1. Show if \( f_1(n) \) is \( O(n^{k_1}) \) and \( f_2(n) \) is \( O(n^{k_2}) \) where \( k_1 \geq k_2 \) then \( f_1(n) + f_2(n) \) is \( O(n^{k_1}) \).

2. The familiar linear search algorithm takes as input a list of values \([v_1, v_2, \ldots, v_n]\) and a search term \( x \) and decides if the \( x = v_i \) for some \( 1 \leq i \leq n \). In the case that the list is empty and \( n = 0 \), the algorithm can immediately reject. You should recall that linear search has time complexity \( f(n) = O(n) \). What is wrong with the following "proof" that linear search has time complexity \( f(n) = O(1) \)?

**Proof.** We will show that linear search has time complexity \( O(1) \) by induction on the size of the number of input values \( n \). For the base case, \( n = 0 \), there is nothing to check and the algorithm clearly runs in \( O(1) \) steps. By the inductive hypothesis, assume that linear search runs in \( O(1) \) steps for lists up to some size \( k < n \). It clearly holds for \( k = 0 \). For the inductive step, consider the case of a list containing \( n \) elements. By the inductive hypothesis, processing the first \( n-1 \) elements can be done in \( O(1) \) time. If \( x \) is in the first \( n-1 \) elements, we accept the input and no further processing is needed. If \( x \) is not in the first \( n-1 \) elements, we spend \( O(1) \) time to processes the last element and accept if \( x = v_n \) and reject otherwise. The total time spent is \( O(1) + O(1) = O(1) \) by the result in previous question. Therefore linear search can be done in \( O(1) \) time.

3. Prove that the complexity class \( P \) is closed under concatenation.

4. Prove that the complexity class \( P \) is closed under Kleene star. Hint: you may want to consider a technique similar to how we built a table in lab 6 for determining if a grammar in Chomsky Normal Form could generate a string \( w \). You probably showed decidable or recognizable languages were closed under Kleene star using non-determinism. Languages in \( P \) can only use deterministic Turing machines that run in polynomial time.

5. Given a graph \( G = (V, E) \) with a set of vertices \( V \) and edges \( E \), we say that \( G \) has 3-clique if there exist three vertices \( a, b, c \in V \) such that there is an edge \( e \in E \) for every pair of unique vertices in \( \{a, b, c\} \). In other words, \( G \) contains a triangle. Show that the language \( L = \{ \langle G = (V, E) \rangle | G \text{ contains a 3-clique} \} \) is in \( P \).