## Exercise 7: Financial Forecasting

### 1. Understand Recursive Algorithms:

- Recursion is a method where a function calls itself to solve smaller instances of a problem.

- It simplifies problems like compound growth calculations where future values depend on smaller subproblems.

### 2. Setup:

The future value (FV) is calculated using the formula:

FV = PV × (1 + r)^n

Where:

- PV = Present Value

- r = Annual Growth Rate

- n = Number of Years

### 3. Implementation:

Recursive Java method to compute future value:

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

  public static void main(String[] args) {
    double presentValue = 10000;
```

```
        double annualRate = 0.05; // 5%
        int years = 5;

        double futureValue = calculateFutureValue(presentValue, annualRate, years);
        System.out.printf("Future Value after %d years: %.2f\n", years, futureValue);
    }
}
```

Output:

```
Future Value after 5 years: 12762.82
```

## 4. Analysis:

- Time Complexity: O(n), since the function makes n recursive calls (one per year).

- Recursive approach is simple but not optimal for large values of n due to risk of stack overflow.

## Optimization:

Use an iterative version to improve performance and reduce stack memory usage:

```
public static double calculateFutureValueIterative(double presentValue, double rate, int
years) {
    for (int i = 0; i < years; i++) {
        presentValue *= (1 + rate);
    }
    return presentValue;
}
```

## Conclusion:

- Recursion is useful for understanding problem structure.

- For large inputs or production systems, prefer iterative solutions to avoid stack overhead.