Another helper function rule:

When the body of a function must shift to a new knowledge domain it should call a helper function to do the work in the new domain.

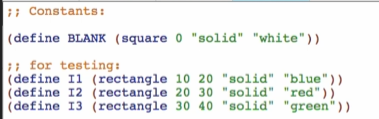
Working through the wish list entry of insert function



**Examples**

Making constants out of examples that you have been using a lot

* Make a section of constants: for testing
* Replace every occurrence for all of the instance of your examples



Back to insert Function

Base case: goes at the beginning of an empty list



Other cases:

First: goes at the beginning of a non-empty list

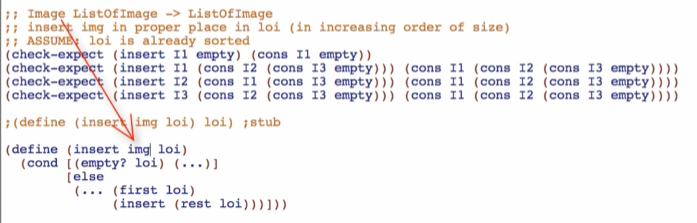
Middle: goes at the middle of a non-empty list

Last: goes at the end of a non-empty list

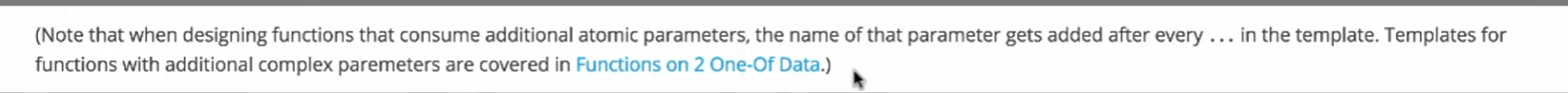


**Template**

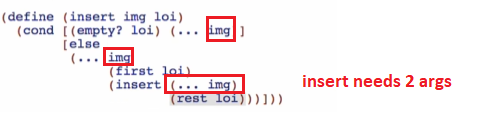
* Rename lst -> loi for consistency
* Rename functions and natural recursions
* Add the 2nd parameter



Go to design recipe page:

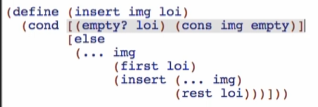


Add the additional img parameter

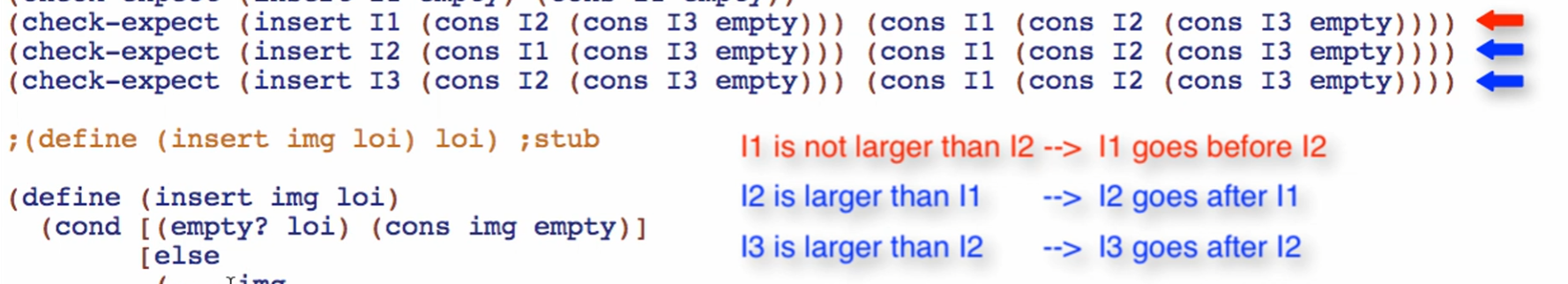


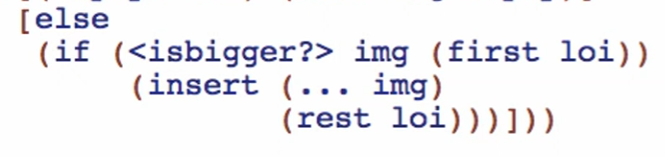
**Code Body:**

Base case:



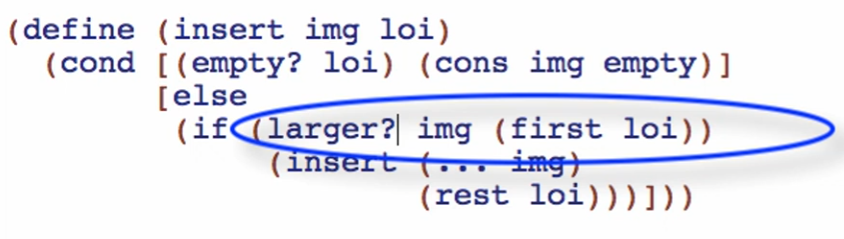
Other cases:



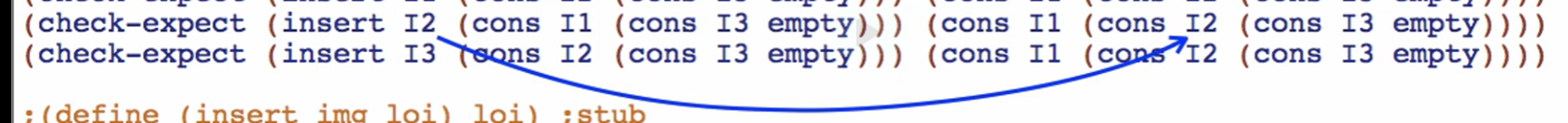


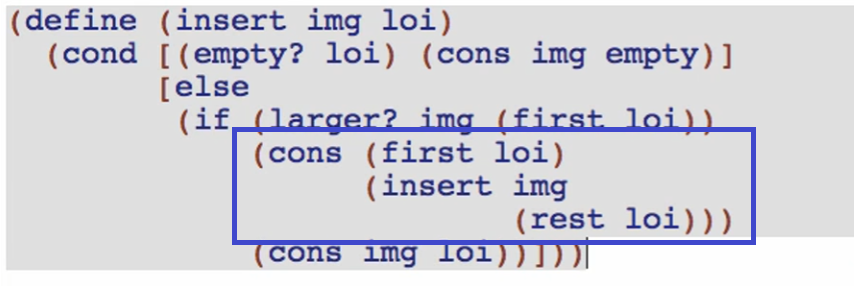
* But this is about comparing the sizes of 2 images, not INSERTING into a sorted list
* There is a **domain knowledge shift**
  + If there is a domain knowledge shift, use a HELPER function!
  + Switching from inserting to comparing of sizes

Wish for larger? Function for the comparing of the sizes of 2 images

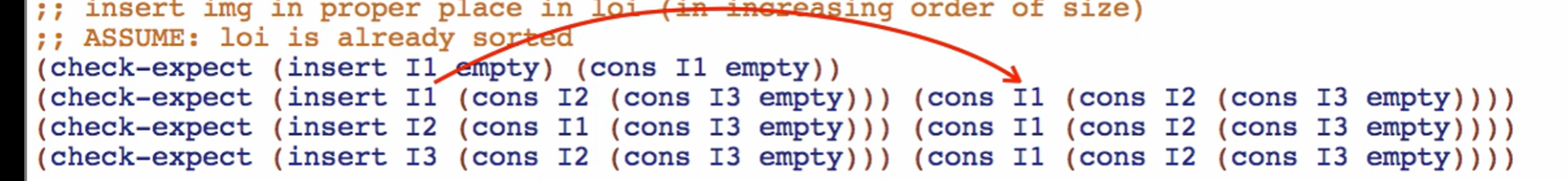


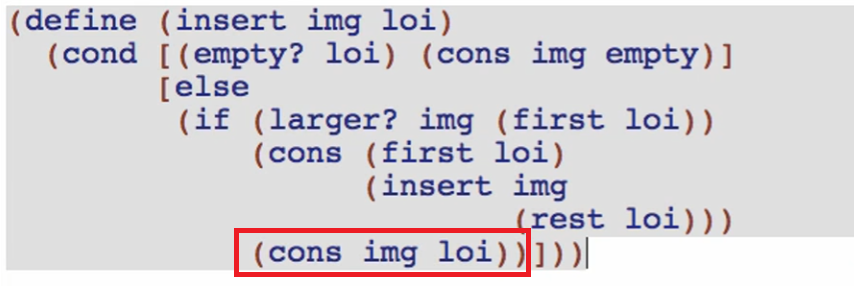
Finish the code body for insert Function





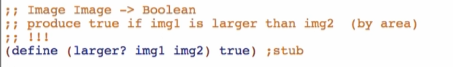
* If img is larger than the first loi
  + put first loi on the front of the list
  + then trust the natural recursion! Use the insert Function: insert img (rest loi)





* otherwise (first loi is larger than img)
  + place img on the front of the list

Make the wish list entry of larger? Function



Check if well-formed!

Overview:

