**1.Create a HTML page with Java Script (built in functions-(date, display, create date object etc.), user defined Functions- (passing parameter, recursive function))**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Date and Recursive Function Example</title>

<script>

// Function to display current date and time

function displayCurrentDateTime() {

let currentDate = new Date();

let dateTimeString = currentDate.toLocaleString();

document.getElementById("datetime").innerText = dateTimeString;

}

// Recursive function to calculate factorial

function factorial(n) {

if (n === 0 || n === 1) {

return 1;

} else {

return n \* factorial(n - 1);

}

}

// Function to handle form submission

function calculateFactorial() {

let num = parseInt(document.getElementById("numInput").value);

let result = factorial(num);

document.getElementById("factorialResult").innerText = `Factorial of ${num} is ${result}`;

}

</script>

</head>

<body>

<h1>Date and Recursive Function Example</h1>

<p>Click the button below to display the current date and time:</p>

<button onclick="displayCurrentDateTime()">Display Date and Time</button>

<p id="datetime"></p>

<hr>

<h2>Factorial Calculation</h2>

<p>Enter a number to calculate its factorial:</p>

<form onsubmit="event.preventDefault(); calculateFactorial();">

<input type="number" id="numInput" required>

<button type="submit">Calculate Factorial</button>

</form>

<p id="factorialResult"></p>

</body>

</html>

**2: Create simple Java Applet programs.(Add two numbers, Factorial of a given no , Check if a string is palindrome or not)**

**JavaScript Examples**

**1. Adding Two Numbers (JavaScript)**

html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Add Two Numbers - JavaScript</title>

<script>

function addNumbers() {

let num1 = parseInt(document.getElementById("num1").value);

let num2 = parseInt(document.getElementById("num2").value);

let sum = num1 + num2;

document.getElementById("result").innerText = `Sum: ${sum}`;

}

</script>

</head>

<body>

<h2>Add Two Numbers</h2>

<label for="num1">Number 1:</label>

<input type="number" id="num1"><br><br>

<label for="num2">Number 2:</label>

<input type="number" id="num2"><br><br>

<button onclick="addNumbers()">Add</button><br><br>

<p id="result"></p>

</body>

</html>

**2. Factorial of a Given Number (JavaScript)**

html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Factorial of a Number - JavaScript</title>

<script>

function calculateFactorial() {

let num = parseInt(document.getElementById("num").value);

let result = factorial(num);

document.getElementById("result").innerText = `Factorial of ${num} is ${result}`;

}

function factorial(n) {

if (n === 0 || n === 1) {

return 1;

} else {

return n \* factorial(n - 1);

}

}

</script>

</head>

<body>

<h2>Factorial of a Number</h2>

<label for="num">Enter a number:</label>

<input type="number" id="num"><br><br>

<button onclick="calculateFactorial()">Calculate Factorial</button><br><br>

<p id="result"></p>

</body>

</html>

**3. Check if a String is Palindrome (JavaScript)**

html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Palindrome Check - JavaScript</title>

<script>

function checkPalindrome() {

let inputString = document.getElementById("inputString").value;

let result = isPalindrome(inputString);

if (result) {

document.getElementById("result").innerText = `${inputString} is a palindrome.`;

} else {

document.getElementById("result").innerText = `${inputString} is not a palindrome.`;

}

}

function isPalindrome(str) {

str = str.toLowerCase().replace(/[^a-zA-Z0-9]/g, ''); // Remove non-alphanumeric characters and convert to lowercase

let len = str.length;

for (let i = 0; i < len/2; i++) {

if (str[i] !== str[len - 1 - i]) {

return false;

}

}

return true;

}

</script>

</head>

<body>

<h2>Palindrome Check</h2>

<label for="inputString">Enter a string:</label>

<input type="text" id="inputString"><br><br>

<button onclick="checkPalindrome()">Check Palindrome</button><br><br>

<p id="result"></p>

</body>

</html>

**Java Applet Examples**

**1. Adding Two Numbers (Java Applet)**

java

import java.applet.Applet;

import java.awt.Graphics;

import java.awt.TextField;

import java.awt.Button;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class AddNumbers extends Applet implements ActionListener {

TextField num1, num2;

Button addBtn;

int result = 0;

public void init() {

num1 = new TextField(10);

num2 = new TextField(10);

addBtn = new Button("Add");

addBtn.addActionListener(this);

add(num1);

add(num2);

add(addBtn);

}

public void actionPerformed(ActionEvent ae) {

int n1 = Integer.parseInt(num1.getText());

int n2 = Integer.parseInt(num2.getText());

result = n1 + n2;

repaint(); // Calls paint() method to redraw the applet

}

public void paint(Graphics g) {

g.drawString("Sum: " + result, 50, 50);

}

}

**2. Factorial of a Given Number (Java Applet)**

java

import java.applet.Applet;

import java.awt.Graphics;

import java.awt.TextField;

import java.awt.Button;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class FactorialApplet extends Applet implements ActionListener {

TextField numField;

Button calculateBtn;

int result = 0;

public void init() {

numField = new TextField(10);

calculateBtn = new Button("Calculate Factorial");

calculateBtn.addActionListener(this);

add(numField);

add(calculateBtn);

}

public void actionPerformed(ActionEvent ae) {

int num = Integer.parseInt(numField.getText());

result = factorial(num);

repaint();

}

private int factorial(int n) {

if (n == 0 || n == 1) {

return 1;

} else {

return n \* factorial(n - 1);

}

}

public void paint(Graphics g) {

g.drawString("Factorial: " + result, 50, 50);

}

}

**3. Check if a String is Palindrome (Java Applet)**

java

import java.applet.Applet;

import java.awt.Graphics;

import java.awt.TextField;

import java.awt.Button;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class PalindromeCheck extends Applet implements ActionListener {

TextField inputField;

Button checkBtn;

String result = "";

public void init() {

inputField = new TextField(20);

checkBtn = new Button("Check Palindrome");

checkBtn.addActionListener(this);

add(inputField);

add(checkBtn);

}

public void actionPerformed(ActionEvent ae) {

String input = inputField.getText();

if (isPalindrome(input)) {

result = "Palindrome!";

} else {

result = "Not a Palindrome!";

}

repaint();

}

private boolean isPalindrome(String str) {

int left = 0;

int right = str.length() - 1;

while (left < right) {

if (str.charAt(left) != str.charAt(right)) {

return false;

}

left++;

right--;

}

return true;

}

public void paint(Graphics g) {

g.drawString("Result: " + result, 50, 50);

}

}

**How to Run Java Applets:**

1. **Compile**: Compile each Java file (AddNumbers.java, FactorialApplet.java, PalindromeCheck.java) using javac command.

bash

javac AddNumbers.java

javac FactorialApplet.java

javac PalindromeCheck.java

1. **Create HTML File**: Create an HTML file to embed and run each applet. Here's a sample HTML file for AddNumbers applet:

html

<!DOCTYPE html>

<html>

<head>

<title>Add Numbers Applet</title>

</head>

<body>

<applet code="AddNumbers.class" width="300" height="200">

Your browser does not support Java applets.

