1. For each of the following languages assume that the alphabet is \{0, 1\}. Give a regular expression that describes that language, and briefly argue why your expression is correct.

(a) All strings that that end in 01 and contain 000 as a substring.

**Solution:** \((0 + 1)^* 000 (0 + 1)^* 01 + (0 + 1)^* 001\)

The first part is the case where the 000 substring and the 01 at the end occur separately. The second part covers the case where they overlap, i.e. when the 001 at the end contributes to both the 000 substring and the 01 termination. 

(b) All strings that do not contain the subsequence 101.

**Solution:** \(0^* 1^* 0^*\)

To not contain the subsequence 101, there can only be one run of 1s if we were to have any 1 at all. Before and after this run of 1, we may have any number of 0s.

(c) A finite language \(L = \{w_1, w_2, \ldots, w_k\}\) where each \(w_i\) is a binary string.

**Solution:** If \(L = \emptyset\) then the regular expression is \(\emptyset\) otherwise if \(L = \{w_1, w_2, \ldots, w_k\}\) for some \(k \geq 1\), the regular expression is \(w_1 + w_2 + \ldots + w_k\).

(d) The complement \(\bar{L}\) of a finite language \(L = \{w_1, w_2, \ldots, w_k\}\). You may want to consider the parameter \(h = \max_{1 \leq i \leq k} |w_i|\), the length of the longest string in \(L\).

**Solution:** If \(L = \emptyset\) then \(\bar{L} = \Sigma^*\) and hence a regular expression for \(\bar{L}\) is \((0 + 1)^*\).

If \(L \neq \emptyset\) but finite let \(h = \max_{w \in L} |w|\). Note that \(h \geq 0\) (and can be 0 if \(L = \emptyset\)). Consider \(\bar{L}_h = \{w \notin L \mid |w| \leq h\}\). We can see that \(\bar{L} = \bar{L}_h \cup \{w \mid |w| > h\}\). Since \(\bar{L}_h\) is a finite language we can construct regular expression for it as in the preceding part. Let \(r_h\) be the regular expression for \(\bar{L}_h\). Then the regular expression for \(\bar{L}\) is \(r_h + (0 + 1)^{h+1} (0 + 1)^*\) since \((0 + 1)^{h+1} (0 + 1)^*\) is a regular expression for all strings with length at least \(h + 1\).

**Rubric:**

+ 2.5: Part (a) correct regex.
+ 2.5: Part (b) correct regex.
+ 2.5: Part (c) correct regex.
+ 2.5: Part (d) correct regex.
2. (a) Let $L$ be the set of all strings in $\{0,1\}^*$ that end in $10$. Describe a DFA over the alphabet $\Sigma = \{0,1\}$ that accepts the language $L$. Argue that your machine accepts every string in $L$ and nothing else, by explaining what each state in your DFA means.

You may either draw the DFA or describe it formally, but the states $Q$, the start state $s$, the accepting states $A$, and the transition function $\delta$ must be clearly specified.

**Solution:** Since this is a relatively simple DFA, we can opt to draw it out explicitly as follows.

![DFA Diagram]

<table>
<thead>
<tr>
<th>State</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>seen no symbols yet, or only read 0 or last two symbols read are 00.</td>
</tr>
<tr>
<td>1</td>
<td>Last symbol read was 1.</td>
</tr>
<tr>
<td>10</td>
<td>Last two symbols read are 10.</td>
</tr>
</tbody>
</table>

(b) Now consider some finite alphabet $\Sigma$ and a string $x = a_1a_2\ldots a_k$ of length $k$ where $a_1,a_2,\ldots, a_k \in \Sigma$. Let $L_x = \{w \in \Sigma^* \mid w \text{ ends in string } x\}$. Describe in formal tuple notation a DFA $M$ that accepts $L_x$. Do not try to optimize number of states. How many states does your DFA have as a function of $k$?

**Note:** We changed $s$ to $x$ in the problem description above to avoid confusion with the start state.

**Solution:** $M = (Q, \Sigma, s, \delta, A)$ for $L$ as follows.

$$
Q := \{q_w \mid w \in \Sigma^*, 0 \leq |w| \leq k\} \\
s := q_\varepsilon \\
A := \{q_x\} \\
\delta(q_w, a) := \begin{cases} \\
q_w \cdot a & \text{if } |w| < k \\
q_z \cdot a & \text{if } |w| = k, \text{ where } w = bz \text{ for } b \in \Sigma, z \in \Sigma^* \\
\end{cases}
$$

