Top 21 C# Coding Interview Questions with Full Code & Explanations

1. Reverse a string without using in-built functions

```
string str = "Hello";
char[] chars = str.ToCharArray();
string reversed = "";
for (int i = chars.Length - 1; i >= 0; i--)
{
    reversed += chars[i];
}
Console.WriteLine("Reversed String: " + reversed);
```

Explanation: Convert string to char array, iterate from end to start, and build reversed string manually.

2. Check if a number is Prime

```
int num = 7;
bool isPrime = true;
if (num <= 1)
    isPrime = false;
for (int i = 2; i <= Math.Sqrt(num); i++)
{
    if (num % i == 0)
    {
        isPrime = false;
        break;
    }
}
Console.WriteLine(isPrime ? "Prime" : "Not Prime");
```

Explanation: A number is prime if not divisible by any number other than 1 and itself.

3. Factorial using Recursion

```
int Factorial(int n)
{
    if (n == 0) return 1;
    return n * Factorial(n - 1);
}
Console.WriteLine(Factorial(5));
```

Explanation: Recursive definition — base case n=0 returns 1; otherwise multiply by n-1 factorial.

4. Count vowels and consonants

```
string str = "Satish";
int vowels = 0, consonants = 0;
str = str.ToLower();
foreach (char ch in str)
{
    if ("aeiou".Contains(ch)) vowels++;
    else if (ch >= 'a' && ch <= 'z') consonants++;
}
Console.WriteLine($"Vowels: {vowels}, Consonants: {consonants}");
```

Explanation: Check each character for vowels and consonants by comparing against 'aeiou'.

5. Find the second largest element in an array

```
int[] arr = { 12, 35, 1, 10, 34, 1 };
int first = int.MinValue, second = int.MinValue;
foreach (int num in arr)
{
    if (num > first)
    {
        second = first;
        first = num;
    }
    else if (num > second && num != first)
        second = num;
}
Console.WriteLine("Second Largest: " + second);
```

Explanation: Track first and second maximum values without sorting.

6. Palindrome check

```
string str = "madam";
string reversed = new string(str.Reverse().ToArray());
Console.WriteLine(str == reversed ? "Palindrome" : "Not Palindrome");
```

Explanation: Reverse the string and compare both values.

7. Fibonacci Series (Recursive + Iterative)

```
int Fibonacci(int n)
{
    if (n <= 1) return n;
    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
int n = 10, a = 0, b = 1, c;
Console.Write(a + " " + b + " ");
for (int i = 2; i < n; i++)
{
    c = a + b;
    Console.Write(c + " ");
    a = b; b = c;
}
Console.WriteLine("\nRecursive:");
for (int i = 0; i < n; i++)
    Console.Write(Fibonacci(i) + " ");
```

Explanation: Iterative adds two previous numbers; recursive uses f(n)=f(n–1)+f(n–2).

8. Bubble Sort

```
int[] arr = { 5, 1, 4, 2, 8 };
for (int i = 0; i < arr.Length - 1; i++)
{
    for (int j = 0; j < arr.Length - i - 1; j++)
    {
        if (arr[j] > arr[j + 1])
        {
            int temp = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
        }
    }
}
```

```
Console.WriteLine(string.Join(", ", arr));
```

Explanation: Repeatedly swap adjacent elements until array becomes sorted.

9. Find duplicate elements in array

```
int[] arr = { 1, 2, 3, 2, 4, 5, 1 };
HashSet<int> seen = new HashSet<int>();
HashSet<int> duplicates = new HashSet<int>();
foreach (int n in arr)
{
    if (!seen.Add(n)) duplicates.Add(n);
}
Console.WriteLine("Duplicates: " + string.Join(", ", duplicates));
```

Explanation: Use HashSet to detect repeated elements efficiently.

10. Find GCD of two numbers

```
int GCD(int a, int b)
{
    return b == 0 ? a : GCD(b, a % b);
}
Console.WriteLine(GCD(54, 24));
```

Explanation: Use Euclidean algorithm recursively until remainder becomes 0.

11. Swap two numbers without temp variable

```
int a = 5, b = 10;
a = a + b;
b = a - b;
a = a - b;
Console.WriteLine($"a={a}, b={b}");
```

Explanation: Arithmetic swap without using a temporary variable.

12. Check if two strings are anagrams

```
string s1 = "listen", s2 = "silent";
bool result = s1.OrderBy(c => c).SequenceEqual(s2.OrderBy(c => c));
Console.WriteLine(result ? "Anagrams" : "Not Anagrams");
```

Explanation: Sort both strings alphabetically and compare.

