Question 1: PDO Grade Update

Task: Write a PHP script that connects to a MySQL database using PDO. The database should use the following parameters: host is `localhost`, database name is `courses`, username is `root`, and password is `root1234`. After establishing the connection, Write a PHP function called `updateStudentGrade` that updates a student's grade in math course. The function should take parameters for the student ID, course name, and new grade. Use prepared statements and return `true` if the update was successful or `false` if not.

Answer

```php
<?php
function updateStudentGrade($studentId, $courseName, $newGrade) {
    $host = 'localhost';
    $dbname = 'courses';
    $username = 'root';
    $password = 'root1234';
    $dsn = "mysql:host=$host;dbname=$dbname;charset=utf8mb4";

    try {
        $pdo = new PDO($dsn, $username, $password);
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        $sql = "UPDATE grades SET grade = :grade
                WHERE student_id = :student_id AND course_name = :course_name";

        $stmt = $pdo->prepare($sql);

        $stmt->bindParam(':grade', $newGrade, PDO::PARAM_INT);
        $stmt->bindParam(':student_id', $studentId, PDO::PARAM_INT);
        $stmt->bindParam(':course_name', $courseName, PDO::PARAM_STR);

        return $stmt->execute();

    } catch (PDOException $e) {
        // In a real application, you would log this error.
        return false;
    }
}?>
```

Question 2: "Remember Me" Cookie

Task: Write a PHP script that implements a "remember me" feature: a) Create a login form with fields for username, password, and a "remember me" checkbox. b) If the user logs in successfully and selects the "remember me" option, set a cookie to store the username. c) Set the cookie to expire after 30 days. d) If the cookie exists and is valid, display the message: “[Username] logged in”. e) If the cookie does not exist, print the message: “Username does not exist”.

Answer

```php
<?php
$cookie_name = "remember_user";

// d) Check if the cookie exists
if (isset($_COOKIE[$cookie_name])) {
    $username = htmlspecialchars($_COOKIE[$cookie_name]);
    echo "[$username] logged in.";
}
// Handle form submission
elseif ($_SERVER["REQUEST_METHOD"] == "POST") {
    // For this example, assume login is successful
    $username = $_POST['username'];

    // b) & c) Check if "remember me" is checked
    if (isset($_POST['remember_me'])) {
        $expiry = time() + (30 * 24 * 60 * 60); // 30 days
        setcookie($cookie_name, $username, $expiry, "/");
        echo "Login successful. Cookie set for 30 days.";
    } else {
        echo "Login successful. No cookie was set.";
    }
    // Refresh to demonstrate the cookie check
    header("Refresh:2; url=" . $_SERVER['PHP_SELF']);
}
// a) & e) Show form if no cookie and no submission
else {
    echo "Username does not exist.";
    display_login_form();
}

function display_login_form() {
    echo '
    <form action="" method="post">
        <p>Username: <input type="text" name="username"></p>
        <p>Password: <input type="password" name="password"></p>
        <p><input type="checkbox" name="remember_me"> Remember Me</p>
        <p><input type="submit" value="Login"></p>
    </form>';
}
?>
```

Question 3: PDO Insert

Task: Write a PHP script that connects to a MySQL database using PDO. The database should use the following parameters: host is `localhost`, database name is `online_courses`, username is `root`, and password is `root1234`. After establishing the connection, create a PHP function named `addNewCourse` which takes three parameters: course name, instructor name, and course duration. Inside the function, insert a new record into the courses table using these values. You must use prepared statements to safely insert the data and protect against SQL injections.

Answer

```php
<?php
function addNewCourse($courseName, $instructorName, $courseDuration) {
    $host = 'localhost';
    $dbname = 'online_courses';
    $username = 'root';
    $password = 'root1234';
    $dsn = "mysql:host=$host;dbname=$dbname;charset=utf8mb4";

    try {
        $pdo = new PDO($dsn, $username, $password);
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        $sql = "INSERT INTO courses (course_name, instructor_name, course_duration)
                VALUES (:course_name, :instructor_name, :course_duration)";

        $stmt = $pdo->prepare($sql);

        $stmt->bindParam(':course_name', $courseName, PDO::PARAM_STR);
        $stmt->bindParam(':instructor_name', $instructorName, PDO::PARAM_STR);
        $stmt->bindParam(':course_duration', $courseDuration, PDO::PARAM_STR);

        return $stmt->execute();

    } catch (PDOException $e) {
        return false;
    }
}
?>
```

Question 4: Distance Calculation

Task: Write a PHP script that calculates how many kilometers a person has walked based on their step count. Create variables for user's name, number of steps walked and average step length in meters (assume 0.8 meters per step). Write a function to convert the total distance to kilometers. Print a personalized message including the result.

