**JavaScript OOP Practice Questions**

**A. Classes and Objects (All Types of Object Creation)**

1. Create an object using **object literal** to store details of a car (brand, model, year). Add a method to display car details.
2. Create an object using **new Object()** to store book details (title, author, price). Add a method to print details.
3. Define a **constructor function** Person that creates objects with properties name, gender and date of birth. Calculate age based on date of birth and display all details. Create 3 objects using it.
4. Create an object using **Object.create()** with a prototype object product having a methods details(), calculate() and display()and it has product id, name, quantity and price. Calculate method calculates total amount and discount (20% of the total amount and payable amount)
5. Define a **class Student** with properties id, name, and marks. Create multiple objects and display details.

**B. Static Members**

1. Create a class MathHelper with a **static method** add(x, y) and multiply(x, y). Demonstrate usage without creating an object.
2. Define a **class Student** with properties id, name, and 3 marks. Calculate total, average, result and grade. Based on the result (‘Pass’ or ‘Fail’) count the number of pass and failure in the class. Create multiple objects and display all the details.

**C. Inheritance**

1. Create a base class Employee with id, name and date of join. Derive a class Manager that adds department. Display details using inheritance.
2. Demonstrate **method overriding**: Create a base class Shape with area() method, then override it in Circle and Rectangle classes.
3. Show an example of **multi-level inheritance**: Person → Employee → Manager.

**D. Abstract Classes (Simulation in JS)**

1. Create an abstract class Vehicle with a method startEngine(). Throw an error if someone tries to instantiate Vehicle directly.
2. Extend the Vehicle class into Car and Bike classes, each implementing its own startEngine() method.
3. Create a Person abstract class with name, date of birth and address and display ( ) function which is abstract. Create Employee class inherits Person class and it contains date of join and department. Calculate age, experience and display all the details using display() along with these data.

**E. Setters & Getters**

1. Create a class Account with account number, customer name, account type and a private balance. Use **get** to return balance and **set** to update balance (with validation).
2. Create a class Temperature with a property Celsius. Provide a getter to return temperature in Fahrenheit by passing the Fahrenheit value.

**F. Polymorphism**

**(i) Overloading (not natively supported, simulate with default values/arguments)**

1. Create a class Calculator with an add() and divide() methods. add() that behaves differently if called with 2, 3, or 4 arguments. Divide() method divides when both inputs are non-zero values. If the second value = 0, throw the error. If more than 2 inputs are passed, error message should be raised to indicate too many inputs to process the data.

**(ii) Overriding**

1. Create a class Person with required properties and display( ) method. Override it in Student and Employee classes. Student class contains student result and grade additionally. Employee class contains department and salary additionally. Override display() method to print the corresponding details in the parent and the derived classes.
2. Create a class Animal with a makeSound() method. Override it in Dog and Cat classes. Write a program to demonstrate runtime polymorphism: call the makeSound() method on different objects stored in an array of Animal type.