**WEEK – 1 Assignments**

**Data structures and Algorithms**

**Mandatory hands-on :-**

**Exercise 7: Financial Forecasting**

**Scenario:**

You are developing a financial forecasting tool that predicts future values based on past data.

**Answers :**

**Recursion** :-

* **Recursion** is a programming technique where a function calls itself to solve a problem by breaking it into smaller, similar sub problems. It continues until it reaches a simple, “base case” that can be solved directly. Recursion simplifies problems like tree traversal, factorial calculation, and sorting by making the code more concise and easier to understand for tasks that naturally repeat in smaller steps.

## **Financial Forecasting with Recursion in C# Implementation :-**

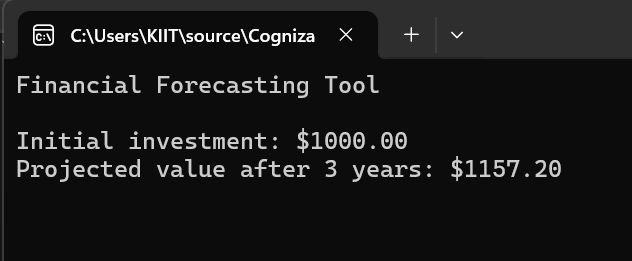
**Codes :-**

**File names :**

**Program.cs :**

| using System;  class Program {  static void Main()  {  Console.WriteLine("Financial Forecasting Tool\n");   // User input example  double initialValue = 1000.0;  double[] growthRates = { 0.05, 0.07, 0.03 }; // 5%, 7%, 3%  int forecastYears = 3;   double futureValue = CalculateFutureValue(  initialValue,  growthRates,  forecastYears  );   Console.WriteLine($"Initial investment: ${initialValue:F2}");  Console.WriteLine($"Projected value after {forecastYears} years: ${futureValue:F2}");   Console.ReadLine();  }   static double CalculateFutureValue(  double currentValue,  double[] growthRates,  int remainingYears)  {  if (remainingYears == 0)  {  return currentValue;  }   int currentIndex = growthRates.Length - remainingYears;  double growthRate = growthRates[currentIndex];  double nextValue = currentValue \* (1 + growthRate);   return CalculateFutureValue(  nextValue,  growthRates,  remainingYears - 1  );  } } |
| --- |

**Output :-**

****

**Analysis :-**

**Time Complexity:**

The recursive algorithm has O(n) time and space complexity, where *n* is the number of forecast years, because it makes one call per year and each call uses stack space.

**Optimization:**

To avoid excessive computation and stack overflow, use an iterative approach (a simple loop) instead of recursion. This keeps the time complexity at O(n) but reduces space complexity to O(1), making it more efficient for large datasets.