Answer

```php
<?php
$userName = "Alex";
$stepsWalked = 12500;
$stepLengthMeters = 0.8;

function convertToKilometers($steps, $stepLength) {
    $totalMeters = $steps * $stepLength;
    return $totalMeters / 1000;
}

$distanceInKm = convertToKilometers($stepsWalked, $stepLengthMeters);

printf(
    "%s, you walked %d steps, which is %.2f kilometers.",
    $userName,
    $stepsWalked,
    $distanceInKm
);
?>
```

Question 5: Dark Mode Toggle

Task: Create a PHP script that implements a "dark mode" preference using cookies. First, display a toggle button to switch between light and dark modes. When clicked, set a cookie to remember the user's preference for 30 days. Check for this cookie when the page loads and apply the appropriate mode. Use inline CSS to change the background and text colors based on the selected mode.

Answer

```php
<?php
$cookie_name = "theme_mode";
$mode = $_COOKIE[$cookie_name] ?? 'light'; // Default to light mode

// Handle toggle request
if (isset($_POST['toggle_mode'])) {
    $mode = ($mode === 'light') ? 'dark' : 'light';
    setcookie($cookie_name, $mode, time() + (30 * 24 * 60 * 60), "/");
    header("Location: " . $_SERVER['PHP_SELF']);
    exit();
}

// Define styles based on the current mode
$style = ($mode === 'dark')
    ? 'background-color: #333; color: #f1f1f1;' // Dark mode styles
    : 'background-color: #f1f1f1; color: #333;'; // Light mode styles

// Generate the page
echo "<body style='{$style}'>";
echo "<h1>Current Mode: " . ucfirst($mode) . "</h1>";
echo '<form method="post"><button name="toggle_mode">Toggle Mode</button></form>';
echo "</body>";
?>
```

Question 6: Movie Eligibility Logic

Task: Create a PHP script that checks a person's eligibility for a movie. The script should have variables for a person's age and whether they have parental permission (true/false). If the person is 18 or older, print "You can watch any movie". If they are under 18 but have parental permission, print "You can watch PG-13 movies". Otherwise, print "You can only watch G-rated movies".

Answer

```php
<?php
function checkEligibility($age, $hasParentalPermission) {
    if ($age >= 18) {
        echo "You can watch any movie.";
    } elseif ($hasParentalPermission) { // Implies age < 18
        echo "You can watch PG-13 movies.";
    } else {
        echo "You can only watch G-rated movies.";
    }
}

$age = 15;
$permission = true;
checkEligibility($age, $permission);
?>
```

Question 7: Search Feature Logic

Task: Write a PHP script for a search feature. If the page is accessed without parameters, show a search form that submits back to the same page. If a 'query' parameter is present, display a message saying "You searched for: [query]" and include a link to clear the search and return to the form.

Answer

```php
<?php
if (isset($_GET['query']) && !empty($_GET['query'])) {
    $searchQuery = htmlspecialchars($_GET['query']);
    echo "You searched for: [$searchQuery]";
    echo '<br><a href="' . $_SERVER['PHP_SELF'] . '">Clear search</a>';
} else {
    echo '
    <form action="" method="get">
        Search: <input type="text" name="query">
        <input type="submit" value="Submit">
    </form>';
}
?>
```

Question 8: GET vs. POST Handling

Task: Create a PHP script that checks if the page was accessed via GET or POST method. If accessed via GET, display a form that submits to the same page via POST, with fields for name, email, and message. If accessed via POST, validate that all fields are filled out, then display the submitted information in a formatted way.

