Data Structures and Algorithms in Python

Assignment 16 - Recursive Algorithms and Fractal Graphics

In this assignment, you will explore the concept of *recursion*, a fundamental technique in algorithm design that involves a function calling itself to solve subproblems of a given problem. Recursive algorithms are particularly powerful in tasks that can be broken down into similar subtasks, such as sorting data (*QuickSort*, *MergeSort*) and generating complex graphical patterns—fractals.

Fractals are intricate geometric shapes that can be split into parts, each of which is a reduced-size copy of the whole. This self-similar property makes fractals excellent subjects for recursive algorithms.

Reading Material

- Read Chapter 4: Recursion from *Data Structures and Algorithms in Python*. Focus on sections 4.1 (Illustrative Examples), 4.2 (Analyzing Recursive Algorithms), and 4.4 (Further examples of Recursion).
- Review the Python `turtle` module documentation: Python Turtle Documentation

Programming

- Implement a recursive fractal pattern using the `turtle` graphics library. You may choose from the following patterns or propose a new one (please discuss with me first):
    - **Sierpiński Triangle:** Described by Waclaw Sierpiński in the 1910s. Comprises a triangular design of nested smaller triangles.
    - **Koch Snowflake:** Described by Helge von Koch in 1904. Features a pattern that builds detailed snowflake outlines through iterative detail enhancements.
    - **Dragon Curve:** Made famous from Michael Crichton's Jurassic Park, folds a line into a complex "space-filling" curve.
    - **H-Tree:** An H-tree is a simple example of a self-similar fractal, where each iteration repeats the structure in a reduced size at the ends of the letter 'H'.
- Use recursive functions to draw the fractals. Ensure your fractals are well-formed with appropriate base cases to stop the recursion.
- Your program should allow the user to specify the depth of recursion as a console input.
- Include comments in your code to explain the purpose of each function and major steps in your code. Also add comments indicating sections of your code that were mostly AI-generated.
- Go beyond these minimum requirements. Examples: implement multiple fractals and add a GUI to let the user choose between them, add color to the fractal patterns to represent the depth of recursion, or an original idea.

Written Questions

- Include screenshots of the output fractals with different levels of recursion. Explain any challenges you faced during the implementation and how you overcame them.
- Complete the following exercises from Chapter 4 in *Data Structures and Algorithms in Python*:
    - R-4.2
    - R-4.3
    - C-4.9 (create a Python file)
    - C-4.14

Submission

- Submit your Python files and your answers to the written questions in a PDF document via CREEKnet.