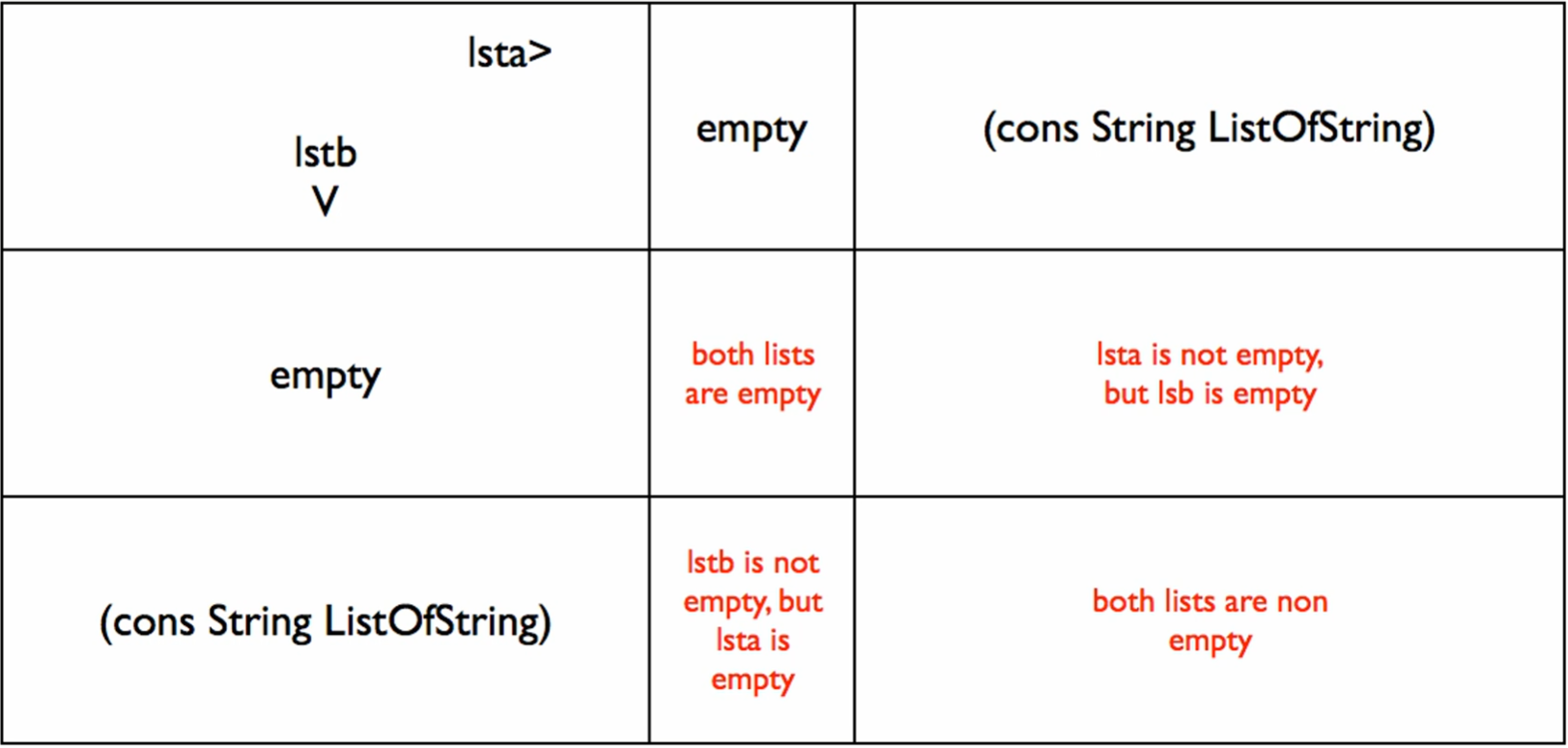
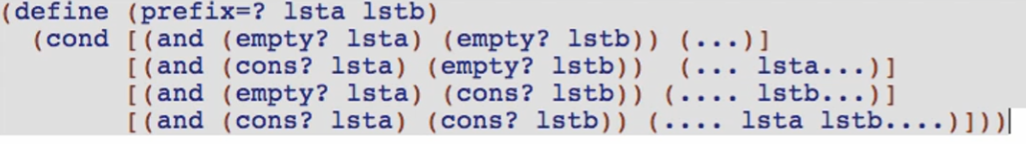
