Question 1. [7 marks] Prove that the language $L = \{ a^n b^{2n} c^{3n} \mid n \geq 0 \}$ is not a context-free language using the pumping lemma for context-free languages.

My Solution:

The Pumping Lemma for context-free languages states that for any context-free language L, there exists a number p (the pumping length) such that any string (s) ∈ (L) with ($|s| \geq p$) divided into five parts, ($s = uvxyz$) satisfying:

- Property 1: $\mathbf{uv^ixy^iz \in L}$ for each ($i \geq 0$)
- Property 2: $|\mathbf{vy}| > \mathbf{0}$ (v and y cannot be empty)
- Property 3: $|\mathbf{vxy}| \leq \mathbf{p}$

Let $w = a^p b^{2p} c^{3p}$ where p is the pumping length. Let $p = 3$. We get, w = aaabbbbbbccccccccc.

Keeping **Property 3** in mind, we have the following cases with **red** indicating the substring |vxy|, **green** indicating the prefix of |vxy| and **blue** indicating suffix of |vxy|.

Case 1: |vxy| doesn't straddle boundaries such that w = aaabbbbbbccccccccc

Following property 1, |vy| cannot be empty because $n \geq 0$, and therefore $|vy| > 0$. Now, for $i = 2$, uv^2xy^2z = a..a..abbbbbbccccccccc which would increase the number of a's without increasing b's and c's disrupting the order $a^p b^{2p} c^{3p}$. This violates property 3 of the pumping lemma.

Case 2a: |vxy| straddles the first boundary such that w = aaabbbbbbccccccccc

Following property 1, |vy| cannot be empty therefore $|vy| > 0$. So, we could have either *v is non-empty a* OR *y is non-empty b* OR *both v, non-empty a and y, non-empty b* OR *either v or y is a non-empty substring abb*. Now, for $i = 2$, uv^2xy^2z = aaa..b..bbbbbbccccccccc which would increase the number of a's and b's without increasing c's disrupting the order $a^p b^{2p} c^{3p}$. This violates property 3 of the pumping lemma.

Case 2b: |vxy| straddles the second boundary such that w = aaabbbbbbccccccccc

Following property 1, |vy| cannot be empty therefore |vy| > 0. So, we could have either *v is non-empty b* OR *y is non-empty c* OR *both v, non-empty b and y, non-empty c* OR *either v or y is a non-empty substring bbc*. Now, for i = 2, uv^2xy^2z = **aaabbbbb..b..ccccccccc** which would increase the number of b's and c's without increasing a's disrupting the order $a^pb^{2p}c^{3p}$. This violates property 3 of the pumping lemma.

Case 3: |vxy| straddles the midpoint such that w = aaabbbbbbcccccccc

Case 3 follows very closely with Case 2b, since the middle point in w also straddles the second boundary between b and c. Therefore, Case 3 violates property 3 of the pumping lemma.

We have shown that there's **no way** to rewrite w = uvxyz which satisfies all three conditions of the pumping lemma. Therefore, L is not context-free.

Question 2. [7 marks] Consider the language L = { $0^n1^n0^m1^m$ | n < m }. Prove that L is not a context-free language using the pumping lemma for context-free languages.

My Solution:

The Pumping Lemma for context-free languages states that for any context-free language L, there exists a number p (the pumping length) such that any string (s) ∈ (L) with (|s| ≥ p) divided into five parts, (s = uvxyz) satisfying:

- Property 1: $\mathbf{uv^ixy^iz \in L}$ for each (i ≥ 0)
- Property 2: **|vy| > 0** (v and y cannot be empty)
- Property 3: **|vxy| ≤ p**

Let w = $0^n1^n0^m1^m$ where m = 2n (This is just an assumption, but we don't really have to establish a relationship between m and n, and we will mention in edge cases where we don't need to consider m = 2n. Since n < m, we assume an m which is two-times bigger than n for simplicity in the proof. We could've also considered m = n – 1 which could've given us the same results). So, w = $0^n1^n0^{2n}1^{2n}$. Letting p be the pumping length and making p = 3. We get, w = $0^31^30^61^6$ = 000111000000111111. Keeping **Property 3** in mind, we have the following cases with **red** indicating the substring |vxy|, **green** indicating the prefix of |vxy| and **blue** indicating suffix of |vxy|.

Case 1: |vxy| doesn't straddles boundaries such that w = 000111000000111111.

Following property 1, |vy| > 0 therefore **v** or **y** or **both** is a non-empty substring of 1's in this case. Taking i = 2, uv^2xy^2z = **0001111000000111111** which would increase the number of 1's without increasing 0's disrupting the order $0^n1^n0^m1^m$. This violates property 3 of the pumping lemma.

Case 2a: |vxy| straddles the first boundary such that w = 000111000000111111.

Following property 1, |vy| > 0 therefore we could have either *v is non-empty 0's* OR *y is non-empty 1's* OR *both v, non-empty 0's and y, non-empty 1's* OR *either v or y is a non-empty substring 011*. Taking i = 2, uv^2xy^2z = **000..1..11000000111111** which results in failing the condition for the language n < m since it would significantly cause imbalance the number of 0's and 1's in the first two substring in w, $0^n1^n0^m1^m$ and therefore **w** will become $0^k1^l0^m1^m$ (for some k not equal to n and l not equal to n). This violates property 3 of the pumping lemma.

Case 2b: |vxy| straddles the third boundary such that w = 000111000000111111.

