1. For each of the following languages over the alphabet $\{0, 1\}$

(a) All strings except 010.

Solution: $(\varepsilon + 0 + 1)^2 + 1(0 + 1)^2 + (0 + 1)0(0 + 1) + (0 + 1)^2 1 + (0 + 1)^4(0 + 1)^*$

We will use the notation r^n to indicate n copies of the regular expression r concatenated together. This is a straightforward modification of the regular expression for "all strings except 000" that we discussed in lab. The expression $(\varepsilon + 0 + 1)^2$ refers to all binary strings of length two or less. The expression $(0 + 1)^4(0 + 1)^*$ represents all strings of length four or greater. The remaining expressions represent strings of length three with at least one character whose value is different from 010. ■

(b) All strings which end in 10 and contain 101 as a substring.

Solution: $(0 + 1)^*101(0 + (0 + 1)^*10)$

Such a string either ends in 1010 (the 101 and the 10 overlap for a character) or has a 101, followed by an arbitrary sequence of characters, followed by 10 (the 101 and 10 don't overlap) ■

(c) All strings in which every nonempty maximal substring of 1s is of length divisible by 3. For instance 0110 and 101110 are not in the language, while 11101111110 is.

Solution: $(0 + 111)^*$

Any sequence of 1s whose length is divisible by 3 must be in $(111)^*$, and thus any string where all sequences of 1s have length divisible by 3 is in $(0+111)^*$. Any string matching this regex will have the length of any maximal substring of 1s divisible by 3, because we can only match 1s three at a time. ■

(d) All strings that do not contain the substring 010

Solution: $1^*(0 + 111^*)^*1^*$

Divide the string into blocks. The initial and final blocks may consist of any number of 1s. Every other block consists of a 0 or a run of at least two 1s. Any run of 1s that appears between two 0s must be internal and so must have length at least two. ■

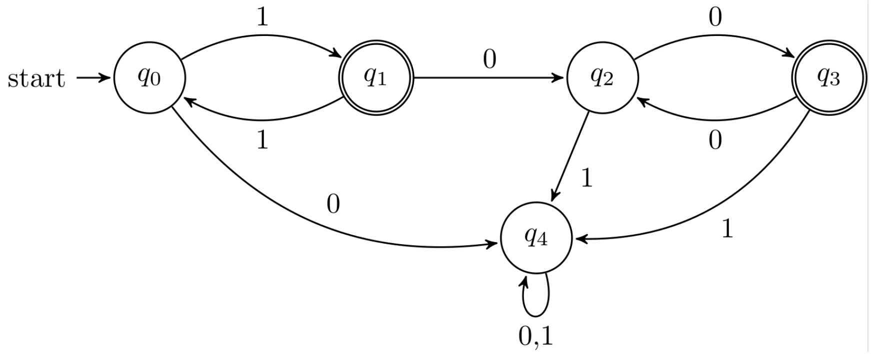
(e) All strings that do not contain the *subsequence* 100.

Solution: $0^*1^*(\varepsilon + 0)1^*$

Any number of 0s before the first 1 don't contribute towards a 100 subsequence. Once we've seen a 1, we can only see one more 0 total (otherwise we'd have the subsequence), but we can have any number of 1s before it or after it (and since we don't need to have a 0 after the 1, we use $0 + \varepsilon$). ■

Rubric: 10 points: 1 for each regular expression + 1 for each explanation These are not the only correct answers!

2. (a) **Solution:** The DFA is drawn below.



q_0: Strings with even number of 1s and no 0s
q_1: Strings with odd number of 1s and no 0s
q_2: Strings with odd number of 1s followed by odd number of 0s
q_3: Strings with odd number of 1s followed by even number of 0s
q_4: Dump state for all other strings.