Answer

```php
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $name = $_POST['name'] ?? '';
    $email = $_POST['email'] ?? '';
    $message = $_POST['message'] ?? '';

    if (!empty($name) && !empty($email) && !empty($message)) {
        echo "<h2>Submission Received:</h2>";
        echo "<p><strong>Name:</strong> " . htmlspecialchars($name) . "</p>";
        echo "<p><strong>Email:</strong> " . htmlspecialchars($email) . "</p>";
        echo "<p><strong>Message:</strong><br>" . nl2br(htmlspecialchars($message)) .
            "</p>";
    } else {
        echo "<h2>Error: All fields are required.</h2>";
        display_form();
    }
} else {
    display_form();
}

function display_form() {
    echo '
    <form action="" method="post">
        <p>Name: <input type="text" name="name"></p>
        <p>Email: <input type="email" name="email"></p>
        <p>Message: <textarea name="message"></textarea></p>
        <p><input type="submit" value="Submit"></p>
    </form>';
}
?>
```

Question 9: PDO Select All

Task: Write a PHP script that connects to a MySQL database using PDO. The database parameters should be: Host: `localhost`, Database name: `students`, Username: `root`, Password: `"admin_1234"`. Create a PHP function called `getAllStudents` that retrieves all records from a 'students' table. The function should return the results as an associative array. Include proper parameter binding.

Answer

```php
<?php
function getAllStudents() {
    $host = 'localhost';
    $dbname = 'students';
    $username = 'root';
    $password = 'admin_1234';
    $dsn = "mysql:host=$host;dbname=$dbname;charset=utf8mb4";

    try {
        $pdo = new PDO($dsn, $username, $password);
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        $sql = "SELECT * FROM students";
        $stmt = $pdo->prepare($sql);
        $stmt->execute();

        return $stmt->fetchAll(PDO::FETCH_ASSOC);

    } catch (PDOException $e) {
        return false;
    }
}
?>
```

Question 10: mysqli Activity Logger

Task: Write a PHP function called `createUserActivity` that logs user activities in a database. Connect to a database (host="localhost", dbname="userportal", username="logger", password="log123") using mysqli. The function should accept parameters for `user_id`, `activity_type` (login, logout, post, etc.), and `timestamp`. Use prepared statements to insert into an 'activity_log' table and handle any potential errors.

Answer

```php
<?php
function createUserActivity($user_id, $activity_type, $timestamp) {
    $conn = new mysqli("localhost", "logger", "log123", "userportal");

    if ($conn->connect_error) {
        return false;
    }

    $sql = "INSERT INTO activity_log (user_id, activity_type, timestamp) VALUES (?, ?, ?)";

    $stmt = $conn->prepare($sql);
    if ($stmt === false) {
        $conn->close();
        return false;
    }

    $stmt->bind_param("iss", $user_id, $activity_type, $timestamp);
    $success = $stmt->execute();

    $stmt->close();
    $conn->close();

    return $success;
}
?>
```

Question 11A-B: Student Score Analysis (Part 1)

Given this array of student test scores:

```php
$scores = [
    ["name" => "Emma", "math" => 85, "science" => 92, "literature" => 78],
    ["name" => "Michael", "math" => 75, "science" => 80, "literature" => 82],
    ["name" => "Sophia", "math" => 92, "science" => 95, "literature" => 90],
    ["name" => "Jacob", "math" => 70, "science" => 65, "literature" => 74],
    ["name" => "Olivia", "math" => 88, "science" => 91, "literature" => 86]
];
```

a) Calculate and print each student's average score across all subjects.

```php
foreach ($scores as &$student) {
    $student['average'] = ($student['math'] + $student['science'] +
        $student['literature']) / 3;
    printf("%s: %.2f\n", $student['name'], $student['average']);
}
unset($student);
```

b) Determine and print the highest score in each subject.

```php
$maxMath = max(array_column($scores, 'math'));
$maxScience = max(array_column($scores, 'science'));
$maxLiterature = max(array_column($scores, 'literature'));
echo "Highest Math: $maxMath\n";
echo "Highest Science: $maxScience\n";
echo "Highest Literature: $maxLiterature\n";
```

c) Find and print all students who scored above 85 in science.

```php
foreach ($scores as $student) {
    if ($student['science'] > 85) {
        echo $student['name'] . "\n";
    }
}
```

d) Calculate the class average for each subject.

```php
$classAvgMath = array_sum(array_column($scores, 'math')) / count($scores);
$classAvgScience = array_sum(array_column($scores, 'science')) / count($scores);
$classAvgLiterature = array_sum(array_column($scores, 'literature')) / count($scores);
printf("Class Avg Math: %.2f\n", $classAvgMath);
printf("Class Avg Science: %.2f\n", $classAvgScience);
printf("Class Avg Literature: %.2f\n", $classAvgLiterature);
```

Question 11-E: Student Score Analysis (Part 2)

