Jonathan McKinsey

The Beauty and Joy of Computing

Lecture #8
Recursion I

Open Source vs Free vs Proprietary Software

- Open Source SW – "Free" as in no cost
- Free SW – "Free" as in Freedom
- Proprietary SW – closed source and costs $$
- Why Open Source and Free?
  Linus' Law - "given enough eyeballs, all bugs are shallow"

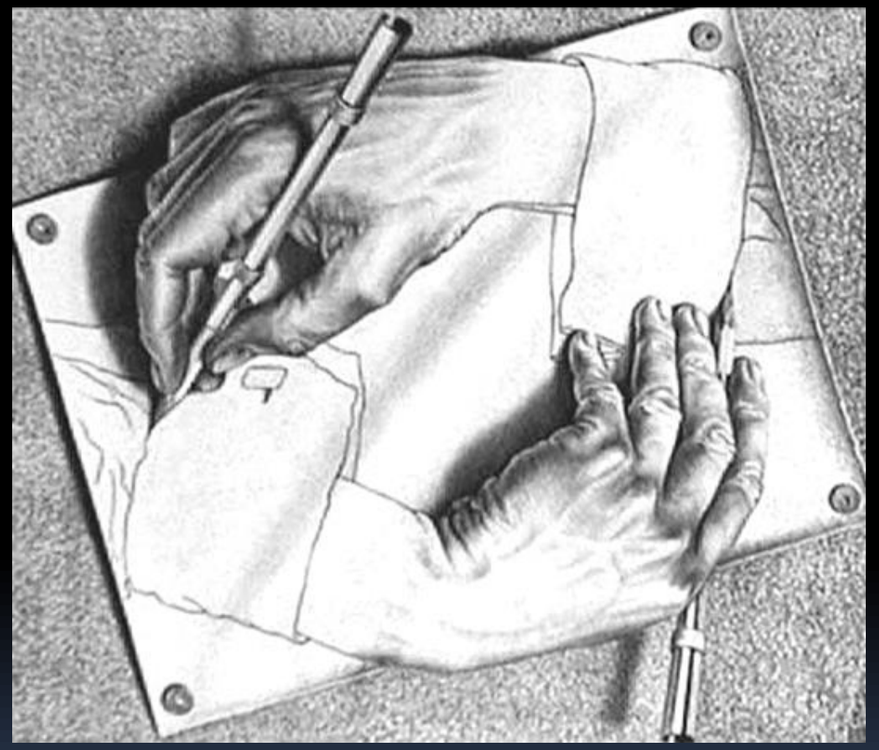
www.gnu.org/philosophy/free-software-for-freedom.en.html
www.theregister.co.uk/2014/12/27/2014_in_open_source_microsofts_cancer_nasdaq_listings_and_a_quiet_dog

www.worldofescher.com/gallery/A13.html

Overview

- Recursion
  - Demo
    - Vee example & analysis
    - Downup
  - You already know it!
  - Definition
  - Trust the Recursion!
  - Conclusion

M. C. Escher : *Drawing Hands*



McKinsey

Recursion: Vee Demo

Recursion: Downup Demo

"I Understood Vee & Downup"

