

Distribute the following questions across the members of your group. You will share your solutions (and most importantly the *method* of your solutions) during the next lecture period. Divide up the questions so that **each** question has at least two solutions from different group members.

1. Refactor the following grammars by consolidating the common prefixes until they are LL(1).

(a)	(b)	(c)
$ \begin{array}{lcl} S & \rightarrow & a\ b\ c\ d\ e\ \$ \\ & & a\ b\ c\ q\ y\ z\ \$ \\ & & a\ b\ c\ q\ r\ s\ \$ \\ & & T\ U\ \$ \\ T & \rightarrow & x\ a\ b \\ & & \lambda \\ U & \rightarrow & G\ z \\ G & \rightarrow & d \\ & & q \end{array} $	$ \begin{array}{lcl} START & \rightarrow & A\ x\ B\ \$ \\ & & C\ \$ \\ & & A\ q\ C\ r\ \$ \\ A & \rightarrow & x\ y\ B \\ B & \rightarrow & g\ h\ m \\ & & g\ k\ n \\ & & g\ h\ q \\ C & \rightarrow & d\ b\ f \\ & & \lambda \end{array} $	$ \begin{array}{lcl} START & \rightarrow & S\ \$ \\ S & \rightarrow & a\ S\ e \\ & & B \\ B & \rightarrow & b\ a\ e\ B\ e \\ & & a\ e\ C \\ & & b\ a\ e\ B\ g \\ C & \rightarrow & c\ c\ C\ e \\ & & c\ c\ B\ d \end{array} $

2. (“Double coverage” for this question can be one group member doing the coding, and another doing the testing.) Incorporate the solution to question 3 of [lga-ll1-parsing.pdf](#) into your group’s “grammar code”; test with [parser-test.tok.cfg](#) and input [parser-test.tok](#) — see also [show_llparse-parser-test.tok.pdf](#).
3. Refactor these grammars’ left-recursive rules to make the grammar LL(1) (some grammars may require common prefix refactoring as well).

(a)	(b)
$ \begin{array}{lcl} S & \rightarrow & Q\ R\ \$ \\ Q & \rightarrow & Q\ x\ Q\ y \\ & & \lambda \\ R & \rightarrow & R\ r\ s\ t\ x\ y \\ & & R\ r\ s\ t\ y\ y\ y \\ & & s\ t \end{array} $	$ \begin{array}{lcl} S & \rightarrow & SUM\ \$ \\ SUM & \rightarrow & SUM\ plus\ PROD \\ & & PROD \\ PROD & \rightarrow & PROD\ mult\ POWER \\ & & POWER \\ POWER & \rightarrow & val\ exp\ POWER \\ & & val \end{array} $
(c)	
$ \begin{array}{lcl} S & \rightarrow & FUNCTIONS\ \$ \\ FUNCTIONS & \rightarrow & FUNCTIONS\ FUNCTION \\ & & FUNCTION \\ FUNCTION & \rightarrow & C \\ & & P \\ & & H \\ C & \rightarrow & type\ id\ oparen\ CPARAMS\ cparen \\ P & \rightarrow & def\ id\ oparen\ PPARAMS\ cparen \\ H & \rightarrow & id\ dblcln\ type\ HPARAMS \\ CPARAMS & \rightarrow & CPARAMS\ comma\ type\ id \\ & & \lambda \\ PPARAMS & \rightarrow & PPARAMS\ comma\ id \\ & & \lambda \\ HPARAMS & \rightarrow & HPARAMS\ rarrow\ type \\ & & type \end{array} $	

Write a CFG for a language with two terminals ($\{a, b\}$) that represents all **non-empty** strings that are palindromes. Is your language LL(1)? If not can you refactor it using common prefix or left recursion refactoring so that it is?