</applet>

</body>

</html>

Replace code attribute with FactorialApplet.class and PalindromeCheck.class for other applets respectively.

1. **Run**: Open the HTML file in a browser that supports Java applets (modern browsers may need additional setup to run Java applets due to security reasons).

**3: Write a PHP program to demonstrate simple tasks (such as: to display ‘hello’) and basic PHP syntax (includes defining variable, constant, data type and basic arithmetic Operations with operators) .**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>PHP Simple Tasks and Syntax</title>

</head>

<body>

<h1>PHP Simple Tasks and Syntax</h1>

<?php

// Displaying 'hello'

echo "<p>Hello, World!</p>";

// Defining variables

$name = "John Doe";

$age = 30;

$height = 175.5;

$isStudent = true;

// Displaying variables

echo "<p>Name: " . $name . "</p>";

echo "<p>Age: " . $age . "</p>";

echo "<p>Height: " . $height . " cm</p>";

echo "<p>Is Student: " . ($isStudent ? 'Yes' : 'No') . "</p>";

// Defining constants

define('PI', 3.14159);

define('GREETING', 'Welcome to PHP!');

// Displaying constants

echo "<p>Value of PI: " . PI . "</p>";

echo "<p>Greeting Message: " . GREETING . "</p>";

// Basic arithmetic operations

$num1 = 10;

$num2 = 5;

echo "<h2>Arithmetic Operations</h2>";

echo "<p>Addition: " . ($num1 + $num2) . "</p>";

echo "<p>Subtraction: " . ($num1 - $num2) . "</p>";

echo "<p>Multiplication: " . ($num1 \* $num2) . "</p>";

echo "<p>Division: " . ($num1 / $num2) . "</p>";

echo "<p>Modulus: " . ($num1 % $num2) . "</p>";

?>

</body>

</html>

**Explanation:**

1. **Displaying 'hello'**:
   * echo "<p>Hello, World!</p>"; prints "Hello, World!" in paragraph tags <p>.
2. **Defining and Displaying Variables**:
   * Variables like $name, $age, $height, and $isStudent are defined and then echoed to display their values.
3. **Defining and Displaying Constants**:
   * Constants are defined using define('NAME', value); syntax. Constants PI and GREETING are defined and then echoed to display their values.
4. **Basic Arithmetic Operations**:
   * Variables $num1 and $num2 are used for demonstrating basic arithmetic operations (+, -, \*, /, %). Results of these operations are displayed using echo.
5. **HTML and PHP Integration**:
   * PHP code is embedded within HTML tags using <?php ... ?> to execute PHP logic and display output within an HTML page.

To run this PHP program:

* Save the code in a .php file (e.g., index.php).
* Place it in a web server's directory (like XAMPP, WAMP, or any PHP-enabled server).
* Open the file in a web browser (http://localhost/index.php) to see the output.

**4.Write a PHP program to demonstrate condition/decision making and loop statement**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>PHP Condition and Loop Statements</title>

</head>

<body>

<h1>PHP Condition and Loop Statements</h1>

<?php

// Demonstrating condition/decision making

$num = 15;

echo "<h2>Condition/Decision Making</h2>";

if ($num > 10) {

echo "<p>$num is greater than 10.</p>";

} elseif ($num == 10) {

echo "<p>$num is equal to 10.</p>";

} else {

echo "<p>$num is less than 10.</p>";

}

// Demonstrating loop statements

echo "<h2>Loop Statements</h2>";

// While loop

echo "<h3>While Loop:</h3>";

$i = 1;

while ($i <= 5) {

echo "Iteration $i<br>";

$i++;

}

// For loop

echo "<h3>For Loop:</h3>";

for ($j = 1; $j <= 5; $j++) {

echo "Iteration $j<br>";

}

// Do-while loop

echo "<h3>Do-While Loop:</h3>";

$k = 1;

do {

echo "Iteration $k<br>";

$k++;

} while ($k <= 5);

?>

</body>

</html>

**Explanation:**

1. **Condition/Decision Making**:
   * The variable $num is set to 15. Depending on its value, different messages are displayed using if, elseif, and else statements.
2. **Loop Statements**:
   * **While Loop**: Iterates from 1 to 5 using a while loop. It initializes $i to 1, checks if $i is less than or equal to 5, and increments $i in each iteration.
   * **For Loop**: Iterates from 1 to 5 using a for loop. It initializes $j to 1, checks if $j is less than or equal to 5, increments $j in each iteration, and executes the loop body.
   * **Do-While Loop**: Iterates from 1 to 5 using a do-while loop. It initializes $k to 1, executes the loop body, increments $k, and then checks if $k is less than or equal to 5 to continue the loop.
3. **HTML and PHP Integration**:
   * PHP code is embedded within HTML tags using <?php ... ?> to execute PHP logic and display output within an HTML page.

To run this PHP program:

* Save the code in a .php file (e.g., index.php).
* Place it in a web server's directory (like XAMPP, WAMP, or any PHP-enabled server).
* Open the file in a web browser (http://localhost/index.php) to see the output.

This PHP program demonstrates condition/decision making with if, elseif, and else statements, and loop statements (while, for, do-while) to iterate and display output in an HTML page.

**5: Write a PHP program to demonstrate mixing of decisions and looping statements with HTML script.**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>PHP Mixing Decisions and Looping with HTML</title>

<style>

.odd {

background-color: #f0f0f0;

}

.even {

background-color: #cccccc;

}

</style>

</head>

<body>

<h1>PHP Mixing Decisions and Looping with HTML</h1>

<?php

// Example array

$numbers = [1, 2, 3, 4, 5];

// Using decision and looping statements

foreach ($numbers as $num) {

if ($num % 2 == 0) {

$class = 'even';

} else {

$class = 'odd';

}

echo "<div class=\"$class\">Number: $num</div>";

}

?>

</body>

</html>

**Explanation:**

1. **HTML Structure**:
   * Basic HTML structure with <head> containing charset definition and title, and <body> containing the main content.
2. **CSS Styling**:
   * <style> tag defines CSS classes .odd and .even for alternating background colors.
3. **PHP Code**:
   * **Array Definition**: $numbers is an array containing [1, 2, 3, 4, 5].
   * **Looping with Decision Making**:
     + foreach loop iterates through each element of $numbers.
     + Inside the loop, if statement checks if the current number ($num) is even ($num % 2 == 0). If true, sets $class to 'even', otherwise to 'odd'.
     + Uses PHP variable interpolation ($class) to dynamically apply CSS classes to <div> elements based on whether the number is odd or even.
4. **HTML and PHP Integration**:
   * PHP code (foreach loop and if statement) is embedded within HTML content to dynamically generate <div> elements with alternating background colors based on the condition ($num % 2 == 0).

