Lab 1: Basic Level Questions

1. **Hello World**
    - Write a PHP script to display "Hello, World!" on the screen.
2. **Variables and Data Types**
    - Create a PHP script that declares a variable `$name` and assigns it your name. Then, display the value of the variable.
3. **String Manipulation**
    - Write a PHP script that concatenates two strings: `$firstName = "John"` and `$lastName = "Doe"`. Display the full name.
4. **Basic Arithmetic**
    - Create a PHP script that calculates the sum, difference, product, and quotient of two numbers (`$a = 10` and `$b = 5`).
5. **Conditional Statements**
    - Write a PHP script that checks if a number is even or odd. Display the result.
6. **Loops**
    - Write a PHP script that prints numbers from 1 to 10 using a `for` loop.
7. **Arrays**
    - Create an array of fruits (`$fruits = ["Apple", "Banana", "Orange"]`) and display each fruit using a `foreach` loop.
8. **String Functions**
    - Write a PHP script that takes a string `$text = "Hello, World!"` and displays the length of the string, the uppercase version, and the lowercase version.
9. **Working with Dates**
    - Write a PHP script that displays the current date in the format `YYYY-MM-DD`.

LAB 2/3: Intermediate Level Questions

1. **Functions**
    a. Write a PHP function called `calculateArea` that takes the length and width of a rectangle as parameters and returns the area.
2. **Form Handling**
    a. Create a simple HTML form with two input fields (name and email) and a submit button. Write a PHP script to handle the form submission and display the submitted data.
3. **File Handling**

    a. Write a PHP script to create a text file named `example.txt` and write the text "This is a sample text file." into it.
4. **Working with Dates**
    a. Write a PHP script that displays the current date and time in the format: `YYYY-MM-DD HH:MM:SS`.
5. **Associative Arrays**
    a. Create an associative array `$student` with keys `name`, `age`, and `grade`. Display the values of the array.
6. **String Functions**
    a. Write a PHP script that takes a string `$text = "Hello, World!"` and displays the length of the string, the uppercase version, and the lowercase version.
7. **Include and Require**
    a. Create two PHP files: `header.php` and `footer.php`. Include these files in a main PHP file to display a complete webpage.
8. **Superglobals**
    a. Write a PHP script that uses the `$_GET` superglobal to retrieve a query parameter `name` from the URL and display a greeting message (e.g., `http://example.com?name=John`).
9. **String Replacement**
    a. Write a PHP script that replaces all occurrences of the word "apple" with "orange" in the string `$text = "I have an apple, and she has an apple too."`.
10. **Array Sorting**
    a. Create an array `$numbers = [5, 2, 9, 1, 7]`. Write a PHP script to sort the array in ascending order and display the sorted array.
11. **Working with JSON**
    a. Write a PHP script that converts an associative array `$data = ["name" => "John", "age" => 30, "city" => "New York"]` into a JSON string and then decodes it back into an array.
12. **File Reading**
    a. Write a PHP script to read the contents of a file `example.txt` and display them on the screen.
13. **Form Handling with Validation**
    a. Create a PHP script that handles a form submission with fields for `name`, `email`, and `password`. Validate the email using `filter_var()` and ensure the password is at least 8 characters long.
14. **Working with Cookies**

a. Write a PHP script that sets a cookie named `theme` with the value `dark` and expires in 7 days. Then, retrieve and display the cookie value.

15. **Session Management**
    a. Create a PHP script that starts a session, stores a variable `$_SESSION['logged_in'] = true`, and displays a message if the user is logged in.
16. **Working with Dates**
    a. Write a PHP script that calculates the difference between two dates (`$date1 = "2023-01-01"` and `$date2 = "2023-12-31"`) and displays the number of days between them.
17. **Working with `explode()` and `implode()`**
    a. Write a PHP script that splits the string `$text = "apple,banana,orange"` into an array using `explode()`, then joins the array back into a string using `implode()` with a hyphen (`-`) as the separator.

