Part01

- **Question: What is the default value assigned to array elements in C#?**
  int, long, short : 0
  float, double, decimal : 0.0
  bool :false
  char : (null character)
  Reference types : null

- **Question: What is the difference between Array.Clone() and Array.Copy()?**
  Array.Clone():
  Creates a shallow copy of the array.
  The new array is independent: changes to the clone do not affect the original array
  Array.Copy():
  Copies elements from one array to another existing array.
  Changes in the copied array affect the original array only if the elements are reference types, because it's a shallow copy of references.

- **Question: What is the difference between GetLength() and Length for multi dimensional arrays?**
  Length:
  Returns the total number of elements in the entire array.
  Ignores the number of dimensions.
  GetLength(dimension):
  Returns the number of elements in a specific dimension.

- **Question: What is the difference between Array.Copy() and Array.ConstrainedCopy()?**
  Array.Copy() :
  may copy some elements before the error occurs.
  Array.ConstrainedCopy():
  ensures that either all elements are copied successfully or none at all.

- **Question: Why is foreach preferred for read-only operations on arrays?**
  foreach preferred for read-only operations because it allows you to easily access each element without modifying the array, making the code simpler

- **Question: Why is input validation important when working with user inputs?**
  Input validation is important to make sure the user enters correct, safe, and usable data, prevent errors, and protect the program.
- **Question: How can you format the output of a 2D array for better readability?**
  You can print a 2D array using nested loops and separate the elements with tabs or spaces so that the rows and columns are easy to read.

- **Question: When should you prefer a switch statement over if-else?**
  Use a switch statement instead of if-else when you are checking one variable against many values. It is easier to read and organize.

- **Question: What is the time complexity of Array.Sort()?**
  The time complexity of Array.Sort() is generally O(n log n) for most cases.

- **Question: Which loop (for or foreach) is more efficient for calculating the sum of an array, and why?**
  for loop is more efficient because it uses the index directly

Part02

- **Define an enum called DayOfWeek with values: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.**
  Write a program that takes an integer input from the user (1-7) and prints the corresponding day using the enum.
  Use Enum.Parse to convert an integer to an enum value.

```
namespace ENumCSharp
{
    enum DayOfWeek
    {
        Saturdy=1,
        Sunday,
        Monday,
        Tuesday,
        Wednesday,
        Thursday,
        Friday
    }
    class Program
    {
        static void Main()
        {
            Console.WriteLine("Enter number from 1 to 7: ");
            int number=int.Parse(Console.ReadLine());

            DayOfWeek day=(DayOfWeek)Enum.Parse(typeof(DayOfWeek),number.ToString());
            Console.WriteLine("day is: " + day);
        }
    }
}
```

- **What happens if the user enters a value outside the range of 1 to 7?**
  Entering a number outside 1–7 will cause an exception, because there is no corresponding value in the DayOfWeek enum.

Part03 Bonus

- **what's the default size of stack and heap and what are the consideration?**
  Stack:
  A fast, temporary memory used to store value types like int, float, bool, and method call information.
  Default size: Typically around 1 MB per thread in .NET (may vary by system).

  Heap:
  Dynamic memory used to store reference types like class, string, object.
  Default size: Depends on available RAM and OS, usually very large (can be several GBs).

- **what is time complexity?**
  Time Complexity is a measure that shows how long an algorithm takes to run depending on the size of the input.

This means that as the input size increases, the execution time may increase depending on the type and design of the algorithm.