To run this PHP program:

* Save the code in a .php file (e.g., index.php).
* Place it in a web server's directory (like XAMPP, WAMP, or any PHP-enabled server).
* Open the file in a web browser (http://localhost/index.php) to see the output.

This PHP program effectively demonstrates mixing decision-making (if, else) and looping statements (foreach) with HTML content to dynamically generate and style elements based on data processing logic within a web page.

6: Write a PHP program to demonstrate call by value, call by reference and recursive function.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>PHP Call by Value, Call by Reference, Recursive Function</title>

</head>

<body>

<h1>PHP Call by Value, Call by Reference, Recursive Function</h1>

<?php

// Function to demonstrate call by value

function incrementByValue($num) {

$num += 10;

echo "<p>Inside incrementByValue function: \$num = $num</p>";

}

$value = 5;

echo "<p>Before calling incrementByValue function: \$value = $value</p>";

incrementByValue($value);

echo "<p>After calling incrementByValue function: \$value = $value</p>"; // $value remains unchanged

// Function to demonstrate call by reference

function incrementByReference(&$num) {

$num += 10;

echo "<p>Inside incrementByReference function: \$num = $num</p>";

}

$value = 5;

echo "<p>Before calling incrementByReference function: \$value = $value</p>";

incrementByReference($value);

echo "<p>After calling incrementByReference function: \$value = $value</p>"; // $value is modified

// Recursive function to calculate factorial

function factorial($n) {

if ($n <= 1) {

return 1;

} else {

return $n \* factorial($n - 1);

}

}

$num = 5;

echo "<p>Factorial of $num is " . factorial($num) . "</p>";

?>

</body>

</html>

**Explanation:**

1. **Call by Value**:
   * incrementByValue($num) function demonstrates call by value. The parameter $num is passed by value, so modifications inside the function do not affect the original variable $value outside the function.
2. **Call by Reference**:
   * incrementByReference(&$num) function demonstrates call by reference. The parameter $num is passed by reference using &, allowing modifications inside the function to directly affect the original variable $value outside the function.
3. **Recursive Function**:
   * factorial($n) function calculates the factorial of a number recursively. It multiplies the current number $n with the factorial of $n-1 until $n reaches 1 (base case).
4. **HTML and PHP Integration**:
   * PHP functions (incrementByValue, incrementByReference, factorial) are embedded within HTML content to execute logic and display output dynamically within a web page.

**Output:**

When you run this PHP program in a web server, you'll see the following output:

bash

Before calling incrementByValue function: $value = 5

Inside incrementByValue function: $num = 15

After calling incrementByValue function: $value = 5

Before calling incrementByReference function: $value = 5

Inside incrementByReference function: $num = 15

After calling incrementByReference function: $value = 15

Factorial of 5 is 120

* **Call by Value**: $value remains unchanged after calling incrementByValue.
* **Call by Reference**: $value is modified after calling incrementByReference.
* **Recursive Function**: Calculates and displays the factorial of 5.

This PHP program effectively demonstrates call by value, call by reference, and a recursive function, illustrating different parameter passing mechanisms and recursion in PHP.

7.Write a PHP Program for String Creating and accessing, String Searching & Replacing String, Formatting String and demonstrating String Related Library function.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>PHP String Operations</title>

</head>

<body>

<h1>PHP String Operations</h1>

<?php

// Creating and accessing strings

$str1 = "Hello, World!";

$str2 = 'PHP is awesome!';

echo "<h2>Creating and Accessing Strings</h2>";

echo "<p>String 1: $str1</p>";

echo "<p>String 2: $str2</p>";

// String length

$len1 = strlen($str1);

$len2 = strlen($str2);

echo "<p>Length of String 1: $len1</p>";

echo "<p>Length of String 2: $len2</p>";

// String concatenation

$concatenatedStr = $str1 . ' ' . $str2;

echo "<p>Concatenated String: $concatenatedStr</p>";

// String searching and replacing

$searchStr = "PHP";

$replaceStr = "JavaScript";

$newStr = str\_replace($searchStr, $replaceStr, $concatenatedStr);

echo "<h2>String Searching and Replacing</h2>";

echo "<p>Original String: $concatenatedStr</p>";

echo "<p>After replacing '$searchStr' with '$replaceStr': $newStr</p>";

// Formatting strings

$price = 49.99;

echo "<h2>Formatting Strings</h2>";

echo "<p>Price: $" . number\_format($price, 2) . "</p>";

// String related library functions

echo "<h2>String Related Library Functions</h2>";

// Convert string to uppercase and lowercase

$uppercaseStr = strtoupper($str1);

$lowercaseStr = strtolower($str2);

echo "<p>Uppercase of '$str1': $uppercaseStr</p>";

echo "<p>Lowercase of '$str2': $lowercaseStr</p>";

// Trim whitespace from a string

$strWithWhitespace = " Trim Test ";

$trimmedStr = trim($strWithWhitespace);

echo "<p>Original String with whitespace: '$strWithWhitespace'</p>";

echo "<p>Trimmed String: '$trimmedStr'</p>";

// Find position of substring in a string

$searchSubstring = "World";

$pos = strpos($str1, $searchSubstring);

if ($pos !== false) {

echo "<p>Position of '$searchSubstring' in '$str1': $pos</p>";

} else {

echo "<p>'$searchSubstring' not found in '$str1'</p>";

}

// Repeat a string

$repeatStr = str\_repeat('PHP ', 3);

echo "<p>Repeated String: $repeatStr</p>";

?>

</body>

</html>

**Explanation:**

1. **Creating and Accessing Strings**:
   * Two strings $str1 and $str2 are created and accessed using PHP's variable interpolation within double quotes (").
2. **String Length**:
   * strlen() function is used to get the length of strings $str1 and $str2.
3. **String Concatenation**:
   * Concatenation operator (.) is used to concatenate $str1 and $str2 with a space in between.
4. **String Searching and Replacing**:
   * str\_replace() function replaces occurrences of $searchStr ("PHP") with $replaceStr ("JavaScript") in $concatenatedStr.
5. **Formatting Strings**:
   * number\_format() function is used to format the $price to two decimal places with a dollar sign.
6. **String Related Library Functions**:
   * **Convert to Uppercase and Lowercase**: strtoupper() and strtolower() functions convert strings to uppercase and lowercase respectively.
   * **Trim Whitespace**: trim() function removes leading and trailing whitespace from $strWithWhitespace.
   * **Find Position of Substring**: strpos() function finds the position of $searchSubstring ("World") in $str1.
   * **Repeat a String**: str\_repeat() function repeats the string 'PHP' three times.

**Output:**

When you run this PHP program in a web server, you'll see the following output:

javascript

Creating and Accessing Strings

String 1: Hello, World!

String 2: PHP is awesome!

Length of String 1: 13

Length of String 2: 15

Concatenated String: Hello, World! PHP is awesome!

String Searching and Replacing

Original String: Hello, World! PHP is awesome!

After replacing 'PHP' with 'JavaScript': Hello, World! JavaScript is awesome!

Formatting Strings

Price: $49.99

String Related Library Functions

Uppercase of 'Hello, World!': HELLO, WORLD!

Lowercase of 'PHP is awesome!': php is awesome!

Original String with whitespace: ' Trim Test '

Trimmed String: 'Trim Test'

Position of 'World' in 'Hello, World!': 7

Repeated String: PHP PHP PHP

8: Write a PHP program for creating index based and associative arrays. Also demonstrate accessing the array and element looping with Index based array

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>PHP Arrays</title>

</head>

<body>

<h1>PHP Arrays</h1>

<?php

// Index based array

$indexArray = array("Apple", "Banana", "Cherry", "Date", "Fig");

// Associative array

$assocArray = array(

"fruit1" => "Apple",

"fruit2" => "Banana",

"fruit3" => "Cherry",

"fruit4" => "Date",

"fruit5" => "Fig"

);

// Accessing elements in index based array

echo "<h2>Accessing Index Based Array</h2>";

echo "<p>First element: " . $indexArray[0] . "</p>";

echo "<p>Third element: " . $indexArray[2] . "</p>";

// Looping through elements in index based array

echo "<h2>Looping through Index Based Array</h2>";

echo "<ul>";

foreach ($indexArray as $fruit) {

echo "<li>$fruit</li>";

}

echo "</ul>";

?>

</body>

</html>

**Explanation:**

1. **Creating Arrays**:
   * **Index Based Array**: $indexArray is created using array("Apple", "Banana", "Cherry", "Date", "Fig").
   * **Associative Array**: $assocArray is created using key-value pairs ("fruit1" => "Apple", "fruit2" => "Banana", etc.).
2. **Accessing Elements**:
   * Elements of the index-based array $indexArray are accessed using their indices ($indexArray[0], $indexArray[2]).
3. **Looping through Index Based Array**:
   * A foreach loop iterates through each element of $indexArray. In each iteration, the current element is stored in $fruit variable, which is then echoed out as list items (<li> tags) within an unordered list (<ul>).

**Output:**

When you run this PHP program in a web server, you'll see the following output:

diff

Accessing Index Based Array

First element: Apple

Third element: Cherry

Looping through Index Based Array

- Apple

- Banana

- Cherry

- Date

- Fig

* **Accessing Index Based Array**: Demonstrates accessing elements by index ($indexArray[0], $indexArray[2]).
* **Looping through Index Based Array**: Uses foreach loop to iterate through each element of $indexArray and display them as list items.

This PHP program effectively demonstrates creating index-based and associative arrays, accessing array elements, and looping through elements in an index-based array to display them in an HTML format.

9: Write a PHP Program to demonstrate associative arrays using each () and for each() and a few related Library functions. Enhance the array program by including decision and loop statements.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>PHP Associative Arrays and Functions</title>

</head>

<body>

<h1>PHP Associative Arrays and Functions</h1>

<?php

// Associative array

$studentScores = array(

"John" => 85,

"Jane" => 92,

"Doe" => 78,

"Smith" => 88,

"Emily" => 95

);

// Demonstrating each() function

echo "<h2>Demonstrating each() Function</h2>";

reset($studentScores); // Resetting array pointer

while ($pair = each($studentScores)) {

echo "<p>{$pair['key']} scored {$pair['value']} marks.</p>";

}

// Demonstrating foreach() loop

echo "<h2>Demonstrating foreach() Loop</h2>";

foreach ($studentScores as $name => $score) {

echo "<p>$name scored $score marks.</p>";

}

// Using library functions with associative arrays

echo "<h2>Using Library Functions</h2>";

// Count number of elements in array

$numStudents = count($studentScores);

echo "<p>Number of students: $numStudents</p>";

// Check if a key exists in the array

$keyToCheck = "Doe";

if (array\_key\_exists($keyToCheck, $studentScores)) {

echo "<p>$keyToCheck exists in the array.</p>";

} else {

echo "<p>$keyToCheck does not exist in the array.</p>";

}

// Find the maximum value in the array

$maxScore = max($studentScores);

echo "<p>Maximum score: $maxScore</p>";

// Calculate the average score

$totalScore = array\_sum($studentScores);

$averageScore = $totalScore / $numStudents;

echo "<p>Average score: " . round($averageScore, 2) . "</p>";

// Using decision and loop statements

echo "<h2>Using Decision and Loop Statements</h2>";

$passingScore = 80;

foreach ($studentScores as $name => $score) {

if ($score >= $passingScore) {

echo "<p>$name has passed with a score of $score.</p>";

} else {

echo "<p>$name has failed with a score of $score.</p>";

}

}

?>

</body>

</html>

**Explanation:**

1. **Associative Array**:
   * $studentScores is an associative array where student names are keys and their scores are values.
2. **Using each() Function**:
   * each($studentScores) retrieves each key-value pair from the associative array until all pairs have been processed. It returns an array with key and value indices.
3. **Using foreach() Loop**:
   * foreach ($studentScores as $name => $score) iterates through each element of the associative array $studentScores, where $name is the key (student name) and $score is the value (student score).
4. **Using Library Functions**:
   * **count()**: Counts the number of elements in the array ($numStudents).
   * **array\_key\_exists()**: Checks if a specific key ($keyToCheck) exists in the array ($studentScores).
   * **max()**: Finds the maximum value (highest score) in the array ($maxScore).
   * **array\_sum()**: Calculates the total sum of all scores in the array ($totalScore).
   * **round()**: Rounds the average score ($averageScore) to two decimal places.
5. **Using Decision and Loop Statements**:
   * foreach loop combined with an if statement checks if each student's score meets or exceeds a passing score ($passingScore). It prints whether each student has passed or failed based on this condition.

