# Prompts Documentation - Assignment 2 (Computational Problem Solving)

## Question 1: Find Missing Numbers in Array

**Prompt Used:**  
How do I find missing numbers in a range from 1 to n given an integer array in C#?

**Response Received:**  
Copilot suggested using a HashSet of input values and looping from 1 to n, checking which numbers are missing.

**Implementation Details:**  
Followed Copilot’s logic. Used HashSet<int> to store array values and checked each value in range [1, n].

**Adjustments:**  
None needed. The logic was clean and optimal.

## Question 2: Sort Array by Parity

**Prompt Used:**  
Sort an array by parity in C# using two-pointer approach

**Response Received:**  
Copilot showed how to use two indices (start, end) and swap values to group evens before odds.

**Implementation Details:**  
Implemented exactly as suggested with slight renaming of variables.

**Adjustments:**  
Added comments for better readability and renamed variables to start and end. Also verified behavior on edge cases like all-even and all-odd arrays.

## Question 3: Two Sum

**Prompt Used:**  
C# dictionary-based solution for Two Sum problem

**Response Received:**  
Suggested using a dictionary to store previously seen numbers and check for complement at each step.

**Implementation Details:**  
Implemented Dictionary<int, int> and used the key check logic.

**Adjustments:**  
Renamed complement to required for clarity. Also confirmed safe return for no-match case using empty array fallback.

## Question 4: Maximum Product of Three Numbers

**Prompt Used:**  
Find maximum product of three integers in a C# array

**Response Received:**  
Copilot showed that we need to sort the array and consider two cases: product of 3 largest vs. 2 smallest and 1 largest.

**Implementation Details:**  
Used Array.Sort() and Math.Max() on the two potential products.

**Adjustments:**  
Added inline comments to explain logic. Validated on mix of positive and negative numbers to ensure both max and min products are considered.

## Question 5: Decimal to Binary Conversion

**Prompt Used:**  
Convert decimal number to binary in C# without using Convert.ToString

**Response Received:**  
Used while loop and Stack<int> to simulate division-by-2 and collect binary digits.

**Implementation Details:**  
Used stack and returned string.Join("", stack).

**Adjustments:**  
None required.

## Question 6: Find Minimum in Rotated Sorted Array

**Prompt Used:**  
Binary search for minimum in rotated sorted array C#

**Response Received:**  
Copilot suggested comparing mid with right and narrowing down the search range.

**Implementation Details:**  
Implemented binary search logic using mid, left, and right variables.

**Adjustments:**  
Added comments explaining the condition when mid > right. Edge cases (fully sorted, single-element) were handled successfully.

## Question 7: Palindrome Number

**Prompt Used:**  
How to check if a number is palindrome in C# without converting to string

**Response Received:**  
Reconstruct the reversed number and compare it with the original.

**Implementation Details:**  
Used loop to reverse the integer and then compare.

**Adjustments:**  
No changes. Logic was straightforward. Verified that logic handles cases like 0 and 10 correctly.

## Question 8: Fibonacci Number

**Prompt Used:**  
C# iterative method to find nth Fibonacci number

**Response Received:**  
Copilot provided loop-based solution using two variables and a temp sum.

**Implementation Details:**  
Used variables first and second to track Fibonacci sequence.

**Adjustments:**  
Minor variable renaming and added base case checks.