Programming Assignment #2
COMP 3200
Fall 2023
Prof. Stucki

Create a file pa2.py containing two functions:

`nfa_eclosure(M, s)`

> Returns the ε-closure of state `s` in the NFA `M`, which is the set of all states reachable from `s` by zero or more ε-transitions. Note that the ε-closure of `s` always includes `s` itself.

`nfa_accepts(M, A, x)`

> Takes an NFA `M`, a set `A` of accepting states, and a string `x`.
> Returns `True` if `M` accepts `x` and `False` otherwise. Here's a sample interaction:

```
>>> M = [{'':{1}, 'a':{2}},      # state 0
         {'':{2}},               # state 1
         {'b':{1,3}},            # state 2
         {'a':{1,3}}]            # state 3
>>> A = {1,3}
>>> nfa_eclosure(M, 0)
set([0, 1, 2])
>>> nfa_accepts(M, A, '')
True
>>> nfa_accepts(M, A, 'babba')
True
>>> nfa_accepts(M, A, 'aaba')
False
```

Of course, your procedure must work with *any* NFA, not just this one. The NFA is represented as a list of dictionaries. Each element of the list corresponds to a state, so the states are numbered starting from 0. State 0 is the start state. The dictionary for a state defines the edges leaving that state: a character (string of length 1) or ε (string of length 0) is mapped to the set of neighboring states reachable by an edge with that label. Edges not present in the NFA have no entries at all.

The string `x` can contain any characters, including ones not mentioned in the NFA, and it may be empty.

To simulate the NFA, just start with `{0}` as the current set of states, then (1) compute the ε-closure of all the states in the current set, then (2) find all states reachable from the current set using one edge labelled with the next character in the string. Once you reach the end of the string, compute the ε-closure of the states one last time, and see if that final set of states contains any accepting states. If at any point the current set of states is empty, then you can reject the string.

Note: it is faster and easier if, inside `nfa_accepts`, you compute the ε-closure for all the states in `M` once and store the results in a local list.