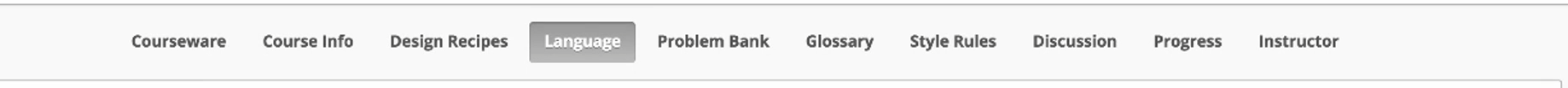
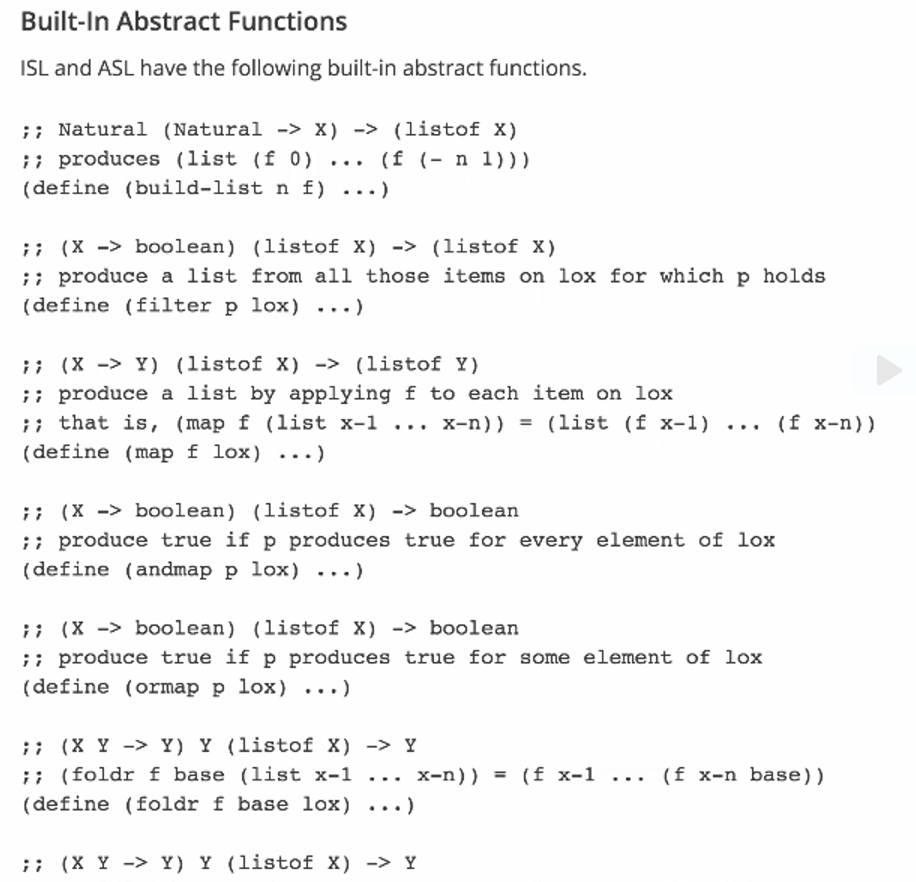
List abstract functions are so useful that they are built into ISL. Using them requires identifying situations when they would be useful.

Where can we find these?

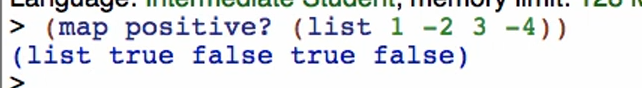


Go to the end of the language page and you can see the list of all the built-in abstract functions

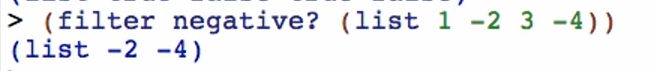


Example uses

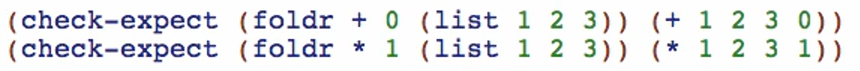
map built-in function

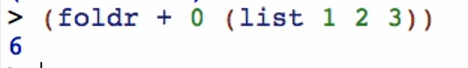


filter built-in function



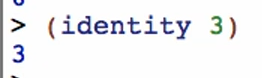
foldr built-in function





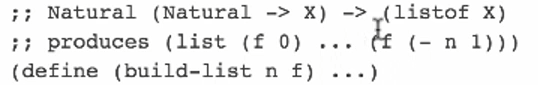
* Abstraction for an operator and a list

