Natural numbers

* Arbitrarily long

(add1 <number>) & (sub1 <number>)

* Add/subtract 1 from a number
* They are like cons and rest



Eg: (add1 5) = 6 ---> … 1, 2, 3, 4, 5 ----> … 1,2,3,4,5,6

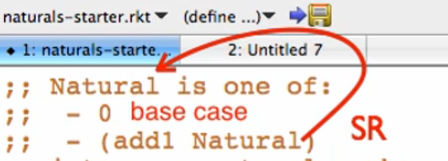
* Like a list 1 element longer
* (cons 6 […1,2,3,4,5])

(sub1 5) = 4 ---> … 1, 2, 3, 4, 5 ----> … 1, 2, 3, 4

* Like a list 1 element shorter
* (cons 5 […1, 2, 3, 4]) = lon
* (rest lon) = […1, 2, 3, 4]

Natural Data Definition

1. Type comment

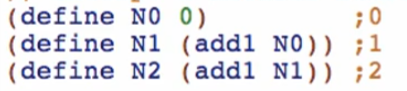


* + Well-formed!

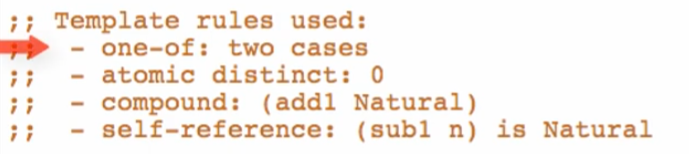
1. Interpretation

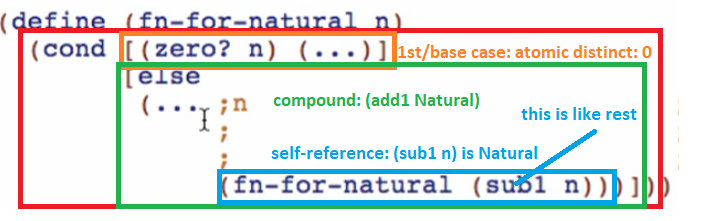


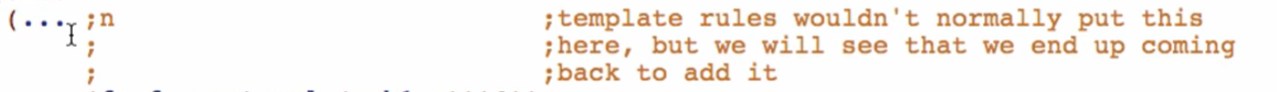
1. Examples



1. Templates

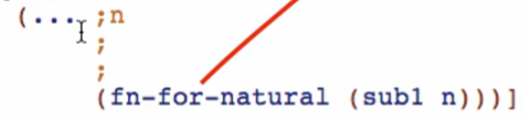






Funny compound list: Naturals don’t have <first> but has <rest>



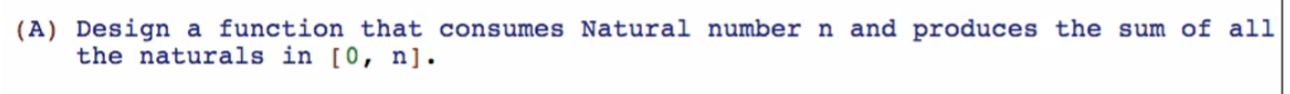


So for this case, first los is just commented like this

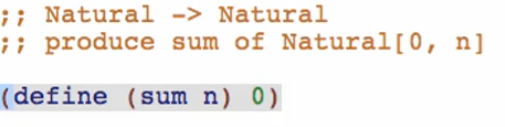


Then rest los takes a self-reference where rest los = sub1 n

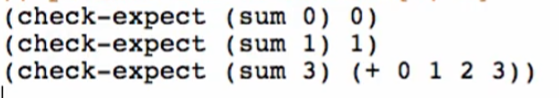
Function Definitions



1. Signature, purpose, and stub

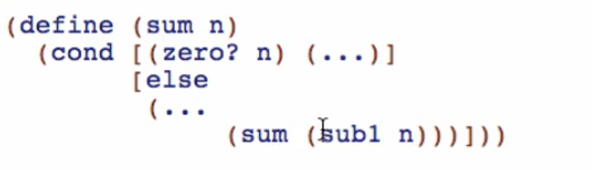


1. Examples



1. Template

Copy template, remove commented out part, and rename the function and all natural recursions



