There are five problems, labeled A-E.

A. Do problem 1.5c from our textbook. The problem intends for you to show two DFAs.

B. Do problem 1.6j from our textbook.

The problems above are routine. On an exam, you are expected to complete such problems quickly. If you want more practice, then do more problems from the book on your own or with the prefect. The next problem is more challenging.

C. Do problem 1.32 from our textbook.

This next problem is about checking the validity of an HTML file. You don't need to know HTML. (It might be better if you didn't.) To keep the problem manageable, we use a highly simplified version of HTML, and we ignore all of the text in the file other than the tags. The HTML file is regarded as a string over the 14-symbol alphabet

$$\Sigma = \{\texttt{html}, \texttt{/html}, \texttt{head}, \texttt{/head}, \texttt{title}, \texttt{/title}, \texttt{body}, \texttt{/body}, \texttt{ul}, \texttt{/ul}, \texttt{li}, \texttt{/li}, \texttt{a}, \texttt{/a}\}.$$

We consider the file to be valid if and only if it meets the following criteria. (Later in this course, we will develop a more precise method of stating such criteria.)

- The file must begin with `html` and end with `/html`. These symbols cannot occur anywhere else in the file.
- Immediately after `html` there must be a *header*, which begins with `head` and ends with `/head`. These two tags cannot occur elsewhere.
- The header must either contain no tags at all, or must contain a single *title*. A title begins with `title`, ends with `/title`, and contains no other tags. These two tags cannot occur elsewhere.
- Immediately after the header there must be a *body*, which begins with `body` and ends with `/body`. These two tags cannot occur elsewhere.
- The body may contain any number (0 or more) of *unordered lists*. An unordered list begins with `ul` and ends with `/ul`.
- An unordered list may contain any number of *list items*. A list item begins with `li` and ends with `/li`.

- Within any list item there can occur any number of `a` tags. Within the body, but outside any unordered lists, there can occur any number of `a` tags. Whenever an `a` tag occurs, the next tag must be a `/a` tag.

D. Draw a DFA that accepts valid HTML and rejects invalid HTML, as just defined. To keep your drawing clear, please adopt the following convention. If a state q has no outgoing transition arrow for a symbol a, then it is understood that the machine rejects its input when it is in state q and sees symbol a.

In this next problem, we are going to implement DFAs in Python 3.x. We need to agree on a format. Let's say that a DFA in Python is a tuple (an immutable list) of five objects:

- Q: Tuple of arbitrary (but hashable) Python objects.
- Σ: Tuple of characters.
- q_0: Python object. Must be one of the elements of Q.
- F: Tuple of Python objects. Must be subset of Q, not necessarily in the same order.
- δ: Dictionary. Each key is a 2-tuple of the form (q, a), where $q \in Q$ and $a \in \Sigma$. Each value is an element of Q. If any pair $(q, a) \in Q \times \Sigma$ is missing from the dictionary's keys, then it is understood that the DFA rejects when it must transition out of state q via symbol a.

E. Write a function `dfa` that takes two inputs — a DFA M and an input string w — runs M on w, and outputs `True` if M accepts and `False` if M rejects. You may assume that M is a well-formed DFA and that the alphabets of M and w match. Print your solution on paper, for submission with the rest of the assignment.

By the way, my implementation is five lines long. Here is some code to help you test your implementation. This M is the "divisible by 3" example from Day 02 of the course.

```
if __name__ == "__main__":
    delta = {("q0", "0"):"q1", ("q0", "1"):"q0",
        ("q1", "0"):"q2", ("q1", "1"):"q1",
        ("q2", "0"):"q0", ("q2", "1"):"q2"}
    m = (("q0", "q1", "q2"), ("0", "1"), "q0", ("q0",), delta)
    for w in ["", "0", "000", "000000", "100", "11", "0111001111"]:
        print(w, dfa(m, w))
```