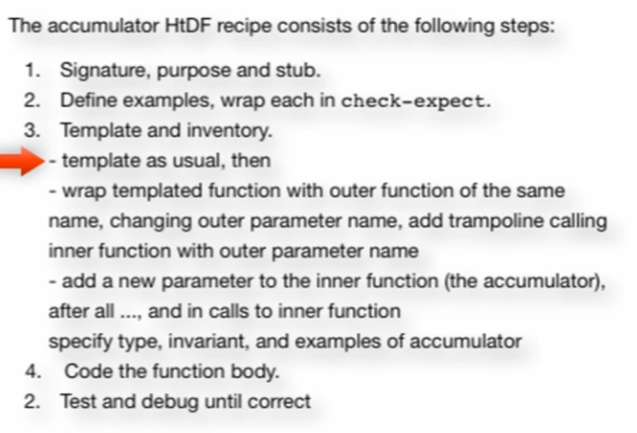
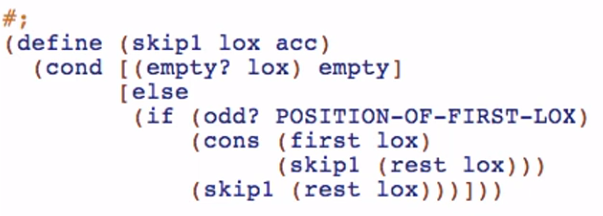
Using a context-preserving accumulator



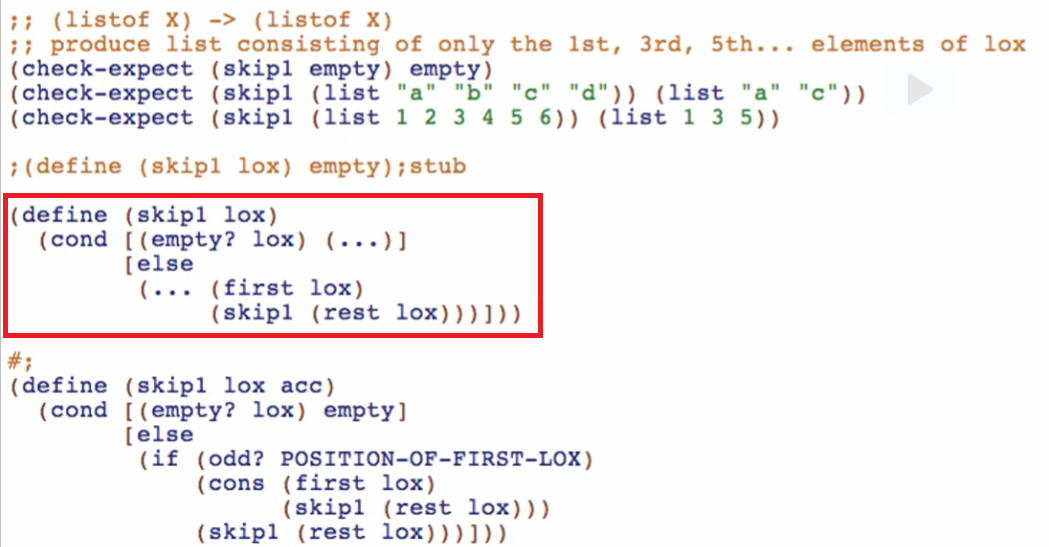
**Backing up in our template:**

Comment out our first template (do not delete)

