**1.Understand Recursive Algorithms**

Recursion is a technique where a function calls itself to solve smaller parts of the same problem. It helps simplify complex problems by breaking them down into smaller, repeatable tasks.

**2.Setup**

We will create a method that calculates the future value of an investment using recursion. The method takes three inputs:  
- Present Value (initial amount)  
- Growth Rate (e.g., 0.1 for 10%)  
- Number of Years

**3.Implementation**

using System;  
  
class FinancialForecast  
{  
 public static double CalculateFutureValue(double presentValue, double rate, int years)  
 {  
 if (years == 0)  
 return presentValue;  
 return CalculateFutureValue(presentValue, rate, years - 1) \* (1 + rate);  
 }  
  
 static void Main(string[] args)  
 {  
 Console.WriteLine("Enter present value:");  
 double presentValue = Convert.ToDouble(Console.ReadLine());  
  
 Console.WriteLine("Enter annual growth rate (e.g. 0.1 for 10%):");  
 double rate = Convert.ToDouble(Console.ReadLine());  
  
 Console.WriteLine("Enter number of years:");  
 int years = Convert.ToInt32(Console.ReadLine());  
  
 double futureValue = CalculateFutureValue(presentValue, rate, years);  
  
 Console.WriteLine($"  
Future Value after {years} years: {futureValue:F2}");  
 }  
}

**4.Analysis**

- The function keeps calling itself, reducing the number of years each time.  
- Once years reach 0, it returns the present value.  
- It then builds up the result with the growth rate applied each time it returns from the previous call.  
  
Time Complexity:  
- The function runs once for each year.  
- So, for n years, it runs n times.  
- Time Complexity = O(n)  
  
Optimization:  
- Recursion is simple but not always efficient.  
- For large values of years, it’s better to use a loop.

**Output:**

