1. (a) Prove that the following languages are not regular by providing a fooling set. You need to provide an infinite set and also prove that it is a valid fooling set for the given language.

   i. $L = \{0^i 1^j 2^k \mid i + j = k + 1\}$.

   **Solution:** Let $F$ be the language $\mathtt{0}^*$.
   Let $x$ and $y$ be arbitrary strings in $F$.
   Then $x = \mathtt{0}^m$ and $y = \mathtt{0}^n$ for some non-negative integers $m \neq n$.
   Let $w = \mathtt{1}^{k+1-m}\mathtt{2}^k$.
   Then $xw = \mathtt{0}^m\mathtt{1}^{k+1-m}\mathtt{2}^k \in L$, , because $m + k + 1 - m = k + 1$.
   And $yw = \mathtt{0}^n\mathtt{1}^{k+1-m}\mathtt{2}^k \notin L$, because $n + k + 1 - m \neq k + 1$.
   Thus, $F$ is a fooling set for $L$.
   Because $F$ is infinite, $L$ cannot be regular. ■

   **Solution (concise):** For all non-negative integers $m \neq n$, the strings $\mathtt{0}^m$ and $\mathtt{0}^n$ are distinguished by the suffix $\mathtt{1}^{k+1-m}\mathtt{2}^k$, because $\mathtt{0}^m\mathtt{1}^{k+1-m}\mathtt{2}^k \in L$ but $\mathtt{0}^n\mathtt{1}^{k+1-m}\mathtt{2}^k \notin L$. Thus, the language $\mathtt{0}^*$ is an infinite fooling set for $L$. ■

   **Solution (concise, alternative fooling set):** Define the fooling set $F = \{\mathtt{0}^n\mathtt{1}^{n+1} \mid n \in \mathbb{N}\}$. Then for distinct $x, y \in F$, $x = \mathtt{0}^i\mathtt{1}^{i+1}$ and $y = \mathtt{0}^j\mathtt{1}^{j+1}$ (where $i \neq j$) are distinguished by the suffix $z = \mathtt{2}^{2i}$, because $xz = \mathtt{0}^i\mathtt{1}^{i+1}\mathtt{2}^{2i} \in L$ but $yz = \mathtt{0}^j\mathtt{1}^{j+1}\mathtt{2}^{2i} \notin L$. Thus, the language $F$ is an infinite fooling set for $L$. ■

   ii. Recall that a block in a string is a maximal non-empty substring of indentical symbols. Let $L$ be the set of all strings in $\{\mathtt{0}, \mathtt{1}\}^*$ that contain two non-empty blocks of $\mathtt{1}$s of unequal length. For example, $L$ contains the strings $\mathtt{01101111}$ and $\mathtt{01001011100010}$ but does not contain the strings $\mathtt{000110011011}$ and $\mathtt{00000000111}$.

   **Solution:** Let $F$ be the language $\mathtt{1}^*$.
   Let $x$ and $y$ be arbitrary strings in $F$.
   Then $x = \mathtt{1}^i$ and $y = \mathtt{1}^j$ for some non-negative integers $i \neq j$.
   Let $w = \mathtt{01}^i$.
   Then $xw = \mathtt{1}^i\mathtt{01}^i \notin L$, because it does not contain two non-empty blocks of 1s of unequal length.
   And $yw = \mathtt{1}^j\mathtt{01}^i \in L$, because $i \neq j$, so it contains two non-empty blocks of 1s of unequal length.
   Thus, $F$ is a fooling set for $L$.
   Because $F$ is infinite, $L$ cannot be regular. ■

**Solution (concise):** For all non-negative integers $i \neq j$, the strings $\mathtt{1}^i$ and $\mathtt{1}^j$ are distinguished by the suffix $\mathtt{01}^i$, because $\mathtt{1}^i\mathtt{01}^i \notin L$ (because it does not contain two non-empty blocks of 1s of unequal length) but $\mathtt{1}^j\mathtt{01}^i \in L$ (because $i \neq j$, so it contains two non-empty blocks of 1s of unequal length). Thus, the language $\mathtt{1}^*$ is an infinite fooling set for $L$. ■

iii. $L = \{0^{n^3} \mid n \geq 0\}$.

**Solution:** Let $F = L = \left\{ \mathtt{0}^{n^3} \;\middle|\; n \geq 0 \right\}$.

Let $x$ and $y$ be arbitrary strings in $F$.