Given this array of student test scores:

```php
$scores = [
    ["name" => "Emma", "math" => 85, "science" => 92, "literature" => 78],
    ["name" => "Michael", "math" => 75, "science" => 80, "literature" => 82],
    ["name" => "Sophia", "math" => 92, "science" => 95, "literature" => 90],
    ["name" => "Jacob", "math" => 70, "science" => 65, "literature" => 74],
    ["name" => "Olivia", "math" => 88, "science" => 91, "literature" => 86]
];
```

e) Add a new "grade" field for each student based on their average score:

```php
foreach ($scores as &$student) {
    if ($student['average'] >= 90) $student['grade'] = 'A';
    elseif ($student['average'] >= 80) $student['grade'] = 'B';
    elseif ($student['average'] >= 70) $student['grade'] = 'C';
    elseif ($student['average'] >= 60) $student['grade'] = 'D';
    else $student['grade'] = 'F';
}
unset($student);
print_r($scores);
```

Question 12: File-Based Guestbook

Task: Create a PHP function that manages a simple file-based guest book. The function should read entries from a file called "guestbook.txt" and display all existing entries on the page.

Answer

```php
<?php
function displayGuestbook() {
    $filename = "guestbook.txt";

    if (!file_exists($filename)) {
        file_put_contents($filename, "Welcome to the guestbook!\n");
    }

    $entries = file($filename, FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);

    echo "<h2>Guestbook Entries:</h2>";
    if (empty($entries)) {
        echo "<p>The guestbook is empty.</p>";
    } else {
        echo "<ul>";
        foreach ($entries as $entry) {
            echo "<li>" . htmlspecialchars($entry) . "</li>";
        }
        echo "</ul>";
    }
}
displayGuestbook();
?>
```

Question 13: Movie Rating Average

Task: Write a PHP function that calculates the average rating of a movie based on five user reviews. Use the provided variables and do not change their values. Print a formatted message displaying the movie title and its average rating: `$movieName = "The Matrix", $rating1 = 5, $rating2 = 4, $rating3 = 3`

Answer

```php
<?php
$movieName = "The Matrix";
$rating1 = 5;
$rating2 = 4;
$rating3 = 3;

function calculateAndPrintAverage($title, ...$ratings) {
    if (count($ratings) === 0) {
        echo "$title has no ratings.";
        return;
    }
    $average = array_sum($ratings) / count($ratings);
    printf("Movie: '%s' | Average Rating: %.2f", $title, $average);
}

// Function is called with only the 3 provided ratings
calculateAndPrintAverage($movieName, $rating1, $rating2, $rating3);
?>
```

Question 14: While Loop Logic

Task: Write a PHP script that uses a while loop to find the first 5 numbers that are both greater than 100 and divisible by 9. Print each number when you find it.

Answer

```php
<?php
echo "First 5 numbers > 100 and divisible by 9:\n";
$found = 0;
$number = 101;

while ($found < 5) {
    if ($number % 9 === 0) {
        echo $number . "\n";
        $found++;
    }
    $number++;
}
?>
```

Question 15: Combining Arrays

Task: Create two arrays: one containing three subject names and another containing the corresponding scores a student received in those subjects. Use the `array_combine()` function to create an associative array with subjects as keys and scores as values. Then calculate and print the average score.

Answer

```php
<?php
$subjects = ["Mathematics", "History", "Physics"];
$scores = [88, 76, 92];

$studentScores = array_combine($subjects, $scores);

$averageScore = array_sum($studentScores) / count($studentScores);

echo "Combined Array:\n";
print_r($studentScores);
printf("\nAverage score: %.2f\n", $averageScore);
?>
```

Question 16: Array Manipulation Methods

Task: Start with this array: `$numbers = [5, 1, 9, 8, 4, 6, 2, 10, 3, 7];` a) Use array method to create a new array containing only odd numbers b) Use array method to create another array with each value squared c) Use array method to calculate the sum of all original numbers d) Find the smallest number in the array e) Print the length of the array

Answer

```php
<?php
$numbers = [5, 1, 9, 8, 4, 6, 2, 10, 3, 7];

// a) Odd numbers
$oddNumbers = array_filter($numbers, fn($n) => $n % 2 !== 0);
echo "a) Odds: " . implode(', ', $oddNumbers) . "\n";

// b) Squared values
$squaredNumbers = array_map(fn($n) => $n ** 2, $numbers);
echo "b) Squares: " . implode(', ', $squaredNumbers) . "\n";

// c) Sum of all
$sum = array_sum($numbers);
echo "c) Sum: " . $sum . "\n";

// d) Smallest number
$smallest = min($numbers);
echo "d) Min: " . $smallest . "\n";

// e) Length of array
echo "e) Length: " . count($numbers) . "\n";
?>
```

Question 17: Reading a File Line by Line

Task: Write a PHP script that reads a text file named "quotes.txt" line by line and displays each quote in an HTML list. The script should handle the case where the file doesn't exist with an appropriate error message.

