Solution 1:

Correct Answer: (D)

Counter example for S1 is \( L = \{01^n \mid n \geq 1 \} \) for which \( L' \) is the set of binary strings with equal number of 0s and 1s which can be proven non-regular even though \( L \) is regular. Counter example for S2 is \( L = \{(01)^n \mid n \geq 1 \} \) for which \( L' \) is the set of binary strings with equal number of 0s, 1s and 2s which can be proven non-context free even though \( L \) is context free.

Solution 2:

Correct Answer: (A, B, C, D)

Here \( L_1, L_2 \) and \( L_3 \) all are context free. For instance CFG for \( L_1 \) is \( S \rightarrow TR, T \rightarrow aTbb \mid \epsilon, R \rightarrow cR \mid \epsilon \).

Similarly one can make CFGs for \( L_2 \) and \( L_3 \). So option A and B are correct, option C is correct because CFGs are closed under union. Although CFGs are not closed under intersection but still option D is correct because \( L_1 \cap L_2 \cap L_3 \) is a null set hence it is also a CFG.

Solution 3:

Correct Answer: (C)

Let the PDA for \( L_1 \) be \( A_1 \) with states \( p_0, p_1, \ldots, p_k \) and DFA for \( L_2 \) be \( A_2 \) with states as \( q_0, q_1, \ldots, q_j \) where \( q_0 \) is the start state. Let \( A \) be the product automaton of \( A_1 \) and \( A_2 \) where every transition \( \delta(p_i q_j, a, X) = (p_i q_j, Y) \) is replaced by \( \delta(p_i q_j, \epsilon, X) = (p_i q_j, Y) \), basically in \( A \) we can move without reading anything and a state \( p_i q_j \) is a final state of \( A \) if \( p_i \) and \( q_j \) are final states of \( A_1 \) and \( A_2 \) respectively.

Now we can use \( A_1 \) and \( A \) to construct the PDA for \( \text{Div}(L_1, L_2) \) and it will be the PDA \( A_1 \) where no state is a final state and on every state \( p_i \) we make an epsilon transition to the state \( p_i q_0 \) of \( A \) without modifying stack content. Basically when we are at some state \( p_i \) we non-deterministically decide that we have seen \( w \) and then to check if there exist a \( x \in L_2 \) such that \( wx \in L_1 \) we move to \( A \).

Let's look at the second part now. We will use almost the same idea. The PDA \( A' \) for \( \text{Prefix}(L_1) \) consist of two copies of \( A_1 \) where in second copy we change every transition \( (p_i, a, X) = (p_j, Y) \) to \( (p_i, \epsilon, X) = (p_j, Y) \), in first copy make all final states non-final states, start state of first copy will be the start state of \( A' \) and final states of second copy will be the final states of \( A' \). Now from every state in first copy we give an \( \epsilon \) transition to the same state in second copy without altering the stack content.

Solution 4:

Correct Answer: (C)

Let's look at why option A is incorrect. Let \( L_1 = \{ \text{Set of binary strings with no 1s or at least two 1s} \} \), \( L_2 = \{0^n10^m \mid n > m \} \) and \( L_3 = \{0^n10^m \mid n < m \} \). Now the complement of language in option A is \( L_1 \cup L_2 \cup L_3 \). Since \( L_1 \) is clearly regular and one make PDA for \( L_2 \) and \( L_3 \) that means that the complement is also CFL.

For option B we are not giving any proof but one can construct PDA for the complement using similar ideas as used in PDA of question 6.

For option D the complement is union of below languages each of which is a CFL thus option D is also incorrect.

\[
\begin{align*}
L_1 &= \{ \text{Set of strings not in form of } a^*b^*c^* \} \\
L_2 &= \{ a^n b^m c^l \mid n \neq m \} \\
L_3 &= \{ a^n b^m c^l \mid n \neq l \} \\
L_4 &= \{ a^n b^m c^l \mid m \neq l \}
\end{align*}
\]
To prove that option $C$ is correct we have to prove that $L = \{ww \mid w \in \{0, 1\}^*\}$ is non CFL using pumping lemma so that every string $s \in L$ of length at least $p$ can be broken into $uvxyz$ such that the following three conditions hold.

