Part01

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

   Comment: ctrl + k + c (or ctrl + /)

   Uncomment: ctrl + k + u (or ctrl + /)

2. Question: Explain the difference between a runtime error and a logical error with examples.

   runtime error: unexpected processes in the code like division over 0

   logical error: a code is compliable but with wrong output (like get an output = 10 and the expected is 9)

3. Question: Why is it important to follow naming conventions such as PascalCase in C#?

   C# is designed as pascal case in general. In addition to make the code readable and making easier collaboration and reducing ambiguity.

4. Question: Explain the difference between value types and reference types in terms of memory allocation.

   Value type: allocated in the stack in memory stack

   Reference type: the object is been allocated in the stack but its address allocated in the heap

5. Question: What will be the output of the following code? Explain why:

```
int a = 2, b = 7;
Console.WriteLine(a % b);
```

   Output = 2
   The remainder of the division 2/7 is 2

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

	&&	&
type	logical	bitwise
treat with	high level (Booleans)	low level (bits)
can treat with non-Boolean	Yeap	Nop
circuit type	short circuit	long circuit
pass through conditions	if some cond is false: also check for the remainder conds	if some cond is false: ignore the remainder

7. Question: Why is explicit casting required when converting a double to an int?
   Sizeof(int) = 4 bytes while sizeof(double) = 8 bytes
   So, this will cause an overflow

8. Question: What exception might occur if the input is invalid and how can you handle it
   `System.FormatException: Input string was not in a correct format.`
   How can handle it: using checked block/statement

9. Question: Given the code below, what is the value of x after execution? Explain why

```
int x = 5;

int y = ++x + x++;
```

