Part01

Q1. What is the purpose of the finally block?

Always run regardless there is an exception or not

Usually, it's used in cleaning data like closing files, deleting pointers like in C, C++,..etc

Q2. How does int.TryParse() improve program robustness compared to int.Parse()?

In case an invalid integer, handles it by returning 0

Q3. What exception occurs when trying to access Value on a null Nullable?

The program throws a System.InvalidOperationException

Q4. Why is it necessary to check array bounds before accessing elements?

To avoid throwing an exception for any reason while program is running which is annoying for any user

Q5. How is the GetLength(dimension) method used in multi-dimensional arrays?

To return the number of elements in the dimensionth elements (0-based)

Q6. How does the memory allocation differ between jagged arrays and rectangular arrays?

	rectangular arr	jagged arr
structure	single contiguous block of memory for all elements	array of references (each points to an array)
access	faster (because of no overhead to go to the reference of something)	slightly slower because of the overhead
wasted memory	may waste some memory: may be some rows/columns aren't fully used	no memory waste: each reference linked with a static memory that allocated in the heap

Q7. What is the purpose of nullable reference types in C#?

In real apps may be a field isn't required and that translated as null in the database, so we'll use nullable reference types in the C# code.

Q8. What is the performance impact of boxing and unboxing in C#?

Memory overhead: Boxing allocates a new object on the heap, which is more expensive than stack allocation.

CPU overhead: Copying the value into a heap object and later casting it back adds extra instructions.

Garbage collection pressure: Each boxed value is a separate heap object, increasing GC workload.

Q9. Why must out parameters be initialized inside the method?

To be specialized over the normal argument in case calling the fnc

And to change the main reference in the body of the fnc

Q10. Why must optional parameters always appear at the end of a method's parameter list?

Imagine with me if it was through the list. Then there will be ambiguity to define where the argument is the same parameter in the fnc or not, need to through an exception or not. Hence, returning bad output

Q11. How does the null propagation operator prevent NullReferenceException?

If an array is null, then the desired property returns null. Else, return the normal property.

But in case using a property without using the null propagation operator and the arr = null -> the program will throw a NullReferenceException

Q12. When is a switch expression preferred over a traditional if statement?

In case a lot of cases

Q13. What are the limitations of the params keyword in method definitions?

Only one params allowed per method

It must be the last parameter

Works only with arrays