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 0^*.
Let x and y be arbitrary strings in F.
Then $x = 0^m$ and $y = 0^n$ for some non-negative integers $m \neq n$.
Let $w = 1^{k+1-m}2^k$.
Then $xw = 0^m 1^{k+1-m} 2^k \in L$, , because $m + k + 1 - m = k + 1$.
And $yw = 0^n 1^{k+1-m} 2^k \notin L$, 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 0^m and 0^n are distinguished by the suffix $1^{k+1-m}2^k$, because $0^m 1^{k+1-m} 2^k \in L$ but $0^n 1^{k+1-m} 2^k \notin L$. Thus, the language 0^* 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 $\{0, 1\}^*$ that contain two non-empty blocks of 1s of unequal length. For example, L contains the strings 01101111 and 01001011100010 but does not contain the strings 000110011011 and 00000000111.

Solution: Let F be the language 1^*.
Let x and y be arbitrary strings in F.
Then $x = 1^i$ and $y = 1^j$ for some non-negative integers $i \neq j$.
Let $w = 01^i$.
Then $xw = 1^i 01^i \notin L$, because it does not contain two non-empty blocks of 1s of unequal length.
And $yw = 1^j 01^i \in L$, 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 1^i and 1^j are distinguished by the suffix 01^i, because $1^i 01^i \notin L$ (because it does not contain two non-empty blocks of 1s of unequal length) but $1^j 01^i \in L$ (because $i \neq j$, so it contains two non-empty blocks of 1s of unequal length). Thus, the language 1^* is an infinite fooling set for L.

■

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

Solution: Let $F = L = \left\{0^{n^3} \;\middle|\; n \geq 0\right\}$.
Let x and y be arbitrary strings in F.
Then $x = 0^{i^3}$ and $y = 0^{j^3}$ for some non-negative integers $i \neq j$.
Let $w = 0^{3i^2+3i+1}$.
Then $xw = 0^{i^3} 0^{3i^2+3i+1} = 0^{i^3+3i^2+3i+1} = 0^{(i+1)^3} \in L$.
And $yw = 0^{j^3} 0^{3i^2+3i+1} = 0^{j^3+3i^2+3i+1} \notin L$, 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 0^{i^3} and 0^{j^3} are distinguished by the suffix 0^{3i^2+3i+1}, because $0^{i^3}0^{3i^2+3i+1} = 0^{i^3+3i^2+3i+1} = 0^{(i+1)^3} \in L$ but $0^{j^3}0^{3i^2+3i+1} = 0^{j^3+3i^2+3i+1} \notin L$, 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 $L \setminus L'$ 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 $L \setminus L'$ is regular.

Solution: L' is a regular language, since it is a finite language.

Let $L'' = L' \cap L$. Since L'' is a subset of L', L'' is also finite, which means L'' is also regular. Then $L = (L \setminus L') \cup L''$. Suppose $L \setminus L'$ is regular. This implies $L = (L \setminus L') \cup L''$ is regular, since the union of two regular language is also regular. (Closure property of regular language). This contradicts the fact that L is not regular. Thus, $L \setminus L'$ is not regular.

For example, let $L = \{0^n 1^n \mid n \geq 0\}$ which is not regular and $L' = \{0,1\}^*$ which is regular. In this case, $L \setminus L' = \emptyset$ is regular.

■

Rubric: On a scale of 10 points:

- 6 points for (a), 2 points for each subquestion:
  - 1 point for a proper setup: an infinite fooling set, x, y which are arbitrary pairs in the fooling set, z which is arbitrary string, and proving exactly one of $\{xz, yz\}$ is in L. No further points if this part is incorrect.
  - 1 point for correctly proving z distinguishes x, y.
  - -0.5 for each minor error.
- 4 points for (b):
  - 3 points for the proof.
  - 1 points for the example.
  - -0.5 each minor error.

2. Describe a context-free grammar for the following languages. Clearly explain how they work and the role of each non-terminal. Unclear grammars will receive little to no credit.

