Context Free Languages

1. For this question, you will prove that context-free languages are NOT closed under intersection. Do this by showing the following:

   - **Part 1:** First, show that $A = \{a^m b^n c^n | m, n \geq 0\}$ is context-free by producing a context-free grammar that generates it.
   - **Part 2:** Do the same, but for language $B = \{a^n b^n c^m | m, n \geq 0\}$
   - **Part 3:** Lastly, find the intersection of these two sets and use the pumping lemma to show that the intersection language is not context-free.

   The grammar for A is:

$$S \rightarrow AB$$
$$A - > aA|\epsilon$$
$$B - > bBc|\epsilon$$

   The grammar for B is similar:

$$S \rightarrow AC$$
$$A - > aAb|\epsilon$$
$$C - > cC|\epsilon$$

   The intersection of A and B is $A \cap B = \{a^n b^n c^n\}$ which was shown to be context-free in class. Thus, context-free languages are not closed under intersection.

2. The grammar below looks like a portion of a reasonable programming language. For this question, you need to first show that this grammar is ambiguous, then re-write the grammar to be unambiguous for the same language. *An ambiguous grammar is one in which there is at least one string that has two unique derivations.*

$$\begin{aligned}
\text{STMT} &\rightarrow \text{ASSIGN}|\text{IF-THEN}|\text{IF-THEN-ELSE} \\
\text{IF-THEN} &\rightarrow \text{if condition then STMT} \\
\text{IF-THEN-ELSE} &\rightarrow \text{if condition then STMT else STMT} \\
\text{ASSIGN} &\rightarrow a := 1
\end{aligned}$$

   The issue here is that we have multiple options for how we place in else statements. We can put in the if-then first and add the else later, or we can put in an if-then-else now and insert a second if-then in between. To make this grammar unambiguous, we redesign it so that the number of ELSE clauses must be decided first, up front, and no other else can be inserted before an ELSE.

$$\begin{aligned}\text{STMT} &\rightarrow \text{ASSIGN}|\text{IF-THEN}|\text{IF-THEN-ELSE}\\ \text{IF-THEN-ELSE} &\rightarrow \text{if condition then ASSIGN else STMT}\\ \text{IF-THEN} &\rightarrow \text{if condition then STMT}\\ \text{ASSIGN} &\rightarrow a := 1\end{aligned}$$

This simple change forces us to end any if-then-else we add.

3. Let us define a new operation using the $\diamond$ symbol as such: if A and B are languages, then $A\diamond B = \{xy|x \in A, y \in B, |x| = |y|\}$. Prove that if A and B are regular languages, then $A\diamond B$ must be a context-free language.

Assume A and B are regular languages, we want to show that $A\diamond B$ is context-free. Given DFAs D_A and D_B (for A and B respectively), we can construct a PDA for $A\diamond B$ as follows:

1. In D_A add a new start state with dummy transition to the old start state that adds a dummy bottom of the stack symbol (\$).
2. For each transition in D_A, push a symbol (X) onto the stack.
3. For final state in A, epsilon transition into the start state of D_B (guess that this is the middle point of the string).
4. For each transition in D_B, pop one instance of symbol X off of the queue.
5. For each final state in D_B, make it a non-final state but add a transition that pops the empty stack symbol (\$) to a new single final state

This PDA recognizes $A\diamond B$. For each transition in A, we push a symbol to count how many characters have been read and epsilon transition to machine D_B on final states of D_A. Machine D_B now pops symbols to ensure that the length of the two strings match. Non-determinism allows us to guess where the midpoint of the string is.