Then $x = \mathtt{0}^{i^3}$ and $y = \mathtt{0}^{j^3}$ for some non-negative integers $i \neq j$.

Let $w = \mathtt{0}^{3i^2+3i+1}$.

Then $xw = \mathtt{0}^{i^3}\mathtt{0}^{3i^2+3i+1} = \mathtt{0}^{i^3+3i^2+3i+1} = \mathtt{0}^{(i+1)^3} \in L$.

And $yw = \mathtt{0}^{j^3}\mathtt{0}^{3i^2+3i+1} = \mathtt{0}^{j^3+3i^2+3i+1} \notin L$, because $i \neq j$ and the equation cannot be constructed as a cube function.

Thus, $F$ is a fooling set for $L$.

Because $F$ is infinite, $L$ cannot be regular. ■

**Solution (concise):** For all non-negative integers $i \neq j$, the strings $\mathtt{0}^{i^3}$ and $\mathtt{0}^{j^3}$ are distinguished by the suffix $\mathtt{0}^{3i^2+3i+1}$, because $\mathtt{0}^{i^3}\mathtt{0}^{3i^2+3i+1} = \mathtt{0}^{i^3+3i^2+3i+1} = \mathtt{0}^{(i+1)^3} \in L$ but $\mathtt{0}^{j^3}\mathtt{0}^{3i^2+3i+1} = \mathtt{0}^{j^3+3i^2+3i+1} \notin L$, because $i \neq j$ and the equation cannot be constructed as a cube function. Thus, $L$ itself is an infinite fooling set for $L$. ■

(b) Suppose $L$ is not regular. Prove that $L \cup L'$ is not regular for any finite language $L'$. Give a simple example of a non-regular language $L$ and a regular language $L'$ such that $L \cup L'$ is regular.

**Solution:**

- If $L'$ is finite, then $L'$ is a regular language, since it is a finite language. Let $L'' = L' \setminus L$. Since $L''$ is a subset of $L'$, $L''$ is also finite, which means $L''$ is also regular. Then $L = (L \cup L') \setminus L''$. Suppose $L \cup L'$ is regular. This implies $L = (L \cup L') \setminus L''$ is regular, since the difference of two regular language is also regular (Closure property of regular language). This contradicts the fact that $L$ is not regular. Thus, $L \cup L'$ is not regular.
- For $L'$ is regular but *infinite*, consider the following example: let $L = \left\{\mathtt{0}^n\mathtt{1}^n \mid n \geq 0\right\}$ which is not regular and $L' = \{\mathtt{0}, \mathtt{1}\}^*$ which is regular. In this case, $L \cup L' = L'$ which is regular. ■

**Rubric:** 10 points.

(a) 6 points, 2 points for each part:
+1 for a correct fooling set, +1 for a correct distinguishing suffix for each pair

(b) 4 points, 2 points for each part

2. Given languages $L_1$ and $L_2$ we define $\mathit{delete}(L_1, L_2)$ to be the language $\{uw \mid uvw \in L_2, v \in L_1\}$ to be the set of strings obtained by "deleting" a string of $L_1$ from a string of $L_2$. For example if $L_1 = \{\texttt{not}\}$ and $L_2 = \{\texttt{CS374isnotfun}, \texttt{not}, \texttt{notnotnot}, \texttt{blah}\}$ then

$$\mathit{delete}(L_1, L_2) = \{\texttt{CS374isfun}, \varepsilon, \texttt{notnot}\}.$$

Prove that if $L_1, L_2$ are regular then $\mathit{delete}(L_1, L_2)$ is also regular. In particular you should describe how to construct an NFA $N = (Q, \Sigma, \delta, s, A)$ from two DFAs $M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, s_2, A_2)$ such that $L(N) = \mathit{delete}(L(M_1), L(M_2))$. You do not need to prove the correctness of your construction but you should explain the ideas behind the construction (see lab 3.5 solutions).