**Output:**

When you run this PHP program in a web server, you'll see the following output:

scss

Demonstrating each() Function

John scored 85 marks.

Jane scored 92 marks.

Doe scored 78 marks.

Smith scored 88 marks.

Emily scored 95 marks.

Demonstrating foreach() Loop

John scored 85 marks.

Jane scored 92 marks.

Doe scored 78 marks.

Smith scored 88 marks.

Emily scored 95 marks.

Using Library Functions

Number of students: 5

Doe exists in the array.

Maximum score: 95

Average score: 87.6

Using Decision and Loop Statements

John has passed with a score of 85.

Jane has passed with a score of 92.

Doe has failed with a score of 78.

Smith has passed with a score of 88.

Emily has passed with a score of 95.

This PHP program effectively demonstrates using associative arrays with each() and foreach() functions, utilizing various library functions for associative arrays, and incorporating decision-making (if statement) and loop statements (foreach loop) for processing array data dynamically within an HTML page.

10: Write a PHP Program to demonstrate Capturing Form and to handle Data Dealing with Multi-value field, Match, Search and Split function.

**HTML Form (index.html):**

html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>PHP Form Handling</title>

</head>

<body>

<h1>PHP Form Handling</h1>

<form action="process\_form.php" method="post">

<label for="name">Name:</label>

<input type="text" id="name" name="name" required><br><br>

<label for="email">Email:</label>

<input type="email" id="email" name="email" required><br><br>

<label for="interests">Interests:</label><br>

<input type="checkbox" id="interest1" name="interests[]" value="Sports">

<label for="interest1">Sports</label><br>

<input type="checkbox" id="interest2" name="interests[]" value="Music">

<label for="interest2">Music</label><br>

<input type="checkbox" id="interest3" name="interests[]" value="Books">

<label for="interest3">Books</label><br><br>

<label for="message">Message:</label><br>

<textarea id="message" name="message" rows="4" cols="50"></textarea><br><br>

<input type="submit" value="Submit">

</form>

</body>

</html>

**PHP Form Processor (process\_form.php):**

php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>PHP Form Processing</title>

</head>

<body>

<h1>PHP Form Processing</h1>

<?php

// Capture form data

$name = $\_POST['name'];

$email = $\_POST['email'];

$interests = isset($\_POST['interests']) ? $\_POST['interests'] : array();

$message = $\_POST['message'];

// Display captured data

echo "<h2>Form Data:</h2>";

echo "<p>Name: $name</p>";

echo "<p>Email: $email</p>";

if (!empty($interests)) {

echo "<p>Interests:</p>";

echo "<ul>";

foreach ($interests as $interest) {

echo "<li>$interest</li>";

}

echo "</ul>";

} else {

echo "<p>No interests selected.</p>";

}

echo "<p>Message:</p>";

echo "<p>$message</p>";

// Example using preg\_match

echo "<h2>Using preg\_match</h2>";

$pattern = "/[0-9]+/";

$str = "Hello 123 World";

if (preg\_match($pattern, $str, $matches)) {

echo "<p>Numbers found in '$str': " . implode(", ", $matches) . "</p>";

} else {

echo "<p>No numbers found in '$str'.</p>";

}

// Example using preg\_split

echo "<h2>Using preg\_split</h2>";

$str = "apple, banana, cherry, date";

$array = preg\_split("/[\s,]+/", $str);

echo "<p>Splitting '$str': " . implode(", ", $array) . "</p>";

?>

</body>

</html>

**Explanation:**

1. **HTML Form (index.html)**:
   * Creates a basic HTML form with fields for name, email, interests (checkboxes), and message.
   * The form action points to process\_form.php which handles form submission.
2. **PHP Form Processor (process\_form.php)**:
   * **Capturing Form Data**: Uses $\_POST to capture form data submitted from index.html.
   * **Displaying Form Data**: Echoes captured data ($name, $email, $interests, $message) within HTML paragraphs and lists.
3. **Handling Multi-value Fields (interests[])**:
   * Uses PHP's $\_POST['interests'] array to handle checkboxes (interests[]) which can have multiple selected values.
4. **Using preg\_match**:
   * Demonstrates preg\_match to find numbers (/[0-9]+/) in a string ($str), and displays matched results using implode().
5. **Using preg\_split**:
   * Demonstrates preg\_split to split a string ($str) by whitespace or comma (/[\s,]+/), and displays the resulting array using implode().

**Usage:**

1. Save index.html and process\_form.php in the same directory on your web server (e.g., XAMPP, WAMP).
2. Open index.html in a web browser (http://localhost/index.html).
3. Fill out the form and submit it.
4. You'll be redirected to process\_form.php, which will process the form data and display the captured data along with examples of using preg\_match and preg\_split.

This PHP program effectively demonstrates capturing form data, handling multi-value fields, and using preg\_match, preg\_split, and other functions to manipulate and process data dynamically within an HTML page.

11: Write a PHP Program for Generating File uploaded form and redirecting a form after submission.

**HTML File Upload Form (index.html):**

html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>File Upload Form</title>

</head>

<body>

<h1>File Upload Form</h1>

<form action="upload\_file.php" method="post" enctype="multipart/form-data">

<label for="file">Select File:</label>

<input type="file" id="file" name="file" required><br><br>

<input type="submit" value="Upload File">

</form>

</body>

</html>

**PHP File Upload Handler (upload\_file.php):**

php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>File Upload Processing</title>

</head>

<body>

<h1>File Upload Processing</h1>

<?php

// Check if file upload form is submitted

if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_FILES["file"])) {

$uploadDir = "uploads/"; // Directory where uploaded files will be stored

$uploadFile = $uploadDir . basename($\_FILES["file"]["name"]); // Path to store uploaded file

// Check if file is uploaded successfully

if (move\_uploaded\_file($\_FILES["file"]["tmp\_name"], $uploadFile)) {

echo "<p>The file ". htmlspecialchars(basename($\_FILES["file"]["name"])) . " has been uploaded.</p>";

echo "<p><a href='upload\_success.php'>Go to Upload Success Page</a></p>";

} else {

echo "<p>Sorry, there was an error uploading your file.</p>";

}

}

?>

</body>

</html>

**Upload Success Page (upload\_success.php):**

php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Upload Success</title>

</head>

<body>

<h1>Upload Success</h1>

<p>Your file has been uploaded successfully.</p>

<!-- Additional content or redirection can be added here -->

</body>

</html>

**Explanation:**

1. **HTML File Upload Form (index.html)**:
   * Provides a simple HTML form (<form>) with a file input (<input type="file">) for users to select a file for upload.
   * The form action (action="upload\_file.php") directs to upload\_file.php for processing the file upload.
2. **PHP File Upload Handler (upload\_file.php)**:
   * Handles the file upload process (move\_uploaded\_file() function).
   * Checks if the form is submitted ($\_SERVER["REQUEST\_METHOD"] == "POST") and verifies if a file is uploaded (isset($\_FILES["file"])).
   * If the file upload is successful, it moves the uploaded file to a specified directory ($uploadDir) and displays a success message with a link to upload\_success.php.
   * If there's an error uploading the file, it displays an error message.
3. **Upload Success Page (upload\_success.php)**:
   * Displays a simple confirmation message that the file has been uploaded successfully.
   * Additional content or redirection logic can be added here as needed.

