Exercise 7: Financial Forecasting

Scenario

You are developing a financial forecasting tool that predicts future values based on past data. This is achieved using a recursive algorithm to model compound growth over a given number of years.

Step 1: Recursive Algorithms Overview

What is Recursion?

Recursion is a programming technique where a method calls itself to solve smaller instances of the same problem. It is especially useful for problems with a **repetitive and self-similar structure**.

Why Use Recursion Here?

In financial forecasting, the future value can be defined in terms of the previous year's value multiplied by a growth factor. This repetitive nature makes recursion an intuitive solution.

Step 2: Setup

A method was created to recursively calculate the future value of an amount, given:

- Current value
- Annual growth rate (as a decimal)
- Number of years to forecast

Step 3: Implementation

Recursive Function

```
public static double PredictFutureValue(double currentValue, double growthRate, int years)
{
    if (years == 0)
        return currentValue;

    return PredictFutureValue(currentValue * (1 + growthRate), growthRate, years - 1);
}
```

Explanation

Each recursive call reduces the number of years by 1, while multiplying the current value by `(1 + growthRate)` to simulate compound growth.

Execution Example (as per your output)

Input Parameter	Value
Initial Value	1500
Annual Growth Rate	0.35 (35%)
Forecast Period	4 years

Output:

```
Predicted value after 4 years: 4982.26
```

This output aligns with compound interest growth:

```
Year 1: 1500 * (1 + 0.35) = 2025.00
Year 2: 2025 * (1 + 0.35) = 2733.75
Year 3: 2733.75 * (1 + 0.35) = 3680.56
Year 4: 3680.56 * (1 + 0.35) = 4972.76 (rounded)
```

Step 4: Analysis

Time Complexity

- **Recursive Depth**: Equals the number of years (n)
- **Time Complexity**: O(n)
- **Space Complexity**: O(n) due to the call stack

Potential Issue

- For very large values of `n` (e.g., 10,000 years), this approach could lead to **stack overflow** due to deep recursion.

Optimization Recommendation

Use an iterative approach or tail-recursion to avoid call stack growth:

Iterative Alternative:

```
public static double PredictFutureValueIterative(double currentValue, double growthRate, int years)
```

```
{
   for (int i = 0; i < years; i++)
   {
      currentValue *= (1 + growthRate);
   }
   return currentValue;
}
```

This version has the same time complexity (O(n)) but constant space complexity (O(1)) and no recursion depth issues.

Conclusion

The recursive forecasting algorithm is simple and works correctly for a moderate number of years. However, in production systems, an iterative version is more robust and scalable. For financial platforms, maintaining high performance and avoiding runtime risks makes the iterative solution preferable.