1. Do the following problems from the textbook:

2.1 [c-g]
2.8 [c-g] Create and test the automata in JFLAP. Submit screenshots of the automaton and the result of a “complete” test set.
2.9. Again, use JFLAP. Note that your DFA should not accept words like suitcase or constant.
2.13 Use JFLAP to do the conversion. Submit a screenshot of each major step.

2. Do exercise 2 (Chapter 1) in the JFLAP book. For each of the FA examples, an example file and a language definition is given. For each example do the following: (i) Find 6 test strings that are incorrectly classified (accepted but they should be rejected or vice versa), (ii) Modify the machine so it is correct, (iii) Demonstrate the correctness of the machine by testing it on the 6 erroneous strings. (iv) Extra credit: describe each incorrect language using English or as a regular expression.
a) ex1.6a “any string in {a,b} with exactly 2 b’s
b) ex 1.6b “any string of a’s whose length is exactly divisible by 2 or 3”
c) ex. 1.6c “any string in {a, b, c} with at least 3 b’s or at least 3 c’s.

3. JFLAP book exercise 6 (Chapter 1). Consider the set of 17 strings S= {bid, pad, hin, bat, pin, pit, hid, han, pid, pan, hat, bad, bit, pat, hit, had}. If S had one more string, you could build a DFA with only eight transitions that recognizes the language S. What is the string? What is the regular expression for the resulting language?

The following two exercises are from JFLAP textbook Chapter 2
4. (Exercise 1) Convert the NFAs in files ex2-nfa2dfa-a through nfa2dfa-f into DFA’s.
5. (Exercise 4) Convert the DFA’s in files ex2-dfa2mindfa-a and ex2-dfa2mindfa-d into minimized DFA’s.

6. Create a minimized DFA for the language “All strings in {a, b, c} that do not contain the substring “abc”. Suggested procedure: 1) Create an NFA for the inverse of that language (all strings that *do* contain the substring. 2) Convert it to a DFA. 3) Switch the non-accepting and accepting states. 4) Minimize the resulting DFA.

Extra credit:
JFLAP book problem 7 (Chapter 1)