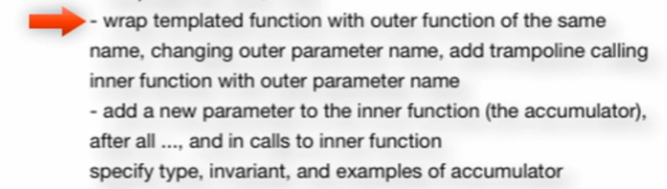


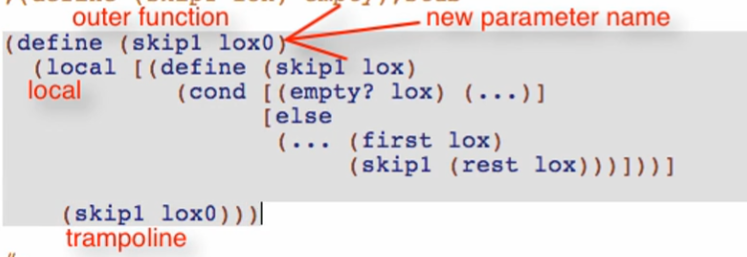
Template using structural recursion

1st step: usual template

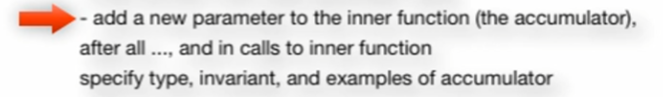


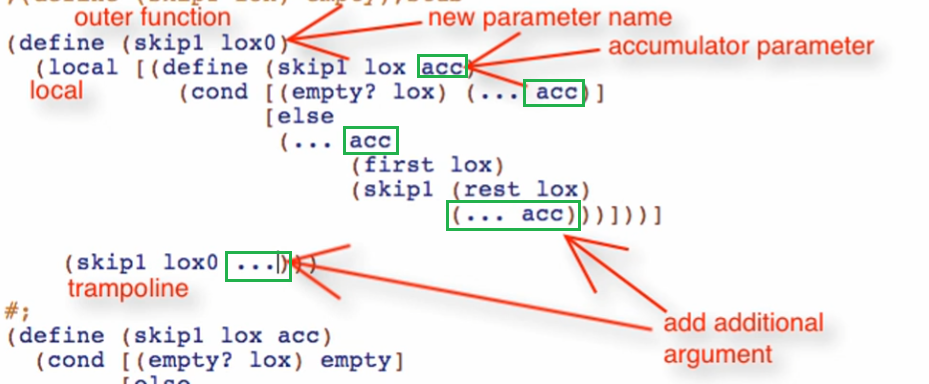
2nd step





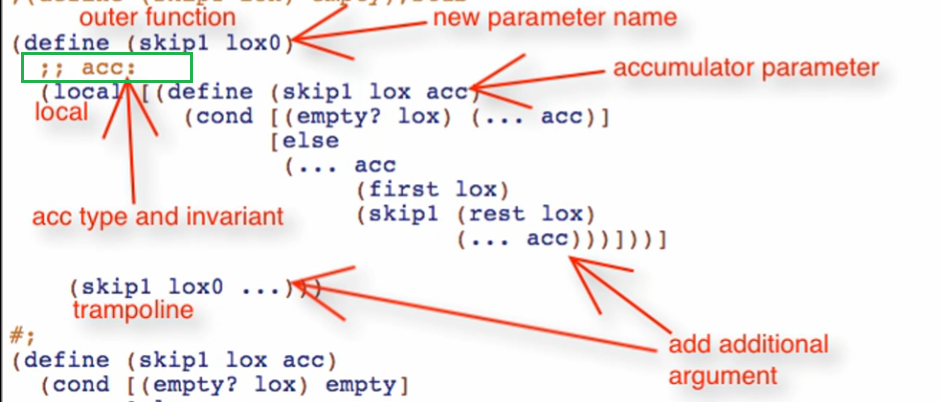
3rd step





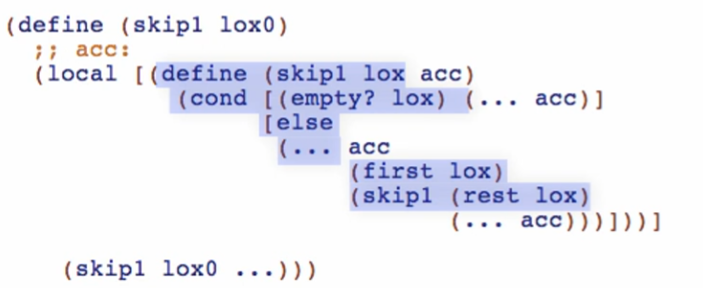
One more thing for the 3rd step



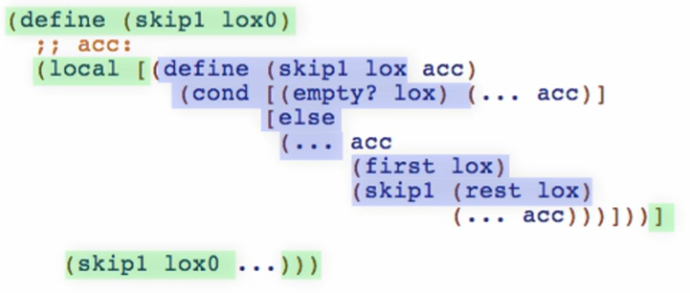


Overview of 3 parts to generating accumulator template:

