**Intermediate C# Problem Solving Programs - Detailed Questions with I/O Examples**

**Category 1: Arrays & Collections (Intermediate) (Problems 1-15)**

1. **Second Largest Element in Array:**
   * **Question:** Given an integer array, find the second largest element without sorting the entire array.
   * **Input:** int[] arr = { 10, 5, 20, 15, 25 };
   * **Output:** 20
   * **Input:** int[] arr = { 5, 5, 5, 5 };
   * **Output:** No second largest element (or handle as per specific requirement)
2. **Frequency of Each Element in Array:**
   * **Question:** Write a program to count the frequency of each element in an integer array. Use a Dictionary or similar approach.
   * **Input:** int[] arr = { 1, 2, 2, 3, 1, 4, 2 };
   * **Output:**
   * 1: 2 times
   * 2: 3 times
   * 3: 1 time
   * 4: 1 time
3. **Rotate Array Elements:**
   * **Question:** Given an array and a number k, rotate the array elements to the right by k positions. (e.g., [1, 2, 3, 4, 5] rotated by 2 becomes [4, 5, 1, 2, 3]).
   * **Input:** int[] arr = { 1, 2, 3, 4, 5 }; int k = 2;
   * **Output:** [4, 5, 1, 2, 3]
4. **Find Missing Number in Sequence:**
   * **Question:** Given an array containing n distinct numbers taken from 0, 1, 2, ..., n, find the one number that is missing from the array.
   * **Input:** int[] arr = { 3, 0, 1 }; (n=3, numbers 0-3)
   * **Output:** 2
   * **Input:** int[] arr = { 9, 6, 4, 2, 3, 5, 7, 0, 1 }; (n=9, numbers 0-9)
   * **Output:** 8
5. **Remove Duplicates from Sorted Array (In-Place):**
   * **Question:** Given a sorted array, remove the duplicates in-place such that each element appears only once and returns the new length. You cannot use extra space for another array.
   * **Input:** int[] nums = { 1, 1, 2, 2, 3, 4, 4, 5 };
   * **Output:** 5 (The array nums should be modified to [1, 2, 3, 4, 5, \_, \_, \_])
6. **Subarray with Given Sum:**
   * **Question:** Given an unsorted array of non-negative integers, find a continuous sub-array which adds to a given sum.
   * **Input:** int[] arr = { 1, 4, 20, 3, 10, 5 }; int sum = 33;
   * **Output:** Sum found between indexes 2 and 4 (elements 20, 3, 10)
   * **Input:** int[] arr = { 1, 2, 3, 7, 5 }; int sum = 12;
   * **Output:** Sum found between indexes 1 and 3 (elements 2, 3, 7)
7. **Find Pair with Given Sum (Two Sum Problem):**
   * **Question:** Given an array of integers, return indices of the two numbers such that they add up to a specific target. Assume that each input would have exactly one solution, and you may not use the same element twice.
   * **Input:** int[] nums = { 2, 7, 11, 15 }; int target = 9;
   * **Output:** [0, 1] (because nums[0] + nums[1] == 9)
8. **Intersection of Two Arrays:**
   * **Question:** Given two integer arrays, return an array of their intersection. Each element in the result must be unique.
   * **Input:** int[] nums1 = { 1, 2, 2, 1 }; int[] nums2 = { 2, 2 };
   * **Output:** [2]
   * **Input:** int[] nums1 = { 4, 9, 5 }; int[] nums2 = { 9, 4, 9, 8, 4 };
   * **Output:** [9, 4] (order doesn't matter)
9. **Union of Two Arrays:**
   * **Question:** Given two integer arrays, return an array of their union. Each element in the result must be unique.
   * **Input:** int[] nums1 = { 1, 2, 2, 3 }; int[] nums2 = { 2, 3, 4, 5 };
   * **Output:** [1, 2, 3, 4, 5] (order doesn't matter)
10. **Diagonal Sum of a Matrix:**
    * **Question:** Calculate the sum of the main diagonal and anti-diagonal elements of a square matrix. For odd-sized matrices, the center element should only be counted once.
    * **Input:**
    * int[,] matrix = {
    * { 1, 2, 3 },
    * { 4, 5, 6 },
    * { 7, 8, 9 }
    * };
    * **Output:** Main Diagonal Sum: 15 (1+5+9), Anti-Diagonal Sum: 15 (3+5+7) (Total unique sum: 21)
11. **Transpose a Matrix:**
    * **Question:** Given a matrix (2D array), return its transpose. The transpose of a matrix flips the matrix over its main diagonal, switching the row and column indices of the matrix.
    * **Input:**
    * int[,] matrix = {
    * { 1, 2, 3 },
    * { 4, 5, 6 }
    * };
    * **Output:**
    * { { 1, 4 },
    * { 2, 5 },
    * { 3, 6 } }
12. **Find Nth Smallest Element:**
    * **Question:** Given an unsorted array and an integer n, find the Nth smallest element in the array.
    * **Input:** int[] arr = { 7, 10, 4, 3, 20, 15 }; int n = 3;
    * **Output:** 7 (The 3rd smallest element)
13. **Count Inversions in an Array:**
    * **Question:** Given an array, count the number of inversions. An inversion is a pair of indices (i, j) such that i < j and arr[i] > arr[j].
    * **Input:** int[] arr = { 2, 4, 1, 3, 5 };
    * **Output:** 3 (Pairs: (2,1), (4,1), (4,3))
14. **Leaders in an Array:**
    * **Question:** An element is a leader if it is greater than all the elements to its right. The rightmost element is always a leader. Find all leaders in a given array.
    * **Input:** int[] arr = { 16, 17, 4, 3, 5, 2 };
    * **Output:** 17, 5, 2
15. **Maximum Subarray Sum (Kadane's Algorithm):**
    * **Question:** Find the contiguous sub-array within an array (containing at least one number) which has the largest sum.
    * **Input:** int[] arr = { -2, 1, -3, 4, -1, 2, 1, -5, 4 };
    * **Output:** 6 (corresponding to the sub-array [4, -1, 2, 1])