(identity x) -> x

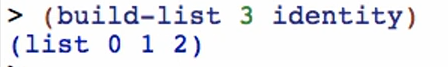


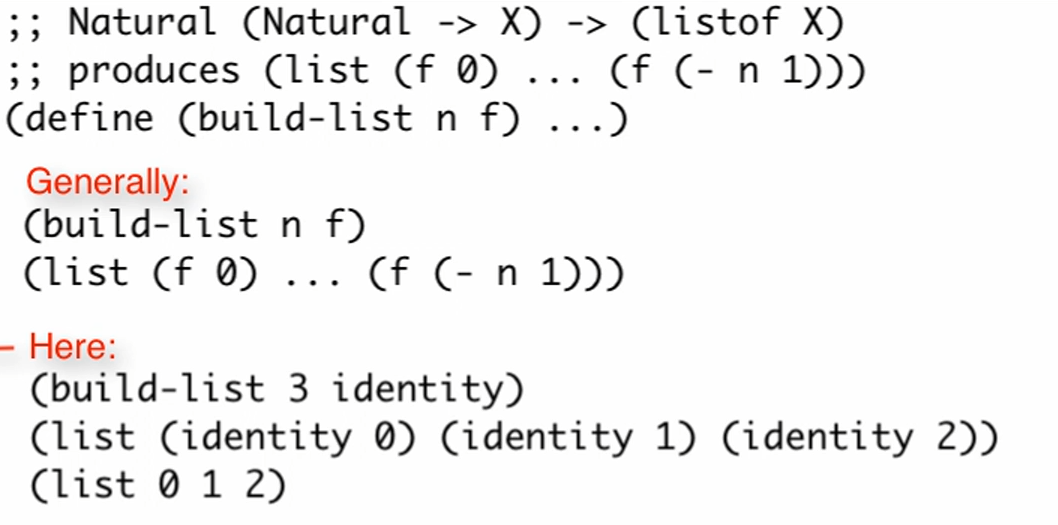
Whatever value you give to that function, it produces the value back