13. Find missing number in array 1..N

```
int[] arr = { 1, 2, 4, 5, 6 };
int n = 6;
int expectedSum = n * (n + 1) / 2;
int actualSum = 0;
foreach (int num in arr) actualSum += num;
Console.WriteLine("Missing Number: " + (expectedSum - actualSum));
```

Explanation: Difference between expected and actual sum gives missing value.

14. Frequency of characters in a string

```
string str = "programming";
Dictionary<char, int> freq = new Dictionary<char, int>();
foreach (char ch in str)
```

```
{
    if (freq.ContainsKey(ch)) freq[ch]++;
    else freq[ch] = 1;
}
foreach (var kv in freq)
    Console.WriteLine($"{kv.Key}: {kv.Value}");
```

Explanation: Dictionary stores character counts for quick lookup.

15. Check if a number is Armstrong

```
int num = 153, sum = 0, temp = num;
while (temp > 0)
{
    int digit = temp % 10;
    sum += digit * digit * digit;
    temp /= 10;
}
Console.WriteLine(sum == num ? "Armstrong" : "Not Armstrong");
```

Explanation: Armstrong number equals sum of cubes of its digits.

16. Merge two sorted arrays

```
int[] arr1 = { 1, 3, 5 };
int[] arr2 = { 2, 4, 6 };
int[] merged = new int[arr1.Length + arr2.Length];
int i = 0, j = 0, k = 0;
while (i < arr1.Length && j < arr2.Length)
{
    if (arr1[i] < arr2[j]) merged[k++] = arr1[i++];
    else merged[k++] = arr2[j++];
}
while (i < arr1.Length) merged[k++] = arr1[i++];
while (j < arr2.Length) merged[k++] = arr2[j++];
Console.WriteLine(string.Join(", ", merged));
```

Explanation: Merge step similar to Merge Sort combining phase.

17. Left rotate array by D positions

```
int[] arr = { 1, 2, 3, 4, 5 };
int d = 2, n = arr.Length;
int[] rotated = new int[n];
for (int i = 0; i < n; i++)
    rotated[i] = arr[(i + d) % n];
Console.WriteLine(string.Join(", ", rotated));
```

Explanation: Shift elements by D positions using modulo arithmetic.

18. First non-repeating character

```
string str = "swiss";
Dictionary<char, int> count = new Dictionary<char, int>();
foreach (char c in str)
{
    if (count.ContainsKey(c)) count[c]++;
    else count[c] = 1;
}
foreach (char c in str)
{
    if (count[c] == 1)
    {
```

```
        Console.WriteLine("First Non-Repeating: " + c);
        break;
    }
}
```

Explanation: Count frequency, then print first character with frequency 1.

19. Linear & Binary Search

```
int LinearSearch(int[] arr, int key)
{
    for (int i = 0; i < arr.Length; i++)
        if (arr[i] == key) return i;
    return -1;
}
int BinarySearch(int[] arr, int key)
{
    int left = 0, right = arr.Length - 1;
    while (left <= right)
    {
        int mid = (left + right) / 2;
        if (arr[mid] == key) return mid;
        if (arr[mid] < key) left = mid + 1;
        else right = mid - 1;
    }
    return -1;
}
int[] arr = { 1, 2, 3, 4, 5, 6 };
Console.WriteLine("Linear Search: " + LinearSearch(arr, 4));
Console.WriteLine("Binary Search: " + BinarySearch(arr, 4));
```

Explanation: Linear checks sequentially; Binary divides sorted array until found.

20. Binary ↔ Decimal conversion

```
string binary = "1010";
int decimalValue = Convert.ToInt32(binary, 2);
Console.WriteLine("Binary to Decimal: " + decimalValue);
int num = 10;
string binaryStr = Convert.ToString(num, 2);
Console.WriteLine("Decimal to Binary: " + binaryStr);
```

Explanation: Use Convert methods for base conversions.

21. Move all zeros to end

```
int[] arr = { 0, 1, 0, 3, 12 };
int index = 0;
for (int i = 0; i < arr.Length; i++)
{
    if (arr[i] != 0)
        arr[index++] = arr[i];
}
while (index < arr.Length) arr[index++] = 0;
Console.WriteLine(string.Join(", ", arr));
```

Explanation: Move non-zero elements forward, fill rest with zeros.