1) Design a C# console app that uses a jagged array to store data for Instagram posts by multiple users. Each user can have a different number of posts,

and each post stores a caption and number of likes.

You have N users, and each user can have M posts (varies per user).

Each post has:

A caption (string)

A number of likes (int)

Store this in a jagged array, where each index represents one user's list of posts.

Display all posts grouped by user.

No file/database needed — console input/output only.

Example output

Enter number of users: 2

User 1: How many posts? 2

Enter caption for post 1: Sunset at beach

Enter likes: 150

Enter caption for post 2: Coffee time

Enter likes: 89

User 2: How many posts? 1

Enter caption for post 1: Hiking adventure

Enter likes: 230

--- Displaying Instagram Posts ---

User 1:

Post 1 - Caption: Sunset at beach | Likes: 150

Post 2 - Caption: Coffee time | Likes: 89

User 2:

Post 1 - Caption: Hiking adventure | Likes: 230

Test case

| User | Number of Posts | Post Captions | Likes |

| ---- | --------------- | -------------------- | ---------- |

| 1 | 2 | "Lunch", "Road Trip" | 40, 120 |

| 2 | 1 | "Workout" | 75 |

| 3 | 3 | "Book", "Tea", "Cat" | 30, 15, 60 |

 Preparation

Create the Employee class as below.

class Employee

    {

        int id, age;

        string name;

        double salary;

        public Employee()

        {

        }

        public Employee(int id, int age, string name, double salary)

        {

            this.id = id;

            this.age = age;

            this.name = name;

            this.salary = salary;

        }

        public void TakeEmployeeDetailsFromUser()

        {

            Console.WriteLine("Please enter the employee ID");

            id = Convert.ToInt32(Console.ReadLine());

            Console.WriteLine("Please enter the employee name");

            name = Console.ReadLine();

            Console.WriteLine("Please enter the employee age");

            age = Convert.ToInt32(Console.ReadLine());

            Console.WriteLine("Please enter the employee salary");

            salary = Convert.ToDouble(Console.ReadLine());

        }

        public override string ToString()

        {

            return "Employee ID : " + id + "\nName : " + name + "\nAge : " + age + "\nSalary : " + salary;

        }

        public int Id { get => id; set => id = value; }

        public int Age { get => age; set => age = value; }

        public string Name { get => name; set => name = value; }

        public double Salary { get => salary; set => salary = value; }

    }

C# Collection-Based Employee Management - Question Set

# ✅ Preparation

Create the Employee class with the following properties and methods:  
- Fields: id, age, name, salary  
- Constructor(s)  
- TakeEmployeeDetailsFromUser method  
- Override ToString() method  
- Use auto-properties

# 🟩 Easy Level

1. 1. Promotion List Collection  
   Create a C# console application with a class named EmployeePromotion that:  
   - Takes employee names in the order of their eligibility for promotion.  
   - Stores names in a collection that maintains insertion order.
2. 2. Find Position in Promotion List  
   - Given an employee name, find and print their position in the promotion list.
3. 3. Trim Excess Memory  
   - Display the current capacity of the list.  
   - Optimize the list to use only the memory required (remove excess capacity).
4. 4. Display Promoted Employees in Ascending Order  
   - Print all promoted employee names sorted in ascending alphabetical order.

# 🟨 Medium Level

1. 1. Store Employee Objects in a Collection  
   - Take employee details using the Employee class.  
   - Store them in a collection that allows retrieval by employee id.  
   - Ensure no duplicate or null IDs.
2. 2. Sort Employees by Salary  
   - Implement IComparable in Employee class.  
   - Store employees in a list and sort based on salary.
3. 3. Find Employee by ID using LINQ  
   - Given an employee ID, use a LINQ query with where to find and print details.
4. 4. Find All Employees with Given Name  
   - Accept a name from the user and find all employees with that name.
5. 5. Find Employees Older Than a Given Employee  
   - Given an employee name or ID, find and list all employees older than them.

# 🟥 Hard Level

1. 1. Complete Employee Management System  
   Build a menu-driven console application that allows the user to:  
   - Add an employee  
   - Print all employee details  
   - Modify an employee’s details (except ID)  
   - Find and print employee details by ID  
   - Delete an employee by ID  
     
   Requirement:  
   Handle all invalid inputs and cases gracefully (e.g., non-existing ID).