**Data structures and Algorithms**

Exercise 7: Financial Forecasting

1. **Understand Recursive Algorithms:**
2. **Explain the concept of recursion and how it can simplify certain problems.**

**Ans: Recursion** is a programming technique where a method calls itself to solve smaller instances of a problem. It simplifies problems that can be broken into similar sub-problems, like computing Fibonacci numbers or traversing trees. In financial forecasting, recursion can model compound growth over time by repeatedly applying a growth rate.

1. **Implementation:**

using System;

class FinancialForecast

{

public static double CalculateFutureValue(double currentValue, double growthRate, int years)

{

if (years == 0)

return currentValue;

return CalculateFutureValue(currentValue \* (1 + growthRate), growthRate, years - 1);

}

static void Main()

{

double initialValue = 10000;

double annualGrowthRate = 0.08;

int forecastYears = 5;

double futureValue = CalculateFutureValue(initialValue, annualGrowthRate, forecastYears);

Console.WriteLine($"Future value after {forecastYears} years: {futureValue:F2}");

}

}

1. **Analysis:**
2. **Discuss the time complexity of your recursive algorithm.**

Ans: The time complexity of this recursive function is **O(n)**, where n is the number of years. Each recursive call reduces the years by 1 until it reaches 0.

1. **Explain how to optimize the recursive solution to avoid excessive computation.**

Ans: Though this recursion is simple and not deeply nested, in more complex financial models, memorization or converting to **iteration** can prevent stack overflows and redundant calculations. An **iterative version** is preferred for larger inputs.

* **OUTPUT**

