(1) Text, chapter 1, exercise 1.4, parts (c) and (f) only. These are to design DFA accepting the respective languages, which are:

(c) $\{ w \in \{ a, b \}^* : \#a(w) \text{ is even} \wedge (\#b(w) = 1 \vee \#b(w) = 2) \}$.

(f) $\{ w \in \{ a, b \}^* : \#a(w) \text{ is odd} \wedge w[|w| - 1] = b \}$.

Here $\#c(w)$ is a helpful notation we will use throughout the course to mean "the number of occurrences of the character c in the string w" and $w[i]$ is ordinary notation for the i-th bit of w (unless otherwise indicated, numbering from 0 like in C/C++/Java). Also $|w|$ means the length of the string w. Thus (c) says that w has an even number of a's and one or two b's, while (f) says that w has an odd number of a's and ends in a b.

The text has you note that since each of these languages is defined via a logical AND at top level, each is the intersection of two "simpler" languages. The text wants you to design DFAs for the "simpler" languages and then use the Cartesian Product construction to combine them. My idea instead is for you to design them with a logical *plan* led by the idea expressed in lecture of writing *comments* for each state that state the logic of what being in that state means. Either way is OK. Use of the "Turing Kit" software is optional; unfortunately its PostScript printing utility has never been fully debugged. (12 + 12 = 24 points, for 48 total on the set)

(2) Text, chapter 1, exercises 1.6ℓ and 1.7c (note that the latter refers to the former). That is, first design a DFA M such that $L(M) = A$, where

$$A = \{ x \in \{0, 1\}^* : \#0(x) \text{ is even } \vee \#1(x) = 2 \}.$$

Then design an NFA N such that $L(N) = A$ where N has at most 6 states. *Finally*—this is not asked in the text—write a regular expression r such that $L(r) = A$. You may find N more helpful than M for building r—and as usual you should put some comments on states that express the logic of the design. (12 + 9 + 6 = 27 pts.)

(3) Text, chapter 1, exercise 1.12: Let $D = (A \cap B) \setminus C$ where

$$\begin{aligned} A &= \{w \in \{a, b\}^* : \#a(w)\%2 == 0\} \\ B &= \{w \in \{a, b\}^* : \#b(w)\%2 == 1\} \\ C &= (a + b)^* ab(a + b)^*. \end{aligned}$$

First work out how this agrees with what the text says—most in particular, subtracting C is the same as intersecting with the set of strings that do not contain the substring ab. Then do:

(a) Build a DFA M with 5 states such that $L(M) = D$.

(b) Build an NFA N with 4 states such that $L(N) = D$.

(c) Give a regular expression r such that $L(r) = D$.

Note that (b) is not asked in the text—and it may be only "kind-of" an NFA. But you can add a remark about whether it helped you focus on what was needed in part (c). (12 + 6 + 9 = 27 pts., for 78 total on the set)