**Exercise 7: Financial Forecasting**

**Q) Explain the concept of recursion and how it can simplify certain problems.**

Recursion is a programming technique where a function calls itself to solve a smaller instance of the same problem. It simplifies problems that have a repetitive or self-similar structure, such as calculating factorials, Fibonacci numbers, or traversing data structures like trees. Recursion helps break down complex problems into manageable subproblems, leading to cleaner and more intuitive code.

**Code:**

**Main Class: -**

public class FinancialForecast {  
 public static double calculateFutureValue(double presentValue,double growthRate,int years){  
 if(years==0){  
 return presentValue;  
 }  
 return *calculateFutureValue*(presentValue\*(1+growthRate),growthRate,years-1);

}  
public static void main(String[] args){  
 double presentValue=1000.0;  
 double growthRate=0.05;  
 int years=10;  
 double futureValue=*calculateFutureValue*(presentValue,growthRate,years);  
 System.*out*.printf("Future Value after %d years: %.2f%n",years,futureValue);  
 }  
}

**Output:**

A screen shot of a computer program

AI-generated content may be incorrect.

**Q) Discuss the time complexity of your recursive algorithm.**

The time complexity of a recursive algorithm depends on how many recursive calls it makes and how much work is done in each call. For example, a naive recursive solution for forecasting based on previous values (like in Fibonacci) may have exponential time complexity, O(2ⁿ), due to repeated calculations of the same subproblems.

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

To avoid excessive computation in recursive algorithms, techniques like memoization or dynamic programming can be used. Memoization stores results of already computed subproblems, so they are not recomputed, thus reducing the time complexity from exponential to linear (e.g., O(n)). This significantly improves performance and makes recursion practical for larger inputs.