“A problems” are for practice only, and should not be turned in.

Problem A1. (i) Given any set $X$, for any two subsets $A, B \subseteq X$, prove that $A \subseteq B$ iff $A \cap B = \emptyset$, where $B$ denotes the complement of $B$ in $X$.

(ii) Sketch an algorithm to test whether $L(G) \subseteq L(D)$, where $G$ is a context-free grammar and $D$ is a DFA.

“B problems” must be turned in.

Problem B1 (80 pts). In this problem, the fundamental property of LR-parsing (due to Knuth) is established.

For simplicity, let us consider context-free grammars without $\epsilon$-rules. Given a reduced context-free grammar $G = (V, \Sigma, P, S')$ augmented with start production $S' \rightarrow S$, where $S'$ does not appear in any other productions, the set $C_G$ of characteristic strings of $G$ is the following subset of $V^*$ (watch out, not $\Sigma^*$):

$$C_G = \{ \alpha \beta \in V^* \mid S' \xrightarrow{\text{rm}}^* \alpha Bv \xrightarrow{\text{rm}} \alpha \beta v, \alpha, \beta \in V^*, v \in \Sigma^*, B \rightarrow \beta \in P \}.$$

In words, $C_G$ is a certain set of prefixes of sentential forms obtained in rightmost derivations: those obtained by truncating the part of the sentential form immediately following the rightmost symbol in the righthand side of the production applied at the last step.

The fundamental property of LR-parsing is that $C_G$ is a regular language. A nondeterministic automaton $N_{C_G}$ accepting $C_G$ can be constructed according to the method described in Section 1 of the handout A Survey of LR-Parsing Methods, etc.. Please, review this construction.
(i) Let $G$ be the following grammar:

$$
S' \rightarrow E \\
E \rightarrow E + T \mid T \\
T \rightarrow T \ast F \mid F \\
F \rightarrow (E) \mid a
$$

with $\Sigma = \{+, \ast, (,), a\}$.

Give the automaton $NC_G$ for the grammar $G$.

(ii) Using the standard algorithms, give a deterministic finite automaton equivalent to $NC_G$. Do not include the “dead state”.

(iii) You shall now prove that $L(NC_G) = C_G$!

(1) Prove the following claim by induction on the length of rightmost derivations:

Claim 1: For any nonterminal $A$, for every rightmost derivation $A \Rightarrow rm \ast \alpha B v \Rightarrow rm \ast \alpha \beta v$, where $v \in \Sigma^*$, $B \in N$, and $\alpha, \beta \in V^*$, if we denote the first production in the above rightmost derivation as $A \rightarrow \delta$, then there is a computation on input $\alpha \beta$ from state $A \rightarrow \ast \delta$ to the final state $B \rightarrow \ast \beta$.

To prove this claim, you will have to show the following (think about it in terms of parse trees). For any nonterminal $A$, every rightmost derivation from $A$ is either of the form

(i) $A \Rightarrow rm \ast \delta$, for some production $A \rightarrow \delta$, in which case $A = B$ and $\delta = \beta$, or of the form

(ii) $A \Rightarrow rm \ast \lambda B_i \rho \Rightarrow rm \ast \lambda B_i w \Rightarrow rm \ast \lambda B_i w_i w \Rightarrow \lambda \alpha_i \beta w_i w$, with $w, w_i \in \Sigma^*$, $A, B, B_i \in N$, $\lambda, \rho, \alpha_i, \beta \in V^*$, and where

$$B_i \Rightarrow rm \ast \lambda \alpha_i B w_i \Rightarrow rm \ast \alpha_i \beta w_i \text{ and } \rho \Rightarrow rm \ast w.$$

Let $B_i \rightarrow \delta_i$ be the first production applied in the rightmost derivation from $B_i$. In the first case, there is a computation in $NC_G$ from state $A \rightarrow \ast \delta$ to the final state $A \rightarrow \ast \delta$ (where again, $A \rightarrow \delta = B \rightarrow \beta$), and in the second case, there is a computation in $NC_G$ from state $A \rightarrow \ast \lambda B_i \rho$ to $B_i \rightarrow \ast \delta_i$ on input $\lambda$, and a computation from state $B_i \rightarrow \ast \delta_i$ to the final state $B \rightarrow \ast \beta$ on input $\alpha_i \beta$.