LAB 4/5: Advanced Level Questions

1. **Sessions**
    a. Write a PHP script that starts a session, stores a variable `$_SESSION['username'] = "JohnDoe"`, and displays the value on another page.
2. **Cookies**
    a. Create a PHP script that sets a cookie named `user` with the value `"Guest"` and expires in 1 hour. Then, retrieve and display the cookie value.
3. **Database Connection**
    a. Write a PHP script to connect to a MySQL database and fetch all records from a table named `users`.
4. **Form Validation**
    a. Create a PHP script that validates a form submission (name, email, and password). Ensure that the email is valid and the password is at least 8 characters long.
5. **File Upload**
    a. Write a PHP script that allows users to upload an image file and saves it to a directory on the server.
6. **Error Handling**
    a. Write a PHP script that uses `try-catch` blocks to handle a division-by-zero error.

7. **Regular Expressions**
    a. Write a PHP script that uses a regular expression to validate an email address.
8. **Object-Oriented PHP**
    a. Create a PHP class `Car` with properties `$make`, `$model`, and `$year`. Add a method `displayInfo()` that returns the car's details.
9. **Working with JSON**
    a. Write a PHP script that encodes an associative array into a JSON string and then decodes it back into an array.
10. **API Integration**
    a. Write a PHP script that makes a GET request to a public API (e.g., https://jsonplaceholder.typicode.com/posts) and displays the response.
11. **Working with cURL**
    a. Write a PHP script that uses cURL to fetch data from a public API (e.g., https://jsonplaceholder.typicode.com/posts) and displays the response.
12. **Object-Oriented PHP (OOP)**
    a. Create a PHP class `BankAccount` with properties `$accountNumber` and `$balance`. Add methods to deposit, withdraw, and display the balance.
13. **Working with Namespaces**
    a. Create a PHP script that demonstrates the use of namespaces. Define a class `User` in a namespace `App\Models` and instantiate it in another file.
14. **Working with Traits**
    a. Create a PHP trait `Loggable` with a method `log($message)` that writes a message to a file. Use this trait in a class `User`.
15. **Working with Composer**
    a. Write a PHP script that uses a package installed via Composer (e.g., `guzzlehttp/guzzle`) to make an HTTP request to an API.
16. **Working with PDO**
    a. Write a PHP script that uses PDO (PHP Data Objects) to connect to a MySQL database and fetch all records from a table named `employees`.
17. **Building a REST API**
    a. Create a simple REST API in PHP that allows users to perform CRUD operations on a `tasks` table in a MySQL database.
18. **Working with Regular Expressions**
    a. Write a PHP script that uses a regular expression to validate a phone number in the format `+1-123-456-7890`.
19. **Working with `glob()`**
    a. Write a PHP script that lists all `.txt` files in a directory using the `glob()` function.

20. **Working with `filter_var()`**
    a. Write a PHP script that uses `filter_var()` to validate an email address and a URL.
21. **Working with `DateTime`**
    a. Write a PHP script that uses the `DateTime` class to calculate the difference between two dates and display the result in years, months, and days.
22. **Building a Pagination System**
    a. Create a PHP script that fetches records from a MySQL table and displays them in a paginated format (e.g., 10 records per page).

Project-Based Questions

1. **Simple Blog System**
    a. Create a simple blog system where users can add, view, and delete blog posts. Use a MySQL database to store the posts.
2. **User Registration and Login**
    a. Build a user registration and login system. Store user data in a MySQL database and use sessions to manage user authentication.
3. **Shopping Cart**
    a. Create a simple shopping cart system where users can add products to their cart and view the total price.
4. **To-Do List**
    a. Build a to-do list application where users can add, update, and delete tasks. Store tasks in a MySQL database.
5. **Weather App**
    a. Create a PHP application that fetches weather data from a public API (e.g., OpenWeatherMap) and displays the current weather for a given city.