To make your answers easier to compare with the solutions, use consecutive nonterminals beginning with S when writing your grammars (S, T, U, etc.).

1. In Pascal both of the following syntax forms can be used for an array type

   `array[`*lower–bound*$_1$`..`*upper–bound*$_1$`, ..., `*lower–bound*$_n$`..`*upper–bound*$_n$`] of` *element–type*`;`

   `array[`*lower–bound*$_1$`..`*upper–bound*$_1$`]...[`*lower–bound*$_n$`..`*upper–bound*$_n$`] of` *element–type*`;`

   What an array's bounds and an array's element type can be are described below.

   Note that an array's lower subscript is not necessarily 0 as in C, Ruby, and Java. For example, here are a few array types in Pascal (all of which have integer bounds):

   ```
   array[1..10] of integer;
   array[5..50] of char;
   array[1..10,10..20] of integer;
   array[1..10][10..20] of integer;
   ```

   The second one has subscripts ranging between 5 and 50 (inclusive), while the last two are two–dimensional arrays.

   Write an unambiguous grammar generating array types similar to Pascal's as specifically described below. (Array types in this problem are based upon, but not completely the same as, Pascal's arrays.)

   - For purposes of this problem, an array type begins with the word `array`, which is followed by the array's dimensions, then the word `of`, then an identifier and a semicolon. Don't worry about indicating where whitespace may appear.
   - An array's dimensions are either a sequence of one or more ranges each inside its own set of square braces, or a single comma–separated list of one or more ranges all together inside one set of square braces.
   - For purposes of this problem a range consists of two constant values of the same type separated by `..` (two adjacent period characters). A range can contain two boolean constants, two identifiers, two character constants, or two integer constants.
   - The only two boolean constants are `true` and `false`. Assume that identifers consist of sequences of one or more occurrences of of the letters `a`, `b`, and `c`, and the characters that can appear as character constants are `x`, `y`, and `z`. A character constant is surrounded by single quotes (`''`). An integer constant consists of a sequence or one or more decimal digits (`0` through `9`), optionally preceded by a single plus (`+`) or minus (`-`) sign.

   Note that your grammar must generate all the tokens (integer constants, character constants, etc.) in array types; the only thing you don't have to worry about is where whitespace may appear.

2. Write unambiguous grammars for the following languages:

   a. $\{ a^{\mathrm{n}} b^{\mathrm{n}} \mid \mathrm{n} \geq 0 \text{ and n is even} \}$

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

   c. $\{ \gamma_1 \gamma_2 \ldots \gamma_{\mathrm{n}} \gamma_{\mathrm{n}} \ldots \gamma_2 \gamma_1 \mid \gamma_i \in \{a, b\},\ 1 \leq i \leq \mathrm{n} \}$

      Note this language is describing all strings that are palindromes, that is, they read the same forward as backwards, over the alphabet $\Sigma = \{a, b\}$.

   d. $\{ a^{\mathrm{m}} b^{\mathrm{n}} a^{\mathrm{m+n}} \mid \mathrm{m} \geq 0 \text{ and n} \geq 1 \}$

e. $\{ a^{\mathrm{n}} b^{\mathrm{m}} a^{\mathrm{n-m}} \mid \mathrm{n} \geq \mathrm{m} \geq 0 \}$

f. All possible sequences of balanced parentheses. Each string in the language is composed of zero or more of the symbols `(` and `)`. Each valid string must have same number of left parentheses as right parentheses, and they must be properly balanced, with every opening left parenthesis paired with a later closing right parentheses. Valid strings may contain nested parentheses. For example, the strings (()) and (()(()())) contain matching balanced parentheses, while (()(()()) and (()))(do not.

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

h. $\{ a^{\mathrm{m}} b^{\mathrm{n}} \mid \mathrm{m} \neq \mathrm{n}$ and m, n $\geq 0 \}$

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

3. 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.