1. Given all the recent hype about ChatGPT, you have decided to start doing work in natural language processing. But before trying to make a high-powered LLM, you've decided to see what you can do with some simpler tools: regular languages and NFAs. Consider the following input alphabet: {the, salty, pirate, pirates, plunder, plunders, ship}.

   (a) Construct a DFA which accepts the following language: {the pirate plunders the ship, the pirates plunder the ship}

   (b) Construct a DFA accepting the language {the pirate plunders the ship, the pirates plunder the ship, the salty pirate plunders the ship, the salty pirates plunder the ship, the salty salty pirate plunders the ship, the salty salty pirates plunder the ship,. . . }

   (c) Construct an NFA accepting the sentences accepted by the automaton in (b) plus those formed by conjoining such sentences with and, e.g. the salty pirate plunders the ship and the salty salty pirates plunder the ship and the pirate plunders the ship

   (d) Construct regular expressions for the automata in (a), (b), (c).

   Solution:

   (a) (the pirate plunders the ship+the pirates plunder the ship)

   (b) the (salty)* pirate(λ+ s) plunder(λ+ s) the ship

   (e) Suppose our alphabet is {the, pirate, pirates, admiral, admirals, sailor, sailors, plunder, plunders, pursue, pursues, hate, hates, ship}

   Consider sentences like the following:

   the pirate plunders the ship

   the pirate the admiral pursues plunders the ship

   the pirate the admiral the sailor hates pursues plunders the ship (i.e. the sailor hates the admiral who pursues the pirate who plunders the ship)

   the pirate plunders the ship

   the pirates the admirals hunt plunders the ship

   the pirates the admirals the sailors hate hunt plunders the ship

   the admirals the sailors pursue

   i.e. sentences of the form (the + [Noun])n + [Transitive Verb]n + the ship

   Is the language that accepts these sentence regular? Why or why not?

2. Prove that $L(r^{**}) = L(r^{*})$

   Proof: ■

3. Show how, given any NFA, you can construct an equivalent one which has no transition arrows into the original state

4. Construct DFAs for the following regular languages over $\{0, 1\}$. Use as few states a possible.

   (a) The set of all strings containing a total of n 1's where $n \equiv 1 \pmod 3$.

   (b) The set of all strings containing exactly two 1's.

   (c) The set of all strings containing a block of at least three consecutive ones:

   (d) $1(010^*1)(0|1)^*$