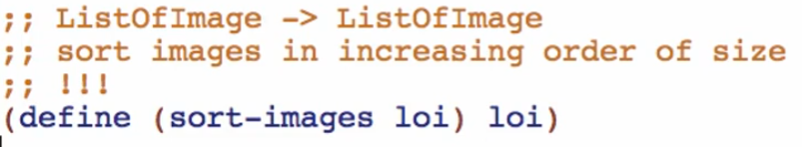
Operating on a List Rule

* When an expression must operate on a list -- and go arbitrarily far into that list -- then it must call a helper function to do that

Working through wish list



**sort-images Function**

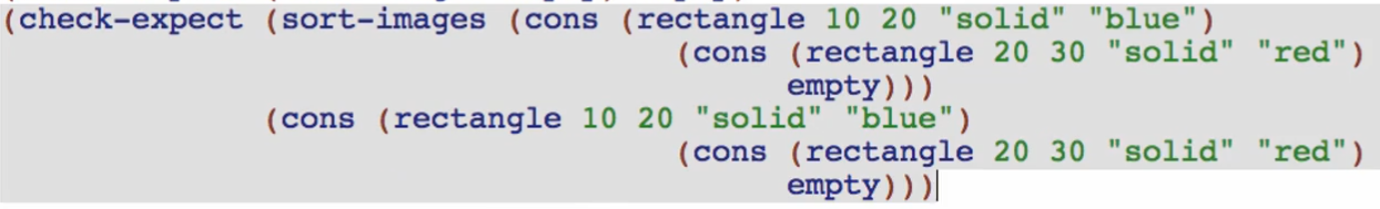
Examples:

Base case:

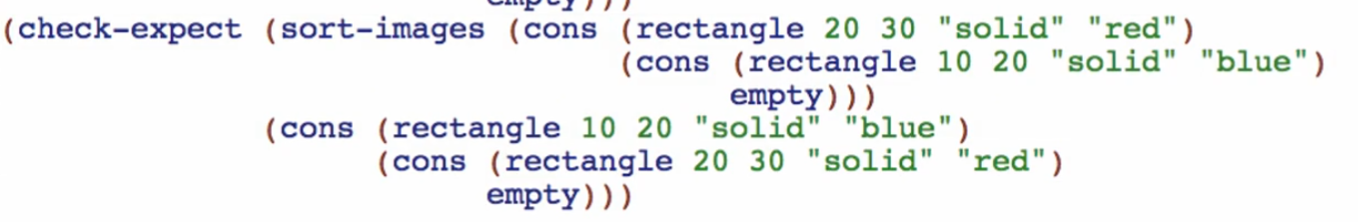


Other cases (at least 2 elements long)

* + Already sorted argument

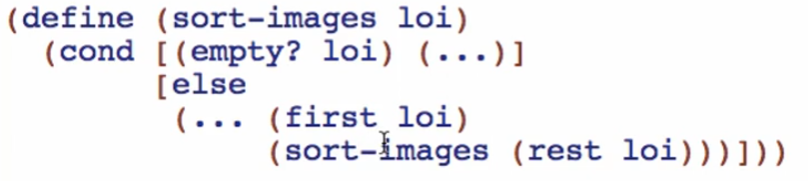


* + Not yet in order argument



Template:

Copy, rename functions and natural recursions



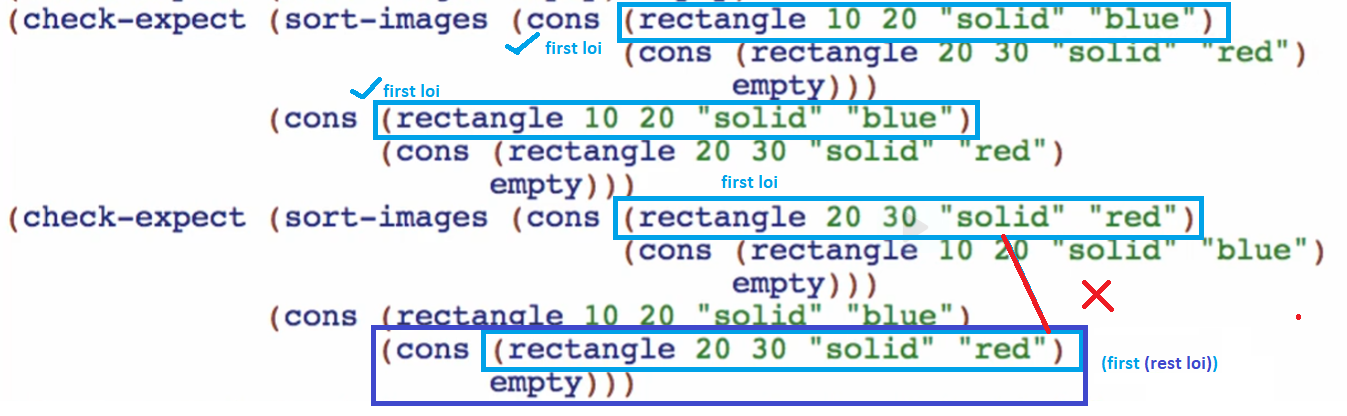
Code Body:

Base case

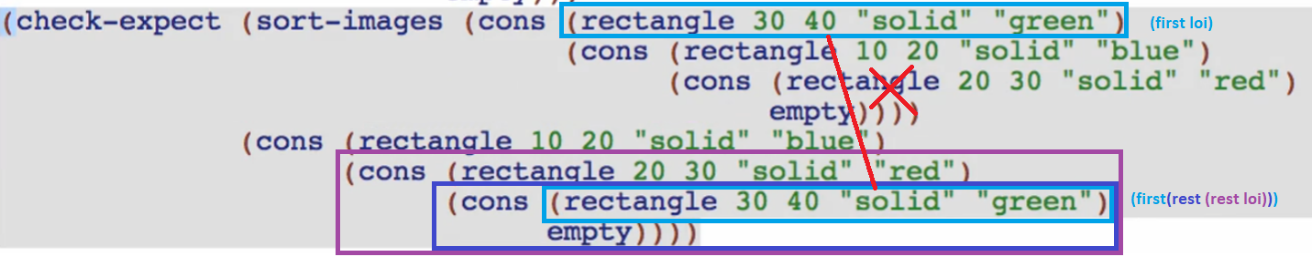


Other cases:

Going back to the examples:

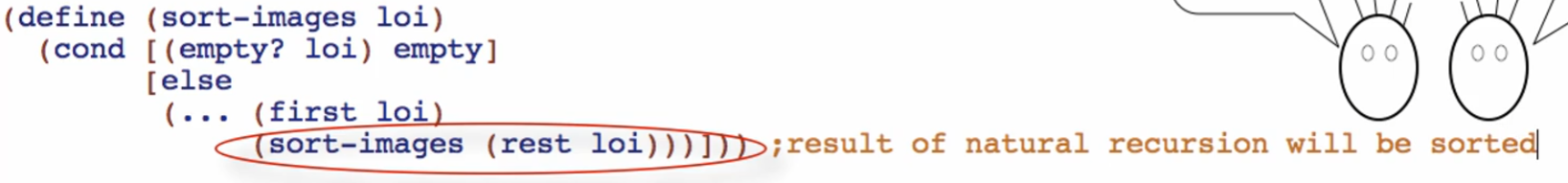


(first loi) isn’t always the first element to be added on the recursive list

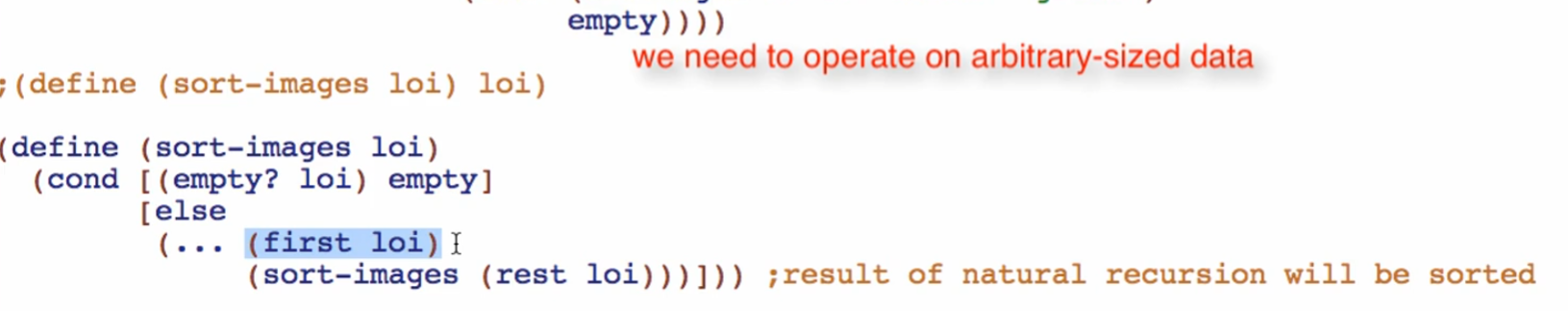


Adding more examples to see other cases where the (first loi) doesn’t come to the first element of the list always

Always trust the natural recursion! Do not do anything here. Trust that the list returned by the natural recursion is always going to be sorted already!