**Solution:** Intuitively, $N$ simulates $M_2$, but insert a string from $L_1$ at a nondeterministically chosen location. To achieve that, $N$ will have three parts: BEFORE, DURING, and AFTER. The BEFORE-machine reads the input, simulates $M_2$, and takes an $\varepsilon$-transition to the DURING-machine. The DURING-machine guesses a valid path from start to an accepting state in $M_1$ while tracking the corresponding path in $M_2$. Finally the AFTER-machine reads in the rest of the input and simulates $M_2$. Note that since we are non-deterministically inserting the string, the DURING-machine will only have $\varepsilon$ transitions in its simulation of $M_1$ and $M_2$.

$$\begin{aligned}
Q &:= Q_1 \times Q_2 \times \{\textsc{Before}, \textsc{During}, \textsc{After}\} \\
s &:= (s_1, s_2, \textsc{Before}) \\
A &:= \left\{(q_1, q_2, \textsc{After}) \;\middle|\; q_2 \in A_2\right\} \\
\delta((q_1, q_2, \textsc{Before}), a) &:= \begin{cases} \left\{(q_1, \delta_2(q_2, a), \textsc{Before})\right\} & a \in \Sigma \\ \left\{(s_1, q_2, \textsc{During})\right\} & a = \varepsilon \end{cases} \\
\delta((q_1, q_2, \textsc{During}), \varepsilon) &:= \begin{cases} \left\{(\delta_1(q_1, a), \delta_2(q_2, a), \textsc{During}) \;\middle|\; a \in \Sigma\right\} & \\ \qquad \cup \left\{(q_1, q_2, \textsc{After})\right\} & \text{if } q_1 \in A_1 \\ \left\{(\delta_1(q_1, a), \delta_2(q_2, a), \textsc{During}) \;\middle|\; a \in \Sigma\right\} & \text{otherwise} \end{cases} \\
\delta((q_1, q_2, \textsc{After}), a) &:= \left\{q_1, \delta_2(q_2, a), \textsc{After})\right\} \quad a \in \Sigma
\end{aligned}$$

All unspecified transitions do not exist. ■

**Solution:** We can simplify the DURING part of the NFA in the previous solution. Instead of non-deterministically choosing a sequence of transitions in the product machine of $M_1$ and $M_2$, we can precompute the set $X_q = \left\{\delta_2^*(q,w) \;\middle|\; w \in L_1\right\}$ for each $q \in Q_2$ by interpreting the product as a *graph* and searching for all states $(p,r)$ reachable from $(s,q)$. In other words, $X_q = \left\{r \in Q_2 \;\middle|\; \exists p \in A_1, (p,r) \text{ is reachable from } (s,q)\right\}$.

$$\begin{aligned}
Q &:= Q_2 \times \{\textsc{Before}, \textsc{After}\} \\
s &:= (s_2, \textsc{Before}) \\
A &:= \left\{(q, \textsc{After}) \;\middle|\; q \in A_2\right\} \\
\delta((q,\textsc{Before}),a) &:= \begin{cases} \left\{(\delta_2(q,a), \textsc{Before})\right\} & a \in \Sigma \\ \left\{(\delta_2^*(q,w), \textsc{After}) \;\middle|\; w \in L_1\right\} = X_q \times \{\textsc{After}\} & a = \varepsilon \end{cases} \\
\delta((q,\textsc{After}),a) &:= \left\{\delta_2(q,a), \textsc{After})\right\} \quad a \in \Sigma
\end{aligned}$$

Note that $X_q$ is a finite subset of $Q_2$ even if $L_1$ is infinite (or even non-regular). Thus an NFA for $\mathit{delete}(L_1, L_2)$ described above always *exists* if $L_2$ is regular, though if $L_1$ is not regular, we have not explained how to actually *compute* $\delta((q,\textsc{Before}),a)$. ■

**Rubric: Standard langage transformation rubric.** For problems worth 10 points:

- \+ 2 for a formal, complete, and unambiguous description of the output automaton, including the states, the start state, the accepting states, and the transition function, as functions of an arbitrary input DFA. The description must state whether the output automaton is a DFA, an NFA without $\varepsilon$-transitions, or an NFA with $\varepsilon$-transitions.
  - – No points for the rest of the problem if this is missing.
- \+ 2 for a brief English explanation of the output automaton. We explicitly do not want a formal proof of correctness, or an English transcription, but a few sentences explaining how your machine works and justifying its correctness. What is the overall idea? What do the states represent? What is the transition function doing? Why these accepting states?
  - – Deadly Sin: No points for the rest of the problem if this is missing.
- \+ 6 for correctness
  - \+ 3 for accepting all strings in the target language
  - \+ 3 for accepting only strings in the target language
  - \- 1 for a single mistake in the formal description (for example a typo)
  - – Double-check correctness when the input language is $\emptyset$, or $\varepsilon$, or $\mathtt{0}^*$, or $\Sigma^*$.

