**Superset id:- 6363303**

**Exercise 7: Financial Forecasting**

**Code:-**

**using System;**

**public class Forecast**

**{**

**public static double Future(double p,double r,int y)**

**{**

**if(y==0)**

**{**

**return p;**

**}**

**return Future(p,r,y-1)\*(1+r);**

**}**

**public static void Print(double p,double r,int y)**

**{**

**Console.WriteLine("\nprediction:-");**

**for(int i=0;i<=y;i++)**

**{**

**double val=Future(p,r,i);**

**Console.WriteLine($"year {i}: ₹ {val:F2}");**

**}**

**}**

**public static void Main()**

**{**

**try**

**{**

**Console.Write("Initial amount: ");**

**double p=Convert.ToDouble(Console.ReadLine());**

**Console.Write("Growth rate(%): ");**

**double r=Convert.ToDouble(Console.ReadLine())/100;**

**Console.Write("Years: ");**

**int y=Convert.ToInt32(Console.ReadLine());**

**double final=Future(p,r,y);**

**Console.WriteLine($"\nFinal Value: ₹ {final:F2}");**

**Print(p,r,y);**

**}**

**catch**

**{**

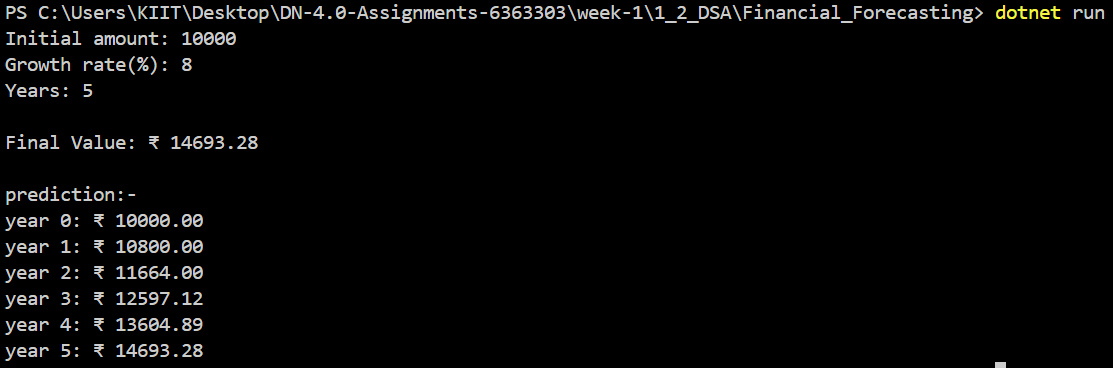
**Console.WriteLine("Invalid input");**

**}**

**}**

**}**

**Output:-**

****

**Recursion:-**

Recursion is a programming technique where a method calls itself to solve smaller parts of the same problem. It’s useful for breaking down problems that have a repetitive structure.

**Time Complexity:-**

The time complexity of this recursive method is **O(n)** where **n** is the number of years. It makes one recursive call per year.

**Optimization:-**

Recursion can lead to **repeated calls and stack overflow** for large n.

You can optimize it using:-

* **Memoization:**-
* Store already computed values in a dictionary or array to avoid recalculating them.
* Reduces time complexity from **exponential** to **linear** in many cases.
* **Iterative Approach-**
* Replace recursion with loops.
* Removes stack usage and improves performance, especially for large n.
* **Tail Recursion-** A special form of recursion that some compilers optimize internally to behave like iteration.

**Ayush Kumar (6363303)**