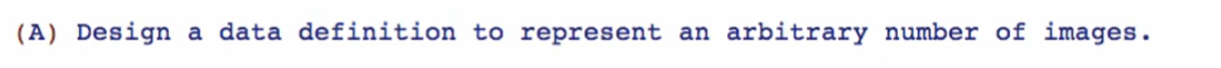
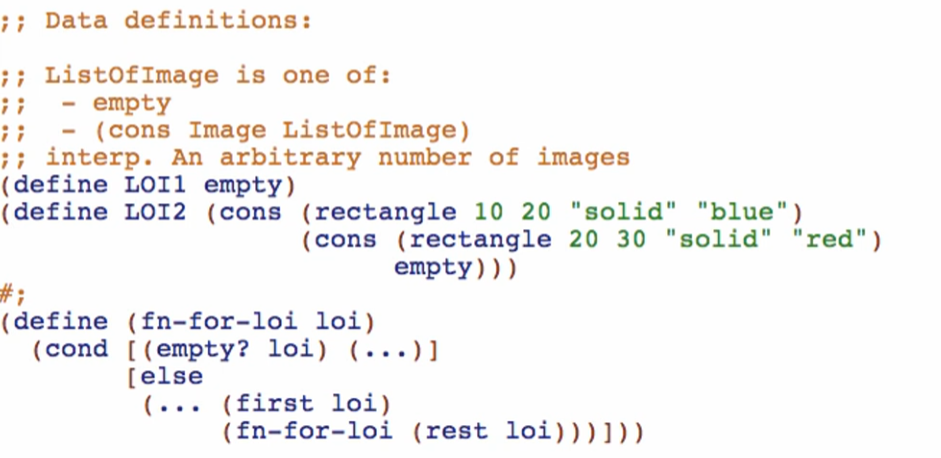


From 1 Function -> 5 Functions



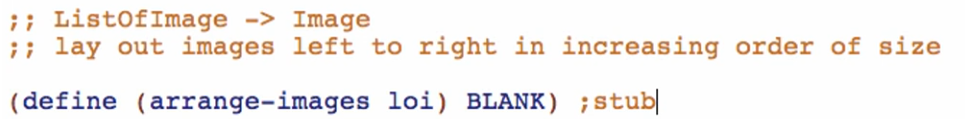
Data definition

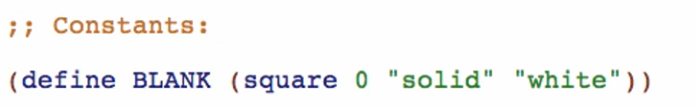


Function definition

1st function: arrange-images

1. Signature, purpose, and stub



Where: 

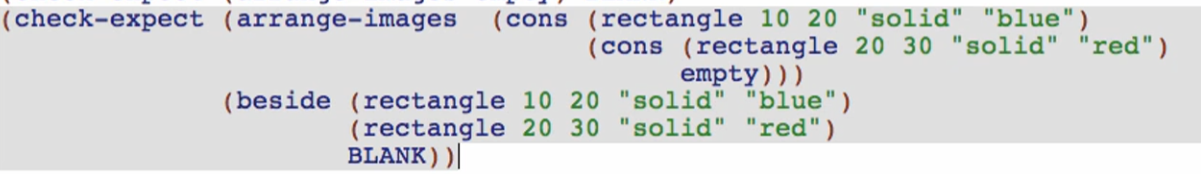
1. Examples

Base case:



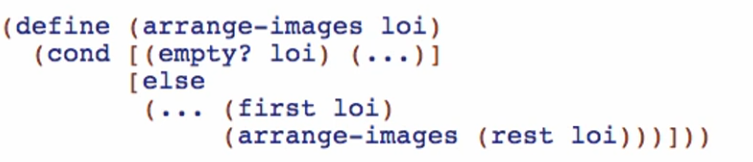
Other cases: (2 elements long)

* Include the full layout of rectangle (not the constants) so you’ll get to see more of the details of your examples

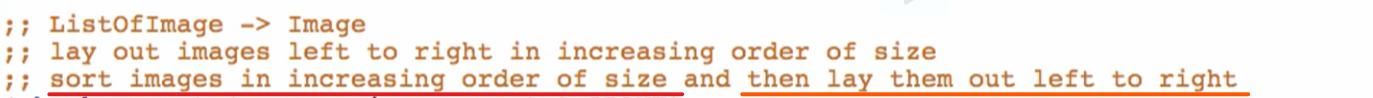


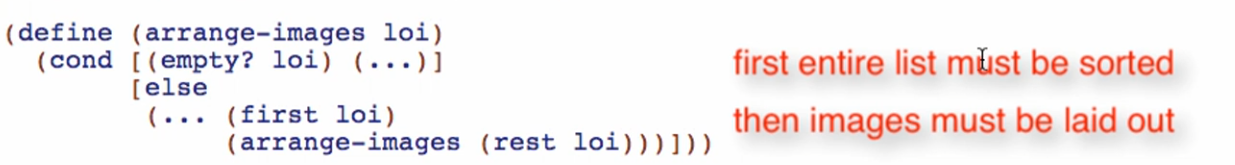
1. Template

Copy, rename function and natural recursions



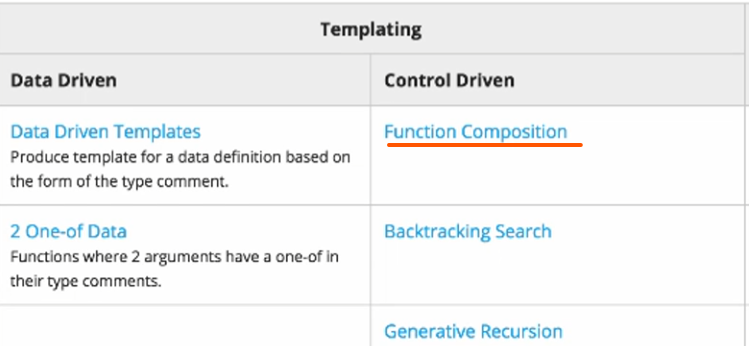
Redefining the purpose:



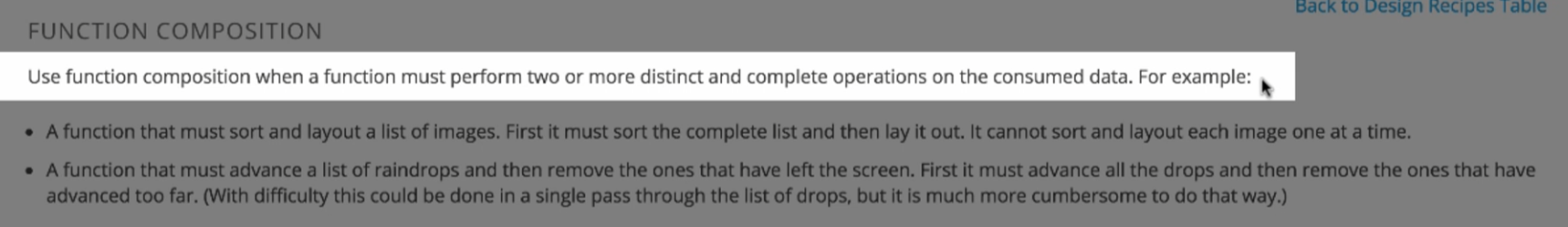


This is called a Function Composition problem!

Go to Design Recipe ->



There is a special entry for function composition



Apply function composition:

Discard entire template:



Replace with function composition:

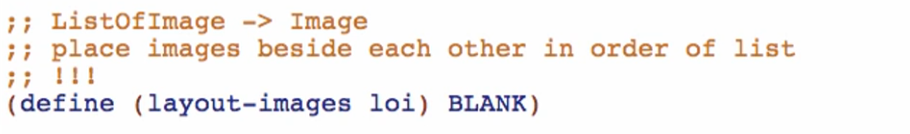


First: sort-images -> sort the entire list of images by size

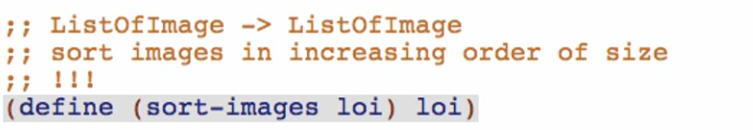
Then: layout-images -> draw the image out based on the sorting

Wish for the functions:

layout-images

**

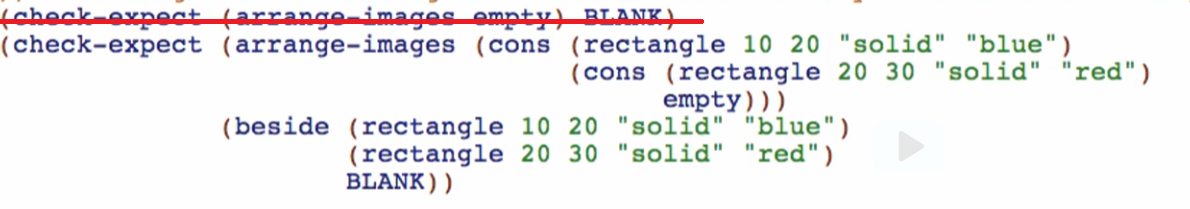
sort-images

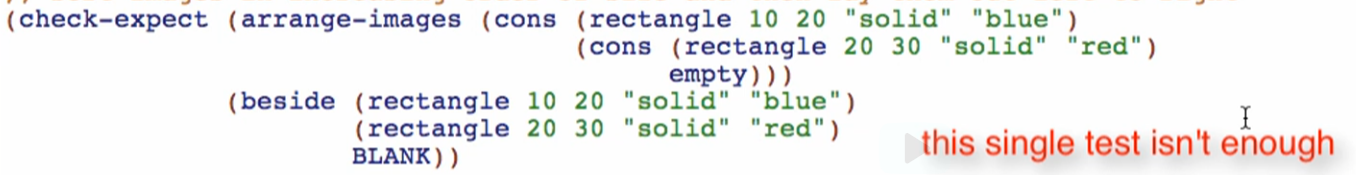


* Making the stub produce its own argument will make the stub a little more useful in at least SOME early tests of arrange-images

Going back to no. 2: Examples

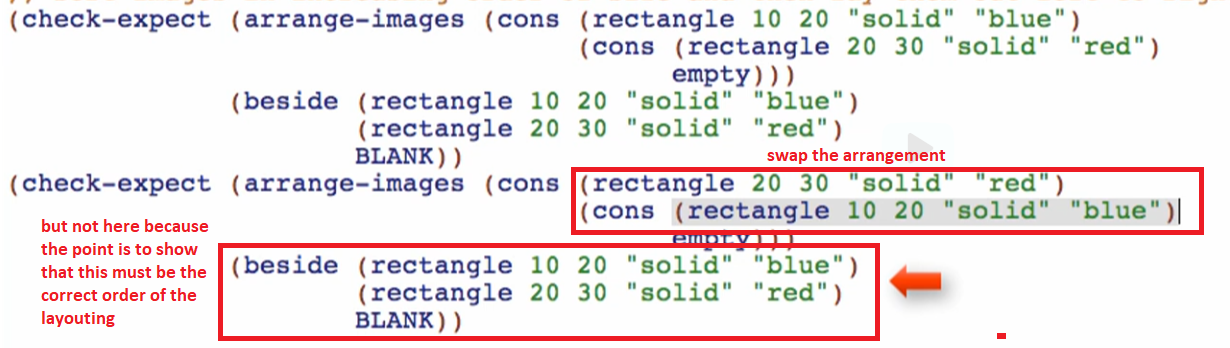
* Tests for arrange-images do not need to show the full laying out and sorting of images. But they do need to show the how the two functions work with each other.
* In function composition, you DON’T NEED to test the BASE CASE





It doesn’t check that the list is both sorted and laid out. This just did the RENDERING/LAYOUTING itself

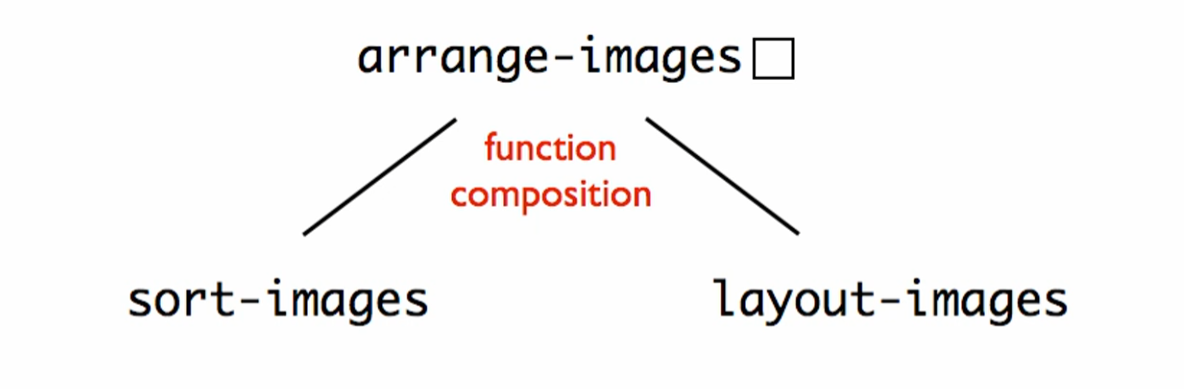
Adding another test



At this point, it checks both the sorting and the laying out

Check if well-formed!

Overview:



Broke down arrange-images to 2 functions with the rule: function composition