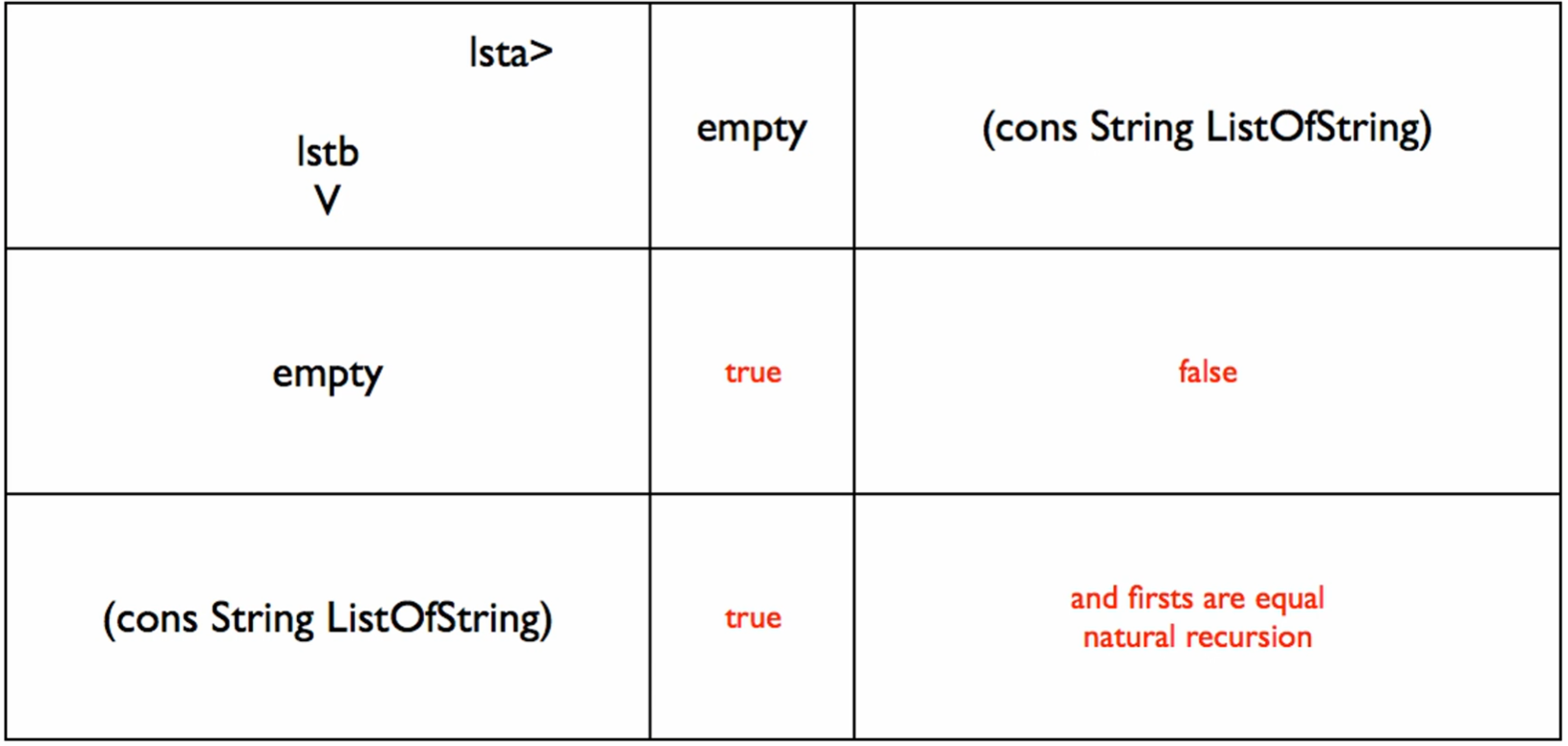
**Templating**