**Category 2: Strings & Characters (Intermediate) (Problems 16-25)**

1. **First Non-Repeating Character in a String:**
   * **Question:** Given a string, find the first non-repeating character in it and return its index. If it doesn't exist, return -1.
   * **Input:** "leetcode"
   * **Output:** 0 (l is at index 0)
   * **Input:** "loveleetcode"
   * **Output:** 2 (v is at index 2)
   * **Input:** "aabb"
   * **Output:** -1
2. **Remove Characters from First String Present in Second String:**
   * **Question:** Given two strings, remove all characters from the first string that are present in the second string.
   * **Input:** string s1 = "geeksforgeeks"; string s2 = "mask";
   * **Output:** "geeforgee"
3. **Check if String is Anagram of Palindrome:**
   * **Question:** Given a string, check if it is an anagram of a palindrome. (A palindrome is a word or phrase that reads the same forwards and backwards. An anagram is a word or phrase formed by rearranging the letters of another).
   * **Input:** "aabbc"
   * **Output:** True (e.g., "abcba")
   * **Input:** "aabbcd"
   * **Output:** False
4. **Longest Common Prefix:**
   * **Question:** Write a function to find the longest common prefix string amongst an array of strings. If there is no common prefix, return an empty string "".
   * **Input:** string[] strs = { "flower", "flow", "flight" };
   * **Output:** "fl"
   * **Input:** string[] strs = { "dog", "racecar", "car" };
   * **Output:** ""
5. **Run Length Encoding:**
   * **Question:** Implement run-length encoding. For example, "AAABBC" should become "A3B2C1".
   * **Input:** "AAABBCDDD"
   * **Output:** "A3B2C1D3"
6. **String Permutations (Simple):**
   * **Question:** Generate all permutations of a given string. (For short strings due to complexity).
   * **Input:** "ABC"
   * **Output:** (Order may vary)
   * ABC
   * ACB
   * BAC
   * BCA
   * CAB
   * CBA
7. **Validate IP Address (Simplified):**
   * **Question:** Given a string, determine if it is a valid IPv4 address (simplified criteria: four numbers separated by dots, each number between 0 and 255 inclusive, no leading zeros for numbers > 0, no extra characters).
   * **Input:** "192.168.1.1"
   * **Output:** True
   * **Input:** "256.1.2.3"
   * **Output:** False
   * **Input:** "192.168.01.1"
   * **Output:** False
   * **Input:** "192.168.1.1."
   * **Output:** False
8. **Group Anagrams:**
   * **Question:** Given an array of strings, group anagrams together. You can return the answer in any order.
   * **Input:** string[] strs = { "eat", "tea", "tan", "ate", "nat", "bat" };
   * **Output:** (Order of groups/strings within groups may vary)
   * [
   * ["ate","eat","tea"],
   * ["nat","tan"],
   * ["bat"]
   * ]
9. **Longest Palindromic Substring:**
   * **Question:** Given a string s, return the longest palindromic substring in s.
   * **Input:** "babad"
   * **Output:** "bab" (or "aba")
   * **Input:** "cbbd"
   * **Output:** "bb"
10. **String Compression (using StringBuilder):**
    * **Question:** Implement a method to perform basic string compression using the counts of repeated characters. For example, the string "aabcccccaaa" would become "a2b1c5a3". If the "compressed" string would not become smaller than the original string, your method should return the original string.
    * **Input:** "aabcccccaaa"
    * **Output:** "a2b1c5a3"
    * **Input:** "abcdef"
    * **Output:** "abcdef"

