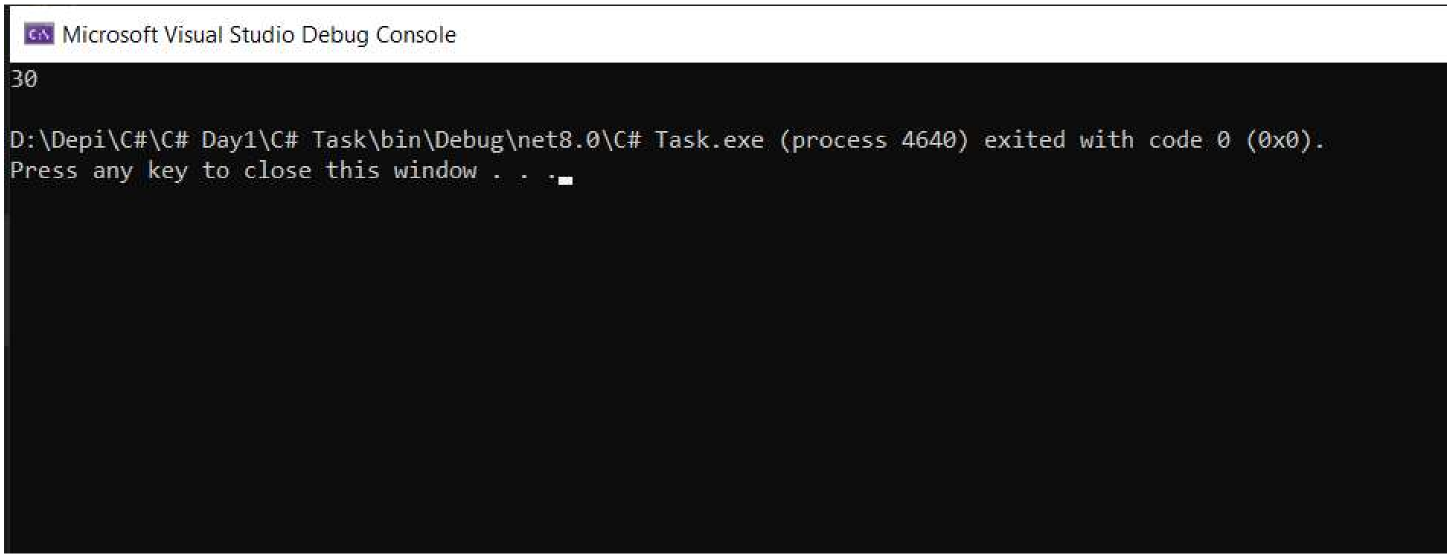
C# Day02 Task

Problem 1



Microsoft Visual Studio Debug Console

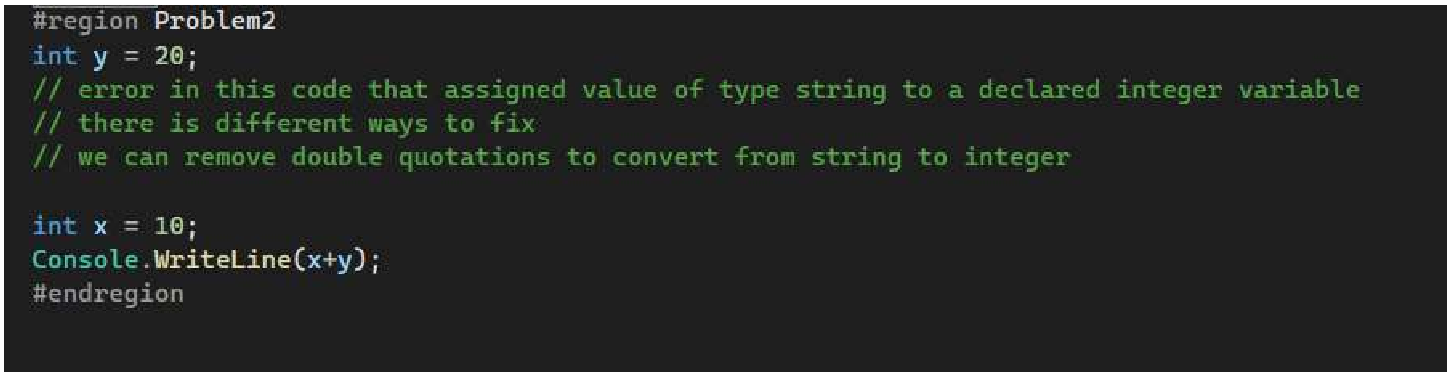
```
30

D:\Depi\C#\C# Day1\C# Task\bin\Debug\net8.0\C# Task.exe (process 4640) exited with code 0 (0x0).
Press any key to close window . . .
```

