**All students** should read §5.4–5.5 of the textbook in preparation for the next lecture.

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.

> **Fair warning**: the grammars for questions don't have the end of source marker \$ on the starting goal rules. This is not wrong per-se, since a grammar is defined by four entities: the list of production rules ($P$), the non-terminals and terminals of the language ($N$ and $\Sigma$), and the starting goal, typically referred to as $S$. When you're simply presented with a list of production rules without the end of source marker, you must infer from the grammar structure **which non-terminal is the starting goal**. Typically this is not difficult, and you are expected to do so for these questions.
> Question 3 presents a particular conundrum, in that *Expr* can't have \$ on the end of its production rules, since it is self recursive. In this case, I simply make a place holder starting goal: $S \rightarrow Expr\,\$$.

1. Recall that **regular expressions** can match any finite number of "matching brackets" so long as their opening and closing patterns are distinct and do not overlap (see show_chl-nested-re-structures.pdf and show_chl-pumping-lemma.pdf ).

   (a) Write a spectactularly simple context free grammar for a language of nothing but opening and closing brackets (symbols $\langle$, $\rangle$) that must be matched recursively nested so that there are the same number of $\langle$s as $\rangle$s.

   (b) Now calculate the **predict sets** for your grammar and show that it is an "LL(1)" language. What does this tell us about the **ambiguity** of this language?

   (c) **(For bragging rights)** can you alter your grammar to allow any number of sequences of recursively nested matching brackets. For instance, your grammar should permit the sentence

   $$\langle\,\langle\,\rangle\,\rangle\,\langle\,\rangle\,\langle\,\langle\,\rangle\,\langle\,\langle\,\rangle\,\rangle\,\rangle$$

   Actually, the challenge here is to make your new language permitting sequences of matching brackets **still LL(1)**.

2. Page 173, question 1
3. Questions 3–4 on page 174
4. Page 175, question 9; by "construct an LL(2) parser", the author means an LL(2) parsing table — so in this case columns are labeled with sets of two sequential terminals.
5. Review the LGA work for lga-predict-sets.pdf , when we gather next in lecture I'll ask to see your code results for some non-trivial but smallish grammars.