**Prompt and Answers**

**Question 1:**

I didn’t use any prompts to solve the first question. Because I know how to deal with lists.

**Question 2:**

*Prompt 1:* what is the meaning of this question "Question 2: Sort Array by Parity Console.WriteLine("Question 2:"); int[] nums2 = { 3, 1, 2, 4 }; int[] sortedArray = SortArrayByParity(nums2); Console.WriteLine(string.Join(",", sortedArray));"

*Answer to prompt:*

**What does "Sort Array by Parity" mean?**

**Parity** refers to whether a number is **even** or **odd**:

* Even number: divisible by 2 → 0, 2, 4, 6, ...
* Odd number: not divisible by 2 → 1, 3, 5, 7, ...

So, **"Sort Array by Parity"** means: 👉 **Rearrange the array so that all even numbers come before all the odd numbers**.  
The relative order of even/odd numbers doesn’t matter (unless specified).

*Prompt 2*: How to add 2 lists

*Answer to prompt:*

To **add (combine) two lists** in C#, you can use the AddRange() method from the List<T> class.

**Question 3:**

I didn’t use any prompts to solve the third question.

**Question 4**:

I didn’t use any prompts to solve the fourth question.

**Question 5:**

*Prompt:* How to convert a number to its binary equivalent mathematically

*Answer to prompt:*

To convert a decimal number to its binary equivalent mathematically, you can use the **repeated division by 2 method**. Here's the step-by-step process:

**Steps for Decimal to Binary Conversion:**

1. **Divide the decimal number by 2.**
2. **Record the remainder** (this will be either 0 or 1) as the least significant bit (LSB).
3. **Divide the quotient from the previous step by 2**, and again record the remainder.
4. **Repeat this process** until the quotient becomes 0.
5. **The binary number is the remainders read from bottom to top.**

**Example: Convert 13 to binary**

1. **Divide 13 by 2:**
   * Quotient = 6, Remainder = 1 (this is the LSB)
2. **Divide 6 by 2:**
   * Quotient = 3, Remainder = 0
3. **Divide 3 by 2:**
   * Quotient = 1, Remainder = 1
4. **Divide 1 by 2:**
   * Quotient = 0, Remainder = 1 (this is the MSB)

Now, read the remainders from bottom to top: **1 1 0 1**, which is **1101** in binary.

**General Algorithm:**

Given a decimal number N:

1. **While N > 0:**
   * **Remainder = N % 2** (this gives either 0 or 1).
   * **N = N / 2** (divide the number by 2).
2. Collect the remainders and read them in reverse order to get the binary equivalent.

**Question 6:**

I didn’t use any prompt for this question.

**Question 7:**

*Prompt:* What is palindrome number?

*Answer to the prompt:*

A **palindrome number** is a number that reads the same backward as forward. In other words, if you reverse the digits of the number, it remains unchanged.

**Example:**

* **121** is a palindrome because reversing it gives **121**.
* **12321** is a palindrome because reversing it gives **12321**.
* **123** is **not** a palindrome because reversing it gives **321**, which is not the same as **123**.

**Characteristics of Palindrome Numbers:**

* The number must remain the same when its digits are reversed.
* Palindrome numbers can be of any length (e.g., one digit, two digits, etc.).

**Palindrome Number Examples:**

* **5**: Single-digit numbers are always palindromes.
* **121**: Reversed is also 121.
* **12321**: Reversed is also 12321.

**Non-Palindrome Example:**

* **123**: Reversed is **321**, which is not the same as 123.

In programming, you can check whether a number is a palindrome by reversing its digits and comparing the reversed number with the original number. If they are equal, it is a palindrome number.

**Question 8:**

I didn’t use any prompt for this question.