Part01

- **Question: What is the purpose of the finally block?**
  finally always runs, even if try/catch throws or return is called

- **Question: How does int.TryParse() improve program robustness compared to int.Parse()?**
  int.TryParse():
  If the user enters invalid input → returns false instead of crashing
  You can handle errors easily

- **Question: What exception occurs when trying to access Value on a null Nullable?**
  When you try to access .Value on a null Nullable<T>, an InvalidOperationException occurs.

- **Question: Why is it necessary to check array bounds before accessing elements?**
  Prevents IndexOutOfRangeException
  Keeps the program safe and robust

- **Question: How is the GetLength(dimension) method used in multi-dimensional arrays?**
  GetLength(dimension) returns the number of elements in the specified dimension of a multi-dimensional array.
  Dimension 0 → number of rows
  Dimension 1 → number of columns

- **Question: How does the memory allocation differ between jagged arrays and rectangular arrays?**
  Rectangular Arrays:
  All rows have the same number of columns
  Stored as a single continuous block in memory
  Jagged Arrays:
  Array of arrays; each row can have different lengths
  Each row is stored separately in memory

- **Question: What is the purpose of nullable reference types in C#?**
  Nullable reference types allow a reference variable to have the value null.

- **Question: What is the performance impact of boxing and unboxing in C#?**
  Boxing:
  Happens when a value type is converted to object or interface type
  Slower than using the value type directly due to heap allocation and copying
  Unboxing:
  Happens when an object is converted back to a value type
  Adds extra overhead due to type checking and copying

- **Question: Why must out parameters be initialized inside the method?**
  Out parameters are used to return values from a method
  The compiler requires that all out parameters are initialized before the method ends

- **Question: Why must optional parameters always appear at the end of a method's parameter list?**
  Optional parameters have default values that the caller can omit.

- **Question: How does the null propagation operator prevent NullReferenceException?**
  ?. allows access to a member only if the object is not null

  If the object is null, the expression returns null safely instead of throwing an exception.

- **Question: When is a switch expression preferred over a traditional if statement?**
  When you have multiple discrete values to check for a single variable.
  Switch expressions are more concise, readable, and less error-prone than multiple if-else statements

- **Question: What are the limitations of the params keyword in method definitions?**
  Type must be a single-dimensional array
  Must be the last parameter in the method
  Only one params parameter is allowed per method