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

   (a) All strings except 101.

   Solution: $(\varepsilon + 0 + 1)^2 + 0(0 + 1)^2 + (0 + 1)1(0 + 1) + (0 + 1)^2 0 + (0 + 1)^4 (0 + 1)^*$
   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 101. ■

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

   Solution: $(0 + 1)^* 000(0 + 1)^* 01 + (0 + 1)^* 0001$
   String that include 000 and end with 01 are of two forms: there can be 000 substrings which are separate from the trailing 01, or they can overlap if the string ends with 0001. ■

   (c) All strings in which every nonempty maximal substring of 0s is of odd length. For instance 1001 is not in the language while 0100010 is.

   Solution: $(\varepsilon + 0(00)^*)(11^* 0(00)^*)^* 1^*$
   The expression $0(00)^*$ represents a block of 0s of odd length. It is important to make sure that before introducing any more blocks of 0s, there is at least one 1 to separate the blocks, hence the expression $(11^* 0(00)^*)^*$. The final 1^* is needed because there can be trailing 1s. ■

   (d) All strings that do not contain the substring 101.

   Solution: $0^*(1 + 000^*)^* 0^*$
   Divide the string into blocks. The initial and final blocks may consist of any number of 0s. Every other block consists of a 1 or a run of at least two 0s. Any run of 0s that appears between two 1s must be internal and so must have length at least two. ■

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

   Solution: $0^* 1^* 0^*$
   Every string that does not contain the subsequence 101 contains at most one run of 1s, possibly preceded and followed by runs of 0s. There cannot be a run of 1s after the first run of 0s that follows a run of 1s.

   ■

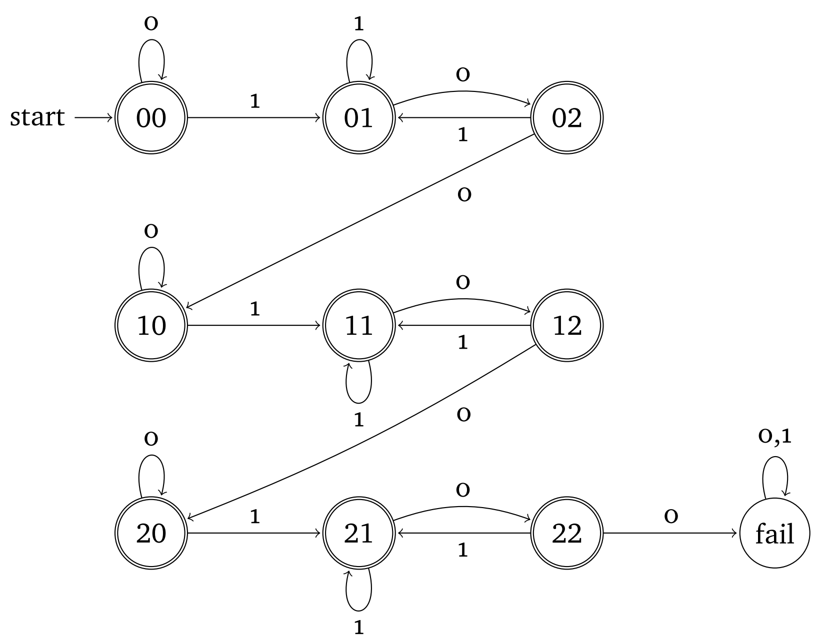
Rubric: 10 points: 1 for each regular expression + 1 for each explanation These are not the only correct answers!

2. Let L be the set of all strings in $\{0, 1\}^*$ that contain at most two occurrences of the substring 100.

(a) Describe a DFA that 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 δ must be clearly specified.

Solution: The following 10-state DFA accepts the language. Every state except *fail* is labeled with a pair of integers (i, j), where i is the number of times we have seen the substring 100, and j is the number of characters of the string 100 (up to 2) we have just read. The machine enters the *fail* state when it sees 100 for the third time.



■

Rubric: 5 points: standard DFA rubric (scaled)

(b) Give a regular expression for L, and briefly argue why the expression is correct.

Solution: $((0^*(1 + 10)^*100) + \epsilon) \cdot ((0^*(1 + 10)^*100) + \epsilon) \cdot (0^*(1 + 10)^*)$.

