PHP Course: Assignment Instructions (Chapters 1-5)

1. Submission Overview and Policy

This document formally establishes the mandatory format and structural requirements for the submission of the PHP course assignment, which covers content from Chapters 1 through 5. All students are required to strictly adhere to these guidelines.

2. General Formatting Specifications

The submitted document must adhere to the following formatting standards:

Specification	**Requirement**
Document Type	PDF format only
Font	Times New Roman, 12 pt
Line Spacing	1.5 lines
Text Alignment	Justified
Margins	Normal (1 inch on all four sides)

3. Assignment Structure

Your submission must be organized as follows:

- The submission must be systematically organized by chapter.
- Each chapter section must be clearly identified (e.g., "Chapter 1: PHP Basics").
- Practical components must include all required PHP code.
- All tasks require visual attachments (screenshots) of both the code and the final output.

4. Practical Tasks

Chapter 1: PHP Basics & Output

- Write a PHP script that prints your full name, university, and current semester.
- Create a script that prints the text “PHP is fun!” 10 times using a loop.
- Create a script that prints numbers from 10 to 1 in reverse order using a loop.
- Display three lines of text using one echo statement and the newline escape sequence \n.

Chapter 2: Variables & Operators

- Create name, age, and city variables and print a sentence (e.g., "My name is [Name], I am [Age] years old and I live in [City].").
- Calculate total savings from given salary and expenses variables.
- Show the sum, difference, product, and division of two numbers.
- Calculate the area of a circle based on a radius variable.
- Convert a temperature from Celsius to Fahrenheit.
- Test the equality of two different variables.
- Demonstrate the use of assignment operators (+=, -=, *=, /=).
- Use logical operators to check if an age is between 18 and 30 (inclusive).

Chapter 3: Control Structures

- Write a PHP program that checks if a number is positive, negative, or zero.
- Create a program that finds whether a number is a multiple of 4 or not.
- Use an if…elseif…else structure to assign a grade (A, B, C, D, Fail) based on marks.
- Write a program that prints all even numbers from 2 to 100 using a loop.

Chapter 4: Arrays

- Count all numbers between 1 and 50 that are divisible by 7.
- Display the sum of all odd numbers from 1 to 100.
- Print only the odd elements from a given indexed array.
- Display an associative array (representing a student record with name, ID, and major) in a neat format.
- Calculate the total and average of 5 prices stored in an array, using loops.
- Sort an array of numbers in ascending order.

Chapter 5: Functions

- Merge two arrays and count the total number of elements.
- Display a 2D array (representing 3 employees with ID, Name, and Position) in an HTML table.
- Find the second largest number in an array.
- Create a function cube() that takes a number as an argument and returns its cube.
- Create a function that takes a name as an argument and returns a message: “Hello [name], welcome!”.
- Create a function that returns the largest of three given numbers.
- Create a function that returns the sum of all even numbers in a given array.
- Create a function that takes two numbers and returns their sum, difference, and product.

5. Visual Evidence Requirements

- **Clarity and Labeling:** All screenshots must be high resolution, clear, and fully legible.
- **Content:** Each capture must visibly contain both the code executed and its associated output.
- **Captioning:** Screenshots must be correctly labeled beneath the image (e.g., Figure 1.1 - PHP Basics Output).

6. File Naming & Submission

1. **Master Folder:** Create a main folder named PHP_Assignment_<YourFullName and Student ID>.
2. **Compiled PDF:** The final report (following all formatting guidelines) should be saved as PHP_Assignment_<YourFullName>.pdf inside this folder.
3. **Source Code:** Include dedicated project folders for all practical tasks.
4. **Screenshots:** Include all source screenshots, clearly organized, within the master folder.
5. **Submission:** Compress the entire master folder into a single **ZIP archive** for upload.

7. Marking Rubric Summary

Criteria	**Description**	**Weighting**
Formatting & Structure	Adherence to academic formatting and organization.	10%
Practical Tasks	Functional code and correct output for all tasks.	60%
Visual Evidence	Clarity and correct labeling of all screenshots.	20%
Originality & Clarity	Demonstration of own work and clarity.	10%