4.26 For a fixed string \( x \), we know that \( \Sigma^*x\Sigma^* \) is a regular expression. By Problem 2.18(a) and its selected solution (see p.132 and 135), we know that \( L(G) \cap (\Sigma^*x\Sigma^*) \) is also context free, hence there exists a context free grammar \( G'_{(G,x)} \) such that \( L(G) \cap (\Sigma^*x\Sigma^*) = L(G'_{(G,x)}) \).

We construct the following TM to decide the language \( C \):

\[ T = "\text{On input } < G, x >, \text{ where } G \text{ is a CFG and } x \text{ is a string.} \]

1. Let the CFG of \( L(G) \cap (\Sigma^*x\Sigma^*) \) be \( G'_{(G,x)} \).
2. Run the decider \( R := R_{E_{CFG}} \) in Theorem 4.8 on \( < G'_{(G,x)} > \).
3. If \( R \) accepts \( < G'_{(G,x)} > \), then \( T \) rejects \( < G, x > \); otherwise, \( T \) accepts \( < G, x > \).

(Correctness:)

Note that \( R \) is the decider for \( E_{CFG} \) so for any description \( < G' > \) of a CFG, \( R \) always stops in either the accept state or the reject state.

If \( x \) is a substring of some \( w \in L(G) \), then \( L(G) \cap (\Sigma^*x\Sigma^*) \neq \emptyset \).

\( \Rightarrow R \) rejects \( < G'_{(G,x)} > \). \( \Rightarrow T \) accepts \( < G, x > \).

If \( x \) is NOT a substring of any \( w \in L(G) \), then \( L(G) \cap (\Sigma^*x\Sigma^*) = \emptyset \).

\( \Rightarrow R \) accepts \( < G'_{(G,x)} > \). \( \Rightarrow T \) rejects \( < G, x > \).

4.28 Since the language \( A \) is recognizable, there exists an enumerator \( E_A \) for \( A \) and hence we can let the output of \( E_A \) be \( \{ < M_1 >, < M_2 >, ... \} \), where every \( M_i \) is a decider. Let \( s_1, s_2, s_3, ... \) be a list of all possible string in \( \Sigma^* \) (see p.180 in the textbook). We will use diagonalization method to construct a language \( D \):

\[ D = \{ s_i \in \Sigma^* : s_i \notin L(M_i) \} \]

Consider this TM to decide \( D \):

\[ T = "\text{On input } x: \]

1. If \( x \notin \Sigma^* \), rejects; otherwise, \( x = s_j \) for some \( j \).
2. Find the description \( < M_j > \) enumerated by \( E_A \), and simulate \( M_j \) on \( s_j \).
3. If \( M_j \) accepts \( s_j \) then \( T \) rejects \( s_j \); otherwise, \( T \) accepts \( s_j \).

(Correctness:)

Note that the step 1 rejects any \( x \notin \Sigma^* \), so obviously step 2 and 3 make \( L(T) = D \).

Besides, for any \( i \in \mathbb{N} \), \( M_i \) can decide if a string in \( \Sigma^* \) belongs to \( L(M_i) \) or not, hence \( T \) is a decider. The most important of all is that for any \( j \), the answer of \( T \) on \( s_j \) is different from \( M_j \), hence \( < T > \notin A \).
5.4. “No.”
Consider the languages 
\( A = \{0^n1^n : n \in \mathbb{N} \} \) and \( B = \{1\} \) over \( \Sigma = \{0, 1\} \). Define the function \( f : \Sigma^* \to \Sigma^* \) by
\[
f(w) = \begin{cases} 
1, & \text{if } w \in A; \\
0, & \text{otherwise.}
\end{cases}
\]
Note that \( f \) is a computable function and \( w \in A \) if and only if \( f(w) \in B \). Hence, \( A \leq_m B \). However, \( A \) is not regular, while \( B \) is regular.

5.23 If \( A \) is decidable, there is a Turing Machine \( M \) to decide it. we define function \( f \) that
\[
f(w) = \begin{cases} 
01, & \text{if } w \in A; \\
10, & \text{if } w \not\in A.
\end{cases}
\]
We can construct a TM to compute the function that it first simulate \( M \), and output 01 if accepted, otherwise output 10. So \( A \leq_m 0^*1^* \) since \( w \in A \) iff \( f(w) \in 0^*1^* \).
Conversely, because \( 0^*1^* \) is decidable, if \( A \leq_m 0^*1^* \), \( A \) is also decidable by Theorem 5.22.

5.25 Consider the language \( J = \{w | \text{either } w = 0x \text{ for some } x \in A_{TM}, \text{ or } w = 1y \text{ for some } y \in A_{TM} \} \) defined in problem 5.24. We show that \( J \) is undecidable and \( J \leq_m J \).
Consider the function \( f(w) = 0w \). By the definition of \( J \), \( w \in A_{TM} \) iff \( f(w) \in J \). It implies that \( A_{TM} \leq_m J \). Since \( A_{TM} \) is undecidable, \( J \) is also undecidable by Corollary 5.23. Now define another function
\[
g(w) = \begin{cases} 
x, & \text{if } w = 0x, \\
0, & \text{if } w = 1x, \\
0\langle M, w \rangle, & \text{otherwise. (Where } M \text{ is a TM accepting all inputs.)}
\end{cases}
\]
such that \( w \in J \) iff \( f(w) \in J \). By the existence of such function, we have \( J \leq_m J \).
Note that here we consider the languages in binary encoding.

5.30.c If \( L(M_1) = L(M_2) \), then \( \langle M_1 \rangle \in ALL_{TM} \) iff \( \langle M_2 \rangle \in ALL_{TM} \) because by the definition of the language, whether a TM \( M \) belongs to \( ALL_{TM} \) depends only on its languages \( L(M) \), so \( ALL_{TM} \) is a property of the TM’s language.
And we can simply construct two TM \( N_1, N_2 \) such that \( N_1 \) always accept all inputs and \( N_2 \) reject all inputs. Since \( N_1 \in ALL_{TM} \) and \( N_2 \not\in ALL_{TM} \), it’s a nontrivial property. By Rice’s theorem it’s undecidable.