a) Strongly agree

b) Agree

c) Neutral

d) Disagree

e) Strongly disagree

M. C. Escher : *Fish and Scales*

McKinsey

Recursion: Definition, You Know It, Trust It

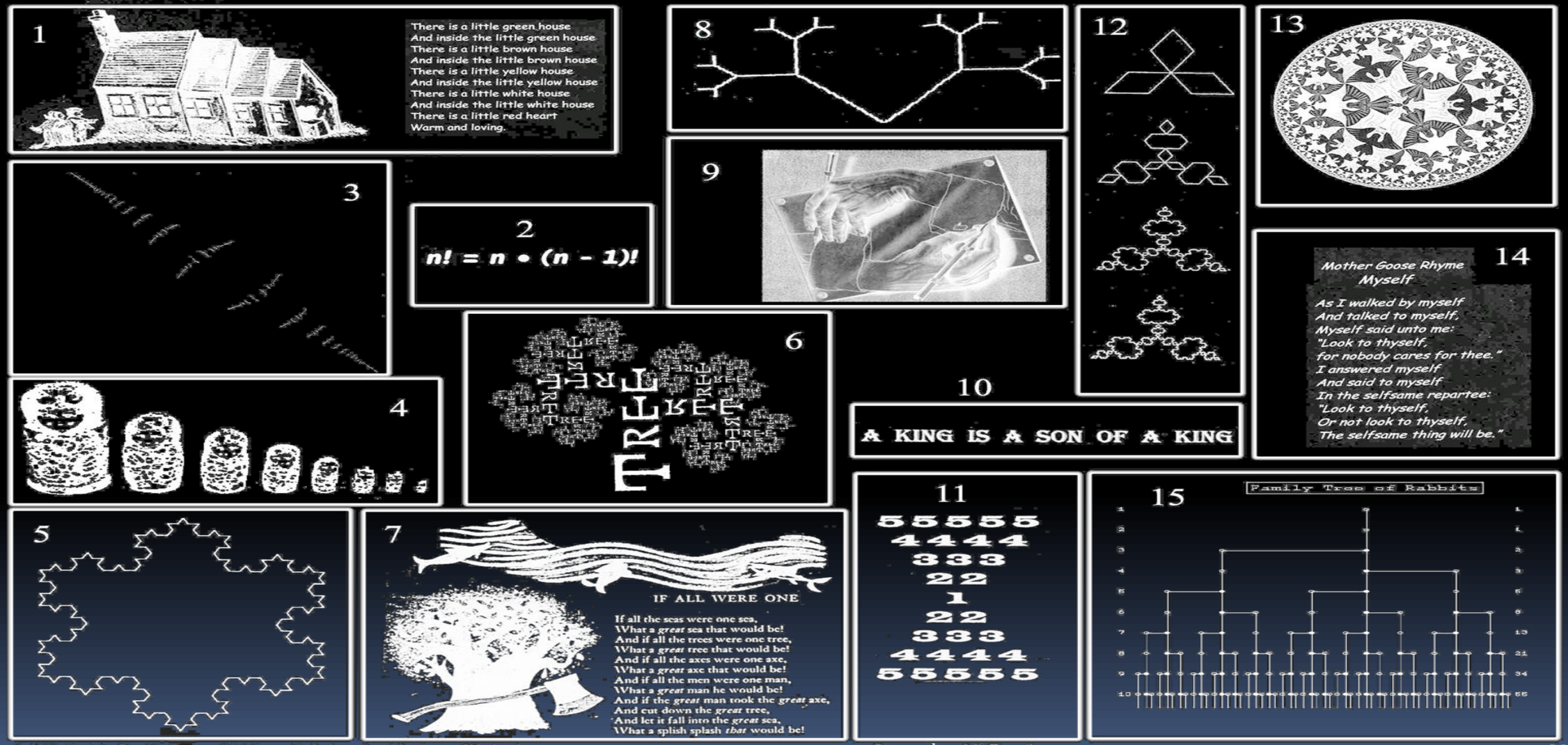
www.catb.org/~esr/jargon/html/R/recursion.html
www.nist.gov/dads/HTML/recursion.html

Definition

- Recursion: (noun) See recursion. ☺
- *An algorithmic technique where a function, in order to accomplish a task, calls itself with some part of the task*
- Recursive solutions involve two major parts:
  - Base case(s), the problem is simple enough to be solved directly
  - Recursive case(s). A recursive case has three components:
    - Divide the problem into one or more simpler or smaller parts
    - Invoke the function (recursively) on each part, and
    - Combine the solutions of the parts into a solution for the problem.
- Depending on the problem, any of these may be trivial or complex.

McKinsey

You already know it!



