Reading: Tuesday’s lecture will finish up induction proofs involving CFGs and cover Chomsky normal form. Chomsky normal form helps in visualizing the proof of the “CFL Pumping Lemma,” so please skip section 2.2 and go on to 2.3. Then since 2.4 is being skipped, please move into Chapter 3. Unlike Chapter 2 but like sections 1.1–1.3, it is good to read all of Chapter 3 “in one gulp” the first time, then focus on certain parts when the lectures hit them.

(1) Consider the following part of a grammar for expressions in Java.

E → E2 ASSGTOP E | E2 //assignment is right-associative!
E2 → E2 BINOP E3 | E3 //BINOPs are left-associative
E3 → +E3 | -E3 | ++P | --P | E4
E4 → P++ | P-- | P
P → (E) | LITERAL | VARIABLE (etc.)
ASSGTOP → = | += | -= (etc.)
BINOP → == | != | + | - | * | / (etc.)
LITERAL → /any number or quoted string etc./
VARIABLE→ /any legal identifier/.

Important for this problem is the supposition that binary + and - has whitespace separating it from any nearby + or - signs in your program text—this is necessary for proper lexing of your program. Unary +, - and the pre- and post-increment operators do not need such whitespace, but they still need separation from each other; e.g. -(a) separate from --a. [For a footnote, programming languages resolve the potential ambiguity of whether e.g. "--a" is parsed as -(a) or as pre-decrement, by using the rule that the longest legal token is selected. Since -- is a longer token than two separate negation signs, "--a" is lexed as pre-decrement. This is not considered an ambiguity in the grammar. In fact, you can do this problem treating ++ and -- as single separate symbols, and then the grammar is formally unambiguous!]

(a) Show how to derive a legal Java expression that has the substring "++ + ++" in it, noting the whitespace around the binary +. (6 pts.)

(b) Prove by “structural induction” that the substring "+ ++ +" can never occur in a legal Java expression, nor any similar one with - in place of + and/or -- in place of ++. (15 pts.)

(2) Consider the grammar G of Problem 2.19 on page 156 (2nd ed., p130; in the 1st ed., it’s problem 2.25):

\[ S \rightarrow aSb \mid bY \mid Ya, \quad Y \rightarrow bY \mid aY \mid \epsilon. \]

Suppose we answer the text’s first question by saying, “G generates the language E of all nonempty strings over the alphabet \{ a, b \}.” This answer has a tiny bug. Fix the bug by adding a rule with no variables on the right-hand side, and prove that the resulting grammar G’ is correct. You may assert without further proof that Y derives all strings, and also note
that $L(G') \subseteq E$ is immediate since $S$ does not derive $\epsilon$. Hence you only need to handle the $S$ cases in proving $E \subseteq L(G')$ (12 pts.)

(3) Let $E = \{ x \in \{a, b\}^* : \#b(x) - \#a(x) \text{ is a multiple of } 3 \}$, where as usual $\#a(x)$ means the number of $a$'s in the string $x$. Let $G$ be the context-free grammar

$$
S \rightarrow aA \mid SbB \mid AB \mid \epsilon \\
A \rightarrow aa \mid bS \mid aB \\
B \rightarrow SaS \mid AA.
$$

(a) Prove that $L(G) = E$. (9+18 = 27 pts. Reasonable proof shortcuts are OK.)

(b) Find (at least) two productions that you can delete from this grammar without changing the language it generates. Justify your answer by explaining why those rules aren’t needed in your proof of $E \subseteq L(G)$ in part (a). (6 pts., for 33 on the problem and 66 on the set)