3. Let $L_k = \{w \in \{0,1\}^* : |w| \geq 2k$ and last $2k$ characters of $w$ has equal number of 0s and 1s$\}$. If $k = 3$ then 010011 and 000111000 are in $L_3$ while 0001111 and 01000110 are not.

   - Describe an NFA for $L_k$ that has $O(k^2)$ states.

**Solution:** The idea is to keep track of the number of 0s and 1s seen so far within the last $2k$ characters. Correspondingly, the states of the NFA are of the form $(i, j)$ where $i$ and $j$ are the number of 0s and 1s seen during this $2k$ character window. At some point, the NFA guesses that the current character being read is $2k$ away from the end of the string, and starts counting the number of 0s and 1s seen within the last $2k$ characters. Note that the number of 0s and 1s are the same when they are both exactly $k$; if either count goes above $k$ we reject.

$$
\begin{aligned}
Q &:= \{(i,j) \mid 0 \leq i \leq k,\ 0 \leq j \leq k\} \\
s &:= (0,0) \\
A &:= \{(k,k)\} \\
\delta((0,0), \texttt{0}) &:= \{(0,0), (1,0)\} \\
\delta((0,0), \texttt{1}) &:= \{(0,0), (0,1)\} \\
\text{For } (i,j) \neq (0,0) \quad \delta((i,j), \texttt{0}) &:= \begin{cases} \{(i+1,j)\} & \text{if } i < k \\ \varnothing & \text{else} \end{cases} \\
\text{For } (i,j) \neq (0,0) \quad \delta((i,j), \texttt{1}) &:= \begin{cases} \{(i,j+1)\} & \text{if } j < k \\ \varnothing & \text{else} \end{cases}
\end{aligned}
$$

Clearly, from the product construction of $Q$ we can see that $|Q|$ is $O(k^2)$. ■

**Solution:** The states are of the form $(i, j)$ where $i$ and $j$ are the number of 0s and 1s seen so far. At any point, the NFA can guess that it did not correctly start counting at the correct location, and reset the counter. The only way to end on the accepting state $(k, k)$ is to start counting at the right place and follow a path of length $2k$ from $(0, 0)$.

$$
\begin{aligned}
Q &:= \{(i,j) \mid 0 \leq i \leq k,\ 0 \leq j \leq k\} \\
s &:= (0,0) \\
A &:= \{(k,k)\} \\
\delta((i,j), \texttt{0}) &:= \begin{cases} \{(i+1,j)\} & \text{if } i < k \\ \varnothing & \text{else} \end{cases} \\
\delta((i,j), \texttt{1}) &:= \begin{cases} \{(i,j+1)\} & \text{if } j < k \\ \varnothing & \text{else} \end{cases} \\
\delta((i,j), \varepsilon) &:= \{(0,0)\}
\end{aligned}
$$

Clearly, from the product construction of $Q$ we can see that $|Q|$ is $O(k^2)$. ■

- Prove that the set of all binary strings of length $k$ is a fooling set for $L_k$. Conclude that any DFA for $L_k$ needs at least $2^k$ states.

**Solution:** Observe that $w \in L_k$ if and only if there are exactly $k$ `0`s in the last $2k$ characters of $w$.

Let $F_k$ be the set of all binary strings of length $k$. Choose an arbitrary pair of distinct $x, y \in F_k$. There are two cases.

- $\#(\mathtt{0}, x) \neq \#(\mathtt{0}, y)$.
  In this case $z = \mathtt{0}^{k-\#(\mathtt{0},x)}\mathtt{1}^{k-\#(\mathtt{1},x)}$ distinguishes $x$ and $y$ with respect to $L$.
- $\#(\mathtt{0}, x) = \#(\mathtt{0}, y)$.
  In this case, let $w$ the longest shared prefix between $x$ and $y$. Without loss of generality, $x = w\mathtt{0}r, y = w\mathtt{1}s$ for some strings $r$ and $s$, where we observe that $\#(\mathtt{0}, r) \neq \#(\mathtt{0}, s)$. Take $z = \mathtt{0}^{k-\#(\mathtt{0},r)}\mathtt{1}^{k-\#(\mathtt{1},r)}$. The last $2k$ characters of $xz$ are exactly $rz$, and $\#(\mathtt{0}, rz) = k$, but the last $2k$ characters of $yz$ are $sz$ where $\#(\mathtt{0}, sz) \neq k$.

