Recursion wrap-up

CSC148, Introduction to Computer Science
Fall 2016
Diane Horton
Base cases are important

• There must be at least one base case.
• There may be more than one.
• Any call to a recursive method must ultimately reach a base case.
  – Otherwise, we have “infinite” recursion.
  – It can’t continue indefinitely, because each recursive call needs a stack frame, and we will stop due to running out of memory.
Recursive cases

• For some methods, a call will recurse just once.
• For others, a call will recurse multiple times.

• For some methods, the size of the problem reduces by 1 on each recursive call.
• For others, it reduces by half.
Communicate via parameters & return

• Each time we call a recursive method:
  – Everything it needs should be sent through parameters
  – Everything it must report back should come through parameters

• Don’t attempt to work around this protocol by using local variables.
  – Each call has its own stack from with its own instance of the local variables.
  – So nothing can accumulate in them across calls.
Recursion vs iteration

• Any problem we can solve with iteration can be solved with just recursion.
  – Some languages have *nothing but* recursion!

• Any problem we can solve with recursion can be solved with just iteration.
  – Recursion doesn’t add “expressive power”

• But some problems have simple, elegant recursive solutions, and only complex non-recursive solutions.
  – Try writing tree traversal with no recursion, *not even through helpers.*
Tips for writing recursive functions

• Think lazy.
  – What smaller instance(s) of the same problem can I ask someone to solve for me?
  – When the problem is so small that even lazy you can do it, write the code directly.

• When writing a recursive case:
  1. Pick a concrete example, and determine what the result should be.
  2. Write down each recursive call and what it should do.
  3. Simply assume each one does do what it should.
  4. Figure out how to combine their results to produce the overall result.