**Usage:**

1. Save index.html, upload\_file.php, and upload\_success.php in the same directory on your web server (e.g., XAMPP, WAMP).
2. Open index.html in a web browser (http://localhost/index.html).
3. Select a file using the file upload form and submit the form.
4. If the file upload is successful, you'll be redirected to upload\_success.php where the upload success message will be displayed.

This PHP program effectively demonstrates how to create a file upload form, handle file uploads using PHP, and redirect to a success page after a successful file upload. Adjustments can be made to handle errors, security checks, and additional processing based on specific requirements.

12: Write a PHP Program to demonstrate how to handle file & directory, Opening and closing a file, Copying, renaming and deleting a file, working with directories, Creating and deleting folder, File Uploading & Downloading.

**Handling Files and Directories in PHP**

**File Operations (file\_operations.php):**

php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>File and Directory Operations</title>

</head>

<body>

<h1>File and Directory Operations</h1>

<?php

// File handling

// Opening and Closing a File

$file = "sample.txt";

$fileHandle = fopen($file, "r") or die("Unable to open file!");

echo "<p>Opened file '$file' successfully.</p>";

fclose($fileHandle);

// Copying a File

$sourceFile = "sample.txt";

$destinationFile = "copied\_sample.txt";

if (copy($sourceFile, $destinationFile)) {

echo "<p>File '$sourceFile' copied to '$destinationFile' successfully.</p>";

} else {

echo "<p>Failed to copy file '$sourceFile'.</p>";

}

// Renaming a File

$oldName = "copied\_sample.txt";

$newName = "renamed\_sample.txt";

if (rename($oldName, $newName)) {

echo "<p>File '$oldName' renamed to '$newName' successfully.</p>";

} else {

echo "<p>Failed to rename file '$oldName'.</p>";

}

// Deleting a File

$fileToDelete = "renamed\_sample.txt";

if (unlink($fileToDelete)) {

echo "<p>File '$fileToDelete' deleted successfully.</p>";

} else {

echo "<p>Failed to delete file '$fileToDelete'.</p>";

}

// Directory handling

// Creating a Directory

$directoryToCreate = "uploads";

if (!is\_dir($directoryToCreate)) {

mkdir($directoryToCreate);

echo "<p>Directory '$directoryToCreate' created successfully.</p>";

} else {

echo "<p>Directory '$directoryToCreate' already exists.</p>";

}

// Deleting a Directory

$directoryToDelete = "uploads";

if (is\_dir($directoryToDelete)) {

rmdir($directoryToDelete);

echo "<p>Directory '$directoryToDelete' deleted successfully.</p>";

} else {

echo "<p>Directory '$directoryToDelete' does not exist.</p>";

}

?>

</body>

</html>

**File Upload and Download (file\_upload\_download.php):**

php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>File Upload and Download</title>

</head>

<body>

<h1>File Upload and Download</h1>

<!-- File Upload Form -->

<h2>File Upload</h2>

<form action="file\_upload\_handler.php" method="post" enctype="multipart/form-data">

<label for="file">Select File to Upload:</label>

<input type="file" id="file" name="file" required>

<br><br>

<input type="submit" value="Upload File">

</form>

<!-- File Download Link -->

<h2>File Download</h2>

<?php

$downloadFile = "sample.txt";

if (file\_exists($downloadFile)) {

echo "<p><a href='file\_download\_handler.php?file=$downloadFile'>Download $downloadFile</a></p>";

} else {

echo "<p>File '$downloadFile' does not exist.</p>";

}

?>

</body>

</html>

**File Upload Handler (file\_upload\_handler.php):**

php

<?php

// File Upload Handling

if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_FILES["file"])) {

$uploadDir = "uploads/"; // Directory where uploaded files will be stored

$uploadFile = $uploadDir . basename($\_FILES["file"]["name"]); // Path to store uploaded file

// Check if file is uploaded successfully

if (move\_uploaded\_file($\_FILES["file"]["tmp\_name"], $uploadFile)) {

echo "<p>The file ". htmlspecialchars(basename($\_FILES["file"]["name"])) . " has been uploaded.</p>";

} else {

echo "<p>Sorry, there was an error uploading your file.</p>";

}

}

?>

**File Download Handler (file\_download\_handler.php):**

php

<?php

// File Download Handling

if (isset($\_GET['file'])) {

$file = $\_GET['file'];

// Check if file exists

if (file\_exists($file)) {

// Set headers for file download

header('Content-Description: File Transfer');

header('Content-Type: application/octet-stream');

header('Content-Disposition: attachment; filename=' . basename($file));

header('Content-Length: ' . filesize($file));

header('Pragma: public');

header('Cache-Control: must-revalidate, post-check=0, pre-check=0');

header('Expires: 0');

readfile($file); // Read the file to download

exit;

} else {

echo "File '$file' not found.";

}

}

?>

**Explanation:**

1. **File Operations (file\_operations.php)**:
   * **Opening and Closing a File**: Uses fopen() to open a file and fclose() to close it.
   * **Copying a File**: Uses copy() to duplicate a file.
   * **Renaming a File**: Uses rename() to change a file's name.
   * **Deleting a File**: Uses unlink() to remove a file from the filesystem.
   * **Creating a Directory**: Uses mkdir() to create a new directory if it doesn't already exist.
   * **Deleting a Directory**: Uses rmdir() to remove a directory if it exists and is empty.
2. **File Upload and Download (file\_upload\_download.php)**:
   * **File Upload Form**: Provides a form (<form>) with a file input (<input type="file">) for users to upload files.
   * **File Upload Handler (file\_upload\_handler.php)**: Handles file uploads using move\_uploaded\_file() and stores uploaded files in the uploads/ directory.
   * **File Download Link**: Displays a download link for sample.txt if it exists, linking to file\_download\_handler.php for downloading.
3. **File Download Handler (file\_download\_handler.php)**:
   * **File Download Handling**: Retrieves the filename from the query string ($\_GET['file']), checks if the file exists (file\_exists()), and sets appropriate HTTP headers to force download (header() functions) using readfile().

