Problem 1:

A -> BAB | B | $\epsilon$
B -> 00 | $\epsilon$

Add a new start symbol.
S -> A
A -> BAB | B | $\epsilon$
B -> 00 | $\epsilon$

Remove $\epsilon$ rules.
S -> A | $\epsilon$
A -> BAB | B | AB | BA | BB
B -> 00

Remove unit rules.
S -> BAB | BA | AB | BB | 00 | $\epsilon$
A -> BAB | BA | AB | BB | 00
B -> 00

Convert to proper form.
S -> BT | BA | AB | BB | CC | $\epsilon$
A -> BT | BA | AB | BB | CC
T -> AB
B -> CC
C -> 0

Problem 2:

(a.) Let $M_c = (Q_c, \Sigma_c, \Gamma_c, \delta_c, q_c, F_c)$ be a PDA that accepts $C$, and $M_r = (Q_r, \Sigma_r, \delta_r, q_r, F_r)$ be a DFA that accepts $R$. Since only one machine uses the stack, we can run the two machines in parallel and accept if both accept. The machine specification is below.

$Q = (Q_c \cup \{q_{rej}\}) \times (Q_r \cup \{q_{rej}\})$
$\Sigma = \Sigma_c \cup \Sigma_r$
$\Gamma = \Gamma_c$
$q = (q_c \times q_r)$
$F = (F_c \times F_r)$
$\delta((q_0, q_1), a, q) = \{(q, \delta_c(q_1, a), \beta) \mid (q, \beta) \in \delta_c(q_0, a, q)\}$
$\delta((q_0, q_1), \epsilon, q) = \{(q, q_1, \beta) \mid (q, \beta) \in \delta_c(q_0, a, q)\}$
Assume $A$ is context-free. Then, $A \cap a^*b^*c^* = \{a^n b^n c^n\}$ is also context-free, a contradiction. See Example 2.20 for a proof of this.

**Problem 3:**

We use a regular grammar to simulate the non-deterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$. For each state $q_i \in Q$, we create a variable $A_i$. The start variable is the variable corresponding to the start state. For each transition $\delta(q_i, a) = P$, we add a rule $A_i \to aA_j$ for every $q_j \in P$. For each accept state $q_i \in F$, we add the rule $A_i \to \epsilon$. Note that an accepting path in $Q$ for a string $s$ corresponds one-to-one with a derivation for the string $s$ in the new grammar, so $L(G) = L(M)$.

**Problem 4:**

Consider the following context-free grammar $G$:

$$
S \to B1B \mid B1S \\
B \to BB \mid 0B1 \mid 1B0 \mid \epsilon
$$

Let $L'$ be the language of strings with a balanced number of ones and zeroes. We first show that $B$ generates all strings in $L'$.

$L(B) \subseteq L'$:

A simple inductive argument shows that $B$ generates only strings with a balanced number of ones and zeroes.

$L' \subseteq L(B)$ (by induction):

Base case: $B$ generates all strings in $L'$ of length two or less (01 and 10).

Inductive step: Consider a string $s$ in $L'$ of length greater than two. If $s$ is of the form $0s'1$ or $1s'0$, then $s'$ must be balanced, and by the inductive hypothesis $B$ generates $s'$, so $B$ can generate $s$. Otherwise, suppose $s$ is of the form $0s'0$. Let the bias at $i$ be the number of zeros appearing in the first $i$ symbols minus the number of ones appearing in the first $i$ symbols. The bias at one is one, while the bias at $|s|-1$ is negative one. Thus, the bias must be zero for some $0 < i < |s|$. Then, the first $i$ symbols form a balanced substring, as do the remaining symbols. Each substring is of length smaller than $|s|$, so by the inductive hypothesis each is generated by $B$ and $s$ is generated by $B \to BB$.

**Problem 5:**

Let $s = a^ib^jc^j$ be any string in $L_1$, where $|s| > 5$. We can always break $s = (u)(v)(x)(y)(z)$ into five pieces which satisfy the basic pumping lemma using the following case analysis.

If $i < j-1$, then $s = (a^ib^{i-1})(b)(c)(c^{j-1})$. 
If \( i = j-1 \), then \( j \geq 2 \) (since \( |s| > 5 \)), and \( s = (a^ib^{j-2})(bb)(\epsilon)(cc)(c^{j-2}) \).
If \( i > j-1 \), then \( i \geq 2 \) (since \( |s| > 5 \)), and \( s = (\epsilon)(a)(\epsilon)(a^i c^j b c^j) \).

**Problem 6:**

(a.): Let \( G \) be a CFG for CFL \( L \). Let \( b \) be the maximum number of symbols in the right-hand side of a rule. We may assume that \( b \geq 2 \). Let \( |V| \) be the number of variables in \( G \). We set \( p \) to be \( b|V|+2 \). We say that an interior node of the parse tree is structural if at least two of its descendants generate a string containing a marked character. If \( w \in L \) has at least \( p \) marked characters, any parse tree for \( w \) will contain a path from root to leaf containing at least \( |V| + 1 \) structural nodes. Let \( \tau \) be the parse tree for \( w \) with the smallest number of nodes. There exists some path from root to leaf containing at least \( |V| + 1 \) structural nodes. Thus, some variable \( R \) will appear more than once on this path as a structural node. We select \( R \) to be a variable that repeats among the lowest \( |V| + 1 \) structural nodes on this path. Let the upper occurrence of \( R \) generate \( vxy \), and let the lower occurrence of \( R \) generate \( x \). Both of these subtrees are generated by the same variable, so we may substitute one for the other to generate. This establishes condition (a). Either \( v \) or \( y \) must have a marked character, since two of the upper occurrence of \( R \)’s children generate marked characters, and only one of these children generates the lower occurrence of \( R \). This establishes condition (b). Finally, since the upper occurrence of \( R \) falls within the bottom \( |V| + 1 \) structural nodes on the path, this variable may generate at most \( b|V|+2 \) marked characters, establishing condition 3.

(b.): Consider the string \( s = a^{p+p!}b^pc^p \), where all of the \( b \)'s and \( c \)'s are marked. We will attempt to breaks \( s \) into five pieces \( uvxyz \) satisfying Ogden’s lemma. There must be at least one marked character in \( v \) and \( y \) together, so one of the two must contain a “\( b \)” or “\( c \)”.
If either contains an “\( a \)” ,neither may contain a “\( c \)”, since otherwise \( vxy \) would have more than \( p \) marked characters. But then, since either \( v \) or \( y \) must contain a “\( b \)”, the pumped string will contain different numbers of \( b \)'s and \( c \)'s, a contradiction. On the other hand suppose neither string contains an “\( a \)”. If \( v \) and \( y \) together contain different numbers of \( b \)'s and \( c \)'s, pumping will yield a string not in the language. If either \( v \) or \( y \) contains both \( b \)'s and \( c \)'s, pumping likewise yields a contradiction. Finally, if \( v \) consists only of \( n \) \( b \)'s, \( n \leq p \), and \( y \) consists only of \( n \) \( c \)'s, \( n \leq p \), then for \( i = p!/n \), we have a contradiction.