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

Ctrl + k + c   for comment

Ctrl + k + u for uncomment

Problem2



```
#region Problem2
int y = 20;
// error in this code that assigned value of type string to a declared integer variable
// there is different ways to fix
// we can remove double quotations to convert from string to integer

int x = 10;
Console.WriteLine(x+y);
#endregion
```

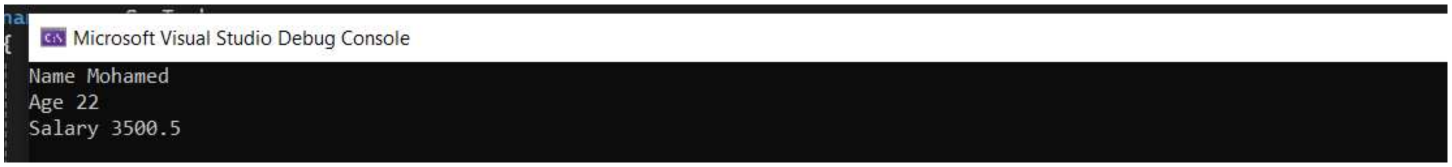
Runtime error : error of syntax of the Language and causes the

Program not to be execute

Logical Error : The program is executed well but the Output

Isn't is the desired causes no fault in compilation process

Problem 3



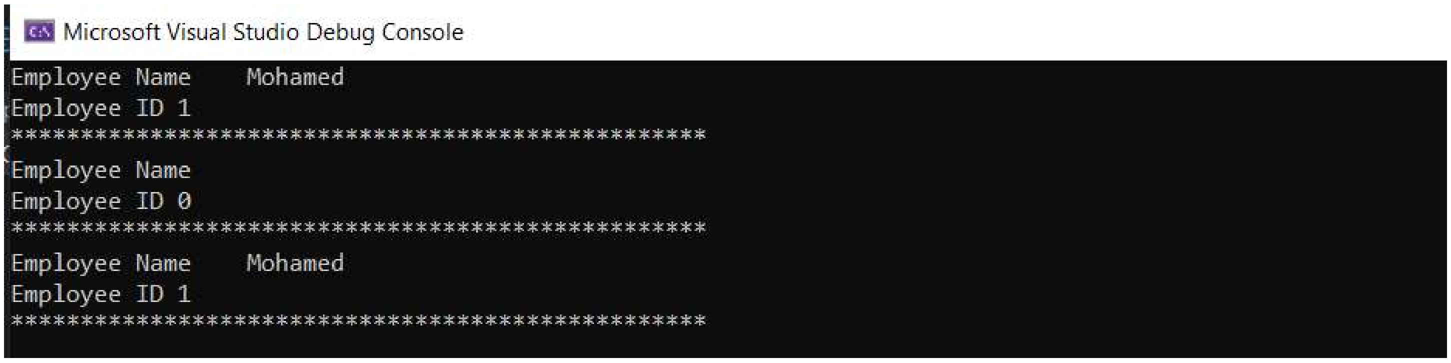
```
Name Mohamed
Age 22
Salary 3500.5
```

Question: Why is it important to follow naming conventions such as Pascal Case in C#?

To ensure that our code is clean, readable, and easy to maintain. Whether it's using Pascal Case for class names, camel case for local variables, or using an underscore prefix for private fields, these conventions help establish a consistent codebase

Problem 4

Write a program to demonstrate that changing the value of a reference type affects all references pointing to that object ?



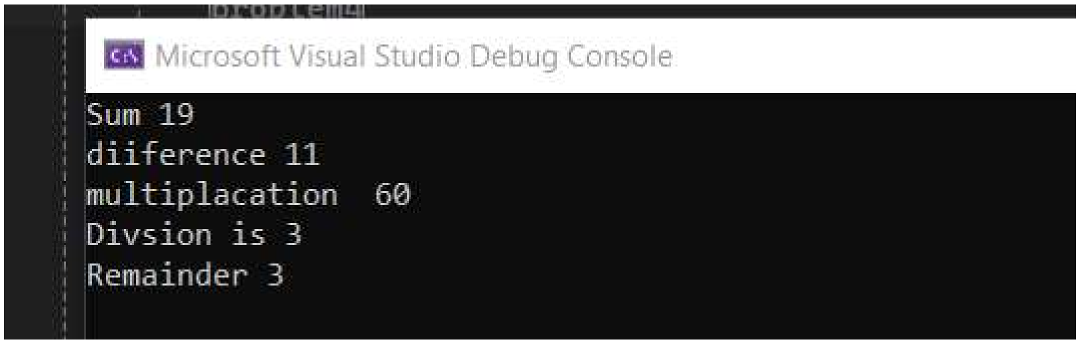
```
Employee Name    Mohamed
Employee ID 1
*****************************************************
Employee Name
Employee ID 0
*****************************************************
Employee Name    Mohamed
Employee ID 1
*****************************************************
```

Explain the difference between value types and reference types in terms of memory allocation ?

