CMSC 330 HOMEWORK EXERCISES #2

1. Give unambiguous grammars for the following languages:

   (a) $\{a^n b^n \mid n \geq 0$ and n is even $\}$

   (b) $\{a^i b^j c^{2i+1} d^k \mid i, j, k \geq 0\}$

   (c) $\{a_1 a_2 ... a_n a_n ... a_2 a_1 \mid a_i \in \{a, b\}, 1 \leq i \leq n\}$

   (d) $\{a^m b^n a^{m+n} \mid m \geq 0$ and $n \geq 1$ $\}$

   (e) $\{a^n b^m a^{n-m} \mid n \geq m \geq 0$ $\}$

   (f) All possible sequences of balanced parentheses

   (g) $\{w \mid w \in \{a, b\}^*$ and w has an odd number of a's and an odd number of b's $\}$.

   (h) $\{w \mid w \in \{a, b\}^*$ and w has an equal number of a's and b's $\}$

   (i) $\{a^m b^n c^p d^q \mid m + n = p + q$ $\}$

   (j) $\{a^m b^n \mid m \neq n$ and $m, n \geq 0$ $\}$

   (k) $\{w \mid w \in \{a, b\}^*$ and w contains no substrings of the form $ab\}$

   (l) $\{w \mid w \in \{a, b\}^*$ and every pair of a's is followed by at least one b $\}$

2. Write a grammar for the language

$$\{a^n b^n a^m b^m \mid m, n \geq 1\} \;\cup\; \{a^n b^m a^m b^n \mid m, n \geq 1\}$$

   If your grammar is ambiguous, prove that it is.

3. Consider a grammar with the following productions:

```
<stmtlist>    ::=  <stmtlist> ; <stmt> | <stmt>
<stmt>        ::=  <if stmt> | skip
<if stmt>     ::=  if b then <stmtlist> <else part> fi
<else part>   ::=  else <stmtlist>  | ∈
```

   Do these productions cure the "dangling" else problem by eliminating the ambiguity of which <if stmt> the final <else part> is associated with? Does the following grammar cure the problem?

```
<Stmt>        ::=  <UStmt> | <CStmt>
<UStmt>       ::=  skip | begin <StmtList> end
<CStmt>       ::=  if b then <UStmt> else <Stmt> | if b then <UStmt>
<StmtList>    ::=  <StmtList> ; <Stmt> | <Stmt>
```

4. Grammars can be used to express concepts like associativity and precedence. For each of the following languages, give unambiguous grammars capturing the appropriate associativity and precedence.

   (a) Operators in APL are right associative and have equal precedence, unless altered by parentheses. The operators +, -, *, / can appear as both monadic (unary) and dyadic (binary) operators. Assignment ($\leftarrow$) is also treated as a binary operator that returns the value assigned as its result. Write an unambiguous context free grammar for APL expressions containing the operands a, b, and c.

   (b) Propositional calculus formulas having operators unary $\sim$ and binary $\supset$, operands p and q, and parentheses. The operators should both be right associative and have their usual precedence.

   (c) C++ expressions with identifiers `a` and `b` and operators `->`, `*`, and `++`. `->` is a left associative binary operator, `*` is a right associative unary operator, and `++` is a right associative postfix operator. The unary operators have precedence lower than the postfix expressions but higher than all binary operators. For example, the expression `*a++` increments `a` before dereferencing it.

5. Context-free grammars cannot express all the restrictions normally placed on programming languages, e.g., that all procedure statements have the same number of actual parameters as the procedure's definition had formals. Consider a LISP-like language which permits a single procedure definition and procedure statement (each with an arbitrary number of parameters) defined by the following grammar:

<S>	::=	(defun p (<Formals>) <Body>) (p <Actuals>)
<Formals>	::=	<Formals> <Id> $\mid \in$
<Actuals>	::=	<Actuals> <Exp> $\mid \in$
<Body>	::=	...
<Id>	::=	...
<Exp>	::=	...

Develop another grammar for this language that enforces the restriction that a call must have the same number of arguments as the declaration (you can treat <Body>, <Id> and <Exp> as non-terminals for this purpose).

Is it possible to generalize this to a language in which arbitrarily many calls to a procedure can occur?