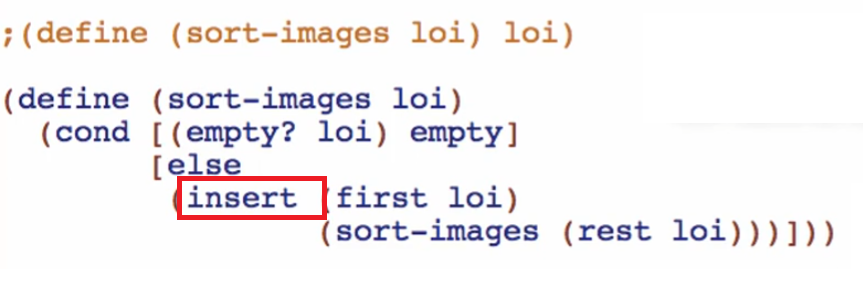


But this part will be needing to operate on an arbitrary-sized data



This is where the operating on list/arbitrary-sized data rule will come

* + We have to use a function to do it, in fact we need to use a RECURSIVE function to do it
  + (first loi) will be TESTED/INSERTED within/inside the (rest loi) thus we are operating inside the arbirary sized (rest loi)

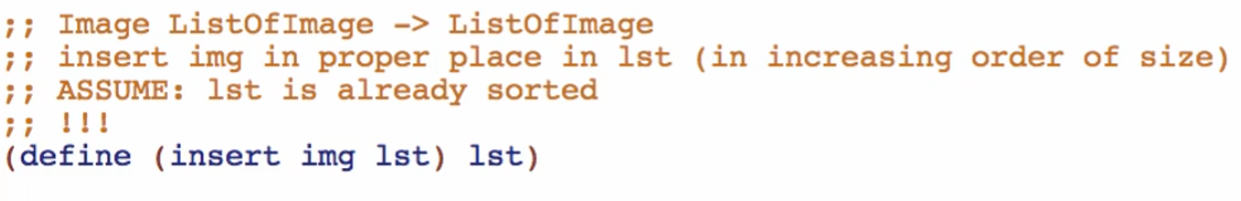




Wish list entry for insert Function

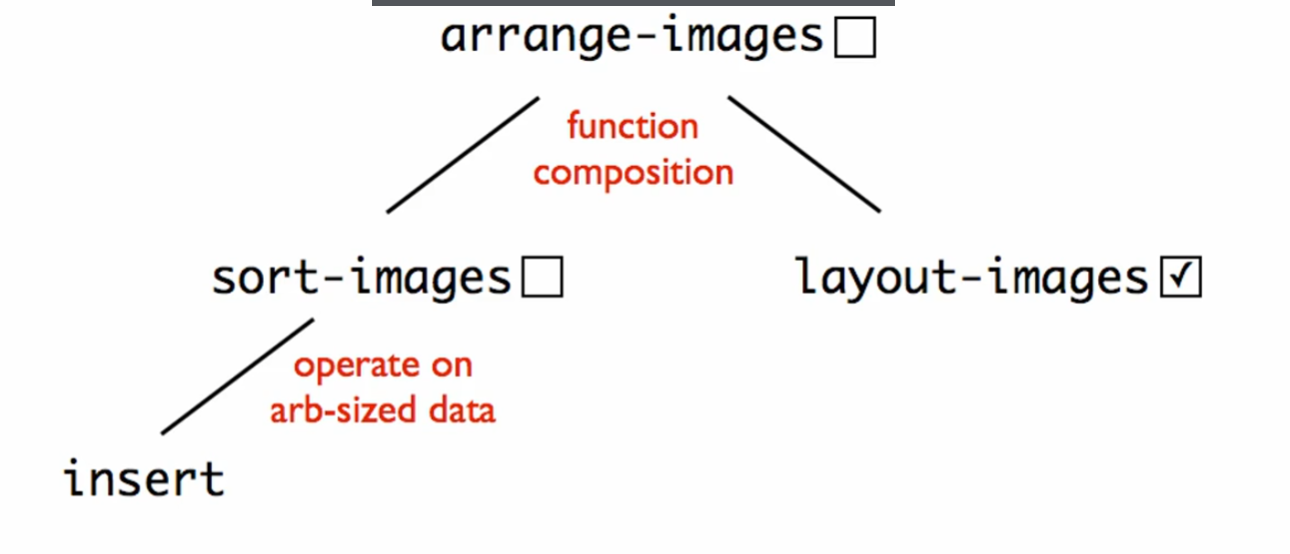






* + This assumption is based on our trusting of the natural recursion!

Overview:



* sort-images require operate on arb-sized data rule, break down to insert Function