**Computational assignment copilot prompts**:

1. Write a C# method to find missing numbers from 1 to n in an unsorted array.

Output:

public static IList<int> FindMissingNumbers(int[] nums)

{

try

{

HashSet<int> set = new HashSet<int>(nums);

List<int> result = new List<int>();

for (int i = 1; i <= nums.Length; i++)

{

if (!set.Contains(i))

result.Add(i);

}

return result;

}

catch (Exception)

{

throw;

}

}

1. Write a C# method that sorts an integer array by parity (even numbers first, odd numbers second).

Output:

public static int[] SortArrayByParity(int[] nums)

{

try

{

int[] result = new int[nums.Length];

int start = 0, end = nums.Length - 1;

foreach (int num in nums)

{

if (num % 2 == 0)

result[start++] = num;

else

result[end--] = num;

}

return result;

}

catch (Exception)

{

throw;

}

}

1. Write a C# function that returns indices of two numbers in an array that add up to a target.

Output:

public static int[] TwoSum(int[] nums, int target)

{

try

{

Dictionary<int, int> map = new Dictionary<int, int>();

for (int i = 0; i < nums.Length; i++)

{

int complement = target - nums[i];

if (map.ContainsKey(complement))

return new int[] { map[complement], i };

if (!map.ContainsKey(nums[i]))

map[nums[i]] = i;

}

return new int[0];

}

catch (Exception)

{

throw;

}

}

1. Write a C# function to return the maximum product of any three numbers in an integer array.

Output:

public static int MaximumProduct(int[] nums)

{

try

{

Array.Sort(nums);

int n = nums.Length;

return Math.Max(nums[0] \* nums[1] \* nums[n - 1],

nums[n - 1] \* nums[n - 2] \* nums[n - 3]);

}

catch (Exception)

{

throw;

}

}

1. Write a C# method to convert a decimal number to its binary representation.

Output:

public static string DecimalToBinary(int decimalNumber)

{

try

{

if (decimalNumber == 0) return "0";

string binary = "";

while (decimalNumber > 0)

{

binary = (decimalNumber % 2) + binary;

decimalNumber /= 2;

}

return binary;

}

catch (Exception)

{

throw;

}

}

1. Write a C# function to find the minimum element in a rotated sorted array.

Output:

public static int FindMin(int[] nums)

{

try

{

int left = 0, right = nums.Length - 1;

while (left < right)

{

int mid = left + (right - left) / 2;

if (nums[mid] > nums[right])

left = mid + 1;

else

right = mid;

}

return nums[left];

}

catch (Exception)

{

throw;

}

}

1. Write a C# function to check whether a number is a palindrome.

Output:

public static bool IsPalindrome(int x)

{

try

{

if (x < 0) return false;

string s = x.ToString();

int left = 0, right = s.Length - 1;

while (left < right)

{

if (s[left++] != s[right--])

return false;

}

return true;

}

catch (Exception)

{

throw;

}

}

1. Write a C# method to return the nth Fibonacci number using iteration.

Output:

public static int Fibonacci(int n)

{

try

{

if (n <= 1) return n;

int a = 0, b = 1;

for (int i = 2; i <= n; i++)

{

int temp = a + b;

a = b;

b = temp;

}

return b;

}

catch (Exception)

{

throw;

}

}