For each string $w$ of length at most $k$ we have a state $q_w$, hence the total number of states is $\sum_{i=0}^{k} |\Sigma|^i$ which is exponential in $k$ if $|\Sigma| \geq 2$. The state $q_w$ has the following meaning: if $|w| < k$ then $w$ is the string seen so far. If $|w| = k$ then $w$ is the string of the last $k$ characters seen so far. It is not hard to check that the transition function maintains this claimed behavior of $M$. The only accept state is $q_x$ since we want to end in $x$. ■
Solution: Here is an alternate solution for a DFA that has only $k+1$ states $q_0, q_1, \ldots, q_k$. The meaning of $q_i$ is that the string seen so far ends in $a_1a_2\ldots a_i$ and moreover there is no $j > i$ such that the string seen so far ends in $a_1a_2\ldots a_j$. No that $q_0$ corresponds to $\varepsilon$.

$$Q := \{q_i \mid 0 \leq i \leq k\}$$
$$s := q_0$$
$$A := \{q_k\}$$
$$\delta(q_i, a) := q_j \text{ where } j = \max\{h \mid a_1a_2\ldots a_h = a_{i+2-h}a_i\}$$

Intuitively, $j$ is the length of the longest prefix of $x$ that is also a suffix of $a_1a_2\ldots a_i a$.

It is harder to convince yourself of the correctness of this but try to prove it by induction.

---

(c) Not to submit. Let $L = \{w_1, w_2, \ldots, w_k\}$ be a finite language. Describe a DFA $M$ that accepts $L$. Describe the states and justify why your DFA accepts $L$.

Solution: Let $h$ be the length of the longest string in $L$. We create a DFA $M = (Q, \Sigma, s, A, \delta)$ for $L$ as follows.

$$Q := \{q_w \mid 0 \leq |w| \leq h\} \cup \{\text{Fail}\}$$
$$s := q_{\varepsilon}$$
$$A := \{q_w \mid w \in L\}$$
$$\delta(q_w, a) := \begin{cases} q_w a & \text{if } |w| < h \\ \text{Fail} & \text{otherwise} \end{cases}$$
$$\delta(\text{Fail}, a) := \text{Fail} \quad \forall a \in \Sigma$$

For each string $w$ of length at most $h$ we have a state $q_w$ and also an additional state Fail, hence the total number of states is $1 + \sum_{i=0}^{h} |\Sigma|^i$ which is exponential in $h$ if $|\Sigma| \geq 2$. Each state represents the string read thus far. The fail state is reached when we have read more than $h$ symbols since such a string cannot be in $L$. The accept states correspond precisely to the strings $w \in L$. ■

Rubric:

• 5 points: Part (a).
  + 3: Correct DFA.
  + 2: Correct explanation.

• 5 points: Part (b).
  + 3: Correct DFA.
  + 2: Correct explanation.
3. Let $L_1$, $L_2$, and $L_3$ be regular languages over $\Sigma$ accepted by DFAs $M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$, $M_2 = (Q_2, \Sigma, \delta_2, s_2, A_2)$, and $M_3 = (Q_3, \Sigma, \delta_3, s_3, A_3)$, respectively.

(a) Describe a DFA $M = (Q, \Sigma, \delta, s, A)$ in terms of $M_1, M_2,$ and $M_3$ that accepts $L = \{ w \mid w \text{ is in at least two of } \{ L_1, L_2, L_3 \} \}$. Formally specify the components $Q, \delta, s,$ and $A$ for $M$ in terms of the components of $M_1, M_2,$ and $M_3$.

Solution: Informally, we will use the product construction to make our DFA, where the accept states are those where at least two of the component states are accept states in their respective component DFAs. This corresponds to the following formal description

$$Q := Q_1 \times Q_2 \times Q_3,$$

$$s := (s_1, s_2, s_3),$$

$$A := \{(q_1, q_2, q_3) \in Q \mid \text{At least two of: } q_1 \in A_1, q_2 \in A_2, q_3 \in A_3\}$$

$$:=(A_1 \times A_2 \times Q_3) \cup (A_1 \times Q_2 \times A_3) \cup (Q_1 \times A_2 \times A_3),$$

$$\delta((q_1, q_2, q_3), a) := (\delta_1(q_1, a), \delta_2(q_2, a), \delta_3(q_3, a)).$$

(b) Prove by induction your solution is correct.