Since words in L cannot contain 01 as a substring, after a 0 appears, a 1 cannot appear anymore. Hence, the words in the language would have all the 1s initially, followed by all the 0s, i.e., the words would be of the form $1^{2k_1+1}0^{2k_2}$. In the above DFA, all the transitions are drawn to obey the listed definitions of the states. Further, since states q_1 and q_3 are the states for strings of the form $1^{2k_1+1}0^{2k_2}$ by their state definitions, these states are made final states. ■

(b) **Solution:** $(11)^*1(00)^*$. As reasoned in 2a, the words in the language are of the form $1^{2k_1+1}0^{2k_2}$, and the above regex accepts only these words. ■

3. (a) **Solution:** We see that $L_1 - L_2 = L_1 \cap \overline{L_2}$. Therefore, $L = L_1 \cap \overline{L_2} \cap (L_4 \cup \overline{L_3}) = (L_1 \cap \overline{L_2} \cap L_4) \cup (L_1 \cap \overline{L_2} \cap \overline{L_3})$. We use the product construction that is similar to what we have seen in class. Here is the formal description of the components of $M = (Q, \Sigma, \delta, s, A)$.

- $Q = Q_1 \times Q_2 \times Q_3 \times Q_4$.
- $s = (s_1, s_2, s_3, s_4)$.
- $\delta : Q \times \Sigma \to Q$ is defined as follows: for each $(q_1, q_2, q_3, q_4) \in Q$ and $a \in \Sigma$, $\delta((q_1, q_2, q_3, q_4), a) = (\delta_1(q_1, a), \delta_2(q_2, a), \delta_3(q_3, a), \delta_4(q_4, a))$.
- $A = (A_1 \times (Q_2 - A_2) \times Q_3 \times A_4) \cup (A_1 \times (Q_2 - A_2) \times (Q_3 - A_3) \times Q_4)$. (Equivalently, $A = \{(q_1, q_2, q_3, q_4) \mid q_1 \in A_1 \wedge q_2 \in Q_2 - A_2 \wedge (q_3 \in Q_3 - A_3 \vee q_4 \in A_4)\}$.)

■

(b) **Solution:** We prove that the above construction is correct. As we did in lecture, we will use $\delta^*(q, w)$ to denote the state that the machine M will reach if started in state $q \in Q$ on input string w. Formally, we have

$$\delta^*(q, w) = \begin{cases} \epsilon & \text{if } w = \epsilon \\ \delta^*(\delta(q, a), x) & \text{if } w = ax \text{ for some symbol } a \in \Sigma \text{ and some string } x \end{cases}$$

The following lemma can be shown by induction on $|w|$ in exactly the same fashion as was done in lecture for the product construction.

Lemma 1. *For any string $w \in \Sigma^*$ and state $q = (q_1, q_2, q_3, q_4) \in Q$,*

$$\delta^*(q, w) = (\delta_1^*(q_1, w), \delta_2^*(q_2, w), \delta_3^*(q_3, w), \delta_4^*(q_4, w)).$$

Assuming the lemma we need to prove that $L(M) = L$.

We will first show that $L(M) \subseteq L$. Suppose $w \in L(M)$. This means that $\delta^*(s,w) \in A$, i.e., $\delta^*(s,w) \in (A_1 \times (Q_2 - A_2) \times Q_3 \times A_4) \cup (A_1 \times (Q_2 - A_2) \times (Q_3 - A_3) \times Q_4)$. Let $\delta^*(s,w) = (q_1,q_2,q_3,q_4)$. By our lemma, this implies that $\delta_1^*(s_1,w) = q_1$, $\delta_2^*(s_2,w) = q_2$, $\delta_3^*(s_3,w) = q_3$, and $\delta_4^*(s_4,w) = q_4$. If $\delta^*(s,w) \in A_1 \times (Q_2 - A_2) \times Q_3 \times A_4$, then we have $q_1 \in A_1, q_2 \in (Q_2 - A_2)$, and $q_4 \in A_4$. Therefore, $\delta_1^*(s_1,w) \in A_1$, $\delta_2^*(s_2,w) \in (Q_2 - A_2)$, and $\delta_4^*(s_4,q) \in A_4$. By definition of the languages, this implies that $w \in (L_1 \cap \overline{L_2} \cap L_4) \subseteq L$. Similarly, if $\delta^*(s,w) \in A_1 \times (Q_2 - A_2) \times (Q_3 - A_3) \times Q_4$, then we have $q_1 \in A_1, q_2 \in (Q_2 - A_2)$, and $q_3 \in Q_3 - A_3$. This implies that $w \in (L_1 \cap \overline{L_2} \cap \overline{L_3}) \subseteq L$. Hence, $w \in L$ in both cases.

