Part 1: The params Keyword

The params keyword allows a method to accept a variable number of arguments of a specific type.

1. **The Summer:** Create a method SumNumbers(params int[] numbers) that returns the sum of all integers passed to it. Ensure it returns 0 if no arguments are provided.
2. **The Sentence Builder:** Write a method CombineWords(params string[] words) that joins an array of strings into a single sentence separated by spaces.
3. **The Multi-Type Challenge:** Create a method that takes a double multiplier as the first parameter and a params double[] values as the second. Multiply every value in the array by the multiplier and return the new list.
4. **The Logger:** Design a LogMessages(string level, params string[] messages) method. It should print the "level" (e.g., "INFO") followed by each message on a new line.
5. **Validation Check:** Why does the following code fail to compile? public void InvalidMethod(params int[] nums, string name) { }

Part 2: Optional Parameters

Optional parameters allow you to provide default values so callers don't have to pass every argument.

6. **The Greeting:** Create a method Greet(string name, string title = "Guest"). If only a name is provided, it should print "Hello, [Title] [Name]".
7. **Price Calculator:** Write a method CalculateTotal(double price, double taxRate = 0.05, double discount = 0). Calculate the final price after adding tax and subtracting the discount.
8. **Named Arguments:** Using the method from Question 7, show how you would call CalculateTotal providing the price and the discount, but skipping the taxRate.
9. **The Shape Maker:** Create a method CreateRectangle(double width, double height = 10). If only one value is passed, the rectangle should be 10 units high by default.
10. **Order of Operations:** Explain why optional parameters must always be defined *after* required parameters in a method signature.