* NOT THE BEST WAY but will work: 4 case cond

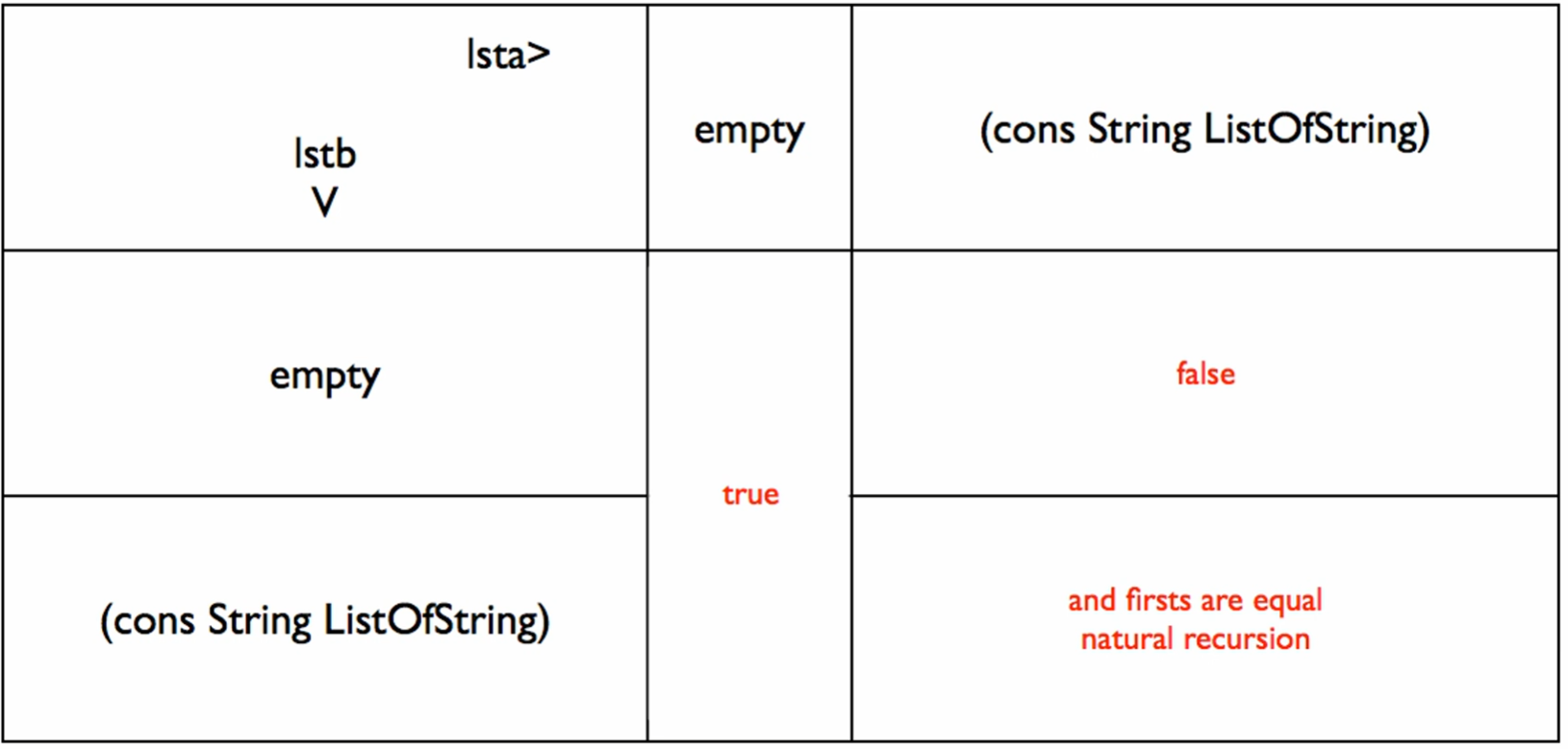


Going back to the cross product table, modify the table by populating it with the ANSWERS for cond. Currently the table is populated with the QUESTIONS for cond (as seen above)



