Operators

Q1. Custom Calculator with History

Create a calculator that supports +, -, *, /, %, and **. Also, maintain a history of operations (as an array of strings) and display them when the user types "history".

Q2. Advanced Comparison Logic

Write a function that takes two arrays and returns true if:

- **They are equal in length.**
- **Each element in array1 is either exactly equal (===) to the corresponding element in array2, or one is string '5' and the other is number 5.**

Use: ==, ===, !=, !==

Q3. Complex Boolean Expression Evaluator

Build a function that accepts 3 boolean values and evaluates a custom condition like:

js
CopyEdit
if ((a || b) && (!c || a && b)) return true;

Now turn this into a reusable boolean logic evaluator with various input sets.

Control Flow

Q4. ATM Withdrawal Simulation

Write a program where:

- **You take amount input from the user.**

- **Use if-else and switch to decide which denominations (2000, 500, 100, 50, 20, 10, 5, 1) to return.**
- **Handle edge cases like invalid input, or insufficient ATM balance.**

Q5. Grading Logic with Bonus System

Write a grading system with:

- **if-else if to assign grades: A, B, C, D, F.**
- **Use switch to give bonus marks if:**
  - **Subject is 'Math' → +10**
  - **'Science' → +5**
  - **Others → +2**

Q6. Nested Loop Pattern Generator

Use nested for, while, or do...while loops to generate patterns like:

1
12
123
1234

Or diamond/star patterns using conditions.

Functions

Q7. Function Pipeline

Create 5 small functions (like add5, square, subtract2, multiply3) and pass the result of one into another using a custom pipeline function.

pipe(5, add5, square, subtract2, multiply3);

Q8. Recursive Arrow Function

Create a recursive arrow function that calculates the factorial of a number without using any loop.

Q9. Function with Variable Parameters (Rest)

Create a function that accepts any number of arguments and:

- **Returns the sum of all even numbers.**
- **Uses .filter, .reduce, and arrow functions.**

IIFE

Q10. IIFE Score Tracker

Create a counter using an IIFE that keeps a private score and gives two functions:

- **increaseScore()**
- **getScore()**

Use closures to prevent external tampering.

Q11. Secure Auth with IIFE

Simulate login credentials using an IIFE:

- **Store username and password privately.**
- **Provide login(user, pass) function to validate.**
- **changePassword(old, new) method that works only if old password is valid.**

Q12. Real-Time Order Summary App

Build a function-driven JS mini app where:

- **User can addItem(price), removeItem(index), and checkout().**
- **Use if-else, switch, ternary to apply discounts based on amount.**
- **Use arrow functions, closures (private cart), loops for rendering.**