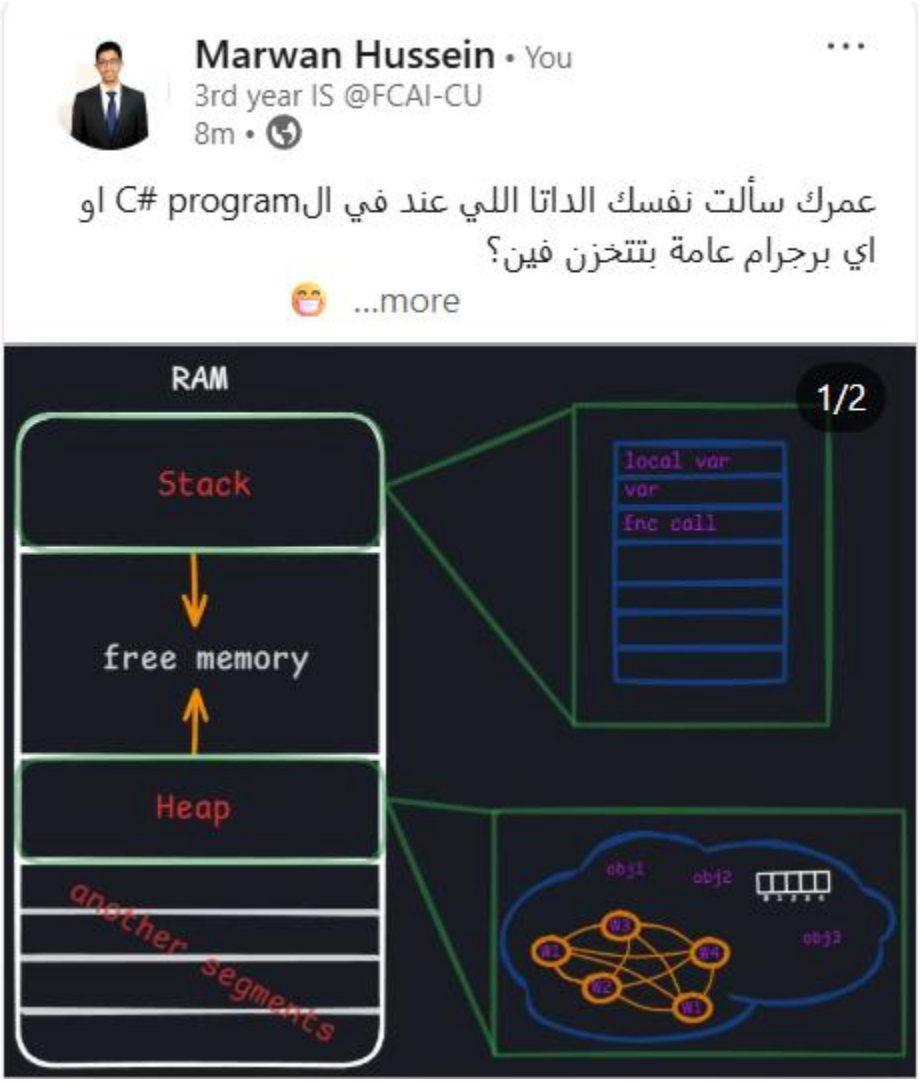
   y = 6 + 6 = 12

   x = 7

   the process was to firstly postfix x (to be 6) then postfix it (to be 6 through the process then become 7)

Part02

1- LinkedIn article about variables allocation in stack and heap for both value and ref types



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

Compiled language: convert the entire code to a machine language producing an .exe file (e.g. C, C++)

Interpreted language: source code is read and executed statement by statement by an interpreter, no separate executable file is produced (e.g. python, js)

C# is hybrid: it’s compiled firstly into an intermediate lang (CIL) then at runtime the CLR interprets (JIT-compiles) that intermediate lang to a machine code

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

Implicit: convert from a smaller dataType to a larger one

Explicit: convert from a larger dataType to a smaller one

Parse: convert a string to any dataType

Convert: converts any dataType to another datatype

Part03

4- Self-study report:

4.1. Creating objects (Reference Types)

4.2. Memory leak

4.3. Garbage Collector

4.4. Customizing Garbage Collector

4.5. Unmanaged resources (database connections, reading files)

4.6. Scientific notation / FPU -> float & double

$x*10^n = xen$

So $1e6 = 1*10^6$ (million)

We use double and float because we may have 1500 that represented as 1.5e3

4.7. When and why to use decimal -> precise

In addition to its large size (16 bytes), other data types make a tiny error (approximation) ➞ may leads to huge errors in some financial or high-precision calculations.

4.8. Difference between float, double, and decimal

Choosing between them is based on speed, size, accuracy.

feature	float	double	decimal
size	4 bytes	8 bytes	16 bytes
precision	6-9 bits	15-17 bit	28-29 bits
type	binary (base-2)	binary (base-2)	decimal (base-10)
performance	fastest	fast	slow
representation	1.2f	1.2d (optional)	1.2m

4.9. Checked block & Unchecked block

checked block: done if no overflow otherwise through an exception

unchecked block: don't check if there is an overflow

```
int x = int.MaxValue;
int y = unchecked(x+1);
// y = int.MinValue (consider it as a circle [mod])
```

But realy it's useless if out the checked block

4.10. Parsing null values (like int.Parse(null))

Return an exception error: `System.FormatException: Input string was not in a correct format.`

4.11. Parse vs Convert (Performance difference when converting from string to other datatypes)
Parse is better in performance -> no overheads for checking if null or not or any exception handling in general unlike Convert

4.12. Stack VS Heap
https://www.linkedin.com/posts/marwan-hussein-568373314_csharp-dotnet-backend-activity-7421550651387641857-0FaG?utm_source=share&utm_medium=member_desktop&rcm=ACoAAE_JsmABvTFVwhoN_Ti9LOJQCQO2JKvt0pM this is my article about this subject

4.13. Deallocation
The garbage deallocator is the responsible for deallocate unreached objects

4.14. bitwise operator:
Firstly we treat in bits level (bits of an obj)
We have 4 bitwise operators (&, |, ^, >>, <<, ~)

2 bits		~ (complement)	& (and)	| (or)	^ (xor)	>> right shift	<< left shift
0	0	11	0	0	0	00	000
0	1	10	0	1	1	00	010
1	0	01	0	1	1	01	100
1	1	00	1	1	0	01	110

5- what meant by C# is managed code
its code is run under the supervision of the CLR that do:

- automatic memory management (GC)
- type safety: prevents illegal type conversions and memory corruption
- exception handling: Unified way to catch and handle runtime errors.
- Security: Enforces code access security and verification.

6- what meant by struct is considered like class before

Both participate in some features like both have attrs (attributes), methods, encapsulation. But the difference is that structs are value type (allocated in the stack) and the class objects allocated in the heap