(a) $\{a^i b^j c^k \mid k = 3(i + j)\}$.

Solution: For every a or b in such a string, we must have 3 c's. We can generate these strings by first adding the a's along with the appropriate number of c's, and then adding the b's, again with 3 c's for each.

$S \to aSccc \mid B$ strings of the form $a^i b^j c^k$, s.t. $k = 3(i + j)$

$B \to bBccc \mid \varepsilon$ strings of the form $b^j c^k$, s.t. $k = 3j$

■

(b) $\{a^i b^j c^k d^\ell \mid i, j, k, \ell \geq 0 \text{ and } i + \ell = j + k\}$.

Solution: Consider following two cases,

- Case 1: $\{a^i b^j c^k d^\ell | i \leq j, i + \ell = j + k\}$
- Case 2: $\{a^i b^j c^k d^\ell | i > j, i + \ell = j + k\}$

For Case 1. Since the number of a's is at most as the number of b's in the string. Therefore, we can represent the beginning of the string as $a^i b^{i+x}$ (i.e., $j = i + x$). Since there are ℓ d's, the string must be in the form of $a^i b^{i+x} c^{\ell-x} d^l$ in order to keep the sum of the number of b's and c's to equal $i + \ell$. We can rewrite this as $a^i b^i$ followed by $b^x c^{\ell-x} d^\ell$. The first group can be generated by $A \to aAb \mid \varepsilon$. And the second group can be generated by $X \to bXd \mid C$ together with $C \to cCd \mid \varepsilon$. Putting these together gives us $L \to AX$, which handles Case 1.

For Case 2, we have $\ell < k$, and the solution is similar to Case 1. But now the grouping is the following form $a^i b^{i-x} c^{\ell+x} d^\ell$. This can be regrouped as $a^i b^{i-x} c^x$ and $c^\ell d^\ell$.

$S \to L \mid M$	strings of the form $a^i b^j c^k d^\ell$, s.t. $i + \ell = j + k$
$L \to AX$	strings of the form $a^i b^j c^k d^\ell$, s.t. $i \leq j$, $i + \ell = j + k$
$A \to aAb \mid \varepsilon$	strings of the form $a^i b^i$, for some $i \geq 0$
$X \to bXd \mid C$	strings of the form $b^j c^{k-j} d^k$, for some $j, k \geq 0$
$C \to cCd \mid \varepsilon$	strings of the form $c^i d^i$, for some $i \geq 0$
$M \to YC$	strings of the form $a^i b^j c^k d^\ell$, s.t. $i > j$, $i + \ell = j + k$
$Y \to aYc \mid A$	strings of the form $a^i b^{i-j} c^j$, for some $i, j \geq 0$

■

(c) $L = \{0, 1\}^* \backslash \{0^n 1^{2n} \mid n \geq 0\}$. In other words, the complement of the language $\{0^n 1^{2n} \mid n \geq 0\}$.

Solution: We take a similar approach to the third problem of lab 4. We claim that L is the union of the language $L_1 = \{0^m 1^n \mid n \neq 2m, m, n \geq 0\}$ and the language $L_2 = (0 + 1)^* 10(0 + 1)^*$. L_1 is contained in L by its definition. L_2 is contained in L because L_2 is the complement of 0^*1^*. 0^*1^* is the union of L_1 and $\{0^n 1^{2n} \mid n \geq 0\}$.

On the other hand, $\forall w \in L$ w is either in L_1 or L_2 by the definition of L. If w is of the form $0^m 1^n$, it must be that $n \neq 2m$, else w would be in $\{0^n 1^{2n} \mid n \geq 0\}$, and thus not in L, thus if w is of the form $0^m 1^{2n}$ it is in L_1. If w is not of the form $0^m 1^n$, then it must contain a 1 followed by a 0, so it is in L_2.