McKinsey

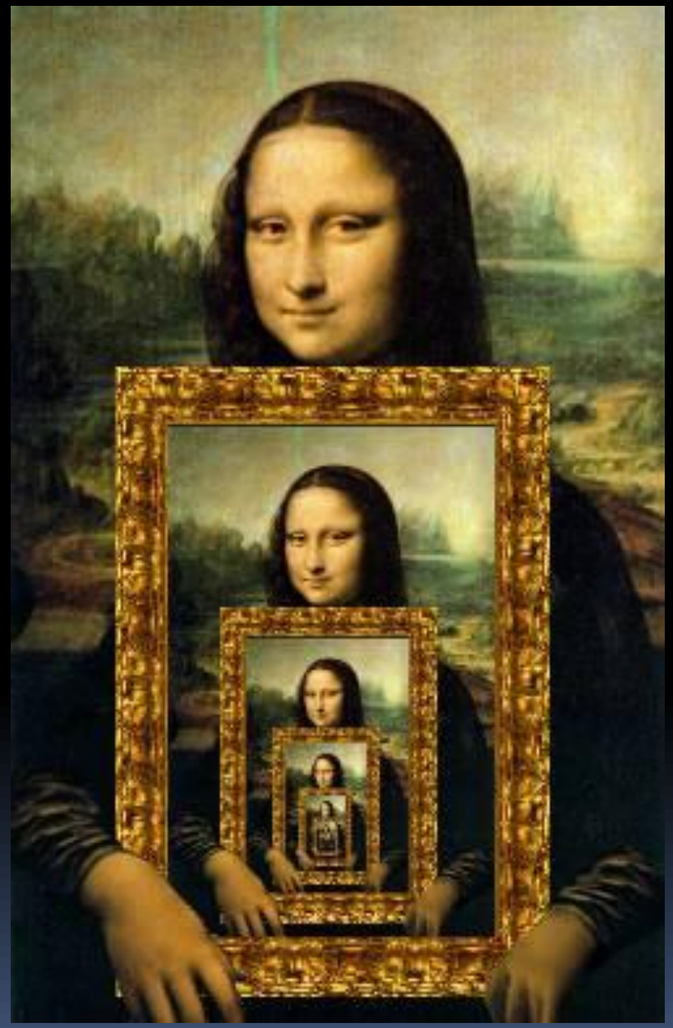
Trust the Recursion

- When authoring recursive code:
  - The base is usually easy: "when to stop?"
  - In the recursive step
    - How can we break the problem down into two:
      - A piece I can handle right now
      - The answer from a smaller piece of the problem
    - Assume your self-call does the right thing on a smaller piece of the problem
    - How to combine parts to get the overall answer?
- Practice will make it easier to see idea

McKinsey

Recursion Versus Iteration

- Recursion is ■ Iteration (i.e. loops)
  a) more powerful than
  b) just as powerful as
  c) more powerful than
  d) just as powerful as



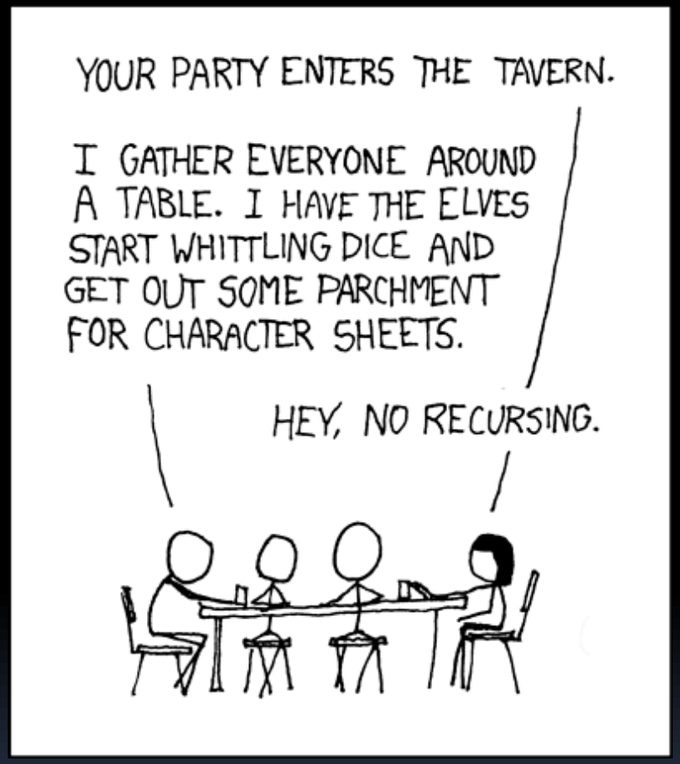
http://www.dominiek.eu/blog/?m=200711

McKinsey

Summary

- Behind Abstraction, **Recursion is the 2nd biggest idea about programming in this course**
- Format (usually) is 2 cases:
  - Base Case
  - Recursive case
    - Divide, Invoke, Combine
- It's most useful when the problem is self-similar
- It's no more powerful than iteration, but **often leads to more concise & better code**

xkcd.com/244/



☑ McKinsey