ASSIGNMENTS 1

1. Write a PHP script that:

- Declares variables of different data types (integer, float, string, boolean).
- Defines and uses constants.
- Performs basic arithmetic operations using operators (addition, subtraction, multiplication, division).
- Displays the result using `echo`.

N.B. Add a user input functionality (using the `$_GET` or `$_POST` methods) to let users input numbers for the operations.

2. Write a PHP script that:

- Asks for the user's age.
- Uses `if•else` statements to determine and display if the user is a child, teenager, adult, or senior citizen.
- Use the `switch` statement to assign a category based on the user's input (e.g., Student, Employee, Retired).

N.B. Use a loop to allow multiple users to input their age until a "stop" keyword is entered.

3. Create a PHP program that:

- Uses a `for` loop to print numbers from 1 to 10.
- Uses a `while` loop to sum numbers from 1 to 100.
- Uses a `do•while` loop to print numbers from 20 to 30.

Create an HTML table using PHP that displays the multiplication table of any number inputted by the user.

N.B. Combine decision•making and loops: Modify the program to ask for a user input number and print its multiplication table using a `for` loop in an HTML table.

4. Write a PHP function `calculateFactorial` that:

- Accepts an integer as an argument and returns its factorial using recursion.

• Create another function `swapValues` that:

  • Takes two variables by reference and swaps their values.

N.B.

• Create a function to check if a number is prime or not, and use a loop to print all prime numbers up to 100.

5. Create a PHP script that:

  • Takes a string as input.

  • Counts the number of words and characters in the string.

  • Searches for a specific word in the string and replaces it with another word.

  • Formats the string (e.g., converts to uppercase or lowercase).

N.B.

• Create a small login form where users input a username and password, and display appropriate messages if the string contains special characters, exceeds a length limit, or is too short.

6. Develop a PHP application that:

  • Takes the user's first name and last name as input.

  • Formats and concatenates them into a full name.

  • Uses functions to count the total characters in the full name.

  • Outputs a greeting message, such as "Hello, [Full Name]! Your name has [x] characters."

N.B.

• Add an option to reverse the string or check if it is a palindrome (ignoring spaces and capitalization).

ASSIGNMENTS:2

1. Write a PHP script that:

- Creates an indexed array of your five favorite fruits and an associative array with names as keys and ages as values.
- Accesses and displays specific elements from each array.
- Uses a `for` loop to iterate through the indexed array and displays each fruit.
- Uses a `foreach` loop to iterate through the associative array and displays each name and age.

N.B.

- Add a section where you remove an element from each array using PHP array functions like `unset()` and print the updated arrays.

2. Create a PHP program that:

- Merges two indexed arrays (e.g., array of fruits and array of vegetables) into one array and prints the result.
- Adds a new element to the merged array using `array_push()`.
- Reverses the merged array using `array_reverse()` and displays the reversed array.

N.B.

- Create a function `findMaxValue` that takes an indexed array of numbers as input and returns the maximum value using built•in functions like `max()`.

3. Build an HTML form with the following fields:

- Name (text input)
- Gender (radio buttons)
- Hobbies (checkboxes)
- Country (dropdown)
- Submit button

Write a PHP script to:

- Capture and display the submitted data, including handling multiple hobbies (checkbox values) using arrays.

• Validate if the name is not empty and display a proper message if it is.

N.B.

• Add an age field and validate if the entered value is a number between 0 and 120. If not, display an error message.

4. Design a simple HTML form that allows users to upload an image file (JPEG/PNG).

• Write a PHP script that:

• Validates if the file is of an acceptable type and within a certain size limit (e.g., less than 2MB).

• Moves the uploaded file to a directory called `uploads` and displays a message confirming the upload.

N.B.

• Check if the uploaded file already exists in the directory and, if so, rename the file by appending a timestamp before saving it.

5. Create a simple contact form with fields for the user's name, email, and message.

• Write a PHP script that:

• Captures the form data and displays a "Thank you" message on the same page.

• Redirects the user to another page called `thankyou.php` after a successful submission using the `header("Location: thankyou.php")` method.

N.B.

