# Q3

For this question, I did the minimisation using the Myhill-Nerode theorem. This minimizing approach is also known as the Table filling method.

Our basic premise is that non-distinguishable vertices can be merged together, and a vertex is distinguishable if, after accepting the same input (for any input), they either get to the end state or not, but for any input, it should never happen that one of them does and one does not.

The steps I have implemented are:

1. First, I created the pairs of all the states involved in the given DFA.
2. This was done in a 2-D table after which I removed the top half.
3. After that, I marked all the pairs (Qa, Qb) such that Qa is the Final state and Qb is a Non-Final State.
4. If there was an unmarked pair (Qa, Qb) such that $\delta$(Qa,x) and $\delta$(Qb,x) is marked, then I marked (Qa, Qb). Here x is an input symbol. I repeated this step until no more marking can be made.
5. If two continuous iterations of the table-filling were the same, I stopped the marking.
6. Finally, I combined all the unmarked pairs and made them a single state in the minimized DFA.

# Q4

For REGEX to NFA, we began by adding brackets around the supplied regex, and instead of implicit concatenation, I inserted a backslash there, so that the final regex is now wrapped in brackets and contains an operator for each operator. As I worked through the concept, I discovered that my problem was simply a variant on the basic PEMDAS rule. I would use letters, concatenation, and union instead of numbers, multiplication, and addition. I started looking for algorithms that use this sort of logic and ultimately came upon the Shunting-Yard Algorithm.

After that I followed the steps in the link below to convert infix to postfix:
https://gregorycernera.medium.com/converting-regular-expressions-to-postfix-notation-with-the-shunting-yard-algorithm-63d22ea1cf88

This simplified the calculation since I now only care about the last two NFAs on the stack that we will be utilising at any given time. So we go to the first operator for the post-fix phrase. If it's a star, we take the final NFA on the stack and use the star operator, which is already a specified step given any NFA by adding a few states and epsilon transitions. In addition, for binary operations like union and concatenation, I pull out the last two NFAs and do the operation on them, implementing the steps in:
https://medium.com/swlh/visualizing-thompsons-construction-algorithm-for-nfas-step-by-step-f92ef378581b

We also needed the usage of an NFA class to store each state of the NFA, and eventually, only one NFA will be left in the stack, which is the solution. Epsilon transitions are handled by the 26th column, which comes immediately after z.