**Usage:**

1. Save all PHP files (file\_operations.php, file\_upload\_download.php, file\_upload\_handler.php, file\_download\_handler.php) in the same directory on your web server (e.g., XAMPP, WAMP).
2. Open file\_operations.php or file\_upload\_download.php in a web browser (http://localhost/file\_operations.php or http://localhost/file\_upload\_download.php).
3. Follow the instructions for handling files and directories, uploading files, and downloading files as demonstrated in each PHP script.

These PHP programs effectively demonstrate how to handle file operations (opening, copying, renaming, deleting), directory operations (creating, deleting), file uploading, and file downloading using PHP. Adjustments can be made to handle errors, enhance security, and meet specific application requirements as needed.

13: Write a PHP Program to Connect PHP with MySQL Database and Perform basic database operation (DML, like Insert, Delete, Update, Select)

**Setting Up MySQL Database**

First, ensure you have MySQL installed and running. Create a database and a table for demonstration purposes:

1. **Create Database**: Assuming you have MySQL installed and running, open a MySQL client (like phpMyAdmin or MySQL Workbench) and execute the following SQL query to create a database:

sql

CREATE DATABASE testdb;

1. **Create Table**: Inside the testdb database, create a table named users with columns for id, name, email, and created\_at:

sql

USE testdb;

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

email VARCHAR(50) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**PHP Program to Connect and Perform Basic Database Operations**

Create a PHP script (database\_operations.php) that connects to MySQL, performs Insert, Delete, Update, and Select operations on the users table.

**database\_operations.php:**

php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>PHP MySQL Database Operations</title>

</head>

<body>

<h1>PHP MySQL Database Operations</h1>

<?php

// Database configuration

$servername = "localhost";

$username = "root"; // Replace with your MySQL username

$password = ""; // Replace with your MySQL password

$dbname = "testdb"; // Replace with your database name

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

echo "<p>Connected successfully to database '$dbname'.</p>";

// Perform Insert Operation

$name = "John Doe";

$email = "john.doe@example.com";

$sql = "INSERT INTO users (name, email) VALUES ('$name', '$email')";

if ($conn->query($sql) === TRUE) {

echo "<p>New record created successfully.</p>";

} else {

echo "<p>Error: " . $sql . "<br>" . $conn->error . "</p>";

}

// Perform Select Operation

echo "<h2>Users List:</h2>";

$selectSql = "SELECT id, name, email, created\_at FROM users";

$result = $conn->query($selectSql);

if ($result->num\_rows > 0) {

echo "<ul>";

while ($row = $result->fetch\_assoc()) {

echo "<li>ID: " . $row["id"] . " - Name: " . $row["name"] . " - Email: " . $row["email"] . " - Created At: " . $row["created\_at"] . "</li>";

}

echo "</ul>";

} else {

echo "<p>No users found.</p>";

}

// Perform Update Operation

$updateSql = "UPDATE users SET email='john.new@example.com' WHERE name='John Doe'";

if ($conn->query($updateSql) === TRUE) {

echo "<p>Record updated successfully.</p>";

} else {

echo "<p>Error updating record: " . $conn->error . "</p>";

}

// Perform Delete Operation

$deleteSql = "DELETE FROM users WHERE name='John Doe'";

if ($conn->query($deleteSql) === TRUE) {

echo "<p>Record deleted successfully.</p>";

} else {

echo "<p>Error deleting record: " . $conn->error . "</p>";

}

// Close connection

$conn->close();

?>

</body>

</html>

**Explanation:**

1. **Database Connection**: Establishes a connection to MySQL using mysqli object-oriented approach.
2. **Insert Operation**:
   * Uses INSERT INTO SQL statement to add a new record ($name and $email) into the users table.
3. **Select Operation**:
   * Retrieves all records from the users table using SELECT SQL statement and displays them in an unordered list (<ul>).
4. **Update Operation**:
   * Uses UPDATE SQL statement to modify the email of the record where name is 'John Doe'.
5. **Delete Operation**:
   * Uses DELETE FROM SQL statement to remove the record where name is 'John Doe'.
6. **Error Handling**: Checks for errors during database operations and displays error messages if any.

**Usage:**

1. Save database\_operations.php in your web server's directory (e.g., htdocs for XAMPP).
2. Open a web browser and navigate to http://localhost/database\_operations.php.
3. The PHP script will execute, connect to the MySQL database (testdb), perform Insert, Select, Update, and Delete operations on the users table, and display results and status messages accordingly.

Ensure you replace placeholders ($username, $password) with your actual MySQL credentials. This example demonstrates basic database operations in PHP using MySQL, and you can expand and customize it based on your specific application requirements and additional security considerations.

14: Write a PHP Program to demonstrate Setting query parameter, Executing query- Join (Cross joins, Inner joins, Outer Joins, Self joins)

**PHP Program Demonstrating SQL Joins and Query Parameters**

**query\_joins.php:**

php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>PHP SQL Joins</title>

</head>

<body>

<h1>PHP SQL Joins</h1>

<?php

// Database configuration

$servername = "localhost";

$username = "root"; // Replace with your MySQL username

$password = ""; // Replace with your MySQL password

$dbname = "testdb"; // Replace with your database name

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

echo "<p>Connected successfully to database '$dbname'.</p>";

// Example data for demonstration

$sql\_create\_tables = "

CREATE TABLE IF NOT EXISTS employees (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

department\_id INT NOT NULL

);

CREATE TABLE IF NOT EXISTS departments (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(50) NOT NULL

);

INSERT INTO departments (name) VALUES

('HR'),

('IT'),

('Finance');

INSERT INTO employees (name, department\_id) VALUES

('John Doe', 1),

('Jane Smith', 2),

('Michael Johnson', 1),

('Emily Davis', 3);

";

if ($conn->multi\_query($sql\_create\_tables) === TRUE) {

echo "<p>Tables created and initial data inserted successfully.</p>";

} else {

echo "<p>Error creating tables: " . $conn->error . "</p>";

}

// Close connection after initial setup (not necessary for execution of joins)

$conn->close();

// Reconnect to execute joins

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Execute SQL Joins

echo "<h2>SQL Joins</h2>";

// Cross Join

echo "<h3>Cross Join</h3>";

$sql\_cross\_join = "