Let R be the subexpression $(0^*(1 + 10)^*)$. Note that R describes the set of all strings that do *not* contain the substring 100, because the subexpression $(1 + 10)^*$ ensures that any 1 can be followed by at most a single 0. Therefore, the subexpression $R' = (R \cdot 100) = (0^*(1 + 10)^*100)$ describes the set of all strings that end with 100 but do not otherwise contain the substring 100. We use two copies of $(R' + \epsilon)$ to capture the two potential occurrences of 100, with the option of ϵ included to account for the cases where less than two occurrences of 100 appear. We end the expression with one more copy of R; once the (up to) two instances of the substring 100 have appeared in any string $w \in L$, the remainder of w can be made up of any string not containing 100 as a substring.

■

Rubric: 5 points = 3 for regular expression + 2 for explanation.

3. Let L_1, L_2, and L_3 be regular languages over Σ accepted by DFAs $M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$, $M_2 = (Q_2, \Sigma, \delta_2, s_1, A_2)$, 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 $L = \{w \mid w \text{ is in exactly two of } \{L_1, L_2, L_3\}\}$. Formally specify the components Q, δ, s, and A for M in terms of the components of M_1, M_2, and M_3.

   (b) Prove by induction that your construction is correct.

Solution: Part (a)
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.

- $Q = Q_1 \times Q_2 \times Q_3$.
- $s = (s_1, s_2, s_3)$
- $\delta : Q \times \Sigma \to Q$ is defined as follows: $\delta((q_1, q_2, q_3), a) = (\delta_1(q_1, a), \delta_2(q_2, a), \delta_3(q_3, a))$ for each $(q_1, q_2, q_3) \in Q$ and $a \in \Sigma$.
- $A = (A_1 \times A_2 \times \bar{A}_3) \cup (A_1 \times \bar{A}_2 \times A_3) \cup (\bar{A}_1 \times A_2 \times A_3)$.
  In set builder notation, this can be written as
  $A = \{(q_1, q_2, q_3) \mid q_1 \in A_1, q_2 \in A_2, q_3 \in \bar{A}_3 \text{ or } q_1 \in A_1, q_2 \in \bar{A}_2, q_3 \in A_3 \text{ or } q_1 \in \bar{A}_1, q_2 \in A_2, q_3 \in A_3\}$.

Part (b) is the proof 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, this is given as

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

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 $q = (q_1, q_2, q_3) \in Q$,*

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

Assuming the lemma we need to prove that

$$L(M) = \{w \mid w \text{ is accepted by exactly two of } M_1, M_2, M_3\}.$$

Let $L = \{w \mid w \text{ is accepted by exactly two of } M_1, M_2, M_3\}$.

We will first show that $L(M) \subseteq L$. Suppose $w \in L(M)$. This means that $\delta^*(s, w) \in A$. Let $\delta^*(s, w) = q = (q_1, q_2, q_3)$. From our definition of A, exactly two of q_1, q_2, q_3 are in A_1, A_2, A_3. We will consider the case that $q_1 \in A_1, q_2 \in A_2, q_3 \notin A_3$. The other cases are similar. From our lemma, this implies that $\delta_1^*(s_1, w) = q_1, \delta_2^*(s_2, w) = q_2, \delta_3^*(s_3, w) = q_3$. Since $q_1 \in A_1, q_2 \in A_2, q_3 \notin A_3$, thus $w \in L(M_1)$ and $w \in L(M_2)$ but $w \notin L(M_3)$. Therefore $w \in L$.

We now prove that $L \subseteq L(M)$. Let $w \in L$. This means that exactly two of M_1, M_2, M_3 accept w. There are again three cases to consider and we will only consider one; w is accepted by M_1 and M_2 but not by M_3. This implies that $\delta_1^*(s_1, w) \in A_1$ and $\delta_2^*(s_2, w) \in A_2$ but $\delta_3^*(s_3, w) \notin A_3$. By our lemma, $\delta^*(s, w) = (\delta_1^*(s_1, w), \delta_2^*(s_2, w), \delta_3^*(s_3, w))$. From our definition of A, we have that $\delta^*(s, w) \in A$, and therefore $w \in L(M)$.

We finish the argument by proving Lemma 1. The proof is by induction on $|w|$.

