Problems A-C will be handed in on paper. Problem D does not need to be handed in, although you should solve the problem anyway. Problem E will be handed in electronically.

Problem 1.45 in our textbook describes the quotient operation on languages. This operation is quite confusing. It will recur a few times during the course. Problem A deals with a strange special case. Problem B gives you an idea of why it is a "quotient". Problem C is more difficult.

A. If $A = \Sigma^*$, then what is A/B? [Hint: There are two possibilities, depending on B.]

B. Find an example of two infinite languages A and B such that $(A/B)B = A$. Also find an example of two languages A and B such that $(A/B)B \neq A$.

C. Do problem 1.45. Your solution will probably involve some kind of description of a DFA or NFA for A/B. Your solution should explain why the automaton accepts all strings in A/B and why it rejects all strings that are not in A/B.

D. Do problem 1.16b in our textbook. Remember that you are not required to hand in your solution. However, it is good practice for Problem E and for exams.

In our Day 02 assignment, I gave you a detailed specification for how to represent a DFA in Python. Now let's make a similar spec for NFAs without ϵ-transitions. In fact, the only difference is this part:

- δ: Dictionary. Each key is a 2-tuple of the form (q, a), where $q \in Q$ and $a \in \Sigma$. Each value is a tuple of Python objects, that is a subset of Q. If any pair $(q, a) \in Q \times \Sigma$ is missing from the dictionary's keys, then it is understood that the associated value is the empty tuple.

E. Write a Python function `dfaFromNFA`. This function takes as input an NFA without ϵ-transitions, as specified above. It returns an equivalent DFA, built using the power set construction. Feel free to use the two helper functions below. In the same file, write tests to demonstrate that your code works correctly. Comment your code.

Hand in your code for Problem E electronically, including its tests, in a single file called `day03.py`. To do this, mount the COURSES file server (instructions are linked near the top of

our course web page), find your CS 254 handin folder, and place your code file there. Note well: Once you place a file in this folder, you will not be able to edit or delete it.

Let me tell you about a pitfall, that you might not see immediately. Your DFA states are going to be sets of NFA states. These sets will be represented as tuples (or lists). Tuples are ordered, whereas sets are not. For example, the tuples `(1, 3, 8)` and `(8, 1, 3)` represent the same set, but Python does not regard them as equal. So you will need to pay attention to the order of elements within certain sets. That's why the documentation for these functions talks about order so much.

```
# Returns the intersection of two sets X and Y.
# Input and output sets are tuples.
# Assumes that the elements are "simple enough to be compared naively".
# The output is ordered to match the order in X.
def intersection(X, Y):
    res = []
    for x in X:
        if x in Y:
            res.append(x)
    return tuple(res)

# Returns the power set of the set X.
# Input and output sets and subsets are tuples.
# The output set itself is not sorted in any way, but
# each subset within the output set is sorted to match the order in X.
def powerSet(X):
    if len(X) == 0:
        return ((),)
    else:
        withoutFirst = powerSet(X[1:])
        withFirst = [(X[0],) + Y for Y in withoutFirst]
        return tuple(withFirst) + withoutFirst
```