Problems [100 points] (same end-of-semester credit as HW 1)

(Submit your homework as in homeworks 1 and 2)

**LL(1) Grammars**

1. [20 points total] The grammar below is not LL(1). Modify the grammar to be LL(1) (and generate the same language). The problems to look out for are [10 points] Not being able to predict which of two rules to apply and [10 points] A recursive definition that doesn't check for a base case before the recursive case(s).

   1. \( S \rightarrow a \ S \)
   2. \( S \rightarrow a \ b \)
   3. \( B \rightarrow B \ b \)
   4. \( B \rightarrow c \)

**LL(1) Table Parsing, Derivations**

2. [35 points total] For the LL(1) grammar

   1. \( S' \rightarrow A \ $\)
   2. \( A \rightarrow B \ C \ A \)
   3. \( A \rightarrow \epsilon \)
   4. \( B \rightarrow b \ B \)
   5. \( B \rightarrow d \)
   6. \( C \rightarrow c \ C \)
   7. \( C \rightarrow \epsilon \)

   a. [8 points] Calculate First(\( S' \)), First(A), First(B), First(C).
   b. [8 points] Calculate Follow(\( S' \)), Follow(A), Follow(B), Follow(C).
   c. [10 points] Write out the Predict table.
   d. [9 points] Show a trace of the parse of \( b \ b \ c \ d \ a \ $ \) Use the format in the Class 13 notes.

**LR(1) and LALR(1) Grammars**

3. [35 points total] For this problem, the grammar is:
1: $S' \rightarrow S \, \$ $
2: $S \rightarrow T \, s \, T \, t$
3: $S \rightarrow U \, t$
4: $T \rightarrow s$
5: $U \rightarrow s$
6: $U \rightarrow \varepsilon$

a. [20 points] Write a deterministic LR(1) CFSM for the grammar: Write it as a table with states, Actions (shift or reduce), and Goto entries. For the states, remember than in an LR(1) parser, the items of the states include a lookahead symbol that gets used when a reduction is done. E.g., the start state includes the items $S' \rightarrow \cdot \, S \, \$, $\varnothing$ and $S \rightarrow \cdot \, T \, s \, T \, t \, , \, \$ (and some others). Use the format from Figure 1 of the Class 17 notes:

<table>
<thead>
<tr>
<th>St#</th>
<th>States</th>
<th>Action</th>
<th>Go To</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{1.0: $S' \rightarrow \cdot , S , $, $\varnothing$, 2.0: $S \rightarrow \cdot , T , s , T , t , , , $}</td>
<td>s</td>
<td>t</td>
</tr>
</tbody>
</table>

b. [5 points] Which LR(1) states (of the parser from part (a)) would you join together to form a LALR(1) parser?

c. [5 points] Show a trace of parsing input $s \, t \, s \, s \, \$ using the LR(1) parser. (Stop when the parse fails.) Use the format from Figure 2 of the Class 17 notes. [11/30]

d. [5 points] Repeat part (c) but use the LALR(1) parser. If the trace is different from the one in part (c), briefly explain why.

Yacc and Lex

4. [10 points] In the Class 18 notes, there's a yacc parser (pp. 6–7) with lex lexer (p.6) for expressions built from natural number constants, parentheses, and the + and * operators.

a. [3 points] The rule for term is above the rule for factor. Is this important? Briefly, why or why not?

b. [4 points] What two lines do you have to add to the yacc parser to be able to add - and / operators to the language?

c. [3 points] Aside from part (b), what else do you need to change with the yacc/lex combination in order to add subtraction and division to the calculator program?