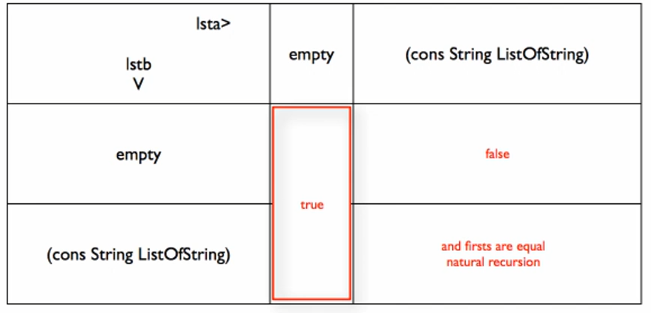
Using this modified cross product table

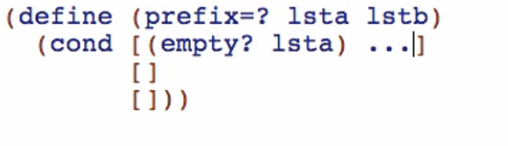
* In this table, there are only 3 cases:



Use a 3 case cond!

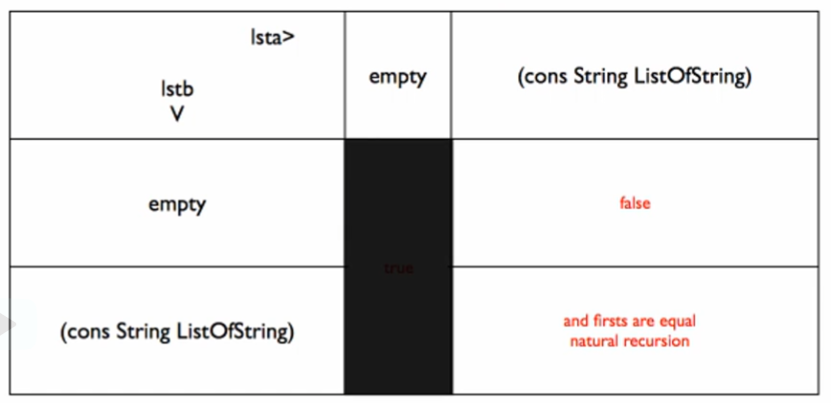
1st case: Start with the biggest box (the one that encompasses most cases)





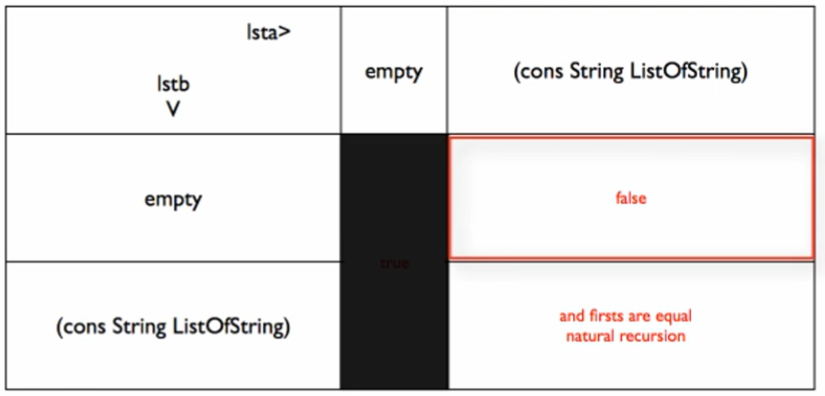
* Since lsta is always empty for this condition

