Abstraction in Education

Arnold L. Rosenberg
Distinguished University Professor Emeritus
University of Massachusetts

It is said that if you *give a person a fish*, you feed him/her for a day. If you *teach the person how to fish*, you feed him/her for a lifetime.

This aphorism has been my roadmap, as a *learner* and a *teacher*.
Inspired by the aphorism, I strive to:

- *educate* rather than *train*
- *abstract* from *concrete and immediate* to *fundamental and essential*

⋆ Inspire each student to *become* a mathematician or an informaticist, rather than just *learn* some mathematics or some informatics.

⋆ Equipped with *understanding* of "*why?*" rather than just "*how?*" a student can confront new situations with courage and creativity.

Examples:

1. Stress the importance of *representations* in computing. For integer n:
   [n "is" a collection of n tokens] $\Rightarrow$ [Summing "is" rearranging tokens]
   [n "is" an $n \times 1$ rectangle] $\Rightarrow$ [Can "approximately" sum via integration]
2. Study parallel computing via *dependency dags*, not programs
   — *identify sources of concurrency.*
   (a) *Expansion-Reduction* computations:
   Shared structure:
   - *Parallel mergesort*
   - *Numerical integration*

   (b) *Parallel-Prefix* computations:
   Shared structure:
   - *Carry-lookahead addition*
   - *Computing paths in graphs*

   (c) *Butterfly-Structured* computations:
   Shared structure:
   - *Convolutions (e.g.,* polynomial multiplication*)*
   - *Odd-even mergesort*
3. Understand the *logical complexity* of computation
   —e.g., *self-referential systems*
   Shared structure:
   - *Self-reference in languages* (Gödel)
   - *Nonexistence of certain infinite sets* (Russell)
   - *"Bigger" and "Smaller" infinities* (Cantor)