In this first problem, you will write a function (in the language of your choice, but preferably Python) that simulates a given DFA on a given input. For this to work, we need to agree on a uniform way of describing DFAs. Let us adopt the convention that the states of a DFA are numbered $q_0, q_1, \ldots, q_{n-1}$, where n is the number of states and q_0 is not necessarily the start state. Similarly, let's agree that the symbols are numbered $a_0, a_1, \ldots, a_{m-1}$, where m is the size of the alphabet. Then one can uniquely specify a DFA by the following data.

- A list `delta` of length n, such that each entry of `delta` is a list of m integers from $\{0, 1, \ldots, n-1\}$. This `delta` is the table for the DFA's transition function δ. Namely, if the machine is in the ith state and sees the jth symbol, then it transitions to the `delta[i][j]`th state.
- A number `s` belonging to the set $\{0, 1, \ldots, n-1\}$, to indicate the start state.
- A list `F` of numbers from $\{0, 1, \ldots, n-1\}$, with no repeats, to indicate the final states.

We have not specified the set Q of states or the alphabet Σ, but we know how big each is from the structure of `delta`, and we know how to compute with them because the states and symbols are uniquely identified as numbers. Thus `delta`, `s`, and `F` essentially encode the DFA.

The input string w to a DFA on m symbols shall be represented as a list of numbers from the set $\{0, 1, \ldots, m-1\}$. For example, the string $w = a_3 a_3 a_1 a_0 a_5$ shall be represented as `[3, 3, 1, 0, 5]`. Finally, our DFA function will output `True` or `False` (or whatever the appropriate analogue is, in your chosen language) rather than Accept or Reject.

A. Write a function `dfa` that simulates a given DFA on a given input. That is, `dfa` takes in four arguments — `delta`, `s`, `F`, and the list `w` — and outputs either `True` or `False`, according to whether the DFA described by `delta`, `s`, `F` would accept or reject that input. Include a short example transcript showing that your code works. (Print out and hand in on paper.)

B. 1.16b

C. 1.32

D. 1.38. Explain how to construct, for any arbitrary all-NFA, an equivalent DFA. List Q, Σ, δ, q_0, and F, rather than drawing a diagram. You need not prove that your construction works.

E. 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).