**Category 3: Object-Oriented Programming (OOP) - Intermediate (Problems 26-35)**

1. **Custom Exception Handling:**
   * **Question:** Create a custom exception class (e.g., InvalidAgeException). Implement a Person class where setting a negative age throws this custom exception. Demonstrate catching it in Main.
2. **Sealed Class/Methods:**
   * **Question:** Demonstrate the use of the sealed keyword for a class (cannot be inherited) and a method (cannot be overridden). Explain when it's useful.
3. **Extension Methods:**
   * **Question:** Write an extension method for the string class called WordCount() that returns the number of words in the string.
   * **Input:** string sentence = "This is a test sentence.";
   * **Output:** sentence.WordCount() should return 5.
4. **Static Constructor and Static Fields:**
   * **Question:** Create a class with a static constructor that initializes a static counter. Demonstrate how it behaves (executes once, before any instance creation or static member access).
5. **Method Chaining:**
   * **Question:** Design a class (e.g., CarBuilder) that allows for method chaining to configure an object before creation (e.g., new CarBuilder().SetMake("Toyota").SetModel("Camry").SetYear(2023).Build();).
6. **Indexer Property:**
   * **Question:** Create a ShoppingCart class that allows accessing Product objects by their name (string) using an indexer, similar to how you would access elements in an array or dictionary (cart["Laptop"]).
7. **Operator Overloading:**
   * **Question:** Create a Vector class with X and Y properties. Overload the + operator to perform vector addition (add corresponding X and Y components).
   * **Input:** Vector v1 = new Vector(1, 2); Vector v2 = new Vector(3, 4); Vector v3 = v1 + v2;
   * **Output:** v3.X = 4, v3.Y = 6
8. **Partial Classes and Methods:**
   * **Question:** Demonstrate the use of partial keyword to split a class definition across multiple files (conceptually, just show two partial class definitions in one file). Also, briefly explain partial methods.
9. **Delegates and Events (Basic):**
   * **Question:** Define a simple delegate for a method that takes no parameters and returns void. Declare an event based on this delegate in a class. Raise the event and demonstrate subscribing a method to it.
10. **Generic Class (Basic):**
    * **Question:** Create a simple generic class Box<T> that can hold any type of item. It should have methods to SetItem(T item) and GetItem(). Demonstrate its use with Box<int> and Box<string>.

**Category 4: Algorithms & Data Structures (Intermediate) (Problems 36-50)**

1. **Merge Sort Algorithm:**
   * **Question:** Implement the Merge Sort algorithm to sort an array of integers.
   * **Input:** int[] arr = { 38, 27, 43, 3, 9, 82, 10 };
   * **Output:** [3, 9, 10, 27, 38, 43, 82]
2. **Quick Sort Algorithm:**
   * **Question:** Implement the Quick Sort algorithm to sort an array of integers.
   * **Input:** int[] arr = { 10, 80, 30, 90, 40, 50, 70 };
   * **Output:** [10, 30, 40, 50, 70, 80, 90]
3. **Implement Queue using Two Stacks:**
   * **Question:** Implement a Queue data structure using two stacks.
   * **Operations:** Enqueue(item), Dequeue(), Peek(), IsEmpty().
4. **Implement Stack using Two Queues:**
   * **Question:** Implement a Stack data structure using two queues.
   * **Operations:** Push(item), Pop(), Peek(), IsEmpty().
5. **Check for Palindrome Linked List (Conceptual/Array Simulation):**
   * **Question:** Given a sequence of characters (simulated as an array for simplicity, as actual linked lists are more advanced), determine if it forms a palindrome.
   * **Input:** char[] list = {'R', 'A', 'C', 'E', 'C', 'A', 'R'};
   * **Output:** True
   * **Input:** char[] list = {'A', 'B', 'C'};
   * **Output:** False
6. **Smallest Substring Containing All Characters:**
   * **Question:** Given a string text and a string pattern, find the smallest substring in text that contains all characters of pattern (characters in pattern don't need to be unique).
   * Input: text = "ADOBECODEBANC", pattern = "ABC"
   * Output: "BANC"
7. **Maximum Profit from Stock (One Transaction):**
   * **Question:** You are given an array prices where prices[i] is the price of a given stock on the i-th day. You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock. Return the maximum profit. If you cannot achieve any profit, return 0.
   * **Input:** int[] prices = { 7, 1, 5, 3, 6, 4 };
   * **Output:** 5 (Buy on day 2 (price=1) and sell on day 5 (price=6), profit = 6-1 = 5)
   * **Input:** int[] prices = { 7, 6, 4, 3, 1 };
   * **Output:** 0
8. **Longest Increasing Subsequence (Length only):**
   * **Question:** Given an unsorted array of integers, find the length of the longest increasing subsequence.
   * **Input:** int[] nums = { 10, 9, 2, 5, 3, 7, 101, 18 };
   * **Output:** 4 (LIS is [2, 3, 7, 101] or [2, 5, 7, 101])
9. **Valid Parentheses (Multiple Types):**
   * **Question:** Given a string s containing just the characters '(', ')', '{', '}', '[', ']', determine if the input string is valid. (This is the advanced balanced brackets problem from the previous list).
   * **Input:** "()[]{}"
   * **Output:** True
   * **Input:** "([{}])"
   * **Output:** True
   * **Input:** "{[()]}"
   * **Output:** True
   * **Input:** "([)]"
   * **Output:** False
   * **Input:** "{["
   * **Output:** False
10. **Number of Islands (Simplified Grid Traversal):**
    * **Question:** Given a 2D binary grid (matrix) of '1's (land) and '0's (water), count the number of islands. An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.
    * **Input:**
    * char[,] grid = {
    * {'1','1','0','0','0'},
    * {'1','1','0','0','0'},
    * {'0','0','1','0','0'},
    * {'0','0','0','1','1'}
    * };
    * **Output:** 3
11. **Rotate Image (90 Degrees Clockwise):**
    * **Question:** You are given an n x n 2D matrix representing an image. Rotate the image by 90 degrees (clockwise) in-place.
    * **Input:**
    * int[,] matrix = {
    * { 1, 2, 3 },
    * { 4, 5, 6 },
    * { 7, 8, 9 }
    * };
    * **Output:**
    * { { 7, 4, 1 },
    * { 8, 5, 2 },
    * { 9, 6, 3 } }
12. **Find all Paths from Top-Left to Bottom-Right in a Grid (without obstacles):**
    * **Question:** Given an m x n grid, find the number of unique paths from the top-left corner (0,0) to the bottom-right corner (m-1, n-1). You can only move down or right.
    * **Input:** m = 3, n = 7
    * **Output:** 28 (using dynamic programming or combinatorics)
13. **Check if a Number is Power of Two:**
    * **Question:** Given an integer n, return true if it is a power of two. Otherwise, return false. An integer n is a power of two if there exists an integer x such that n == 2^x. (Solve without loops/recursion if possible).
    * **Input:** 1 -> True
    * **Input:** 16 -> True
    * **Input:** 3 -> False
    * **Input:** 0 -> False
    * **Input:** -16 -> False
14. **Find Peak Element:**
    * **Question:** A peak element is an element that is strictly greater than its neighbors. Given a 0-indexed integer array nums, find a peak element and return its index. If the array contains multiple peaks, return the index to any of the peaks. (Consider edge cases for first/last elements).
    * **Input:** int[] nums = { 1, 2, 3, 1 };
    * **Output:** 2 (index of 3)
    * **Input:** int[] nums = { 1, 2, 1, 3, 5, 6, 4 };
    * **Output:** 1 (index of 2) or 5 (index of 6)
15. **Container With Most Water:**
    * **Question:** You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the i-th line are (i, 0) and (i, height[i]). Find two lines that together with the x-axis form a container, such that the container contains the most water. Return the maximum amount of water a container can store.
    * **Input:** int[] height = { 1, 8, 6, 2, 5, 4, 8, 3, 7 };
    * **Output:** 49 (between height 8 at index 1 and height 7 at index 8)

