Up solve questions

1. What is the shortcut to comment/uncomment a selected block of code in Visual Studio?

- **Shortcut**:
  - **Comment**: Ctrl + K followed by Ctrl + C.
  - **Uncomment**: Ctrl + K followed by Ctrl + U.
- **Note**: Works for most languages in Visual Studio, including C#.

2. Difference Between Runtime Error and Logical Error

Runtime Error	**Logical Error**
Occurs during program execution.	Program runs but produces incorrect results.
Example: Division by zero (DivideByZeroException).	**Example**: Using + instead of * in a calculation.
Causes: Invalid operations, missing files, etc.	**Causes**: Flawed algorithm or incorrect logic.

3. Importance of PascalCase in C#

- **Readability**: Clearly distinguishes class names (MyClass), methods (CalculateTotal), and properties.
- **Convention**: Adheres to C# coding standards, improving code maintainability.
- **Avoids Conflicts**: Differentiates types from variables (e.g., Person vs. person).

4. Value Types vs. Reference Types (Memory Allocation)

Value Types	Reference Types
Stored on the **stack** (fast access).	Stored on the **heap** (reference on the stack).
Directly hold their data.	Hold a reference/memory address to data.
Examples: int, char, struct.	**Examples**: string, class, array.
Assignment: Creates a copy.	**Assignment**: Copies the reference (shared data).

5. Output of a % b When a = 2, b = 7

- **Output**: 2
- **Explanation**: The modulus operator (%) returns the remainder after division.
  - 2 ÷ 7 = 0 with a remainder of 2.

6. && (Logical AND) vs. & (Bitwise AND)

&&	&
Evaluates boolean expressions.	Performs bitwise operations on integers.
Short-circuits: Skips the right operand if the left is false.	Always evaluates both operands.
Example: if (isValid && isReady)	**Example**: result = 5 & 3 (binary 0101 & 0011 = 0001).

7. Why Explicit Casting is Required for double to int

- **Reason**: double can store decimals, while int cannot. Explicit casting ((int)myDouble) ensures the developer acknowledges potential data loss (e.g., truncating 3.14 to 3).

8. Handling Invalid Input Exceptions

- **Exception**: FormatException (e.g., parsing non-numeric input with int.Parse()).
- **Handling**: Use try-catch blocks:

```
try {

    int = int.Parse(Console.ReadLine());

} catch (FormatException  ex) {

Console .WriteLine("Invalid input! Enter a number.");

}
```

9. Value of x After int x = 5; int y = ++x + x++;

- **Final Value of x**: 7
- **Explanation**:
  1. ++x (prefix): Increments x to 6 before evaluation.
  2. x++ (postfix): Uses x = 6 in the expression, then increments to 7.
     - **Calculation**: y = 6 + 6 = 12, x becomes 7.