Hence, we have that $F_k$ is a valid fooling set for $L_k$. Since each string in any valid fooling set for $L_k$ must correspond to a distinct state in any DFA that accepts $L_k$, any DFA for $L_k$ must have at least $|F_k| = 2^k$ states. ■

- **Extra credit:** Improve size of the fooling set for $L_k$ or prove that there is a DFA for $L_k$ with $2^k$ states.

**Solution:** Let $F_{k+1}$ be the set of all binary strings of length $k+1$. Choose an arbitrary pair of distinct $x, y \in F_k$. There are two cases.

- $x = au$ and $y = bv$ where $a, b \in \{\mathtt{0}, \mathtt{1}\}$ and $u, v$ are distinct elements of $F_k$. In the previous part, we proved that $F_k$ is a fooling set for $L_k$, so let $z$ be the distinguishing suffix found above for for $u, v$. We know that $|z| \geq k$, so that last $2k$ characters of $xz$ and $yz$ are in fact substrings of $uz$ and $vz$, respectively, so $z$ is also a distinguishing suffix for $x, y$.
- Without loss of generality, $x = \mathtt{0}u$ and $y = \mathtt{1}u$ where $u \in F_k$
  If $\#(\mathtt{0}, u) \leq \#(\mathtt{1}, u)$, set $z = \mathtt{0}^{k-1-\#(\mathtt{0},u)}\mathtt{1}^{\#(\mathtt{0},u)}$. Clearly $|z| = k-1$, so $|xz| = |yz| = 2k$. Now $\#(\mathtt{0}, xz) = 1 + \#(\mathtt{0}, u) + k - 1 - \#(\mathtt{0}, u) = k$, so $xz \in L_k$. But $\#(\mathtt{0}, yz) = \#(\mathtt{0}, u) + k - 1 - \#(\mathtt{0}, u) = k-1$, so $yz \notin L_k$.
  If $\#(\mathtt{0}, u) > \#(\mathtt{1}, u)$, set $z = \mathtt{1}^{k-1-\#(\mathtt{1},u)}\mathtt{0}^{\#(\mathtt{1},u)}$. Similar analysis shows that $xz \notin L_k$ but $yz \in L_k$.

Hence $F_{k+1}$ is a fooling set for $L_k$, and since $|F_{k+1}| = 2^{k+1}$, $F_{k+1}$ is a larger fooling set than $F_k$.

---

We claim (without proof) that the largest size of any fooling set for $L_k$ and, correspongly, the smallest number of states in a DFA for $L_k$, is actually $\binom{2(k+1)}{k+1} - 1$. We leave it up to the reader to verify this claim by describing a fooling set and DFA of this size for $L_k$. ■

**Rubric:**

- 5 points. Scaled DFA design rubric:
  - 1 point for an unambiguous description of the NFA, including the states set $Q$, the start state $s$, the accepting states $A$, and the transition function $\delta$.
  - 2 points for *briefly* explaining the purpose of each state in English. This is how you argue that your NFA is correct.
  - 2 points for correctness.
    - -0.5 for a single mistake: a single misdirected transition, a single missing or extra accepting state, rejecting exactly one string that should be accepted, or accepting exactly one string that should be accepted.
    - -1 for incorrectly accepting/rejecting more than one but a finite number of strings.
    - -2 for incorrectly accepting/rejecting an infinite number of strings
- 5 points.
  - 4 points for the proof that the set is a fooling set:
    - +1 for correctly considering *arbitrary* $x, y \in F$.
      - No credit for the proof unless both $x$ and $y$ are *always* in $F$.
      - No credit for the proof unless $x$ and $y$ can be any *strings* in $F$.
    - +1 for correctly describing a suffix $z$ that distinguishes $x$ and $y$.
    - +1 for proving either $xz \in L$ or $yz \in L$
    - +1 for proving either $yz \notin L$ or $xz \notin L$, respectively.
  - 1 point for connecting the size of the set to lower bounding the size of a DFA for the language.
- Record extra credit for any correct fooling set of size strictly greater than $2^k$.
  Record extra extra credit if a (correct) fooling set of size $\binom{2(k+1)}{k+1} - 1$ was found.