Solution: We begin by defining the extended transition function (similar to the definition in the notes)

$$\delta^*(q, w) := \begin{cases} q & \text{if } w = \epsilon, \\ \delta^*(\delta(q, a), x) & \text{if } w = ax. \end{cases}$$

We let $\delta^*$ denote the extended transition function for $M$ and $\delta_1^*, \delta_2^*, \delta_3^*$ as the extended transition functions for $M_1, M_2,$ and $M_3$ respectively. We now prove the following claim for use later

Claim 1. Let $(q_1, q_2, q_3) \in Q$. Then for all $w \in \Sigma^*$

$$\delta^*((q_1, q_2, q_3), w) = (\delta_1^*(q_1, w), \delta_2^*(q_2, w), \delta_3^*(q_3, w)).$$

Proof: We proceed by induction on $|w|$. For our base case, $|w| = 0$, so $w = \epsilon$. Thus, by the definition of $\delta^*$, we have that $\delta^*((q_1, q_2, q_3), \epsilon) = (q_1, q_2, q_3) = (\delta_1^*(q_1, \epsilon), \delta_2^*(q_2, \epsilon), \delta_3^*(q_3, \epsilon)).$

We now proceed to our inductive step, where we assume that $|w| > 0$ and that the claim holds for every string $x$ such that $|x| < |w|$. Since $|w| > 0$ we may assume that $w = ax$ for some symbol $a$ and string $x$ where $|x| < |w|$. Thus,

$$\begin{align*}
\delta^*((q_1, q_2, q_3), ax) &= \delta^*(\delta((q_1, q_2, q_3), a), x) \\
&= \delta^*(\delta_1(q_1, a), \delta_2(q_2, a), \delta_3(q_3, a)), x) \\
&= (\delta_1^*(\delta_1(q_1, a), x), \delta_2^*(\delta_2(q_2, a), x), \delta_3^*(\delta_3(q_3, a), x)) \\
&= (\delta_1^*(q_1, ax), \delta_2^*(q_2, ax), \delta_3^*(q_3, ax)) \quad \text{by definitions of } \delta_1^*, \delta_2^*, \delta_3^*.
\end{align*}$$

This completes our induction and proves the claim. \qed
Now we show that our DFA $M$ accepts the language $L$. Consider some string $w = a_1 \ldots a_n$ accepted by our new DFA $M$. If we let $s = (s_1, s_2, s_3)$ be the starting state of $M$, then $\delta^*(s, w) \in A$. Applying the claim, $\delta^*(s, w) = (\delta^*_1(s_1, w), \delta^*_2(s_2, w), \delta^*_3(s_3, w))$, and by the definition of $A$, then at least two of $\delta^*_1(s_1, w) \in A_1, \delta^*_2(s_2, w) \in A_2, \delta^*_3(s_3, w) \in A_3$. Without loss of generality, we write that that $\delta^*_1(s_1, w) \in A_1, \delta^*_2(s_2, w) \in A_2$. Thus, $w \in L_1, L_2$ (as $w$ is accepted by $M_1$ and $M_2$), and therefore $w \in L$ (as $w$ is in at least two of $\{L_1, L_2, L_3\}$).

For the reverse direction, take some $w \in L$, and so without loss of generality, we have that $w \in L_1, L_2$. Then, $\delta^*_1(s_1, w) \in A_1, \delta^*_2(s_2, w) \in A_2$, and by our claim, $(\delta^*_1(s_1, w), \delta^*_2(s_2, w), \delta^*_3(s_3, w)) = \delta^*((s_1, s_2, s_3), w)$, meaning that $\delta^*((s_1, s_2, s_3), w) \in A$ from the definition of $A$. Therefore, $w$ is accepted by $M$.

By both of our previous statements, $L = L(W)$ as desired.

**Rubric:**

- **Part (a), correct construction.** + 4 points
- **Part (b).** 6 points
  - **Standard induction rubric (included at end).** + 5 points
  - **Rest of proof (using result from induction).** + 1 point

**Rubric (induction):** For problems worth 10 points:

- **1 for explicitly considering an arbitrary object** + 1 point
- **2 for a valid strong induction hypothesis**
  - **Deadly Sin!** Automatic zero for stating a weak induction hypothesis, unless the rest of the proof is perfect.
  - **2 for explicit exhaustive case analysis**
    - No credit here if the case analysis omits an infinite number of objects. (For example: all odd-length palindromes.)
    - −1 if the case analysis omits an finite number of objects. (For example: the empty string.)
    - −1 for making the reader infer the case conditions. Spell them out!
    - No penalty if cases overlap (for example:
      - **1 for cases that do not invoke the inductive hypothesis ("base cases")**
        - No credit here if one or more "base cases" are missing.
      - **2 for correctly applying the stated inductive hypothesis**
        - No credit here for applying a different inductive hypothesis, even if that different inductive hypothesis would be valid.
      - **2 for other details in cases that invoke the inductive hypothesis ("inductive cases")**
        - No credit here if one or more "inductive cases" are missing.