**Category 5: General Logic & Problem Solving (Intermediate) (Problems 51-60)**

1. **Find Duplicate in Array (without extra space/modifying array):**
   * **Question:** Given an array nums containing n + 1 integers where each integer is between 1 and n (inclusive), prove that at least one duplicate number must exist. Assume that there is only one duplicate number, find the duplicate number. You must solve the problem without modifying the array nums and uses only constant extra space.
   * Input: int[] nums = { 1, 3, 4, 2, 2 };
   * Output: 2
   * **Input:** int[] nums = { 3, 1, 3, 4, 2 };
   * **Output:** 3
2. **Product of Array Except Self:**
   * **Question:** Given an integer array nums, return an array answer such that answer[i] is equal to the product of all the elements of nums except nums[i]. You must write an algorithm that runs in O(n) time and without using the division operation.
   * **Input:** int[] nums = { 1, 2, 3, 4 };
   * **Output:** [24, 12, 8, 6]
3. **Move Zeros to End:**
   * **Question:** Given an integer array nums, move all 0's to the end of it while maintaining the relative order of the non-zero elements. You must do this in-place without making a copy of the array.
   * **Input:** int[] nums = { 0, 1, 0, 3, 12 };
   * **Output:** [1, 3, 12, 0, 0]
4. **FizzBuzz (Advanced):**
   * **Question:** Write a program that prints numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz". Implement this using a cleaner design (e.g., separate methods for checks or using Func<int, string>).
   * **Input:** (Implicit: numbers 1 to 100)
   * **Output:** (Partial)
   * 1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, FizzBuzz, ...
5. **Pangram Checker:**
   * **Question:** A pangram is a sentence containing every letter of the alphabet at least once. Write a program to check if a given string is a pangram.
   * **Input:** "The quick brown fox jumps over the lazy dog"
   * **Output:** True
   * **Input:** "The quick brown fox jumps over the lazy cat"
   * **Output:** False
6. **Reverse Words in a String:**
   * **Question:** Given a string s, reverse the order of words. Each word is separated by a single space. You may assume no leading or trailing spaces.
   * **Input:** "the sky is blue"
   * **Output:** "blue is sky the"
   * **Input:** " hello world "
   * **Output:** "world hello" (Handle multiple spaces and trim)
7. **Check if two strings are Rotations of each other:**
   * **Question:** Given two strings, s1 and s2, check if s2 is a rotation of s1 using only one call to a substring method (or equivalent string check).
   * **Input:** s1 = "waterbottle", s2 = "erbottlewat"
   * **Output:** True
   * **Input:** s1 = "abc", s2 = "bca"
   * **Output:** True
   * **Input:** s1 = "abc", s2 = "acb"
   * **Output:** False
8. **Trailing Zeros in Factorial:**
   * **Question:** Given an integer n, return the number of trailing zeroes in n!.
   * **Input:** 3 (3! = 6)
   * **Output:** 0
   * **Input:** 5 (5! = 120)
   * **Output:** 1
   * **Input:** 25
   * **Output:** 6 (25! has 6 trailing zeros)
9. **Find the Single Number:**
   * **Question:** Given a non-empty array of integers nums, every element appears twice except for one. Find that single one. Your algorithm should have a linear runtime complexity and use only constant extra space.
   * **Input:** int[] nums = { 2, 2, 1 };
   * **Output:** 1
   * **Input:** int[] nums = { 4, 1, 2, 1, 2 };
   * **Output:** 4
10. **Missing Ranges:**
    * **Question:** Given a sorted integer array nums that does not contain duplicates, where the range of elements is [lower, upper] inclusive, return its missing ranges.
    * **Input:** nums = [0, 1, 3, 50, 75], lower = 0, upper = 99
    * **Output:** ["2", "4->49", "51->74", "76->99"]