Answer

```php
<?php
$filename = "quotes.txt";

if (file_exists($filename) && is_readable($filename)) {
    $fileHandle = fopen($filename, "r");
    if ($fileHandle) {
        echo "<ul>\n";
        while (($line = fgets($fileHandle)) !== false) {
            echo "  <li>" . htmlspecialchars(trim($line)) . "</li>\n";
        }
        echo "</ul>\n";
        fclose($fileHandle);
    } else {
        echo "Error: Could not open the file.";
    }
} else {
    echo "Error: The file '$filename' was not found.";
}
?>
```

Question 18: Email Signature Generator

Task: Write a PHP script that generates a personalized email signature. Create variables for name, job title, company, phone number, and email address with values of your choice. Format these details into a professional email signature with HTML tags for styling. Print the complete signature and also show the PHP code that generated it.

Answer

```php
<?php
$name = "Jane Doe";
$jobTitle = "Senior Web Developer";
$company = "Innovatech Solutions";
$phone = "+1 (555) 123-4567";
$email = "jane.doe@innovatech.com";

$signature = <<<HTML
<div style="font-family: sans-serif; border-top: 1px solid #ccc; padding-top: 5px;">
    <p><strong>{$name}</strong><br>
    <em>{$jobTitle}</em><br>
    {$company}<br>
    P: {$phone} | E: {$email}</p>
</div>
HTML;

echo $signature;

echo "<h2>PHP Code Used:</h2>";
highlight_file(__FILE__);
?>
```

Question 19: File System Functions

Task: Match each PHP file system function with its correct description.

Answer

`filesize()`	→	Returns the size of a file in bytes
`opendir()`	→	Opens a directory handle for reading
`file_exists()`	→	Checks whether a file or directory exists
`unlink()`	→	Deletes a file from the file system
`mkdir()`	→	Creates a new directory
`is_dir()`	→	Checks whether the specified path is a directory
`scandir()`	→	Returns an array of files and directories inside a directory
`fopen()`	→	Opens a file or URL for reading/writing
`fwrite()`	→	Writes data to an open file
`fclose()`	→	Closes an open file pointer
`filemtime()`	→	Returns the last modification time of a file
`rename()`	→	Renames a file or directory
`copy()`	→	Copies a file
`rewind()`	→	Rewinds the position of a file pointer to the beginning

Question 20: Tip Calculator

Task: Create a function called `calculateTip` that accepts two parameters: the bill amount and the service quality ("excellent", "good", or "fair"). The function should return a 20% tip for excellent service, 15% for good service, and 10% for fair service. Call the function with a bill of $84.00 and "good" service, and print the tip amount.

Answer

```php
<?php
function calculateTip($billAmount, $serviceQuality) {
    switch (strtolower($serviceQuality)) {
        case 'excellent': return $billAmount * 0.20;
        case 'good':    return $billAmount * 0.15;
        case 'fair':    return $billAmount * 0.10;
        default:        return 0.0; // No tip for unspecified service
    }
}

$bill = 84.00;
$service = "good";
$tipAmount = calculateTip($bill, $service);

printf("Tip for a $%.2f bill with '%s' service: $%.2f.", $bill, $service, $tipAmount);
?>
```

Question 21: Item Price Calculation

Task: Write a PHP script that creates three variables to store the name, price, and tax rate of an item. The item name must be "Bluetooth Mouse", the price must be 12, and the tax rate should be 0.08 (8%). Using these variables, calculate the final total of the purchased item by adding the tax to the price. Finally, print a checkout message formatted as follows: "Thank you for purchasing the [item]! Your total is $[price]." Make sure the total includes the tax.

Answer

```php
<?php
$itemName = "Bluetooth Mouse";
$price = 12.00;
$taxRate = 0.08;

$total = $price * (1 + $taxRate);

printf(
    "Thank you for purchasing the %s! Your total is $%.2f.",
    $itemName,
    $total
);
?>
```

Question 22: Advanced Array Manipulation

Task: Start with the array `$fruits = ["Apple", "Banana", "Orange", "Grape", "Pear"];` and perform the following operations in sequence.