Following property 1, |vy| > 0 therefore we could have either *v is non-empty 1's* OR *y is non-empty 0's* OR *both v, non-empty 1's and y, non-empty 0's* OR *either v or y is a non-empty substring 110*. Taking i = 2, uv^2xy^2z = **00011..1..000000111111** which results in failing the condition for the language n < m since it would significantly increase the number of 1's in the second substrings in w, $0^n1^n0^m1^m$. This violates property 3 of the pumping lemma.

Case 2c: |vxy| straddles the third boundary such that w = 000111000000111111.

Following property 1, |vy| > 0 therefore we could have either *v is non-empty 0's* OR *y is non-empty 1's* OR *both v, non-empty 0's and y, non-empty 1's* OR *either v or y is a non-empty substring 001*. Taking i = 2, uv^2xy^2z = **000111000000..0..111111**.

This doesn't result in the failing the condition for the language n < m since it would not increase the number of 0's and 1's in the first two substring in w, 0^n1^n and wouldn't disrupt the order in $0^n1^n0^m1^m$. However, in the last two substring 0^m1^m, uv^ixy^iz would imbalance the number of 0's and 1's and therefore **w** will become $0^n1^n0^k1^l$ (for some k not equal to m and l not equal to m). This violates property 3 of the pumping lemma.

Case 3: |vxy| straddles the midpoint such that w = 000111000000111111.

Following property 1, |vy| > 0 therefore **v** or **y** or **both** is a non-empty substring of 0's in this case. Taking i = 2, uv^2xy^2z = **00011100..0..000111111** which would increase the number of 0's without increasing 1's in the third substring causing imbalance in the order $0^n1^n0^m1^m$ where **w** will become $0^n1^n0^k1^m$ (for some k not equal to m). This violates property 3 of the pumping lemma.

We have shown that there's **no way** to rewrite w = uvxyz which satisfies all three conditions of the pumping lemma. Therefore, L is not context-free.

Question 3a. Consider the language L = { 0^{3n} | n ≥ 0 } over Σ = {0}. In other words, L contains strings which have a multiple of 3 number of 0's. (e.g. ε, 000, 000000, 000000000, ...). Give a high-level description of a Turing machine which recognizes L.

<u>My Solution:</u>

To describe a Turing machine that recognizes the language L = {0^{3n} | n ≥ 0 } where Σ = {0}, we need to construct a high-level plan for the machine's operation.

Here's the outline of the Turing machine:

1. **Initialization**: The Turing machine begins in the start state and moves its head to the rightmost end of the input string.
2. **Scan and Counting**: The Turing machine scans the input tape from right to left, counting the number of '0's encountered. It can do this by transitioning between states while moving left along the tape and incrementing a counter for each '0' it encounters.
3. **Determining Multiples of 3**: Once it reaches the leftmost end of the input string, the Turing machine checks whether the count of '0's is a multiple of 3. It does this by utilizing the counter it has maintained throughout the scanning process.

4. **Acceptance/Rejection**: If the count is a multiple of 3, the Turing machine accepts the input string. Otherwise, it rejects the input.

Here's a more detailed description of the Turing machine's states and transitions:

- **Q0 (Start)**: Initial state. Move the head to the rightmost end of the input.
- **Q1 (Counting)**: While in state Q1, scan the input tape from right to left, incrementing a counter for each '0' encountered until reaching the leftmost end of the input string.
- **Q2 (Check)**: After reaching the left end of the input, check whether the count is a multiple of 3.
- **Q3 (Accept)**: If the count is a multiple of 3, move to the accept state and halt.
- **Q4 (Reject)**: If the count is not a multiple of 3, move to the reject state and halt.

The transition function will be designed to facilitate this process, moving between states based on the symbols read and the current state of the machine. This high-level description outlines the logic and flow of the Turing machine that recognizes the language $L = \{0^{3n} \mid n \geq 0\}$. The specifics of transitions, states, and tape movements would require further detailed design and implementation.

Question 3b. Consider the language $L = \{ 0^{3n} \mid n \geq 0 \}$ over $\Sigma = \{0\}$. In other words, L contains strings which have a multiple of 3 number of 0's. (e.g. ε, 000, 000000, 000000000, ...). Give the formal description (i.e. a state machine and 7-tuple definition) of a Turing machine which recognizes L.

My Solution:

To formally describe a Turing machine that recognizes the language $L = \{ 0^{3n} \mid n \geq 0 \}$, we need to define its components using a 7-tuple notation: $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$.

1. Q: Set of states: $\{ q_0, q_1, q_2, q_3, q_4 \}$
   - q_0: Start state
   - q_1: Counting state
   - q_2: Check state
   - q_3: Accept state
   - q_4: Reject state

2. Σ: Input alphabet: { 0 }
3. Γ: Tape alphabet: { 0,⊔ } (includes Σ and special blank symbol).
4. δ: Transition function: δ
   - $\delta(q_0, 0) = (q1, \sqcup, R)$ (Move right and start counting)
   - $\delta(q_1, 0) = (q1, 0, L)$ (Count '0's and move left)
   - $\delta(q_1, \sqcup) = (q2, \sqcup, R)$ (Transition to check state)
   - $\delta(q_2, 0) = (q2, 0, R)$ (Stay in check state)
   - $\delta(q_2, \sqcup) = (q3, \sqcup, S)$ (Accept if count is a multiple of 3)
   - $\delta(q_2, 0) = (q4, 0, S)$ (Reject if count is not a multiple of 3)
5. q_0: Start state
6. q_{accept} : Accept state
7. q_{reject} : Reject state

Here we have outlined the states, input alphabet, tape alphabet, transition function, start state, accept state, and reject state of the Turing machine that recognizes the language $L = \{ 0^{3n} \mid n \geq 0 \}$.