build-list Function

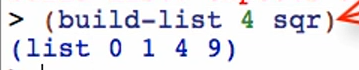


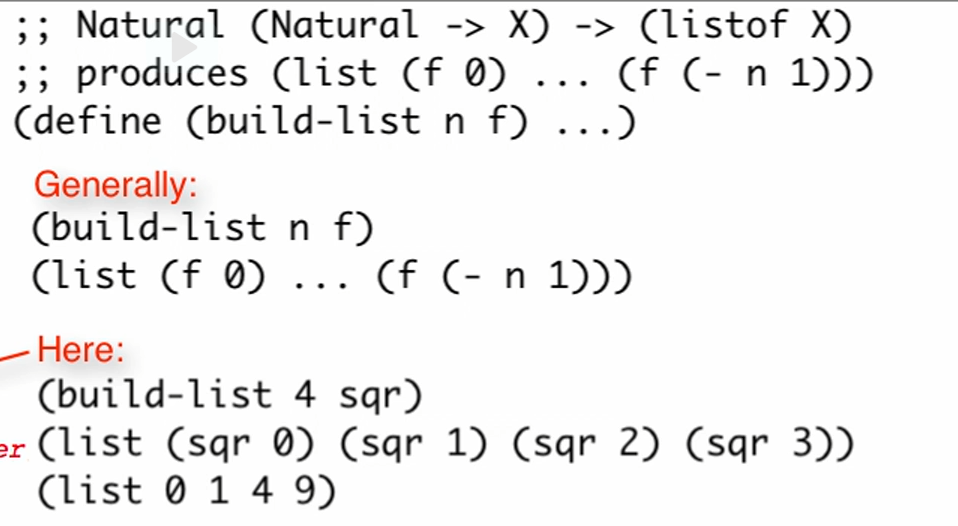
Using identity with build-list

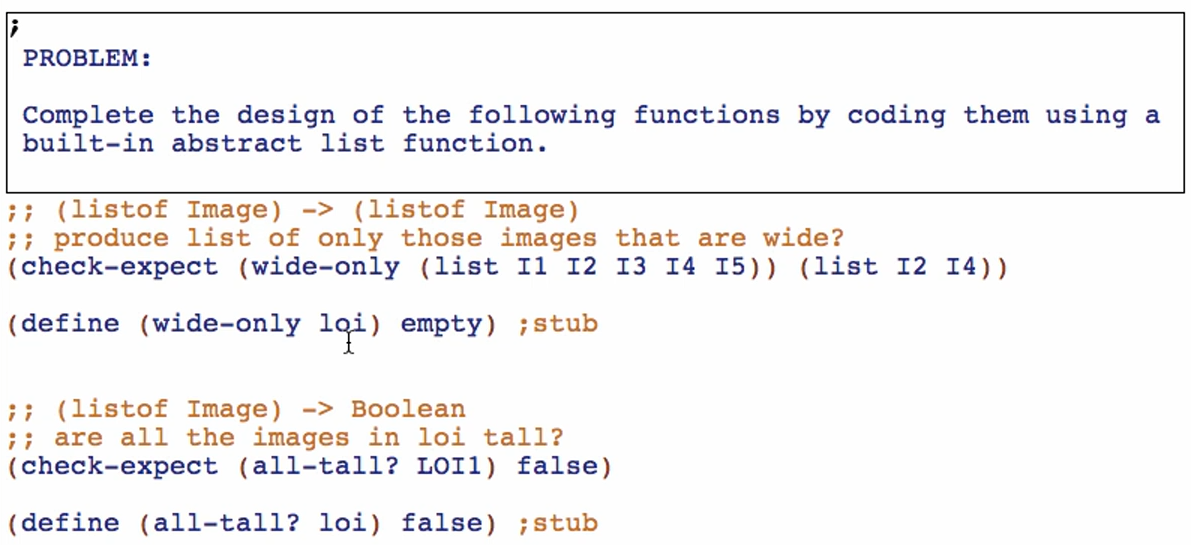




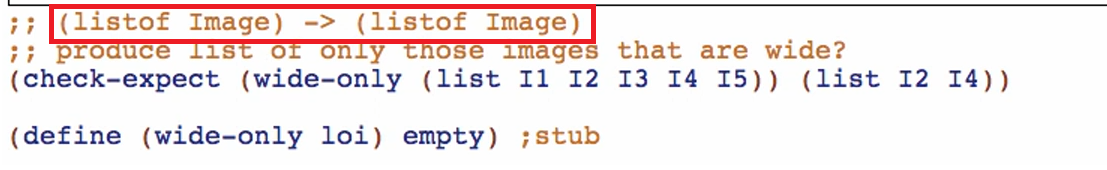
Using other functions with build-list

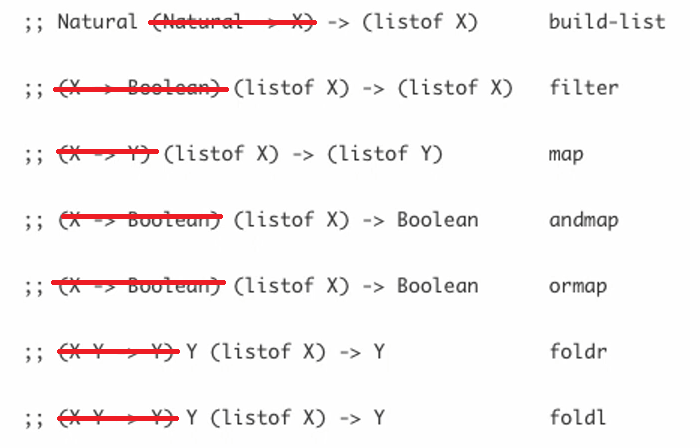


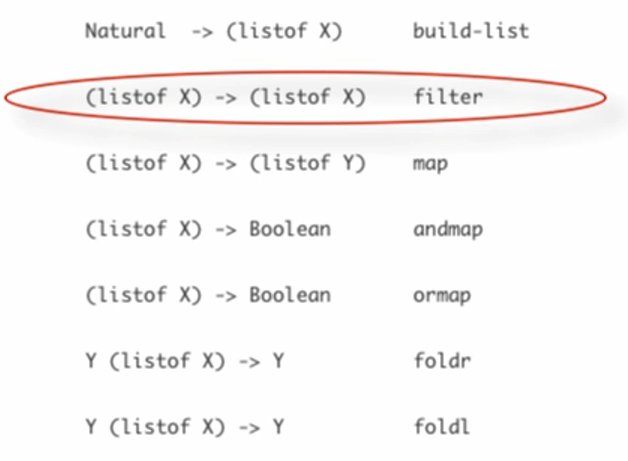




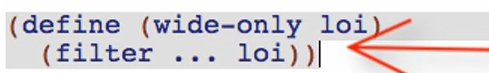
1. Check the signature of the function to be abstracted and match them to the signatures of the built-in abstract functions minus their function arguments/parameters

Eg: 

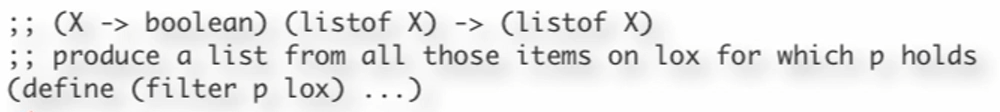


 ->

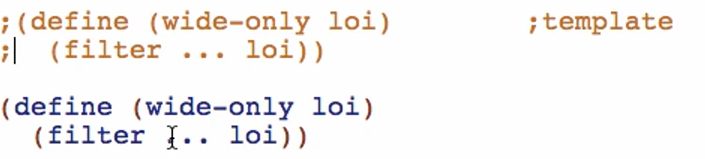
1. Create a template for the filter function:



This the template for filter Function

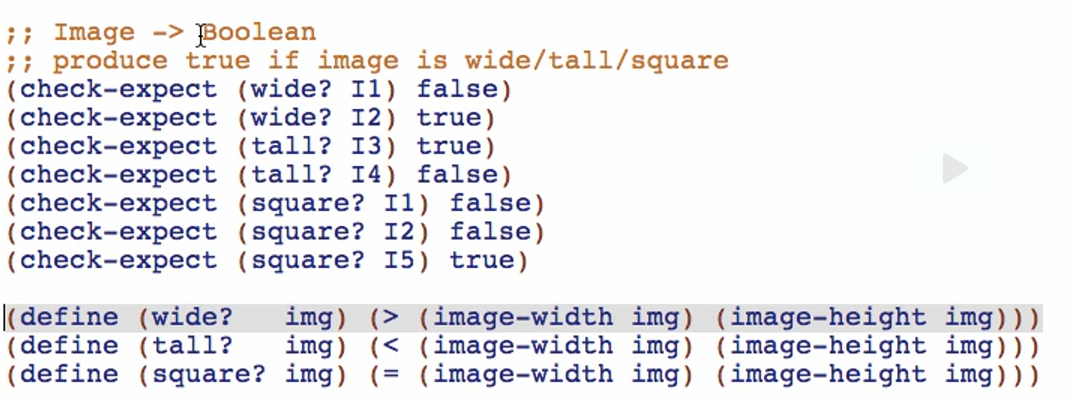


1. Copy and fill in the code body

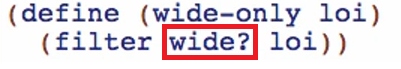




Therefore we need a (X -> Boolean) type of function. Check above or through the code base for a function with that kind of type



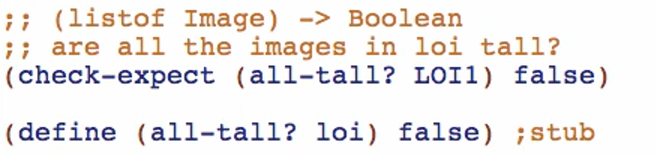
Put your found function in the code body of filter function



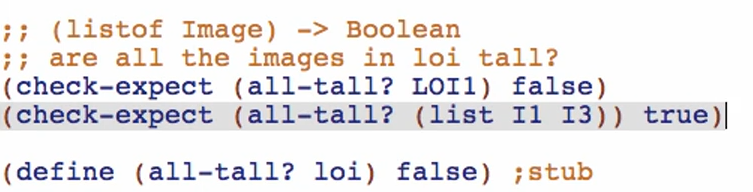
“one-liner” – short and simple functions (not necessary on one line)

Note: base case test ISN’T NEEDED for built-in functions

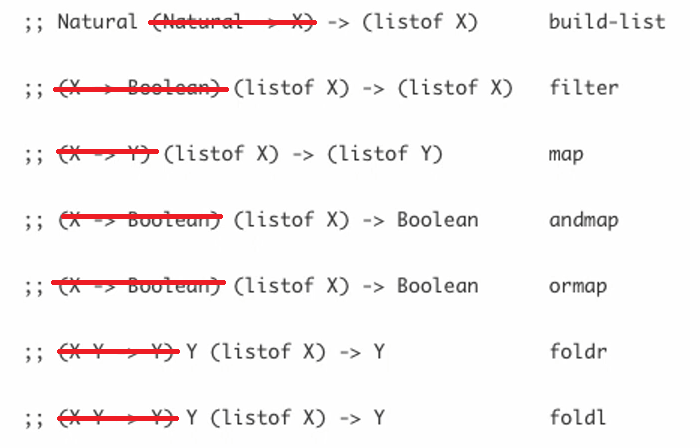
Let’s try another one



Add tests first for 2 long



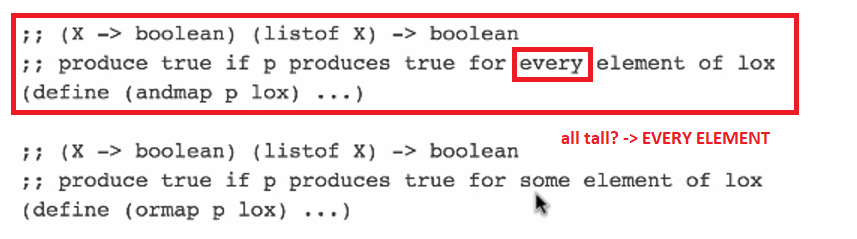
Get the signatures for built-in functions, remove the signatures for function parameters/arguments



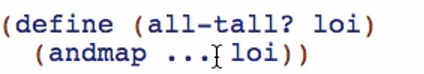
Match the signature of the function all-tall?



Check descriptions in Language:



Using andmap template:



Check for existing functions through the code body or through built-in functions