**Category 6: LINQ & Functional Programming Basics (Problems 61-70)**

1. **Filter and Project with LINQ:**
   * **Question:** Given a list of Product objects (each with Name, Price, Category), use LINQ to:
     1. Filter products where Price is greater than 50.
     2. Select only the Name and Price of these products into an anonymous type.
   * **Input (Code Example):**

C#

public class Product

{

public string Name { get; set; }

public decimal Price { get; set; }

public string Category { get; set; }

public Product(string name, decimal price, string category)

{

Name = name;

Price = price;

Category = category;

}

}

List<Product> products = new List<Product>

{

new Product("Laptop", 1200, "Electronics"),

new Product("Mouse", 25, "Electronics"),

new Product("Keyboard", 75, "Electronics"),

new Product("Book", 15, "Books")

};

* + **Output (Conceptual):**
  + Name: Laptop, Price: 1200
  + Name: Keyboard, Price: 75

*(Actual output would be printed to console, e.g., by iterating through the resulting collection)*

1. **Grouping with LINQ:**
   * **Question:** Group the Product objects from the previous question by Category and print the count of products in each category.
   * **Input (Code Example):** (Same products list as P61)
   * **Output (Conceptual):**
   * Electronics: 3 products
   * Books: 1 product
2. **Ordering with LINQ:**
   * **Question:** Sort the Product objects by Price in descending order, then by Name in ascending order for products with the same price.
   * **Input (Code Example):**

C#

List<Product> products = new List<Product>

{

new Product("Laptop", 1200, "Electronics"),

new Product("Mouse", 25, "Electronics"),

new Product("Keyboard", 75, "Electronics"),

new Product("Book", 15, "Books"),

new Product("Headphones", 75, "Electronics") // Added for demonstration of secondary sort

};

* + **Output (Conceptual - expected order of printed product names and prices):**
  + Laptop (1200)
  + Headphones (75)
  + Keyboard (75)
  + Mouse (25)
  + Book (15)

1. **Aggregate Functions with LINQ (Sum, Min, Max, Average):**
   * **Question:** Calculate the total price, maximum price, minimum price, and average price of all products in the list.
   * **Input (Code Example):** (Same products list as P61)
   * **Output (Example values):**
   * Total Price: 1315
   * Max Price: 1200
   * Min Price: 15
   * Average Price: 328.75
2. **All/Any with LINQ:**
   * **Question:** Check if All products have a price greater than 10. Check if Any product has "Electronics" as its category.
   * **Input (Code Example):** (Same products list as P61)
   * **Output (Conceptual):**
   * All products > 10: False
   * Any product is Electronics: True
3. **Distinct with LINQ:**
   * **Question:** Given a list of integers with duplicates, use LINQ to get a new list containing only the distinct elements.
   * **Input (Code Example):**

C#

List<int> numbers = new List<int> { 1, 2, 2, 3, 4, 3, 5, 1 };

* + **Output (Conceptual - list elements):**
  + [1, 2, 3, 4, 5]

1. **SelectMany (Flattening) with LINQ:**
   * **Question:** Given a list of Department objects, where each department has a List<Employee> (with Name property), use SelectMany to get a flat list of all employee names across all departments.
   * **Input (Code Example):**

C#

public class Employee

{

public string Name { get; set; }

public Employee(string name) { Name = name; }

}

public class Department

{

public string Name { get; set; }

public List<Employee> Employees { get; set; }

public Department(string name, List<Employee> employees) { Name = name; Employees = employees; }

