Coding Set 5

---

1. Student Result Management System

Concepts: Array of objects, data input/output, aggregation logic

Problem:

Create a class Student with members: rollNo, name, marks[5], and total.

- Use an array of Student objects to store data of n students.
- Calculate and display the total marks and the student with the **highest total**.

Hint:
Loop through array, call member functions for input and calculation.

---

2. Employee Salary Slip Generator

Concepts: Constructor overloading, array of objects, computation logic

Problem:

Define a class Employee with data members id, name, basicSalary, hra, da, and netSalary.

- Overload constructors: one default and one parameterized.
- Store records of n employees in an array.
- Calculate and display netSalary = basicSalary + hra + da.
- Display employee(s) earning above ₹50,000.

---

3. Library Book Management

Concepts: Array of objects, searching, string handling

Problem:

Create a class Book with members: bookID, title, author, and price.

- Read details for n books into an array.
- Implement a function to **search for a book by author name**.
- Display all books written by that author.

Extension:
Allow partial match search using string::find().

---

4. Cricket Player Statistics

Concepts: Array of objects, sorting based on member variable

Problem:

Design a class Cricketer with members: playerName, matches, runs, average.

- Read data for n players.
- Calculate average = runs / matches.
- Sort and display players in **descending order of average runs**.

Hint:
Use a **sorting algorithm (like bubble sort)** with object swapping.

---

5. Inventory Stock Update

Concepts: Array of objects, operator overloading

Problem:

Define a class Item with members: code, name, and quantity.

- Overload the + operator to **add quantities** of items with the same code.
- Create an array of items.
- Merge stock from two inventories (two arrays of objects) into a third array.

Hint:
Use operator overloading with an array-based merging logic.

---

6. Flight Reservation System

Concepts: Dynamic array of objects, object filtering

Problem:

Create a class Flight with data members: flightNo, source, destination, seatsAvailable.

- Allocate flights dynamically (new Flight[n]).
- Display all flights going to a particular destination entered by user.
- Update seat count when a booking is made.

Hint:
Use functions for searching and seat modification.

---

7. Car Dealership Records

Concepts: Array of objects, aggregation, string matching

Problem:

Make a class Car with data members: model, company, price.

- Store n car objects.
- Display all cars of a particular company.
- Find and display the **most expensive car** in the array.

Extension:
Allow case-insensitive search using transform().

---

8. Online Shopping Cart

Concepts: Array of objects, composition, discount logic

Problem:

Define a class Product with attributes: id, name, price, discount.

- Create an array of objects representing cart items.
- Compute the **total amount payable** after applying discounts.
- Display products sorted by **final price (after discount)**.

Hint:
Use (price - (price * discount / 100)) for each product.
Implement a function to compute the total bill.

---

9. Complex Number Operations using Array of Objects

Concepts: Array of objects, operator overloading, aggregation

Problem:

Create a Complex class with data members: real and imag.

- Overload the + and * operators.
- Create an array of Complex objects.
- Compute the **sum** and **product** of all complex numbers in the array.

Hint:
Use loop:

sum = sum + arr[i];

product = product * arr[i];

---

10. Student Result Ranking System

Concepts: Array of objects, sorting, encapsulation, file handling (optional)

Problem:

Design a class Student with members:
rollNo, name, marks[3], total, and grade.

- Calculate total and grade (A for ≥90, B for 80–89, etc.).
- Sort the array by total marks (descending order).
- Display the **rank list** (1st, 2nd, 3rd...).
- (Optional) Write the rank list to a text file.

Hint:
Use encapsulation (private members, public access functions).
Use a **sorting algorithm** and maintain ranks dynamically.