$S \to T \mid X$	$\{0,1\}^* \setminus \{0^n 1^{2n} \mid n \geq 0\}$
$T \to 0T11 \mid A \mid B \mid C$	$\{0^m 1^n \mid n \neq 2m, m, n \geq 0\}$
$A \to 1 \mid A1$	1^+
$B \to 0 \mid B0$	0^+
$C \to 1 \mid B1$	0^*1
$X \to Z10Z$	$(0+1)^*10(0+1)^*$
$Z \to \varepsilon \mid 0Z \mid 1Z$	$(0+1)^*$

■

Rubric: (a) 3 points
- 2 for a correct grammar (This is not the only correct solution)
- 1 for a clear explanation of the grammar
- if the solution is not understandable and no explanation, give 0.

(b) 4 points
- 1 for correctly identify two cases.
- 2 for a correct grammar. (This is not the only correct solution.)
- 1 for a clear explanation of the grammar.
- if the solution is not understandable and no explanation, give 0.

(c) part
- 2 for a correct grammar. (These are not the only correct solutions.)
- 1 for a clear explanation of the grammar.
- if the solution is not understandable and no explanation, give 0.

3. **Solution:** We prove $insert(L_1, L_2)$ is regular by constructing a NFA N such that $L(N) = insert(L_1, L_2)$. Let $M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, s_2, A_2)$ be DFAs recoginizing the languages L_1 and L_2 respectively. The construction idea is that for any input string x, the NFA N will guess where to split the string into $x = uvw$ such that $v \in L_1$ and $uw \in L_2$. Naturally, N will follow the transitions of M_2 when reading u, until the position where it guesses v starts and go to the start state of M_1 from there. Then it follows the transitions of M_1 from s_1 to an accepting state while reading v, from which point it goes back to the state of M_2 where it left and continues to follow transitions of M_2 afterwards as it reads w.

To realize this idea, we will use two copies of M_2 and $|Q_2|$ copies of M_1. We need two copies of M_2 to distinguish whether the insertion of a string from L_1 has been made. We need $|Q_2|$ copies of M_1 to "remember" the state of M_2 from which we transitioned into M_1 and the state to which we will transition back. Inside each copy of M_1 (respectively M_2), the transition functions are the same as in M_1 (respectively M_2). The copies of M_1 have no transition between each other, neither do the two copies of M_2. We add epsilon transitions from each state q of the first copy of M_2 to (and only to) the start state of the copy of M_1 associated with q and we add epsilon transitions from every accepting state of that copy of M_1 to the corresponding state q of the second copy of M_2.

Formally, we define $N = (Q, \Sigma, \delta, s, A)$ such that

$$
\begin{aligned}
Q &= (Q_1 \times Q_2) \cup (Q_2 \times \{\mathit{before}, \mathit{after}\}), \\
s &= (s_2, \mathit{before}), \\
A &= \{(q, \mathit{after}) \mid q \in A_2\}, \\
\delta((q,l),a) &= \{(\delta_2(q,a), l)\} \quad \forall q \in Q_2, a \in \Sigma, l \in \{\mathit{before}, \mathit{after}\}, && (1) \\
\delta((p,q),a) &= \{(\delta_1(p,a), q)\} \quad \forall p \in Q_1, q \in Q_2, a \in \Sigma, && (2) \\
\delta((q,\mathit{before}),\epsilon) &= \{(s_1, q)\} \qquad \forall q \in Q_2, && (3) \\
\delta((p,q),\epsilon) &= \{(q, \mathit{after})\} \quad \forall p \in A_1, q \in Q_2. && (4)
\end{aligned}
$$

■

Rubric: On a scale of 10 pts:

- 5pts for explaining the construction idea:
  - 1pt for using the guessing power (epsilon transitions) of NFA to split the string,
  - 2pts for using two copies of M_2 to distinguish between "before" and "after" the insertion in the string,
  - 2pts for using copies of M_1 to remember the state to which the NFA shall go back.
- 5pts for definition of NFA:
  - 1pt for Q,
  - 1pt for s and A,
  - 1pt for transition functions (1) and (2) together,
  - 1pt each for transition functions (3) and (4).