}

List<Department> departments = new List<Department>

{

new Department("HR", new List<Employee> { new Employee("Alice"), new Employee("Bob") }),

new Department("IT", new List<Employee> { new Employee("Charlie"), new Employee("David") })

};

* + **Output (Conceptual - list elements):**
  + ["Alice", "Bob", "Charlie", "David"]

1. **Custom Sorting with OrderBy and Lambda:**
   * **Question:** Given a list of strings, sort them by their length in ascending order, then alphabetically for strings of the same length.
   * **Input (Code Example):**

C#

List<string> words = new List<string> { "apple", "cat", "banana", "dog", "ant" };

* + **Output (Conceptual - list elements in order):**
  + ["cat", "dog", "ant", "apple", "banana"]

*(Assuming standard alphabetical order, "ant" would come before "apple")*

1. **Skip and Take with LINQ (Paging):**
   * **Question:** Simulate paging for a list of products. Given a pageSize and pageNumber, return the products for that specific page.
   * **Input (Code Example):**

C#

List<Product> allProducts = new List<Product>

{

new Product("Product1", 10, "A"), new Product("Product2", 20, "A"),

new Product("Product3", 30, "B"), new Product("Product4", 40, "A"),

new Product("Product5", 50, "C"), new Product("Product6", 60, "B"),

new Product("Product7", 70, "A"), new Product("Product8", 80, "C"),

new Product("Product9", 90, "B"), new Product("Product10", 100, "A")

};

int pageSize = 3;

int pageNumber = 2; // For the second page (1-indexed)

* + **Output (Conceptual - products for the second page):**
  + Product4 (40)
  + Product5 (50)
  + Product6 (60)

1. **Joining Collections with LINQ:**
   * **Question:** Given a list of Order objects (with OrderId, ProductId, and Quantity) and a list of ProductInfo objects (with ProductId and ProductName), join them to display OrderId, ProductName, and Quantity for each order.
   * **Input (Code Example):**

C#

public class Order

{

public int OrderId { get; set; }

public int ProductId { get; set; }

public int Quantity { get; set; }

public Order(int orderId, int productId, int quantity) { OrderId = orderId; ProductId = productId; Quantity = quantity; }

}

public class ProductInfo

{

public int ProductId { get; set; }

public string ProductName { get; set; }

public ProductInfo(int productId, string productName) { ProductId = productId; ProductName = productName; }

}

List<Order> orders = new List<Order>

{

new Order(101, 1, 2),

new Order(102, 3, 1),

new Order(103, 1, 1),

new Order(104, 2, 3)

};

List<ProductInfo> productsInfo = new List<ProductInfo>

{

new ProductInfo(1, "Laptop"),

new ProductInfo(2, "Mouse"),

new ProductInfo(3, "Keyboard")

};

* + **Output (Conceptual):**
  + Order ID: 101, Product: Laptop, Quantity: 2
  + Order ID: 102, Product: Keyboard, Quantity: 1
  + Order ID: 103, Product: Laptop, Quantity: 1
  + Order ID: 104, Product: Mouse, Quantity: 3

**Category 7: Advanced Data Structures & Algorithms (Problems 71-85)**

1. **Count Set Bits (Hamming Weight):** \* **Question:** Write a function that takes an unsigned integer and returns the number of '1' bits it has (also known as the Hamming weight). \* **Input:** uint n = 00000000000000000000000000001011; (binary representation of 11) \* **Output:** 3 \* **Input:** uint n = 00000000000000000000000010000000; (binary representation of 128) \* **Output:** 1
2. **Majority Element:** \* **Question:** Given an array nums of size n, return the majority element. The majority element is the element that appears more than n / 2 times. You may assume that the majority element always exists in the array. \* **Input:** int[] nums = { 3, 2, 3 }; \* **Output:** 3 \* **Input:** int[] nums = { 2, 2, 1, 1, 1, 2, 2 }; \* **Output:** 2
3. **Find Kth Largest Element in an Array:** \* **Question:** Given an integer array nums and an integer k, return the kth largest element in the array. Note that it is the kth largest element in the sorted order, not the kth distinct element. \* **Input:** int[] nums = { 3, 2, 1, 5, 6, 4 }; int k = 2; \* **Output:** 5 (Sorted: [1,2,3,4,5,6], 2nd largest is 5) \* **Input:** int[] nums = { 3, 2, 3, 1, 2, 4, 5, 5, 6 }; int k = 4; \* **Output:** 4 (Sorted: [1,2,2,3,3,4,5,5,6], 4th largest is 4)
4. **Longest Substring Without Repeating Characters:** \* **Question:** Given a string s, find the length of the longest substring without repeating characters. \* **Input:** "abcabcbb" \* **Output:** 3 (The longest substring is "abc") \* **Input:** "bbbbb" \* **Output:** 1 (The longest substring is "b") \* **Input:** "pwwkew" \* **Output:** 3 (The longest substring is "wke". Note that "pwke" is a subsequence, not a substring)
5. **Spiral Matrix Traversal:** \* **Question:** Given an m x n matrix, return all elements of the matrix in spiral order. \* **Input:**
6. int[,] matrix = {
7. { 1, 2, 3 },
8. { 4, 5, 6 },
9. { 7, 8, 9 }
10. };