1. Structural recursion template



1. Wrapping function in outer function, local, and trampoline

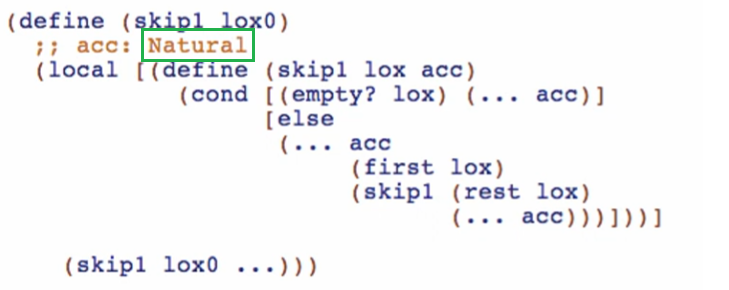


1. Adding additional accumulator parameter



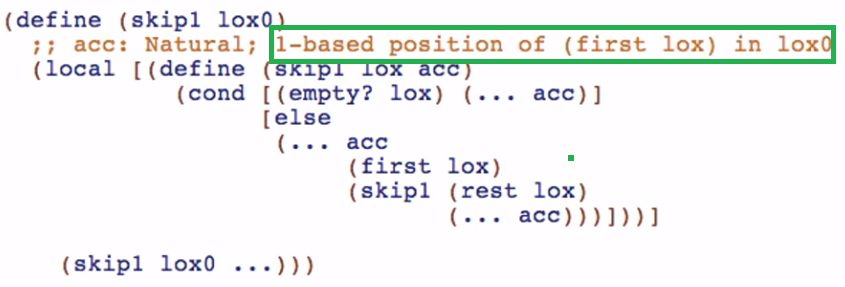


Signature



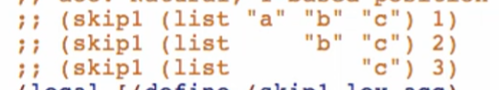
* We want our accumulator to represent the current position of our element in the list

Purpose



* This is where the purpose of lox0 is highlighted. So, we can say that lox is the CURRENT list and lox0 is the ORIGINAL lox

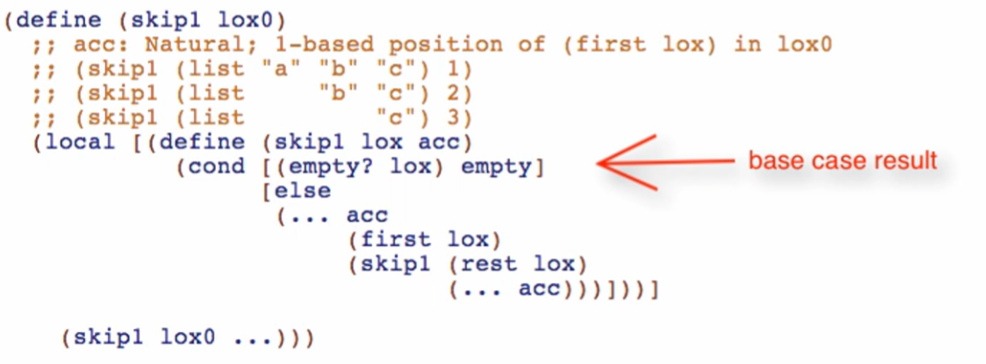
Examples



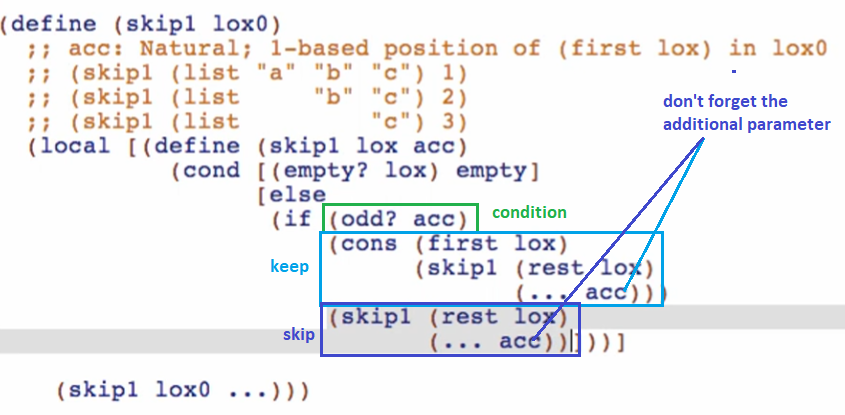
* Note: we used 1-based not 0-based indexing