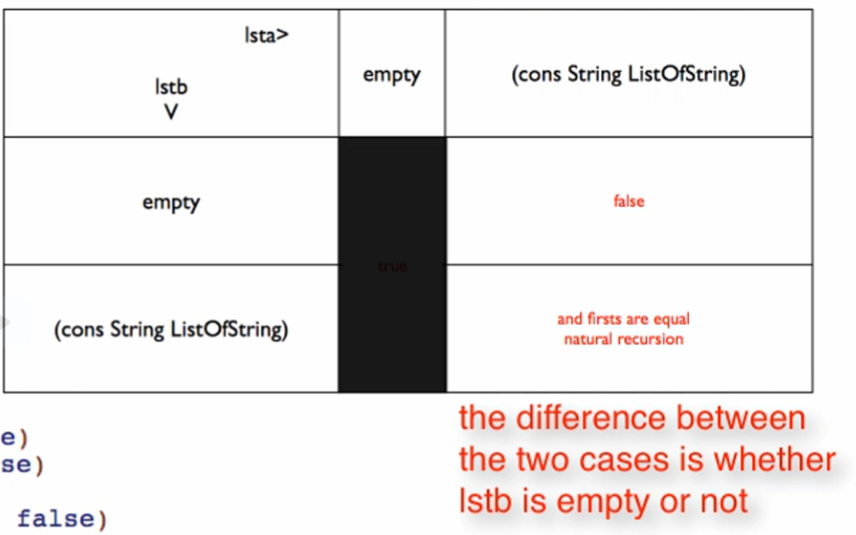
Then cross out:



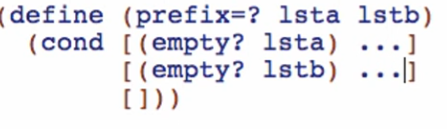
* Means we can now ignore the possibility of lsta being empty because it is already handled!

2nd case: next with the simplest answer

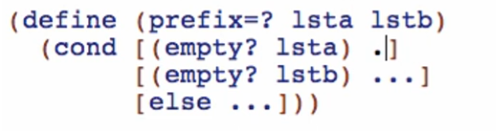




* The only difference for this is whether lstb is empty! We can use this as the question

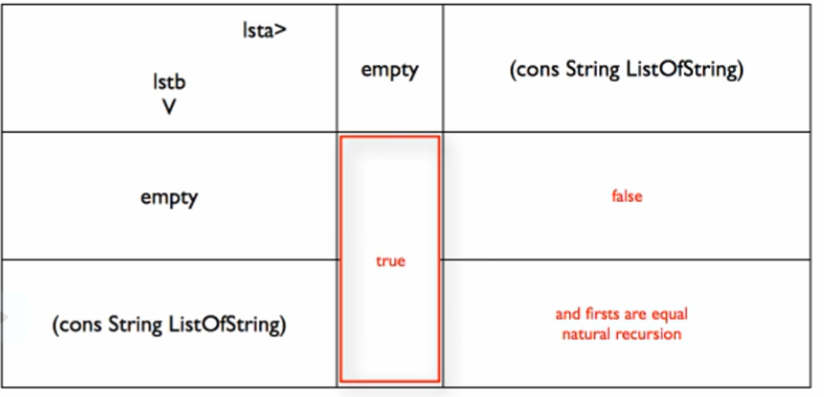


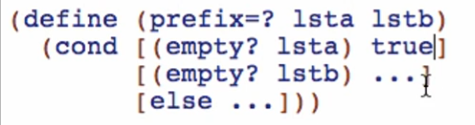
3rd case, last case: we can now use else!