- **Induction Hypothesis:** For all $n \geq 0$, for any string w of length n, for all $(q_1, q_2, q_3) \in Q$, we have that

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

- **Base case:** Let w be an arbitrary string of length 0. $w = \varepsilon$, since there is only one such string. Thus

$$\begin{aligned}
\delta^*((q_1,q_2,q_3),\epsilon) &= (q_1,q_2,q_3) && \text{by definition of } \delta^*(\cdot)\\
&= (\delta_1^*(q_1,\epsilon), \delta_2^*(q_2,\epsilon), \delta_3^*(q_3,\epsilon)) && \text{by definitions of } \delta_i^* \text{ for } 1 \leq i \leq 3
\end{aligned}$$

- **Inductive Step:** Let w be an arbitrary string of length $n > 0$. Assume inductive hypothesis holds for all strings u of length $< n$. Then $w = au$ for some $a \in \Sigma$ and $u \in \Sigma^*$ where $|u| < n$. Let (q_1,q_2,q_3) be an arbitrary state in Q. Let $q_i' = \delta_i(q_i, a)$. From the definition of $\delta(\cdot)$, we have that $\delta((q_1,q_2,q_3),a) = (q_1',q_2',q_3')$. Then we have

$$\begin{aligned}
\delta^*((q_1,q_2,q_3),w) &= \delta^*(\delta((q_1,q_2,q_3),a),u)\\
&\qquad \text{from the definition of } \delta^*(\cdot)\\
&= \delta^*((q_1',q_2',q_3'),u)\\
&\qquad \text{since } \delta((q_1,q_2,q_3),a) = (q_1',q_2',q_3')\\
&= (\delta_1^*(q_1',u), \delta_2^*(q_2',u), \delta_3^*(q_3',u))\\
&\qquad \text{by the inductive hypothesis since } |u| < n\\
&= (\delta_1^*(\delta_1(q_1,a),u), \delta_2^*(\delta_2(q_2,a),u), \delta_3^*(\delta_3(q_3,a),u))\\
&\qquad \text{since } q_i' = \delta_i(q_i,a) \text{ for } 1 \leq i \leq 3\\
&= (\delta_1^*(q_1,au), \delta_2^*(q_2,au), \delta_3^*(q_3,au))\\
&\qquad \text{by definition of } \delta_i^*(\cdot) \text{ for } 1 \leq i \leq 3\\
&= (\delta_1^*(q_1,w), \delta_2^*(q_2,w), \delta_3^*(q_3,w))\\
&\qquad \text{since } w = au.
\end{aligned}$$

We used in the second equation above that for $1 \leq i \leq 3$, $q_i' = \delta_i(q_i, a)$.

■

Rubric: 5 points for the construction: 1 point each for Q, δ, q_0 and 2 points for F. 5 points for the proof. 2 points for the inductive part and 3 points for deducing the equivalence of the languages from the inductive part.

Rubric (DFA design): For problems worth 10 points:

- 2 points for an unambiguous description of a DFA, including the states set Q, the start state s, the accepting states A, and the transition function δ.
  - **For drawings:** Use an arrow from nowhere to indicate s, and doubled circles to indicate accepting states A. If $A = \emptyset$, say so explicitly. If your drawing omits a reject state, say so explicitly. **Draw neatly!** If we can't read your solution, we can't give you credit for it,.
  - **For text descriptions:** You can describe the transition function either using a 2d array, using mathematical notation, or using an algorithm.
  - **For product constructions:** You must give a complete description of the states and transition functions of the DFAs you are combining (as either drawings or text), together with the accepting states of the product DFA.
- **Homework only:** 4 points for *briefly* and correctly explaining the purpose of each state *in English*. This is how you justify that your DFA is correct.
  - For product constructions, explaining the states in the factor DFAs is enough.
  - **Deadly Sin:** ("Declare your variables.") No credit for the problem if the English description is missing, *even if the DFA is correct*.
- 4 points for correctness. (8 points on exams, with all penalties doubled)
  - -1 for a single mistake: a single misdirected transition, a single missing or extra accept state, rejecting exactly one string that should be accepted, or accepting exactly one string that should be accepted.
  - -2 for incorrectly accepting/rejecting more than one but a finite number of strings.
  - -4 for incorrectly accepting/rejecting an infinite number of strings.
- DFA drawings with too many states may be penalized. DFA drawings with *significantly* too many states may get no credit at all.
- Half credit for describing an NFA when the problem asks for a DFA.