1. What is the shortcut to comment and uncomment a selected block of code in Visual Studio?
   - To Comment: ctrl+K+C
   - To Uncomment: ctrl+K+U

2. Explain the difference between a runtime error and a logical error
   - Runtime error is an error that only being detected during runtime like trying to add int and char Console.WriteLine("A" + 10);
   - Logical error is unexpected BUG in the code like:
   - ```
     int x = 15;
     int y = 10;
     if(x < y)
         Console.WriteLine("X is greater");
     else
         Console.WriteLine("Y is greater");
     ```
   - The error here in the condition I typed smaller than instead of greater than so the first condition that is the true one will not be succeed

3. Why is it important to follow naming conventions such as PascalCase in C#?
   To make it easy to understand for other developers who will work on the project.

4. Explain the difference between value types and reference types in terms of memory allocation.
   The value type stored in the stack but the reference is stored in the heap

5. What will be the output of the following code? Explain why:
   ```
   int a = 2, b = 7;
   Console.WriteLine(a % b);
   ```
   The output is 2 because 7 is greater than 2 so the result of the division is 0 and remind the 2.

6. How does the && (logical AND) operator differ from the & (bitwise AND) operator?

   Logical AND: is the one we are using in logical operations like in if statement or loop condition and it need the both conditions to be true.

   Bitwise AND: is working more with integers and it add them according to their bit value like 2(0010) & 6(0110) equals to 2(0010).

7. Why is explicit casting required when converting a double to an int?
   Double is 8 bytes but int is 4 bytes converting int to double is easy so double can contain int but on the other hand int cannot contain double so we use explicit casting.

8. What exception might occur if the input is invalid and how can you handle it
   1. FormatException if the input is not number.
   2. ArgumentNullException if there is no input found.
   3. OverFlowException if the input is too small or too big.
   - We can handle it with try catch method.

9. Given the code below, what is the value of x after execution? Explain why
   - int x = 5; int y = ++x + x++;

   It will print 12 cause the 5 in ++x will increase to 6 then be added to the other x who is 6 now then it increases again but after the operation is done.
   ```

Part02

1. what's the difference between compiled and interpreted languages and in this way what about C#?
    - Compiled Language: Code is translated before execution (at compile time), Faster, because the code is precompiled to machine code, generates an executable (.exe), Most errors caught at compile time, like C, C++, Rust, Go

    - Interpreted Language: Code is translated during execution (at runtime), Slower, because it's interpreted line by line, no separate binary; runs through an interpreter, Errors may appear at runtime, like Python, JavaScript, Ruby.

    - C# is neither fully compiled nor purely interpreted — it's a hybrid.
      C# is compiled to Intermediate Language (IL), then interpreted (JIT compiled) at runtime.
      Source code (.cs) compiled by C# compiler, just-In-Time (JIT) Compilation
      CLR (Common Language Runtime) compiles IL to native machine code at runtime, execution on your machine.

2. Compare between implicit, explicit, Convert and parse casting
    - Implicit Casting: Automatically done by the compiler, no data loss expected, from smaller to larger types (int to long).

    - Explicit Casting: Requires cast syntax (type), may lose data or cause runtime errors, used when converting from larger to smaller types (double to int).

    - Convert Class: Uses .NET Convert methods (Convert.ToInt32()), handles nulls, Booleans, and formats, can throw exceptions if conversion fails.

    - Parse Method: Available in types like int.Parse, double.Parse, etc.
      Only works on strings
      Throws exception if input is invalid or null