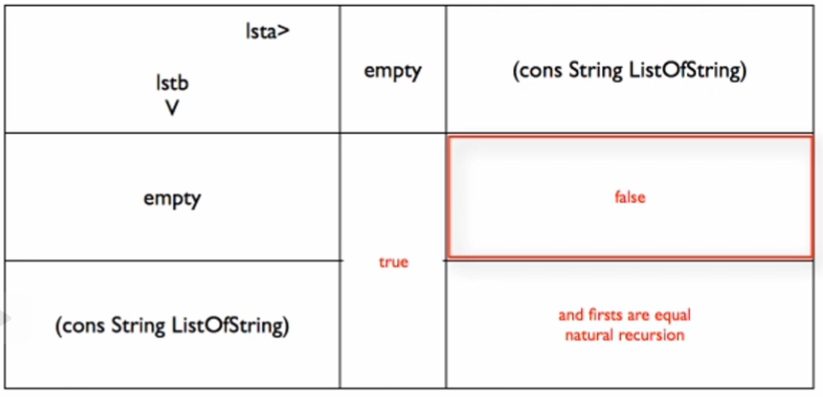
**Code body:**

1st case: if lsta is empty

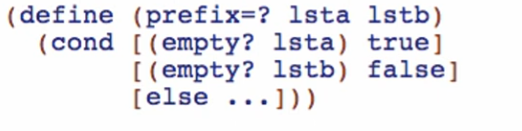
****

****

2nd case, if lstb is empty:

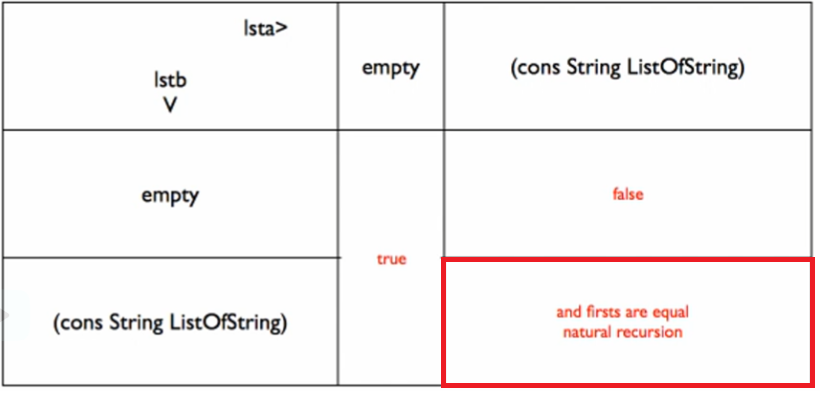


* We already handled lsta being empty so no need for that! We only need false answer!

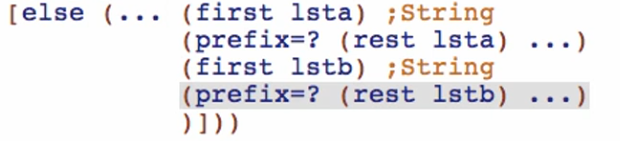


3rd case:

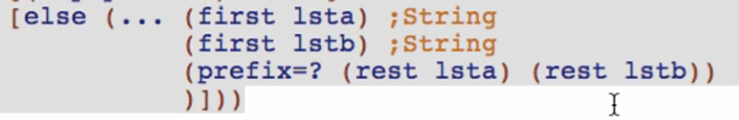
Templating the 2 lists



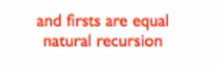
Compound and natural recursion:



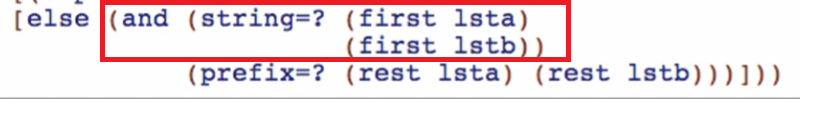
* But prefix=? needs lstb so the final template would be:



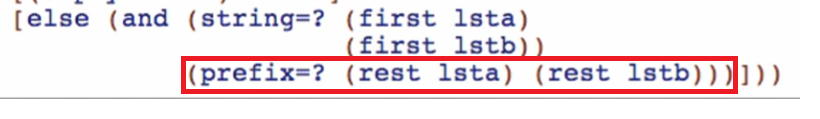
Coding the 3rd case:



Firsts are equal:



Natural recursion:



Run and debug!

Simplifying at a **Model Level**

* Using the cross product of type comments to code the function