Solving an Easy Problem

- What are the input types? What is the output type? Give example input/output pairs.
- Which input represents the domain of the recursion, *i.e.,* which input becomes *smaller*? How is problem size defined?
- What function is used to produce smaller problem instances?
- What is the output value when the problem is *smallest*?

Solving an Easy Problem (contd.)

- How can a problem instance be reduced to one or more *smaller* problem instances? What function creates the output value?
- Is your case analysis correct and complete?
- If an input can be of more than one type, *e.g.,* sometimes an atom, sometimes a pair, then you will need to provide a case for each type.

Solving a Hard Problem

- Identify one (or more) sub problems that would make the hard problem into an easy problem if solved.
- Give example input/output pairs for helper functions which would solve the sub problems.
- Define the helper functions and test your solutions.
- If any of the sub problems are hard themselves then identify additional helper functions which would permit you to solve *them.*

Debugging Imperative Programs

- An imperative program is understood by the programmer as a process which transforms the state of an abstract machine.
- The state of the abstract machine is comprised of the values of variables and the contents of the stack and heap.
- By observing how the values of variables change over time, the programmer verifies that the process is defined correctly.

Debugging Functional Programs

- A functional program is understood by the programmer as the definition of the solution to a problem.
- A functional programmer fixes errors by reformulating this definition using new terms.
- These terms are the solutions of sub problems each of which can be independently verified by testing.
- A functional program is debugged by rewriting it using simpler and simpler pieces until each piece is demonstrably correct.

Solving an Easy Problem

- What are the input types? What is the output type? Give example input/output pairs.
- Which input represents the domain of the recursion, *i.e.,* which input becomes *smaller*? How is problem size defined?
- What function is used to produce smaller problem instances?
- What is the output value when the problem is *smallest*?

Solving an Easy Problem (contd.)

- How can a problem instance be reduced to one or more *smaller* problem instances? What function creates the output value?
- Is your case analysis correct and complete?
- If an input can be of more than one type, *e.g.,* sometimes an atom, sometimes a pair, then you will need to provide a case for each type.

Solving a Hard Problem

- Identify one (or more) sub problems that would make the hard problem into an easy problem if solved.
- Give example input/output pairs for helper functions which would solve the sub problems.
- Define the helper functions and test your solutions.
- If any of the sub problems are hard themselves then identify additional helper functions which would permit you to solve *them.*

Debugging Imperative Programs

- An imperative program is understood by the programmer as a process which transforms the state of an abstract machine.
- The state of the abstract machine is comprised of the values of variables and the contents of the stack and heap.
- By observing how the values of variables change over time, the programmer verifies that the process is defined correctly.

Debugging Functional Programs

- A functional program is understood by the programmer as the definition of the solution to a problem.
- A functional programmer fixes errors by reformulating this definition using new terms.
- These terms are the solutions of sub problems each of which can be independently verified by testing.
- A functional program is debugged by rewriting it using simpler and simpler pieces until each piece is demonstrably correct.