Software Metrics Course Assignment 1: Measurement Instruments

Definitions:

- Physical LoC: Total number of lines in the source file, including code, blank lines, and comments.
- Logical LoC: Approximate count of executable statements in the program. For C/Java/Python. This includes semicolons, return statements, braces; for Python, lines ending with ´:´ or containing assignments.
- McCabe Complexity: Number of independent paths through a function. Calculates as 1 plus the sum of decision points (if, for, while, case, catch, etc.) in the function. The McCabe complexity of a script is the total complexity of all the functions of the script.
- Fan-in: Number of functions that call the function. Fan-in for a repo is the sum of the Fan-in of all functions across the module.
- Fan-out: Number of function calls made by the function. Fan-out for a repo is the sum of the Fan-out of all functions across the module.

Summary of measurement of the provided programs:

Program name	Physical LoC	Logical LoC	McCabe Complexity	Fan-in	Fan-out
fibonacci_1.c	35	26	6	1	5
fibonacci_2.c	70	26	6	1	5
fibonacci_3.c	54	25	8	2	5
fibonacci_4.c	79	25	8	2	5
fibonacci_4.java	82	26	9	3	8
fibonacci_4.py	72	18	4	1	1
fibonacci_5.py	39	9	2	0	0

Summary of the measurement of the *Linux* kernel:

- **Repo-wise**

Physical LoC	35946381
Logical LoC	6991016
McCabe Complexity	4437448
Fan-in	1363654
Fan-out	4885563

- **Module-wise**

Module	Physical LoC	Logical LoC	McCabe Complexity	Fan-in	Fan-out
linux/init	4549	1091	691	254	837
linux/crypto	100541	18555	7741	2913	14601

--------------	---------------	------------	---------------	------------	-------------

- The full results are available in the results/linux_results.json file.

Summary of measurement of *Pandas* repo:

- **Repo-wise**

Physical LoC	644892
Logical LoC	179758
McCabe Complexity	244820
Fan-in	30615
Fan-out	178889

- **Module-wise**

Module	Physical LoC	Logical LoC	McCabe Complexity	Fan-in	Fan-out
pandas/	788	196	60	21	107
pandas/web	506	185	37	12	153
--------------	---------------	------------	---------------	------------	-------------

The full results are available in the results/pandas_results.json file.

Summary of measurement of *Apache Hadoop* repo:

- **Repo-wise**

Physical LoC	3164998
Logical LoC	830011
McCabe Complexity	205752
Fan-in	109562
Fan-out	1006365

- **Module-wise**

Module	Physical LoC	Logical LoC	McCabe Complexity	Fan-in	Fan-out
hadoop/yarn/applications/distributedshell	3133	1094	120	59	1416
hadoop/yarn/applications/unmanagedamlauncher	3696	516	182	73	461
--------------	---------	---------	-----------	----	-------

The full results are available in the results/hadoop_results.json file.

General Observations and Patterns:

1. **Overall Complexity and Size:**
    - **Linux kernel** (C, ~36 million physical LOC) has **extremely high McCabe complexity, Fan-in, and Fan-out**, since it's a large system with many interdependent modules.
    - **Apache Hadoop** (Java, ~3.1 million LOC) shows high Fan-out relative to Fan-in, probably because of the fact that Java functions tend to call many other functions but are less reused across modules.
    - **Pandas** (Python, ~645k LOC) has lower complexity overall, with moderate Fan-in and Fan-out. This reflects Python's concise syntax and high-level abstractions.
2. **Language Trends**
    - **C programs** (Linux kernel, Fibonacci examples) tend to have **high Fan-out** and higher McCabe complexity for similar logical LOC because low-level control structures (if, for, while, switch) are more verbose and explicit.
    - **Java projects** (Hadoop) show **higher Fan-in** in some modules due to object-oriented design and reuse through classes and method calls.
    - **Python projects** (Pandas, Fibonacci scripts) generally have **lower McCabe complexity** per module because Python uses concise expressions and fewer explicit branches.

N.B: Although Pandas has fewer than 1 million LOC, it was included for comparison because it is one of the largest and most widely used Python repositories. It provides a useful contrast in language-specific complexity against the Linux kernel (C) and Hadoop (Java).