Homework 6: Context Free Grammars, Parsing

Required Problems

1. (a) Consider the following grammar:

$$E \rightarrow E+T \mid E-T \mid T \quad (1)$$
$$T \rightarrow T*F \mid T/F \mid F \quad (2)$$
$$F \rightarrow (E) \mid id \quad (3)$$

Why is this grammar not LL? Give a rightmost derivation and parse tree for the following expression:
$7-8*(3+2)/4+11$
Note that all the numbers are ids, as in class, and I'm asking for BOTH the derivation and the parse tree (which are not always identical).

(b) Now consider an equivalent LL grammar:

$$E \rightarrow TE' \quad (4)$$
$$E' \rightarrow +TE' \mid -TE' \mid \epsilon \quad (5)$$
$$T \rightarrow FT' \quad (6)$$
$$T' \rightarrow *FT' \mid /FT' \mid \epsilon \quad (7)$$
$$F \rightarrow (E) \mid id \quad (8)$$

Give a leftmost derivation and parse tree for the same expression:
$7-8*(3+2)/4+11$
Again, note that all the numbers are ids, as in class, and I'm asking for BOTH the derivation and the parse tree (which are not always identical).

2. (a) Write a context free grammar to accept the set of all regular expressions. (Hint: you need to make rules for each of our regex operations- alternation, concatenation, the kleene star, and you need to accept parentheses to provide grouping.)

For example, the regular expression a^*b^* is is the set of all strings starting with zero or more a's and ending with zero or more b's. The string "a^*b^*" should be accepted by your grammar, not the set of strings described by the regex. Similarly, the regular expression $(a|b)^*$ is the set of all strings containing any number of a's and b's. Again, your job is to accept the regex string "$(a|b)^*$".

(b) Is your grammar LL?

3. Consider the following LL grammar:

$$S \rightarrow aB \mid bA \mid \epsilon \tag{9}$$
$$A \rightarrow bAA \mid aS \tag{10}$$
$$B \rightarrow aBB \mid bS \tag{11}$$

   (a) Compute the FIRST and FOLLOW sets for each nonterminal.

   (b) Using the FIRST and FOLLOW sets, generate the predictive parsing table.

   (c) Show the parsing action (including matches, stack, input, and action columns) for the string *baab*\$. Note that if your parsing does not work (which it should for this one, unless I've made a mistake), you should simply show the parsing action up to the point where it gets stuck.