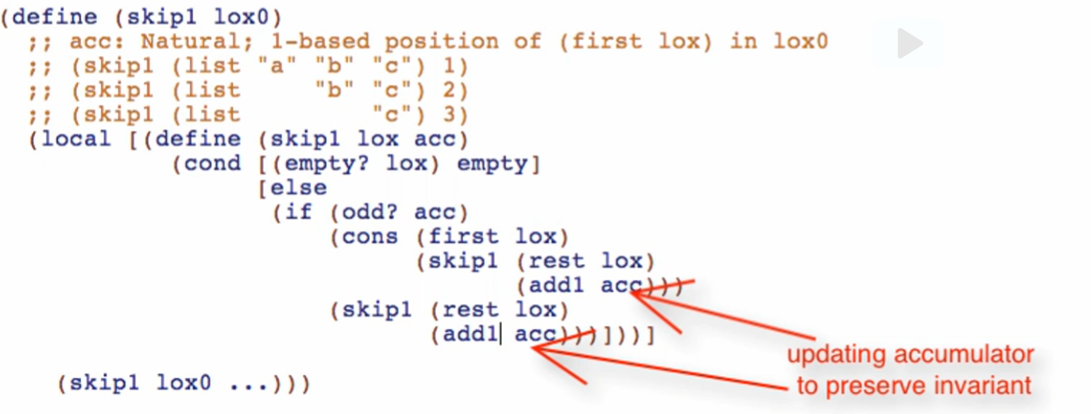
**Working through the code body**

Base case result

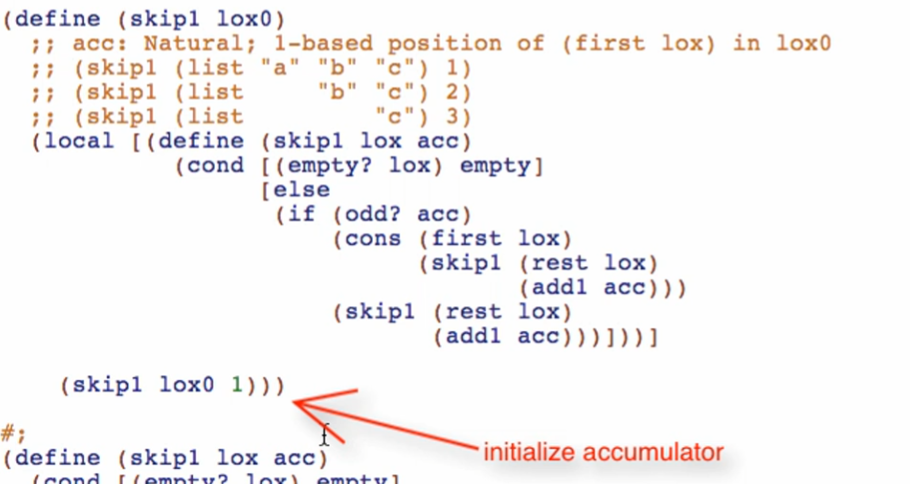


Other cases





* This is just like i++, index++

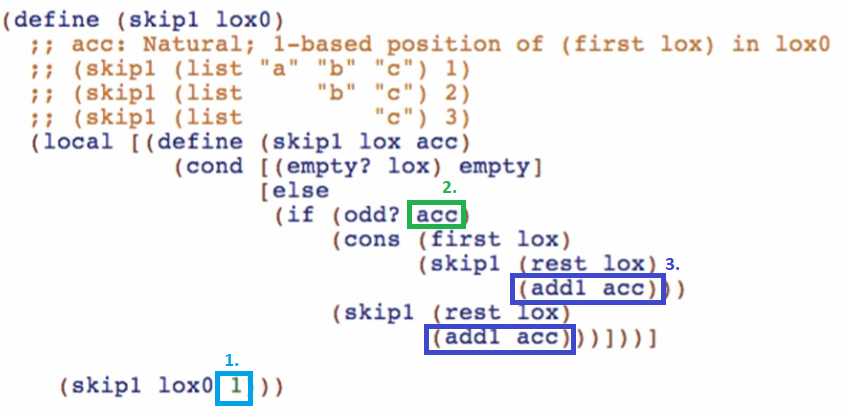


* This is just like int i = 0 or int index = 0 when you are using fori loop

Run and debug!

You always need 3 things for the accumulator to work

1. Initial value
2. Exploit/use the accumulator value
3. Update the accumulator per natural recursion





* Accumulator Invariant
  + Something that’s always true about the accumulator
  + Even if the accumulator’s exact value varies