\* \*\*Output:\*\* `[1, 2, 3, 6, 9, 8, 7, 4, 5]`

1. **Pascal's Triangle:** \* **Question:** Given an integer numRows, return the first numRows of Pascal's triangle. In Pascal's triangle, each number is the sum of the two numbers directly above it. \* **Input:** numRows = 5 \* **Output:**
2. [
3. [1],
4. [1,1],
5. [1,2,1],
6. [1,3,3,1],
7. [1,4,6,4,1]
8. ]
9. **Set Matrix Zeroes:** \* **Question:** Given an m x n integer matrix, if an element is 0, set its entire row and column to 0's. You must do this in-place. \* **Input:**
10. int[,] matrix = {
11. { 1, 1, 1 },
12. { 1, 0, 1 },
13. { 1, 1, 1 }
14. };

\* \*\*Output:\*\*

```

{ { 1, 0, 1 },

{ 0, 0, 0 },

{ 1, 0, 1 } }

```

1. **Happy Number:**
   * **Question:** Write an algorithm to determine if a number n is "happy". A happy number is a number defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits. Repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. Those numbers for which this process ends in 1 are happy.
   * **Input:** 19
   * **Output:** True (19 -> 1^2 + 9^2 = 82 -> 8^2 + 2^2 = 68 -> 6^2 + 8^2 = 100 -> 1^2 + 0^2 + 0^2 = 1)
   * **Input:** 2
   * **Output:** False
2. **Ugly Number:**
   * **Question:** An ugly number is a positive integer whose prime factors are limited to 2, 3, and 5. Given an integer n, return true if n is an ugly number.
   * **Input:** 6
   * **Output:** True (6 = 2 \* 3)
   * **Input:** 14
   * **Output:** False (14 = 2 \* 7)
   * **Input:** 1
   * **Output:** True (By convention, 1 is an ugly number.)
3. **Implement Contains for a Custom Generic List:**
   * **Question:** Given your CustomStack<T> (or similar CustomList<T>), add a Contains(T item) method that checks if an item exists in the list.
   * **Input:** CustomList<string> myList = new CustomList<string>(); myList.Add("Apple"); myList.Add("Banana");
   * **Output:** myList.Contains("Apple") -> True, myList.Contains("Orange") -> False
4. **Smallest Common Multiple (LCM of multiple numbers):**
   * **Question:** Find the smallest positive number that is evenly divisible by all of the numbers from 1 to n.
   * **Input:** n = 10
   * **Output:** 2520 (LCM of 1,2,3,4,5,6,7,8,9,10)
5. **Sieve of Eratosthenes:**
   * **Question:** Implement the Sieve of Eratosthenes algorithm to find all prime numbers up to a given limit n.
   * **Input:** n = 30
   * **Output:** [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
6. **Circular Array Loop:**
   * **Question:** You are playing a game involving a circular array of non-zero integers nums. Each nums[i] denotes the number of steps to advance or go backward. A positive value means a forward move, while a negative value means a backward move. A circular array loop is a sequence of indices k\_0, k\_1, ..., k\_{p-1} such that:
     + Each nums[k\_i] has the same sign.
     + abs(nums[k\_i]) != n (where n is the array length) for any k\_i.
     + The sequence forms a cycle when starting from k\_0, and the cycle length is greater than 1.
   * Return true if there is a circular array loop in nums, or false otherwise.
   * **Input:** int[] nums = { 2, -1, 1, 2, 2 };
   * **Output:** True (Cycle 0 -> 2 -> 3 -> 0)
   * **Input:** int[] nums = { -1, 2 };
   * **Output:** False (Cycle length 1 for -1, no valid cycle for 2)

