Variables, Constants, and Data Types

1. **Dynamic Type Conversion**: Write a program that accepts two inputs from a user. If both are numeric, multiply them; if one is a string, concatenate the values; if both are strings, sort them alphabetically. Handle type conversion dynamically.
2. **Constant Functionality**: Create a constant EXCHANGE_RATE for converting between USD and EUR. Write a function that dynamically converts an array of amounts from USD to EUR and vice versa, based on user input.
3. **String Manipulation**: Write a PHP program that accepts a multi-sentence paragraph as input, splits the sentences into an array, and then reverses the order of the words in each sentence, printing the result in one string.
4. **Data Types and Error Handling**: Write a program that dynamically detects and outputs the data types of different variables passed to it. If an invalid type (like resource) is encountered, throw an exception.

Operators

5. **Complex Conditional Operations**: Write a PHP script that accepts a number. If the number is divisible by 3 and 5, print "FizzBuzz". If divisible by 3, print "Fizz". If divisible by 5, print "Buzz". However, if the number is prime, print "Prime" regardless of its divisibility.
6. **Bitwise Operators and Puzzles**: Write a PHP program that uses bitwise operators to determine if two integers are equal or not without using comparison operators (== or ===).
7. **Logical Operators in Complex Conditions**: Create a function that accepts a series of boolean conditions and performs multiple logical operations (AND, OR, NOT) on them to evaluate a complex decision tree, returning the final result.

Conditional Statements

8. **Nested Conditional Logic**: Create a dynamic discount calculator where the discount depends on user loyalty (if-else tree). If the user has been a member for over 5 years, they get a 30% discount; for 3-5 years, they get 20%; less than 3 years, they get 10%. Additional conditions include whether they are premium members, which increases their discount by 10%.
9. **Ternary Challenges**: Write a program that uses multiple nested ternary operators to evaluate a series of complex conditions based on user input (like age, membership status, and amount of purchase) and returns a personalized message based on the results.
10. **Switch Statement Puzzle**: Using a switch statement, create a simple calculator that takes a string input (like "5 + 2" or "8 * 3") and calculates the result. Handle errors for unsupported operations or invalid input formats.

Loops and Iteration

11. **Nested Loops and Patterns**: Write a PHP script that prints the following pattern using nested loops for any user-defined size:

Copy code

1

1 2

1 2 3

1 2 3 4

The size of the pyramid should be dynamically provided by the user.

12. **Dynamic Loop Generation**: Write a program that accepts an integer from the user, then generates an array of random integers between 1 and 100. Calculate the sum of the integers using a for loop, but skip every second number in the array using the continue statement.
13. **Fibonacci Sequence with Loops**: Write a PHP function that generates and returns the Fibonacci sequence up to a user-specified number using a while loop.
14. **Prime Number Finder (Optimized Loop)**: Write a PHP function that takes a number as input and returns an array of all prime numbers between 1 and the given number. Optimize the loop to minimize the number of iterations (hint: loop only until the square root of the number).

Arrays

15. **Array Sorting**: Write a PHP program that takes an associative array of product names and their prices, then sorts the array by price (highest to lowest) while maintaining the key-value associations.
16. **Multi-Dimensional Array Manipulation**: Given a multi-dimensional array that represents students and their exam scores across subjects, write a function that finds the student with the highest average score and prints their name and average.
17. **Array Search with Conditions**: Create a function that accepts an array of numbers and a threshold. The function should search the array and return all numbers greater than the threshold. Use array functions and loops in combination.
18. **Array Mapping and Filter Challenge**: Write a program that takes an array of words and filters out all words that contain vowels. Then, apply a function to the resulting array to reverse each word, and print the modified array.
19. **Associative Array Merge**: Given two associative arrays of student names and their scores, merge the two arrays into one while keeping the higher score for students who appear in both arrays.

Functions

20. **Recursive Function for Palindrome**: Write a recursive function to check if a given string is a palindrome. The function should only use recursion (no loops) and return true if the string is a palindrome, false otherwise.
21. **Anonymous Function with Array Processing**: Write an anonymous function that is passed to the array_filter() function to filter an array of numbers. The anonymous function should only return numbers that are divisible by both 3 and 7.
22. **Closure and Callback Functions**: Write a function that accepts an array of numbers and a callback function as parameters. The function should apply the callback to each element of the array and return the transformed array. Use a closure to define the callback.
23. **Function Chaining**: Implement a class with methods that manipulate strings (e.g., uppercase, reverse, add prefix/suffix). The methods should be chainable, allowing multiple operations on a string in a single statement.
24. **Dynamic Function Creation**: Write a function that accepts the name of a mathematical operation (like add, subtract, multiply, or divide) and dynamically creates a function that performs the corresponding operation on two numbers. Invoke this dynamic function.
25. **Higher-Order Functions**: Write a PHP function that takes another function as an argument and applies it to every element in a 2D array of numbers. The higher-order function should return a transformed 2D array.
26. **Memoization with Recursive Functions**: Write a recursive PHP function that computes the nth Fibonacci number. Use memoization to store intermediate results and improve performance for large n.