SELECT e.name AS employee\_name, d.name AS department\_name

FROM employees e

CROSS JOIN departments d;

";

executeAndDisplayQuery($conn, $sql\_cross\_join);

// Inner Join

echo "<h3>Inner Join</h3>";

$sql\_inner\_join = "

SELECT e.name AS employee\_name, d.name AS department\_name

FROM employees e

INNER JOIN departments d ON e.department\_id = d.id;

";

executeAndDisplayQuery($conn, $sql\_inner\_join);

// Left Outer Join

echo "<h3>Left Outer Join</h3>";

$sql\_left\_outer\_join = "

SELECT e.name AS employee\_name, d.name AS department\_name

FROM employees e

LEFT JOIN departments d ON e.department\_id = d.id;

";

executeAndDisplayQuery($conn, $sql\_left\_outer\_join);

// Right Outer Join

echo "<h3>Right Outer Join</h3>";

$sql\_right\_outer\_join = "

SELECT e.name AS employee\_name, d.name AS department\_name

FROM employees e

RIGHT JOIN departments d ON e.department\_id = d.id;

";

executeAndDisplayQuery($conn, $sql\_right\_outer\_join);

// Self Join (for demonstration)

echo "<h3>Self Join</h3>";

$sql\_self\_join = "

SELECT e1.name AS employee\_name, e2.name AS manager\_name

FROM employees e1

LEFT JOIN employees e2 ON e1.manager\_id = e2.id;

";

executeAndDisplayQuery($conn, $sql\_self\_join);

// Function to execute query and display results

function executeAndDisplayQuery($conn, $sql) {

$result = $conn->query($sql);

if ($result === FALSE) {

echo "<p>Error executing query: " . $conn->error . "</p>";

} else {

if ($result->num\_rows > 0) {

echo "<ul>";

while ($row = $result->fetch\_assoc()) {

echo "<li>" . $row["employee\_name"] . " - " . $row["department\_name"] . "</li>";

}

echo "</ul>";

} else {

echo "<p>No results found.</p>";

}

}

}

// Close connection

$conn->close();

?>

</body>

</html>

**Explanation:**

1. **Database Setup**: Creates tables employees and departments with initial data using CREATE TABLE and INSERT INTO SQL statements.
2. **Connecting to Database**: Uses mysqli to connect to MySQL database (testdb) and checks for connection errors.
3. **Executing SQL Joins**:
   * **Cross Join**: Demonstrates all combinations of rows from employees and departments.
   * **Inner Join**: Retrieves rows where there is a match between employees.department\_id and departments.id.
   * **Left Outer Join**: Retrieves all rows from employees and matching rows from departments.
   * **Right Outer Join**: Retrieves all rows from departments and matching rows from employees.
   * **Self Join**: For demonstration purposes, joins employees table with itself to show hierarchical relationships (manager\_id referencing id).
4. **Function executeAndDisplayQuery**: Helper function to execute SQL queries ($sql) and display results ($result).
5. **Error Handling**: Checks for errors during SQL query execution and displays error messages if any.

**Usage:**

1. Save query\_joins.php in your web server's directory (e.g., htdocs for XAMPP).
2. Open a web browser and navigate to http://localhost/query\_joins.php.
3. The PHP script will execute, connect to the MySQL database (testdb), perform various SQL joins (Cross, Inner, Left Outer, Right Outer, Self), and display results accordingly.

Make sure to replace placeholders ($username, $password) with your actual MySQL credentials. This example demonstrates how to use PHP to execute different types of SQL joins and display their results, leveraging the flexibility and power of MySQL relational database management system. Adjustments can be made based on specific application requirements and additional SQL functionalities needed.

15: Write a PHP Program to demonstrate statement/process like exception and error, Try, catch, throw, Error tracking and debuggingTop of Form

**PHP Program Demonstrating Exception Handling and Error Tracking**

**exception\_handling.php:**

php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>PHP Exception Handling</title>

</head>

<body>

<h1>PHP Exception Handling</h1>

<?php

// Example function that throws an exception

function divideNumbers($numerator, $denominator) {

if ($denominator == 0) {

throw new Exception("Division by zero error.");

}

return $numerator / $denominator;

}

// Example try-catch block to handle exceptions

try {

$result = divideNumbers(10, 0);

echo "<p>Result of division: $result</p>";

} catch (Exception $e) {

echo "<p>Caught exception: " . $e->getMessage() . "</p>";

}

// Example of error handling with error\_reporting and error\_log

error\_reporting(E\_ALL);

ini\_set('display\_errors', 1);

// Example of debugging with var\_dump and error\_log

$array = [1, 2, 3];

$undefinedVariable = $array[5]; // Accessing an undefined index

var\_dump($undefinedVariable); // Outputs NULL and generates a notice

// Logging errors to a file

error\_log("Error: Undefined index accessed in exception\_handling.php", 3, "error.log");

echo "<p>Check error.log for logged errors.</p>";

?>

</body>

</html>

**Explanation:**

1. **Exception Handling (try, catch, throw)**:
   * **divideNumbers Function**: Defines a function that attempts to divide two numbers ($numerator and $denominator). If $denominator is zero, it throws a new Exception with a custom message.
   * **try-catch Block**: Uses a try block to execute the divideNumbers function and a catch block to handle any Exception that may be thrown. If an exception occurs (division by zero), it catches the exception and displays the error message using $e->getMessage().
2. **Error Tracking and Debugging**:
   * **error\_reporting and ini\_set**: Sets error reporting to display all errors (E\_ALL) and enables displaying errors (display\_errors) in the script.
   * **var\_dump**: Demonstrates debugging by using var\_dump to output the value of an undefined variable ($undefinedVariable). This operation generates a notice because $array[5] does not exist.
   * **Logging Errors**: Uses error\_log to log an error message ("Undefined index accessed in exception\_handling.php") to a file (error.log). The third parameter (3) specifies logging to a file, and the file path (error.log) specifies where to log errors.

**Usage:**

1. Save exception\_handling.php in your web server's directory (e.g., htdocs for XAMPP).
2. Open a web browser and navigate to http://localhost/exception\_handling.php.
3. The PHP script will execute, demonstrating exception handling with try, catch, and throw statements for division by zero error, error reporting and debugging with var\_dump, and error logging to error.log.

Make sure your PHP configuration (php.ini) allows displaying errors (display\_errors = On) for development purposes. This example showcases how to handle exceptions, track errors, and debug PHP scripts effectively, ensuring robust error handling and debugging practices in PHP applications. Adjustments can be made based on specific error handling requirements and debugging scenarios in your application.

Top of Form

Bottom of Form

Bottom of Form