**Category 8: Practical Applications & Utilities (Problems 86-100)**

1. **Date and Time Manipulation:**
   * **Question:** Get the current date and time. Add 5 days and 3 hours to it. Subtract 2 months from the result. Print all steps in a readable format.
   * **Input:** (Current Date/Time - e.g., 2025-07-18 14:43:09)
   * **Output:**
   * Current: 2025-07-18 14:43:09
   * After +5 days, +3 hours: 2025-07-23 17:43:09
   * After -2 months: 2025-05-23 17:43:09
2. **Stopwatch (Performance Measurement):**
   * **Question:** Use System.Diagnostics.Stopwatch to measure the execution time of a simple loop that counts from 1 to 1,000,000.
   * **Input:** (Implicit)
   * **Output:** Loop execution time: X ms (X will vary)
3. **Parsing CSV Data:**
   * **Question:** Given a simple CSV string (e.g., "Name,Age,City\nAlice,30,New York\nBob,25,London"), parse it into a list of objects or a structured collection (e.g., List<Dictionary<string, string>>).
   * **Input:** string csvData = "Name,Age,City\nAlice,30,New York\nBob,25,London";
   * **Output:** (Conceptual representation)
   * [
   * {"Name": "Alice", "Age": "30", "City": "New York"},
   * {"Name": "Bob", "Age": "25", "City": "London"}
   * ]
4. **Basic Logging Utility:**
   * **Question:** Create a simple static Logger class with methods like LogInfo(string message), LogWarning(string message), LogError(string message). Each method should print the message along with a timestamp and the log level (e.g., "[INFO] [2025-07-18 14:45:00] My info message").
   * **Input:** Logger.LogInfo("Application started.");
   * **Output:** [INFO] [YYYY-MM-DD HH:MM:SS] Application started.
5. **URL Parser (Simple):**
   * **Question:** Given a URL string (e.g., "[https://www.example.com/path/to/page?id=123&name=test#section](https://www.google.com/url?sa=E&source=gmail&q=https://www.example.com/path/to/page?id=123%26name=test#section)"), extract the protocol, domain, path, query parameters (as a dictionary), and fragment.
   * **Input:** "https://www.example.com/path/to/page?id=123&name=test#section"
   * **Output:**
   * Protocol: https
   * Domain: www.example.com
   * Path: /path/to/page
   * Query Params: { "id": "123", "name": "test" }
   * Fragment: section
6. **Data Validation (Email Regex):**
   * **Question:** Write a function IsValidEmail(string email) that uses regular expressions to validate if a given string is a basic valid email address format. (e.g., name@domain.com).
   * **Input:** "test@example.com" -> True, "invalid-email" -> False, "user@domain" -> True (simple regex)
   * **Output:** True or False
7. **Serialization to JSON (Basic):**
   * **Question:** Using System.Text.Json (or Newtonsoft.Json), serialize a simple C# object (e.g., an instance of your Person class) into a JSON string.
   * **Input:** new Person { Name = "Alice", Age = 30 };
   * **Output:** {"Name":"Alice","Age":30}
8. **Deserialization from JSON (Basic):**
   * **Question:** Deserialize the JSON string created in the previous problem back into a Person object.
   * **Input:** string jsonString = "{\"Name\":\"Alice\",\"Age\":30}";
   * **Output:** Person object with Name = "Alice", Age = 30
9. **Asynchronous Programming (Basic async/await):**
   * **Question:** Write a simple async method DownloadDataAsync() that simulates a long-running operation (e.g., Task.Delay(3000)). Demonstrate calling it with await and show that the UI/console doesn't freeze during the delay.
   * **Input:** (Implicit)
   * **Output:**
   * Starting download...
   * (3 second pause, but console responsive if running in a loop)
   * Download complete!
10. **Generic Repository Pattern (Conceptual):**
    * **Question:** Design a basic generic interface IRepository<T> with methods like GetById(int id), GetAll(), Add(T entity), Update(T entity), Delete(int id). Briefly show how you'd implement a concrete UserRepository or ProductRepository based on it.
    * **Input:** (Conceptual design, no direct I/O for this one)
    * **Output:** (Code structure demonstrating interface and class implementation)
11. **Unit Conversion Utility:**
    * **Question:** Create a simple utility class UnitConverter with static methods for common conversions, e.g., ConvertCelsiusToFahrenheit(double celsius) and ConvertKilometersToMiles(double kilometers).
    * **Input:** UnitConverter.ConvertCelsiusToFahrenheit(25);, UnitConverter.ConvertKilometersToMiles(100);
    * **Output:** 77 (Fahrenheit), 62.1371 (Miles)
12. **Calculate Age from Birthdate:**
    * **Question:** Write a method that takes a DateTime representing a birthdate and calculates the current age of the person.
    * **Input:** DateTime birthDate = new DateTime(1990, 7, 18); (assuming current date is 2025-07-18)
    * **Output:** 35
13. **Simple Console Menu Application:**
    * **Question:** Create a basic console application that presents a menu to the user (e.g., 1. Add Numbers, 2. Check Even/Odd, 3. Exit). Use a while loop and switch statement to handle user choices until they choose to exit.
    * **Input:** 1, then 5, 3, then 3
    * **Output:** (Interactive console output based on menu options)
14. **Read Environment Variable:**
    * **Question:** Read and print the value of a common environment variable (e.g., PATH or USERNAME) using C#.
    * **Input:** (Implicitly reads system environment variable)
    * **Output:** PATH: C:\...\; USERNAME: [YourUsername]
15. **Generic Method for Array Reversal:** \* **Question:** Create a generic method ReverseArray<T>(T[] array) that reverses the elements of any type of array (e.g., int[], string[]). \* **Input:** int[] intArray = { 1, 2, 3, 4 }; string[] stringArray = { "A", "B", "C" }; \* **Output:** intArray becomes [4, 3, 2, 1], stringArray becomes ["C", "B", "A"]