Part01

- **What is the purpose of the finally block?**

The finally statement is used with (try & catch) lets you execute code, after try...catch, regardless of the result**.**

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

Both Convert a string representation of number to an integer, If the string cannot be converted, then the int.TryParse method returns false i.e. a Boolean value, whereas int.Parse returns an exception.
Limitations of throwing an execption is terminating the program and Exceptions can be costly in terms of performance.
So ,int.TryParse is better in program robustness.

- **What exception occurs when trying to access Value on a null Nullable<T>?**

InvalidOperationException :Accessing its Value property when it does not contain a value throws an InvalidOperationException because it's an invalid operation.

- **Why is it necessary to check array bounds before accessing elements?**

Accessing an array element outside its bounds will result in an IndexOutOfRangeException in C#. This exception can cause your program to crash if not handled properly.This can prevent unpredictable behavior and potential data corruption.

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

It is used to determine the number of elements along a specified dimension in a multi-dimensional array.
Ex:

```
int[,] arr = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
int r = arr.GetLength(0);                // no. of rows
int c = arr.GetLength(1);               // no. of Columns
```

- ## How does the memory allocation differ between jagged arrays and rectangular arrays?
- Jagged Arrays

A jagged array is an array of arrays. This means that the elements of a jagged array are themselves arrays, which can vary in length. Each sub-array is stored separately in memory, and only the references to these sub-arrays are stored in the main array.
-Memory is allocated for the main array, which contains references (pointers) to the sub-arrays. Each sub-array is then allocated independently,

- Rectangular Arrays

A rectangular array (or multi-dimensional array) is a single contiguous block of memory where elements are laid out in a grid-like structure. All rows and columns have the same length.
-tend to be more memory-efficient for uniform grids because they allocate a single contiguous block of memory.

- ## What is the purpose of nullable reference types in C#?
- avoid NullReferenceException errors by providing a way to express the nullability of reference types explicitly.
- Nullable reference types help you write safer, clearer, and more robust code by explicitly defining the nullability of reference types and providing compiler support for null safety.

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

Memory: Boxing allocates memory on the heap for the new object, which increases memory usage and may lead to more frequent garbage collections.

CPU Overhead: Both boxing and unboxing involve additional CPU cycles for copying values and performing type checks.

Garbage Collection: Frequent boxing can lead to increased garbage collection overhead, as more objects are created on the heap.

Cache Efficiency: Value types are typically stored on the stack, which is faster to access. Boxing moves them to the heap, potentially reducing cache efficiency.

Why must out parameters be initialized inside the method?

to ensure that they have a definite value before the method returns. This provides a clear contract between the method and the caller, ensuring that the caller can rely on the fact that out parameters will be properly assigned a value.

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

to avoid ambiguity and ensure the compiler can correctly interpret which arguments are being passed explicitly and which ones are optional.
To ensures that there is no confusion between required and optional arguments when calling a method. Since The compiler processes method calls by matching provided arguments to parameters based on their position. Keeping optional parameters at the end simplifies this process and ensures correct interpretation of the arguments.

How does the null propagation operator prevent NullReferenceException?

by safely accessing members and methods on potentially null objects. This operator is represented by ?. and ?[].

When you use the null propagation operator, the expression is only evaluated if the object is not null. If the object is null, the expression short-circuits and returns null instead of throwing a NullReferenceException.

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

Performance: switch statements can be more efficient than if statements. This is because switch statements can be optimized by the compiler into a jump table or a binary search, depending on the number of cases. This can lead to faster execution, especially when dealing with a large number of cases.

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

The params parameter must be the last parameter in a method's parameter list.
A method can only have one params parameter.
The params keyword can only be applied to an array parameter. You cannot use it with other types of collections.
You cannot provide default values for params parameters.
Using params can introduce a performance overhead due to the creation of an array to hold the arguments.
All arguments passed to the params parameter must be of the same type as the array element type.