Problem B1 (60 pts). Consider the grammar $G$ for arithmetic expressions with $\Sigma = \{+, *, (, )\}$, given by

0: $S \rightarrow E$
1: $E \rightarrow E + T$
2: $E \rightarrow T$
3: $E \rightarrow T \ast F$
4: $T \rightarrow F$
5: $F \rightarrow (E)$
6: $F \rightarrow a$

1) Construct the deterministic characteristic DFA $DC_G$ and the action and goto tables for the grammar $G$ using the methods described in Chapter 4 of the Notes (pages 87–98). You should obtain a DFA with 12 states (not counting an error dead state) whose states defined by their core items and action and goto tables and shown below.

1: $S \rightarrow .E$
2: $S \rightarrow E.$
   $E \rightarrow E. + T$
3: $E \rightarrow T.$
   $T \rightarrow T. * F$
4: $T \rightarrow F.$
5: $F \rightarrow (.E)$
6: $F \rightarrow a.$
Which states are final states?

(2) Using the computation of $SLR(1)$ lookahead sets using the FOLLOW sets, it can be shown that the shift and reduce actions for accepting states yield the following parsing tables (you do not need to justify this):
Show the parsing actions on inputs $a + a \ast a$ and $(a + a) \ast a$ using the format used in the notes on page 98.

(3) (Extra Credit (50 points))

Implement the shift-reduce algorithm (as described in the notes, pages 96-98) and test it on the grammar of part (1) for various input strings. Explain clearly what is the format for the input. You may want to encode shift entries as positive integers and reduce entries as negative integers. The output should consist of the sequence of parsing actions following the format of the notes.

**Problem B2 (40 pts).** Give a context-free grammar for the language over the alphabet \{a, b, c\} given by

$L = \{xy | x \neq y, x, y \in \{a, b\}^*\}$.

*Hint.* At first glance, this seems impossible. Think nondeterministically. You need to figure out how to express that $x \neq y$ in such a way that you can write grammar rules that enforce this condition. Obviously, this is the case if $|x| < |y|$ or $|y| < |x|$. Another possibility is that $x$ and $y$ differ by some symbol in the same position (scanning from left to right).

If you do it “right,” your choice of productions should yield a justification of the correctness of your grammar.

**Problem B3 (20 pts).** Prove that the extending pairing function $\langle x_1, \ldots, x_n \rangle_n$ defined in the notes (see Section 6.1 of the notes) satisfies the equation

$\langle x_1, \ldots, x_n, x_{n+1} \rangle_{n+1} = \langle x_1, \langle x_2, \ldots, x_{n+1} \rangle_n \rangle$.

Compute $\langle 2, 5, 7, 17 \rangle_4$ (this integer has 10 digits).

**Problem B4 (10 pts).** Explain (concisely) why the composition of two total recursive functions (of a single argument) computed by RAM programs is also total recursive.

**Problem B5 (30 pts).** Prove that the function, $f : \Sigma^* \rightarrow \Sigma^*$, given by $f(w) = www$ is RAM computable by constructing a RAM program ($\Sigma = \{a, b\}$).

**Problem B6 (80 pts).** Write a computer program implementing a RAM program interpreter. You may want to assume that the instructions have five fields

<table>
<thead>
<tr>
<th>N</th>
<th>X</th>
<th>opcode</th>
<th>j</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>X</td>
<td>opcode</td>
<td>j</td>
<td>N1</td>
</tr>
</tbody>
</table>
with $j \in \{1, \ldots, k\}$, where $k$ is the number of symbols in $\Sigma$, and that the opcodes are

```
add  tail  clr  assign  gotoa  gotob  jmpa  jmpb  continue
```

where `gotoa` corresponds to jump above, `gotob` to jump below, `jmpa` corresponds to jump above if condition is satisfied, and `jmpb` to jump below if condition is satisfied. Depending on the opcode, some of the fields may be irrelevant (set them to 0).

The number of input registers is $n$ (so your memory must have at least $n$ registers), and the total number of registers is $p$. The number $k, n, p$ are input to your interpreter, as well as the program to be executed (a sequence of instructions), and the $n$ input strings $w_1, \ldots, w_n$. Assume that line numbers are integers. For the symbols in $\Sigma$, you may either assume that they are the integers 1, 2, \ldots, $k$ (if $k > 9$, don’t forget spaces to separate the characters) or you may supply a list of the characters that you wish to use as input (for example, $a, b, c, \ldots, z$).

Your program should output.

1. The input RAM program $P$

2. The input strings $w_1, \ldots, w_n$ to the RAM program $P$.

3. The value of the function being computed.

4. The sequence consisting of the memory contents and the current program counter as your interpreter executes the RAM program.

Test your interpreter on several RAM programs (and input strings), including the program of B5.

**TOTAL: 240 points + 50 points Extra credit**