SET-2

21. Find All Permutations of a String

Problem: Return all permutations of a string

```
using System;
class Program
{
   static void Main()
   {
      string s = Console.ReadLine();
      Permute(s.ToCharArray(), 0, s.Length - 1);
   }
   static void Permute(char[] arr, int l, int r)
   {
      if (l == r)
      {
         Console.WriteLine(new string(arr));
         return;
      }
      for (int i = l; i <= r; i++)
      {
         Swap(arr, l, i);
         Permute(arr, l + 1, r);
         Swap(arr, l, i); // backtrack
      }
   }
   static void Swap(char[] arr, int i, int j)
   {
      char temp = arr[i];
      arr[i] = arr[j];
      arr[j] = temp;
   }
}
```

22. Find Longest Increasing Subsequence

Problem: Return length of longest increasing subsequence

```
using System;
using System.Linq;
class Program
{
  static void Main()
  {
    Console.Write("Enter numbers: ");
    int[] nums = Console.ReadLine().Split().Select(int.Parse).ToArray();

    int n = nums.Length;
    int[] dp = new int[n];
    Array.Fill(dp, 1);

    for (int i = 0; i < n; i++)
      for (int j = 0; j < i; j++)
        if (nums[i] > nums[j])
          dp[i] = Math.Max(dp[i], dp[j] + 1);

    Console.WriteLine("Length of LIS: " + dp.Max());
}
}
```

23. Find Next Greater Element

Problem: For each element, find the next greater element to its right

```
using System;
class Program
{
  static void Main()
  {
    Console.Write("Enter numbers separated by spaces: ");
    string[] parts = Console.ReadLine().Split(' ');
    int n = parts.Length;
    int[] arr = new int[n];
    int[] result = new int[n];
    for (int i = 0; i < n; i++)
      arr[i] = int.Parse(parts[i]);
    for (int i = 0; i < n; i++)
    {
      result[i] = -1;
      for (int j = i + 1; j < n; j++)
      {
        if (arr[j] > arr[i])
        {
          result[i] = arr[j];
          break;
        }
      }
    }
    Console.Write("Next Greater Elements: ");
    for (int i = 0; i < n; i++)
      Console.Write(result[i] + " ");
  }
}
```

24. Find All Elements Appearing More Than n/3 Times

Problem: Return elements appearing more than n/3 times.

```
using System;
class Program
{
  static void Main()
  {
    Console.Write("Enter numbers: ");
    string[] parts = Console.ReadLine().Split(' ');
    int n = parts.Length, limit = n / 3;
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) arr[i] = int.Parse(parts[i]);

    Console.Write("Elements appearing more than n/3 times: ");
    for (int i = 0; i < n; i++)
    {
      int count = 0;
      for (int j = 0; j < n; j++) if (arr[j] == arr[i]) count++;
      bool dup = false;
      for (int k = 0; k < i; k++) if (arr[k] == arr[i]) dup = true;
      if (count > limit && !dup) Console.Write(arr[i] + " ");
    }
  }
}
```

25. Find All Unique Combinations That Sum to Target

Problem: Return all combinations of numbers that sum to a target.

```
using System;
using System.Collections.Generic;

class Program
{
    static void Main()
    {
        Console.Write("Enter numbers: ");
        int[] nums = Array.ConvertAll(Console.ReadLine().Split(), int.Parse);

        Console.Write("Enter target: ");
        int target = int.Parse(Console.ReadLine());

        List<List<int>> result = new List<List<int>>();
        FindCombinations(nums, target, new List<int>(), 0, result);

        Console.WriteLine("Combinations:");
        foreach (var combo in result)
            Console.WriteLine("[" + string.Join(", ", combo) + "]");
    }

    static void FindCombinations(int[] nums, int target, List<int> current, int index, List<List<int>> result)
    {
        if (target == 0) // Found a valid combination
        {
            result.Add(new List<int>(current));
            return;
        }

        for (int i = index; i < nums.Length; i++)
        {
            if (nums[i] <= target) // Only pick numbers that don't exceed remaining target
            {
                current.Add(nums[i]); // Choose the number
                FindCombinations(nums, target - nums[i], current, i, result); // Recurse with reduced target
                current.RemoveAt(current.Count - 1); // Backtrack
            }
        }
    }
}
```