✅ Core C# Concepts to Revise (Beginner to Intermediate)

# 1. C# Basics

* • Syntax (case sensitivity, semicolon, braces)
* • Main() method and structure of a program
* • Variables and Constants
* • Data Types (int, float, double, char, string, bool)
* • Type Conversion (Convert, Parse, ToString, casting)

# 2. Control Statements

* • if, else if, else
* • switch statement
* • for, while, do...while loops
* • break, continue, goto

# 3. Functions/Methods

* • Declaring and calling methods
* • Parameters and return values
* • ref, out, params
* • Method overloading
* • Recursion

# 4. Object-Oriented Programming (OOP)

* • Classes and Objects
* • Constructors (default, parameterized)
* • Inheritance
* • Polymorphism (method overriding and overloading)
* • Encapsulation (private, public, protected)
* • Abstraction (abstract class, interface)
* • this and base keyword
* • static members

# 5. Exception Handling

* • try, catch, finally, throw
* • Custom exceptions

# 6. Arrays and Collections

* • Single-dimensional and multi-dimensional arrays
* • List<T>, Dictionary<TKey, TValue>, Queue<T>, Stack<T>, HashSet<T>
* • LINQ basics on collections

# 7. Strings

* • Common string methods (Split, Replace, Substring, Contains, etc.)
* • String interpolation ($"Hello {name}")
* • StringBuilder

# 8. Structs, Enums, Tuples

* • struct vs class
* • enum usage
* • Tuple<T1, T2> and value tuples

# 9. File I/O

* • Reading and writing text files using File, StreamReader, StreamWriter

# 10. Advanced Concepts

* • Delegates and Events
* • Anonymous methods and Lambda expressions
* • LINQ (basic queries: Select, Where, GroupBy)
* • async and await (async programming basics)
* • Nullable types (int?, ?., ??)
* • Extension methods

# 11. .NET Concepts

* • What is CLR, CLS, CTS
* • Compilation process in .NET
* • Assemblies and DLLs
* • App.config / web.config basics
* • Connection strings

📘 C# Data Structures & Algorithms (DSA) Roadmap

# 1. Core Data Structures in C#

* • Array – Fixed size, indexed (int[], string[])
* • List – Dynamic array (List<T>)
* • Linked List – Efficient insert/delete (LinkedList<T>)
* • Stack – LIFO (Stack<T>)
* • Queue – FIFO (Queue<T>)
* • HashSet – Unique values, unordered (HashSet<T>)
* • Dictionary – Key-value pairs (Dictionary<K, V>)
* • SortedList / SortedDictionary – Sorted key-value pairs (SortedList<K, V>)

# 2. Key Algorithms (with C# examples)

* • Searching: Linear Search, Binary Search
* • Sorting: Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, List<T>.Sort()
* • Recursion: Fibonacci, Factorial, Tower of Hanoi
* • String Manipulation: Palindrome, Anagram check, Reverse string/words, Substring search
* • Hashing: Count frequency of chars/words, Detect duplicates using HashSet, Two Sum using Dictionary
* • Stack & Queue Problems: Valid Parentheses, Min Stack, Implement Queue using Stacks
* • Linked List Problems: Reverse a LinkedList, Detect loop (Floyd’s Algorithm), Merge two sorted LinkedLists
* • Trees: Inorder, Preorder, Postorder, BFS, DFS, Height/Depth, BST insert/search/delete
* • Graphs: Adjacency List, BFS, DFS, Topological Sort, Dijkstra’s Algorithm
* • Dynamic Programming: Fibonacci with memoization, LCS, Knapsack Problem, Grid path problems
* • Backtracking: N-Queens, Sudoku Solver, Subset/Permutation generation