Question1:

For numeric types = 0

For bool = false

For reference types = null

Question2:

Clone: create new object in new and different address but with same data as the original.

Copy: it copy everything in the same object with same address.

Question3:

Length: total number of elements.

GetLength(): size of specific dimension.

Question4:

Array.Copy():standard method, may leave the destination array partially changed if something goes wrong.

Array>Constrainedcopy():safer method, ensures that either the copy succeeds completely or the destination array stays untouched.

Question5:

foreach :

is simpler and safer when you only need to read elements.

It automatically handles indexing, so you don’t risk going out of bounds.

It makes the code cleaner and more readable.

But you cannot modify the array elements inside a foreach loop that’s why it’s ideal for read-only operations.

Question6:

Prevents program crashes from invalid or unexpected input.

Protects against security risks (e.g., malicious input).

Ensures the program only works with correct, meaningful data.

Improves user experience by giving clear feedback when input is wrong.

Question7:

Use tabs (\t) or spaces to align columns neatly.

Print each row on a separate line to resemble a matrix.

Optionally, add labels or headers if needed (e.g., row/column numbers).

For larger arrays, consider using string formatting (like Console.Write("{0,4}", value)) to ensure consistent column width.

Question8:

Use switch when you are checking one variable against many constant values (like month numbers).

It makes the code cleaner, easier to read, and faster to understand compared to a long chain of if-else.

Use if-else when conditions are complex, involve ranges, or logical expressions .

Question9:

In C#, Array.Sort() uses an introspective sort (Introsort) algorithm, which is a combination of QuickSort, HeapSort, and InsertionSort.

The average time complexity is O(n log n).

The worst-case time complexity is also O(n log n) (because it switches to HeapSort if QuickSort recursion gets too deep).

The best case (already sorted or nearly sorted arrays) can approach O(n) due to InsertionSort optimization.

Question10:

Efficiency: Both for and foreach loops have the same time complexity O(n), since they iterate through all elements once.

Practical difference:

for gives you index control (you can access numbers[i] and manipulate positions).

foreach is simpler and cleaner when you only need to read values.