• In the `thankyou.php` file, display the name of the user dynamically based on the form submission data.

6. Create an HTML form that lets the user choose their favorite colors (up to three) from a list of checkboxes and submit it.

• Write a PHP script that:

• Captures the selected colors.

• Generates an HTML table where each row has the color name and a cell filled with the corresponding color.

N.B.

• Allow users to input their own color values (using a text input) and dynamically add those colors to the array when generating the table.

7. Create an HTML form that lists various products (e.g., books, electronics) with checkboxes for users to select.

- Write a PHP script that:
  - Captures the selected products in an array.
  - Displays the total number of products selected and the list of selected products.

N.B.

• Assign prices to each product and calculate the total price based on the user's selection.

ASSIGNMENTS 3

1. Create a PHP script that:

- Creates a text file called `myfile.txt`.
- Write a message (e.g., "Welcome to PHP File Handling!") to the file.
- Reads the content of the file and displays it on the webpage.
- Closes the file after reading.

N.B.

• Add an option for the user to input their own message via a form, which is then written to the file.

2. Write a PHP script that:

- Copies `myfile.txt` to a new file named `copiedfile.txt`.
- Renames `copiedfile.txt` to `renamedfile.txt`.
- Deletes the `renamedfile.txt` file upon user confirmation.

N.B.

• Create a small HTML interface with buttons to perform each operation (copy, rename, delete) using PHP.

3. Develop a PHP application that:

- Creates a directory named `uploads` if it doesn't already exist.
- Deletes the `uploads` directory when a button is clicked.

N.B.

- List all files and subdirectories inside the `uploads` directory on the webpage.

4. Create an HTML form with an input field for uploading files.

- Write a PHP script that:
  - Validates the file type (e.g., only allows image files like PNG, JPEG).
  - Moves the uploaded file to the `uploads` directory.
  - Displays a message confirming the upload and shows a preview if it's an image.

N.B.

- Add a feature that lists all uploaded files with download links.

5. Create a login form where users can input their username and password.

- Write a PHP script that:
  - Starts a session and stores the username as a session variable.
  - Sets a cookie for the user's login status that expires in 24 hours.
  - Redirects the user to a welcome page displaying the username if login is successful.

N.B.

- Implement a logout button that destroys the session and deletes the cookie.

6. Create a page where users can select products (e.g., laptop, phone, tablet) and add them to their cart.

- Store the selected products in a session variable.
- Display the cart contents on a separate page with the total number of items and a button to clear the cart (destroy the session).

N.B.

• Add a feature to remove individual items from the cart using sessions.

7. Write a PHP script that:

• Sets a cookie for the user's preferred background color when they choose a color from a dropdown menu.

• On subsequent visits, the page uses the cookie to set the background color automatically.

N.B.

• Add an option for the user to reset their preferences, which deletes the cookie.

8. Write a PHP script that:

• Connects to a MySQL database called `shop`.

• Displays a message confirming the successful connection or an error message if the connection fails.

N.B.

• Implement a form for users to input their database credentials (host, username, password) dynamically.

9. Create a database table called `products` with fields: `id`, `name`, `price`, and `category`.

• Write PHP scripts to:

• Insert a new product into the table using a form.

• Update the price of a product based on its ID.

• Delete a product based on its ID.

• Select and display all products in a table format.

N.B.

• Add a search form that allows users to search for products by name or category.

10. Create two tables: `orders` (fields: `order_id`, `product_id`, `quantity`, `order_date`) and `customers` (fields: `customer_id`, `name`, `email`).

• Write PHP scripts to:

- Insert data into both tables.
- Perform an inner join to display orders along with customer names.
- Perform an outer join to show all customers and their orders (if any).
- Perform a self-join on the `customers` table to find customers with the same name.

N.B.

• Implement a page where users can select a customer from a dropdown and see all their orders using joins.

11. Build a PHP based product catalog:

- Connect to the `shop` database and retrieve all products from the `products` table.
- Display the products in a grid layout with details like name, price, and category.
- Implement pagination if the number of products exceeds 10 per page.

N.B.

• Add a filter option to display products based on categories and price range using SQL queries.