT NULL,

Emp\_Phone varchar(255) NOT NULL,

Dept\_No TEXT NOT NULL,

Dept\_Name varchar(255) NOT NULL,

Salary INTEGER NOT NULL

);

-- insert

INSERT INTO EMPLOYEE VALUES (0001, 'Ramesh', 'GNoida','9855498465', '3445', 'Sales',25000);

INSERT INTO EMPLOYEE VALUES (0002, 'Suresh', 'GNoida','98565498465', '0072', 'Sales',75000);

INSERT INTO EMPLOYEE VALUES (0003, 'Rajesh', 'GNoida','9855497865', '2324', 'Sales',28000);

INSERT INTO EMPLOYEE VALUES (0004, 'Shyamu', 'BSB','9853698465', '8883', 'Sales',35000);

INSERT INTO EMPLOYEE VALUES (0005, 'Ramu', 'BSB','9855498235', '74568', 'Sales',96000);

INSERT INTO EMPLOYEE VALUES (0006, 'Mahesh', 'GNoida','9851678465', '1238', 'Sales',25000);

INSERT INTO EMPLOYEE VALUES (0007, 'Chaman', 'BSBS','9856723465', '7634', 'D10',51000);

highest salary emp

SELECT \* FROM employee GROUP BY salary ORDER BY salary DESC LIMIT 0,1;

lowest salary emp

SELECT \* FROM employee GROUP BY salary ORDER BY salary ASC

LIMIT 0,1;

Implement the employee*Rank* function that returns the employee at the given rank.

**Question 4:**

Write a javascript program that accepts text input from a text element. Use comma as a separator segregate all the numbers & characters from the text element into javascript array & show them as two different list in the form on UL LI list.

<html>

<body>

<div id = "char\_html'></div>

<div id = "digit\_html'></div>

</body>

</html>

<script>

var val = 'abc123456';

var str = val.split('');

var characters = '';

var digits = '';

for (var i = 0; i < str.length; i++) {

if(isNaN(str[i]))

{

characters += "<li>" + str[i] + "</li>";

}

else

{

digits += "<li>" + str[i] + "</li>";

}

}

$('#char\_html').html("<ul>"+characters+"</ul>");

$('#digit\_html').html("<ul>"+digits+"</ul>");

</script>

**Question 5:**

Write a php program that will validate the form elements with following elements.

Text Input: Should only allow alphanumeric.

Int Input: should only allow integer values.

String: Can accept all characters except \*,|\)

Email: Can only accept company emails. Not public emails.

WYSIWYG: Needs to validate for any known security attacks.

validate\_form\_elements($\_POST, $value)

{

$error = array();

if(!empty($\_POST['text']) && ctype\_alnum($value))

{

$error['text'] = "Only alphanumeric characters are allowed!";

}

if(!empty($\_POST['int']) && ($value < 0 || !filter\_var($value, FILTER\_VALIDATE\_INT))

{

$error['int'] = "Only integer values are allowed!";

}

if(!empty($\_POST['string']) && preg\_match('/[\*()\/\/,|]/', $value))

{

$error['string'] = "Special characters are not allowed!";

}

if(!empty($\_POST['email']) && (!preg\_match("/(hotmail|gmail|yahoo)/i", $value) || !filter\_var($value, FILTER\_VALIDATE\_EMAIL)))

{

$error['email'] = "Only accept company emails. Not public emails!";

}

return $error;

}