Answer to Exercise 6.1

1. Straightforward, only the key steps are given. See below for an example of a complete proof. If n is the constant of the pumping lemma, consider a specific word $a^n b^m c^{n+m}$, where $m \geq 0$. This word clearly belongs to L_1, and it is sufficiently long to satisfy the requirements of the lemma. Once split into three parts according to the lemma, the middle part is going to be non-empty and it is going to consist of a's only. If the middle part is repeated, the resulting string is not going to have the right balance of a's, b's, and c's, and thus it is not going to belong to L_1.
2. Assume that L_2 is a regular language. Then, according to the pumping lemma for regular languages, there exists a constant n such that any word $w \in L_2$ that has length at least n ($|w| \geq n$) can be split into three parts, $w = xyz$, as follows:
   1. $y \neq \epsilon$
   2. $|xy| \leq n$
   3. $\forall k \in \mathbb{N} . xy^k z \in L_2$

   Consider the word[11] $w = a^{4n} b^{2n} c^n$. Clearly $w \in L_2$. Moreover $|w| = 7n \geq n$. The prerequisites of the lemma are thus fulfilled, and we know it is possible to split w into three parts x, y, and z satisfying the conditions of the lemma. Because our chosen word w starts with $4n$ a's, and because the combined length of the two first parts, x and y, is at most n, we know that x and y consists solely of a's. Thus $x = a^i$ and $y = a^j$ for some $i, j \in \mathbb{N}$ such that $i + j \leq n$. Furthermore, because y is not empty according to the lemma, we know that $j > 0$. z is whatever remains of w; i.e., $z = a^{(4n-i-j)} b^{2n} c^n$.
   Now consider words of the form $xy^k z$. According to the pumping lemma, these words belong to L_2 for any value of k. Let us consider a specific value for k, for example $k = 0$. The word $xy^0 z$ should belong to L. But $xy^0 z = xz = a^i a^{(4n-i-j)} b^{2n} c^n = a^{(4n-j)} b^{2n} c^n$. Because $j > 0$, it is now clear that there are *fewer* than twice as many a's as b's in this word, which thus *cannot* belong to L_2. We have reached a contradiction, and our assumption that L_2 is a regular language must be wrong. Thus L_2 is *not* a regular language, QED.

[11] Note how w is chosen: it is one specific word that obviously is a word in L_2, but that depends on the constant n in such a way that the length of w is at least n *whatever* n is. It is *completely wrong* to assume that n is some particular value! All we know is that n exists. Also, note that the structure of w was chosen to facilitate the proof. For example, a word $(aaaabbc)^n$ is also a word in L_2 with length at least n. But in this case, it is not going to be possible to show that *all* ways to divide the word into three parts that satisfy the constraints of the lemma are going to lead to a contradiction. By first establishing that n must be at least 7 (which can be done), we will see that a division of v into $x = \epsilon$, $y = aaaabbc$, and $z = (aaaabbc)^{n-1}$ always is a possibility, which in turn means that all words of the form $xy^k z$ *do* belong to L, and we do not get any contradiction.

Answer to Exercise 7.1

1. (a)

$$\begin{array}{rcll} S & \underset{G}{\Rightarrow} & X & \text{by } S \to X \\ & \underset{G}{\Rightarrow} & \epsilon & \text{by } X \to \epsilon \end{array}$$

or

$$\begin{array}{rcll} S & \underset{G}{\Rightarrow} & Y & \text{by } S \to Y \\ & \underset{G}{\Rightarrow} & \epsilon & \text{by } Y \to \epsilon \end{array}$$

When giving derivation sequences in a context-free grammar, it is normally *not* necessary to justify every single step as it is fairly obvious which production is being used. The justified derivation sequence above were just given for explanatory purposes. Thus, an answer like the following is perfectly OK too:

$$S \underset{G}{\Rightarrow} X \underset{G}{\Rightarrow} \epsilon$$

or even

$$S \Rightarrow X \Rightarrow \epsilon$$

as it is clear which grammar we are referring to from the context. However, to make it very clear how derivations work, we will give explicit justifications here.

(b)

$$\begin{array}{rcll} S & \underset{G}{\Rightarrow} & X & \text{by } S \to X \\ & \underset{G}{\Rightarrow} & aXb & \text{by } X \to aXb \\ & \underset{G}{\Rightarrow} & aaXbb & \text{by } X \to aXb \\ & \underset{G}{\Rightarrow} & aabb & \text{by } X \to \epsilon \end{array}$$

(c)

$$\begin{array}{rcll} S & \underset{G}{\Rightarrow} & Y & \text{by } S \to Y \\ & \underset{G}{\Rightarrow} & cYd & \text{by } Y \to cYd \\ & \underset{G}{\Rightarrow} & ccYdd & \text{by } Y \to cYd \\ & \underset{G}{\Rightarrow} & cccYddd & \text{by } Y \to cYd \\ & \underset{G}{\Rightarrow} & cccddd & \text{by } Y \to \epsilon \end{array}$$

2. No, $aaaddd \notin L(G)$. From the start symbol S, we can either derive X or Y. However, from X it is only possible to derive strings $a^n b^n$, while from Y it is only possible to derive strings $c^n d^n$, neither of which are words starting with a's and ending with d's.
3. $L(G) = \{a^n b^n \mid n \in \mathbb{N}\} \cup \{c^n d^n \mid n \in \mathbb{N}\}$

Answer to Exercise 7.2

The following is one possible grammar generating L:

$$\begin{array}{rcl} S & \rightarrow & XY \mid Z \\ X & \rightarrow & abXba \mid abX_1ba \\ X_1 & \rightarrow & bcX_1cb \mid bccb \\ Y & \rightarrow & dY \mid \epsilon \\ Z & \rightarrow & ddY \end{array}$$

S, X, X_1, Y, and Z are nonterminal symbols, S is the start symbol, and a, b, c, and d are terminal symbols.

(The following explanation is very detailed to make it easy to follow what is going on. An explanation of the key ideas is sufficient for full marks as long this explanation reflects a clear understanding of the construction.)

The grammar was constructed as follows. There are three constituent sets in the definition of the language L. Call them

$$\begin{array}{rcl} L_X & = & \{(ab)^m(bc)^n(cb)^n(ba)^m \mid m, n \geq 1\} \\ L_Y & = & \{d^n \mid n \geq 0\} \\ L_Z & = & \{d^n \mid n \geq 2\} \end{array}$$

Introduce a non-terminal for each, such that the set is generated by using that non-terminal as a start symbol; i.e. the nonterminal X corresponds to the set L_X etc. Then observe that L is the concatenation of the sets L_X and L_Y in union with L_Z; i.e. $L = L_X L_Y \cup L_Z$. This is captured by the two productions:

$$S \rightarrow XY \mid Z$$

The set L_Y, i.e. zero, one, or more d's, is described by the following recursive productions:

$$Y \rightarrow dY \mid \epsilon$$

The set L_Z is similar, except there has to be at least two d'd. We can obtain such a set by simply prefixing all words in the set L_Y by two d's. as follows:

$$Z \rightarrow ddY$$

(Obviously, there is nothing wrong by doing it from scratch, not "reusing" the productions for Y.)

Finally, as the words in L_X are strings with a balanced nesting of one pair of substrings (bc and cb) inside another (ab and ba), we need to introduce a "helper" non-terminal to deal with the inner nesting. Keeping in mind that each substring pair should occur at least once, we obtain:

$$\begin{array}{rcl} X & \rightarrow & abXba \mid abX_1ba \\ X_1 & \rightarrow & bcX_1cb \mid bccb \end{array}$$