Conclude that $C_G$ is a subset of $L(NC_G)$.

(2) Prove the following claim by induction on the number of $\epsilon$-transitions in a computation in $NC_G$:

Claim 2: For any state $A \rightarrow \ast \delta$, if there is a computation on input $\gamma$ to some final state $B \rightarrow \ast \beta$, then there is some rightmost derivation $A \Rightarrow rm \ast \alpha B v \Rightarrow \ast \alpha \beta v$, such that, the production applied in the first rightmost derivation step is $A \rightarrow \delta$, and $\gamma = \alpha \beta$.

For this, prove the following:
i) Either $\gamma = \delta$ and the computation is from state $A \rightarrow \cdot \delta$ to state $A \rightarrow \delta \cdot$, or

(ii) $\delta$ is of the form $\lambda B_i \rho$, $\gamma$ is of the form $\lambda \alpha_i \beta$, and there is a computation on input $\alpha_i \beta$ from some state of the form $B_i \rightarrow \cdot \delta_i$ to the final state $B \rightarrow \beta \cdot$, and a rightmost derivation as in Claim 1.

Conclude that $L(N_{CG})$ is a subset of $CG$, thus establishing that $CG = L(N_{CG})$.

Problem B2 (40 pts). Use the pumping lemma (or Ogden’s lemma) to show that the following languages are not context-free:

$L_1 = \{a^m b^n c^p \mid 1 \leq m < n < p\}$

$L_2 = \{a^n b^n c^p \mid n, p \geq 1, p \neq n\}$

**Hint.** Let $m$ be the constant of Ogden’s Lemma. For $L_1$, pick $w = a^m b^{m+1} c^{m+2}$ with the $a$’s marked. For $L_2$, let $p = m!$, pick $w = a^{2p} b^{2p} c^p$ with the $c$’s marked.

Problem B3 (40 pts). (1) Prove that the function, $f: \Sigma^* \rightarrow \Sigma^*$, given by

$$f(w) = w^R$$

is RAM computable. ($\Sigma = \{a_1, \ldots, a_N\}$).

(2) Prove that the function, $f: \Sigma^* \rightarrow \Sigma^*$, given by

$$f(w) = www$$

is RAM computable. ($\Sigma = \{a_1, \ldots, a_N\}$).

Problem B4 (100 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 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). Assume that line numbers are integers.

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 programs of B3.

**Problem B5 (20 pts).** Prove that the following properties of partial recursive functions are undecidable:

(a) A partial recursive function is a constant function.

(b) Two partial recursive functions $\varphi_x$ and $\varphi_y$ are identical.

(c) A partial recursive function $\varphi_x$ is equal to a given partial recursive function $\varphi_a$.

(d) A partial recursive function diverges for all input.

**Problem B6 (60 pts).** Let $A$ be any $p \times q$ matrix with integer coefficients and let $b \in \mathbb{Z}^p$ be any vector with integer coefficients. The 0-1 integer programming problem is to find whether the system

$$Ax = b$$

has any solution, $x \in \{0, 1\}^q$.

(i) Prove that the 0-1 integer programming problem is in $\mathcal{NP}$.

(ii) Prove that the 0-1 integer programming problem is $\mathcal{NP}$-complete by providing a polynomial-time reduction from the bounded-tiling problem. **Do not try to reduce any other problem to the 0-1 integer programming problem.**

*Hint.* Given a tiling problem, $((T, V, H), \hat{s}, \sigma_0)$, create a 0-1-valued variable, $x_{mnt}$, such that $x_{mnt} = 1$ iff tile $t$ occurs in position $(m, n)$ in some tiling. Write equations or inequalities expressing that a tiling exists and then use “slack variables” to convert inequalities to equations. For example, to express the fact that every position is tiled by a single tile, use the equation

$$\sum_{i \in T} x_{mnt} = 1,$$

for all $m, n$ with $1 \leq m \leq 2s$ and $1 \leq n \leq s$.

(iii) Prove that the restricted 0-1 integer programming problem in which the coefficients of $A$ are 0 or 1 and all entries in $b$ are equal to 1 is also $\mathcal{NP}$-complete.

**TOTAL: 340 points.**