Aspect	Value Types	Reference Types
Memory Allocation	Stored directly on the stack, containing the actual data.	Stored on the heap, with a reference (pointer) to the data stored on the stack.
Data Storage	Each variable has its own copy of the data.	Variables hold a memory address pointing to the same data on the heap.
Assignment Behavior	Copying a variable creates a new, independent copy of the data.	Copying a variable copies the reference, pointing to the same data.
Modification Impact	Changes to one variable do not affect others.	Changes to the object via one reference affect all references to it.
Examples	`int`, `float`, `struct` (e.g., `int x = 5; int y = x; y = 10;` `x` stays 5).	Objects, arrays, strings (e.g., `List<int> a = new List<int>{1,2}; List<int> b = a; b.Add(3);` `a` changes).
Performance	Faster access (stack-based), suited for small, fixed-size data.	Slower access (heap-based), suited for complex, dynamic data; may involve garbage collection.
Size	Fixed size, typically small.	Variable size, can be large and dynamic.

Problem 5

Problem: Create a program that calculates the following using variables x = 15 and y = 4


Microsoft Visual Studio Debug Console

```
Sum 19
diiference 11
multiplacation  60
Divsion is 3
Remainder 3
```

Problem 6

Question: What will be the output of the following code? Explain why: int a = 2, b = 7; Console.WriteLine(a % b);

The Result will be 2

Question: How does the && (logical AND) operator differ from the & (bitwise AND) operator?

purpose and Functionality:

- **Logical AND (&&):** Evaluates two Boolean expressions and returns true only if both operands are true. It operates at the logical level, typically used in conditional statements (e.g., if statements). It short-circuits, meaning if the first operand is false, the second operand is not evaluated.
  - Example: if (a > 0 && b > 0) checks if both a and b are positive.
  - Result: true or false.
- **Bitwise AND (&):** Performs a binary operation on the individual bits of two integer operands, comparing each corresponding bit. If both bits are 1, the result is 1; otherwise, it's 0. It does not short-circuit and evaluates both operands.
  - Example: 5 & 3 (binary: 101 & 011) results in 1 (binary: 001).
  - Result: An integer.

Problem7

Question: Why is explicit casting required when converting a double to an int?

Because converting Bigsize datatype to lowersize DataType Will Make Losses in the data

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

Solution

X is incremented (++x) = > 6 and (x++) 6

The summation = 1 2

And the final value of x is 7

Part 02

2- what's the difference between compiled and interpreted languages and in this way what about Csharp?

Aspect	Compiled Languages	Interpreted Languages	C#
Definition	Translated into machine code before execution.	Executed line-by-line by an interpreter at runtime.	Primarily compiled, but uses an intermediate step (IL code) executed by a runtime (CLR).
Execution Speed	Generally faster, as code is pre-translated to machine code.	Slower, due to real-time interpretation.	Fast, as IL is compiled to machine code at runtime (JIT compilation).
Portability	Less portable; requires recompilation for different platforms.	Highly portable; runs on any platform with the appropriate interpreter.	Portable across platforms with .NET runtime (e.g., Windows, macOS, Linux).
Error Detection	Errors caught during compilation, before execution.	Errors often detected during runtime, which can interrupt execution.	Errors caught at compile-time (C# to IL) and runtime (IL to machine code or execution).
Development Speed	Slower development due to compilation step.	Faster development, as code can be run immediately without compilation.	Balanced; compilation is fast, and tools like Visual Studio enhance development speed.
Examples	C, C++, Rust, Go.	Python, JavaScript, Ruby, PHP.	C# (compiled to IL, then JIT-compiled to machine code).
Code Distribution	Distributed as executable binaries.	Distributed as source code or bytecode, requiring an interpreter.	Distributed as IL (assemblies), executed by .NET CLR.
Memory Usage	Generally lower, as no interpreter is needed during execution.	Higher, as interpreter runs alongside the program.	Moderate; CLR manages memory, with JIT compilation optimizing performance.
Use Case	System programming,	Scripting, web	Enterprise apps, web

	performance-critical applications (e.g., OS, games).	development, rapid prototyping.	(ASP.NET), games (Unity), cross-platform apps (.NET MAUI).

3- Compare between implicit, explicit, Convert and parse casting

Aspect	Implicit Casting	Explicit Casting	Convert	Parse
Definition	Automatic conversion, no data loss.	Manual conversion, possible data loss.	Uses `Convert` class for broad type conversion.	Converts strings to specific types.