Problems A-D will be handed in on paper. Problem E will be handed in electronically: Mount the COURSES file server, 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.

A. 1.16b

B. 1.32

C. 1.45. This problem is significantly harder than the others, I think. You may want to use the result of 1.31 (which you do not have to prove).

D. In our Day 02 assignment, I gave you a detailed specification for how to represent a DFA in Python. How would an NFA be specified? You do not need to handle ϵ-transitions. Try to make your NFA spec as similar to my DFA spec as possible, and just tell me the parts that are different.

There are two versions of Problem E. They are worth equal credit. Choose one of them to hand in. Doing both earns you no extra credit. Write tests to demonstrate that your code works correctly. Comment your code. Hand in your code electronically, including its tests, in a single file called `day02.py`.

EA. Write Python functions `dfaIntersection`, `dfaUnion`, and `dfaComplement`. These functions take in one or two DFAs (as appropriate), and return a single DFA for the intersection, union, or complement of the languages of the input DFAs.

EB. Write a Python function `dfaFromNFA`. This function takes as input an NFA without ϵ-transitions, as specified in Problem D above. It returns an equivalent DFA, built using the power set construction. You may find the helper functions below useful.

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.

```
# 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
```