Answer

```php
<?php
$fruits = ["Apple", "Banana", "Orange", "Grape", "Pear"];
echo "Initial: " . implode(', ', $fruits) . "\n";

// a) Use array_push() to add "Mango" to the end
array_push($fruits, "Mango");

// b) Then use array_unshift() to add "Kiwi" to the beginning
array_unshift($fruits, "Kiwi");
echo "After push/unshift: " . implode(', ', $fruits) . "\n";

// c) After that, use array_pop() to remove the last element
//    and use array_slice() to get only the first 3 elements
array_pop($fruits);
$fruits = array_slice($fruits, 0, 3);
echo "After pop/slice: " . implode(', ', $fruits) . "\n";

// d) Finally, reverse the array using array_reverse() and print
$fruits = array_reverse($fruits);
echo "After reverse: " . implode(', ', $fruits) . "\n";

// e) Replace the 3rd and 4th elements of the main array
//    (re-initialized) with "Papaya" and "Peach"
$fruits = ["Apple", "Banana", "Orange", "Grape", "Pear"];
array_splice($fruits, 2, 2, ["Papaya", "Peach"]);
echo "After splice on original: " . implode(', ', $fruits) . "\n";
?>
```

Question 23: Login Throttling

Task: Write a PHP script that limits the number of login attempts using sessions. First, create a login form with username and password fields. Track the number of failed login attempts in a session variable. After 3 failed attempts, display a message that says, "Too many failed attempts" and prevent further login attempts.

Answer

```php
<?php
session_start();
$max_attempts = 3;

// Initialize counter if not set
if (!isset($_SESSION['login_attempts'])) {
    $_SESSION['login_attempts'] = 0;
}

// Handle a login attempt
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // For demonstration, every attempt fails
    $_SESSION['login_attempts']++;
}

// Check if locked out
if ($_SESSION['login_attempts'] >= $max_attempts) {
    echo "Too many failed attempts.";
} else {
    // Show the login form
    $remaining = $max_attempts - $_SESSION['login_attempts'];
    echo '
    <form action="" method="post">
        <p>Login (' . $remaining . ' attempts remaining)</p>
        <p>Username: <input type="text" name="username"></p>
        <p>Password: <input type="password" name="password"></p>
        <p><input type="submit" value="Login"></p>
    </form>';
}?>
```

Question 24: PDO Search with LIKE

Task: Write a PHP script that connects to a MySQL database using PDO. The database should use the following parameters: host is localhost, database name is university, username is root, and password is root1234. After establishing the connection, create a PHP function called `searchStudents` that searches for students based on a name query. The function should use a LIKE clause to find partial matches in either first name or last name fields. Return the matching records as an associative array. Handle the case where no students are found.

Answer

```php
<?php
function searchStudents($query) {
    $dsn = "mysql:host=localhost;dbname=university;charset=utf8mb4";
    try {
        $pdo = new PDO($dsn, 'root', 'root1234');
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        $sql = "SELECT * FROM students
                WHERE first_name LIKE :query OR last_name LIKE :query";

        $stmt = $pdo->prepare($sql);
        $searchTerm = '%' . $query . '%';
        $stmt->bindParam(':query', $searchTerm, PDO::PARAM_STR);
        $stmt->execute();

        $results = $stmt->fetchAll(PDO::FETCH_ASSOC);
        return empty($results) ? [] : $results;

    } catch (PDOException $e) {
        return []; // Return empty array on error or if no students are found
    }
}
?>
```

Question 25: Age Calculation

Task: Write a function called `calculateAge` that accepts a birthdate in the format 'YYYY-MM-DD' and returns the person's age in years. Use the PHP date functions to perform the calculation. Test your function with the birthdate '2000-09-20' and print the resulting age.

Answer

```php
<?php
function calculateAge($birthdate) {
    try {
        $birthDateObj = new DateTime($birthdate);
        $currentDateObj = new DateTime('now');
        $ageInterval = $currentDateObj->diff($birthDateObj);
        return $ageInterval->y;
    } catch (Exception $e) {
        return "Error: Invalid date format.";
    }
}

$birthdate = '2000-09-20';
$age = calculateAge($birthdate);
echo "Age for birthdate $birthdate: $age years.";
?>
```