1. Code body

Base case



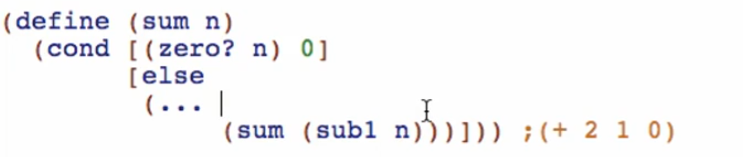
Going back to check-expect



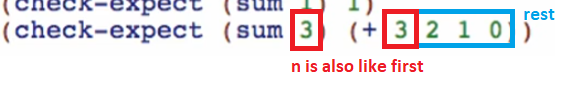
Because the natural recursion will be like this:

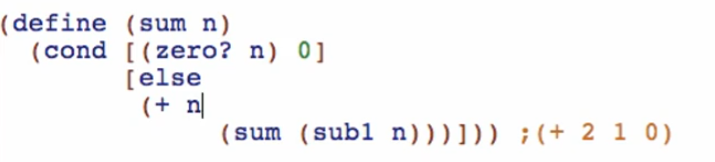


Then for the missing first:

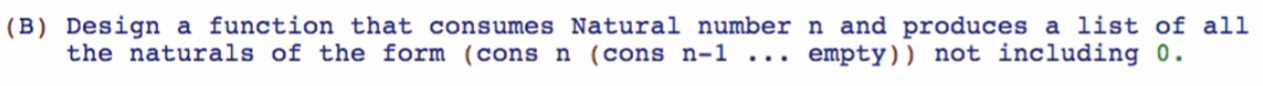


The missing first is just n:



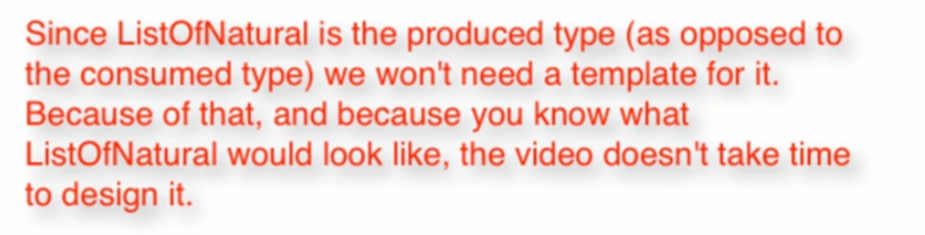


1. Test and debug!

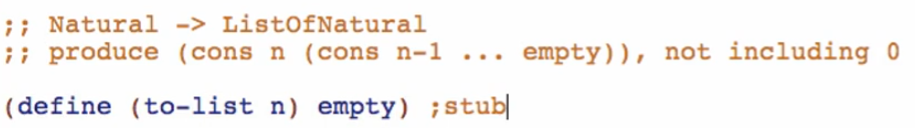


1. Signature, purpose, and stub

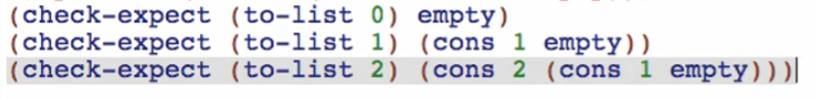




* + Note: produced/return types don’t need/not necessary for data definitions!



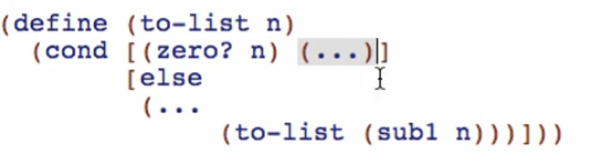
1. Examples



Run and check if well-formed

1. Template

Copy from data definition, take out commented out part, rename functions and all natural recursions



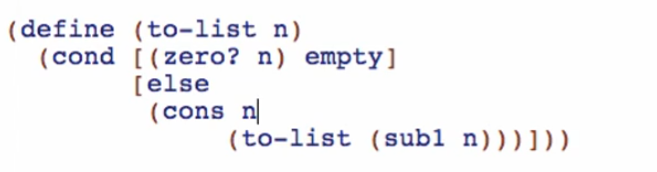
1. Code body

Base case result:



Rest case:





1. Test and debug

We ended up adding “n” as the “first” to both code bodies. We can augment our original template to add n as first specifically for Natural numbers only!

