1-3. Let us consider an automaton for recognizing words starting with a pound sign #. For simplicity, let us only consider letters m and n. This automaton has 3 states: start (s), correct (c), and sink (si). Start is the starting state, correct is the only final state. The transitions are as follows:

- from the state s, symbol # leads to c, symbols m and n lead to si;
- from c, any symbol leads to c; and
- from si, every symbol leads to si.

1. Trace, step-by-step, how this finite automaton will check whether the following two strings start with #:
   - the word #mn (which this automaton should accept)
   - the word m# (which this automaton should reject).

2. Use the above tracing to find the parts x, y, and z of the word #m#n corresponding to the Pumping Lemma. Check that the "pumped" word xyyz will also be accepted by this automaton.

3. Write down the tuple \(<Q, \Sigma, \delta, q_0, F>\) corresponding to this automaton:
   - Q is the set of all the states,
   - \(\Sigma\) is the alphabet, i.e., the set of all the symbols that this automaton can encounter;
   - \(\delta: Q \times \Sigma \rightarrow Q\) is the function that describes, for each state \(q\) and for each symbol \(s\), the state \(\delta(q, s)\) to which the automaton that was originally in the state \(q\) moves when it sees the symbol \(s\) (you do not need to describe all possible transitions this way, just describe two of them);
   - \(q_0\) is the starting state, and
   - \(F\) is the set of all final states.
1. $\#mn^k\| \quad \text{not accepted}$

2. $x\ y\ y\ \Rightarrow\ #\ m\ n\#$

\[ s\ c\ c\ c\ c\ c\ c \]

*Pumped word*

\[ #m\ m\ m\ n \]

\[ s\ c\ c\ c\ c\ c\ c\ c\]

$\text{accepted}$
3. \( \langle Q, \Sigma, \delta, q_0, F \rangle \)

\( Q = \{s, c, si\} \)

\( \Sigma = \{\#, m, n\} \)

\( \delta = \begin{array}{ccc}
\# & c & c & s_i \\
 s & c & c & s_i \\
 m & s_i & c & s_i \\
n & s_i & c & s_i \\
\end{array} \)

\( q_0 = s \)

\( F = \{\Sigma, c\} \)
4-5. Let A be the automaton described in Problem 1-3. Let B be an automaton that accepts all the strings that contain only # and m but not any other symbols. This automaton has two states: the start state which is also a final state, and the sink state. The transitions are as follows:

- from the start state, # or m leads back to the start state, n leads to the sink;
- from the sink state, any symbol leads back to the sink.

4. Use the algorithm that we had in class to describe the following two new automata:
- the automaton that recognizes the union A U B of the two corresponding languages, and
- the automaton that recognizes the intersection of the languages A and B.

5. Test these two new automata step-by-step on the following words:
- test the union automaton on the example of the word m# (that it should accept);
- test the intersection automaton on the example of the words #mn (that it should reject).
5. \[ m \# \]

Accepted by Union

\[
\begin{align*}
& S_{A} B \quad S_{A} S_{B} \quad S_{A} S_{B} \\
& S_{A} S_{B} \quad C_{A} S_{B} \quad C_{A} S_{B} \quad C_{A} S_{B} \\
& \quad \quad \text{rejected by intersection}
\end{align*}
\]
6. Use the general algorithm that we learned in class to design a non-deterministic finite automaton that recognizes the language \((m \cup n \cup \#)^*\):

- first, describe the automata for recognizing \#, \(m\), and \(n\);
- then, combine them into the automata for recognizing the union \(m \cup n\), \((m \cup n) \cup \#\), and the Kleene star \((m \cup n \cup \#)^*\);
- finally, combine the automaton for \# with an automaton for the Kleene star into an automaton for recognizing the desired composition of the two languages.

7. Use the general algorithm to transform the resulting non-deterministic finite automaton into a deterministic one.
6. Use the general algorithm that we learned in class to design a non-deterministic finite automaton that recognizes the language #(m U n U #)*:

- first, describe the automata for recognizing #, m, and n;
- then, combine them into the automata for recognizing the union m U n, (m U n) U #, and the Kleene star (m U n U #)*;
- finally, combine the automaton for # with an automaton for the Kleene star into an automaton for recognizing the desired composition of the two languages.

7. Use the general algorithm to transform the resulting non-deterministic finite automaton into a deterministic one.
8-9. Use a general algorithm to transform the finite automaton B from Problem 4-5 into the corresponding regular expression.

8-9. B.

\[
R_{ij} = R_{ij} \cup (R_{ik} R_{kj}^* R_{kj})
\]

\[
R_{s+t}s = R_{s+t}s \cup (R_{s+t} R_{s+t}^* R_{s+t} s)
\]

\[
\emptyset \cup (\emptyset \ldots)
\]

\[
R_{s+t}^e = R_{s+t}^e \cup (R_{s+t}^e R_{s+t}^e R_{s+t}^e)
\]

\[
\emptyset \cup (\emptyset \ldots)
\]

\[
R_{s+t}^s = R_{s+t}^s \cup (R_{s+t}^s R_{s+t}^s R_{s+t}^s)
\]

\[
(\# Um) \cup (\# Um) \cup (\# Um)
\]
\[ R_{stf} = R_{stf} \cup (R_{sts} R_{ss} R_{sf}) \]
\[ \emptyset \cup (\emptyset \cup (#um)^* \emptyset) \]
\[ \emptyset \cup (#um)^* \]
\[ (#um)^* \]
10. Prove that the language $L$ of all the words that have equal number of m's and n's is not regular.

$L = \{ \text{Equal num of m's and n's} \}$

**Pumping lemma**

$\exists p \forall w (\text{len}(w) \geq p \Rightarrow x, y, z (w = x y z \land \text{len}(y) > 0 \land \text{len}(x y) \leq p \land \forall i (x y^i z \in L))$.

Assume $L$ is regular.

Let's take $w = m^p n^p = \underbrace{m m \ldots m}_p \underbrace{n n \ldots n}_p$

$\text{len}(w) = 2p \geq p$

So, we can represent $w$ as $x y z$. Since $\text{len}(x y) \leq p$, $y$ is among the first $p$ symbols, so $y$ contains only m's. So, when we go from $xyz$ to $x y y z$, we have more m's than n's. So $x y y z$ is not in the language $L$. But by Pumping lemma, $x y y z$ is in the language $L$. Therefore, there is a contradiction so the language is not regular.