We now prove that $L \subseteq L(M)$. Let $w \in L$. This means that $w \in (L_1 \cap \overline{L_2} \cap L_4) \cup (L_1 \cap \overline{L_2} \cap \overline{L_3})$. There are again two cases to consider and we will only consider one: $w \in L_1 \cap \overline{L_2} \cap L_4$. By definition of the languages, we know $\delta_1^*(s_1,w) \in A_1$, $\delta_2^*(s_2,w) \in Q_2 - A_2$, and $\delta_4^*(s_4,w) \in A_4$. By our lemma, $\delta^*(s,w) = (\delta_1^*(s_1,w), \delta_2^*(s_2,w), \delta_3^*(s_3,w), \delta_4^*(s_4,w)) \in (A_1 \times (Q_2 - A_2) \times Q_3 \times A_4)$. From our definition of A, $(A_1 \times (Q_2 - A_2) \times Q_3 \times A_4) \subseteq A$, which implies that $\delta^*(s,w) \in A$, and therefore $w \in L(M)$ by definition of the language.

We finish the argument by giving the proof of Lemma 1. The proof is by induction on the length of the input string. Let w be any string of length n.

- **Base case**: Consider $n = 0$. It follows that $w = \epsilon$. Then for any state $q = (q_1,q_2,q_3,q_4) \in Q$, we have

$$\begin{aligned}\delta^*((q_1,q_2,q_3,q_4),\epsilon) &= (q_1,q_2,q_3,q_4) \\ &= (\delta_1^*(q_1,\epsilon), \delta_2^*(q_2,\epsilon), \delta_3^*(q_3,\epsilon), \delta_4^*(q_3,\epsilon)).\end{aligned}$$

- **Inductive Hypothesis**: Assume that for any string w of length k with $k < n$, we have for all $(q_1,q_2,q_3,q_4) \in Q$,

$$\delta^*((q_1,q_2,q_3,q_4),w) = (\delta_1^*(q_1,w), \delta_2^*(q_2,w), \delta_3^*(q_3,w), \delta_4^*(q_4,w)).$$

- **Inductive Case**: Consider $n > 0$. Then $w = ax$ for some $a \in \Sigma$ and $x \in \Sigma^*$ where $|x| < n$. Let $q = (q_1,q_2,q_3,q_4) \in Q$. Then we have

$$\begin{aligned}\delta^*((q_1,q_2,q_3,q_4),w) &= \delta^*(\delta((q_1,q_2,q_3,q_4),a),x) \\ &\quad \text{(by definition of } \delta^*\text{)} \\ &= \delta^*((\delta_1(q_1,a), \delta_2(q_2,a), \delta_3(q_3,a), \delta_4(q_4,a)),x) \\ &\quad \text{(by definition of } \delta\text{)}\end{aligned}$$

Let $q_i' := \delta_i(q_i,a)$, for $1 \le i \le 4$. Therefore,

$$\begin{aligned}\delta^*((q_1,q_2,q_3,q_4),w) &= \delta^*((q_1',q_2',q_3',q_4'),x) \\ &= (\delta_1^*(q_1',x), \delta_2^*(q_2,x), \delta_3^*(q_3',x), \delta_4^*(q_4',x)) \\ &\quad \text{(by inductive hypothesis since } |x| < n\text{)} \\ &= (\delta_1^*(\delta(q_1,a),x), \delta_2^*(\delta(q_2,a),x), \delta_3^*(\delta(q_3,a),x), \delta_4^*(\delta(q_4,a),x)) \\ &= (\delta_1^*(q_1,ax), \delta_2^*(q_2,ax), \delta_3^*(q_3,ax), \delta_4^*(q_4,ax)) \\ &\quad \text{(by definition of } \delta_i^* \text{ for each } 1 \le i \le 4\text{)} \\ &= (\delta_1^*(q_1,w), \delta_2^*(q_2,w), \delta_3^*(q_3,w), \delta_4^*(q_4,w)) \\ &\quad \text{(since } w = ax\text{)}.\end{aligned}$$

This concludes the proof for our lemma and hence the correctness of the construction.

■

Rubric: 5 points for the construction: 1 point each for Q, δ, s and 2 points for A. 5 points for the proof. 2 points for the lemma, including 1 point for the base case and the inductive hypothesis and 1 point for the inductive case. 3 points for deducing the equivalence of the languages from the inductive part.