- $uv^ixy^i \in L$ for $i \geq 0$.
- $|vy| > 0$
- $|vxy| \leq p$

Let's take $s = 0^p1^p0^p1^p \in L$. Now consider all cases one by one depending on the positioning of $vxy$.

**Case 1:** $vxy$ is completely contained in the first half of $s$, in which case $uv^2xy^2z \notin L$ as its second half starts from 1 while string starts from 0.

**Case 2:** $vxy$ is completely contained in the second half, in which case $uv^2xy^2z \notin L$ as its first half ends in a 0 while string ends at 1.

**Case 3:** $vxy$ is overlapping first and second half, in which case $uv^0xy^0z \notin L$ as it will be equal to $0^p1^j0^k1^p$ such that $j \neq k$.

**Solution 5:**

**Correct Answer:** (C)

Counter example for option $A$ is string 00 that doesn’t get accepted, for $B$ it is string 111 which is not getting accepted and for $C$ it is 010 which also is not getting accepted.

To see why option $C$ is correct notice that string accepted by PDA is always of odd length because we are pushing $X$ for every symbol seen at $q_1$ and popping $X$ for every symbol seen at $q_2$, but while moving from $q_1$ we are reading 0 and we are not pushing or popping anything from stack. And since in the middle we are reading only 0 thus PDA accepts only odd length strings whose middle element is 0.

**Solution 6:**

**Correct Answer:** (B)

Counter example for option $A$ and $C$ is string $ab$ that doesn’t get accepted and for $D$ it is $abb$ which also is not getting accepted.

Consider the two special cases of strings where number of $a$ is twice the number of $b$. First consider the case where all $as$ precede $bs$ in which case using $(a, \epsilon \rightarrow A)$ transition we first put as many $As$ on stack as $a$ symbol we have seen and then when $b$ starts we pop two $As$ everytime we see a $b$ using $(b, A \rightarrow \epsilon)$ and $(\epsilon, A \rightarrow \epsilon)$. Similarly one can see the other case where all $bs$ precede $as$ where we use transitions $(b, \epsilon \rightarrow B), (\epsilon, \epsilon \rightarrow B)$ and $(a, B \rightarrow \epsilon)$ to accept such strings.

For other general cases we are not giving the proof but advise you to work yourself on such strings. However we are giving three conditions which will be satisfied at any point of time.

- If number of $as$ read so far is less than twice the number of $bs$ then there will be a thread of execution where the stack contains only $Bs$ and the number of $Bs$ will be equal to $(2 \times \text{Number of bs read so far}) - (\text{Number of as read so far})$.
- If number of $as$ read so far is greater than twice the number of $bs$ then there will be a thread of execution where the stack contains only $As$ and the number of $As$ will be equal to $(\text{Number of as read so far}) - (2 \times \text{Number of bs read so far})$. 

2
• If number of $a$s read so far is equal to twice the number of $b$s then there will be a thread of execution where the stack contains just $\#$.

**Solution 7:**

**Correct Answer: (C, D)**

It can be easily observed that the grammar produces strings which can be broken into substrings of type $a^n b^n$. So first is incorrect because it cannot be broken into such types.

Option B is also incorrect because although string $aaabbbabbbabbb \in L(G)$ the only left most derivation is $S \rightarrow SS \rightarrow aTbS \rightarrow aaTbbbS \rightarrow aaabbbT \rightarrow aaabbbTaTb \rightarrow aaabbbabbb$ hence it has only one parse tree.

Option C is correct because the string $abaabbaaabbb$ definitely belongs to $L(G)$ and we can get two parse because once applying $S \rightarrow SS$ we can make first $S \rightarrow T$ and second $S \rightarrow SS$ or we can make first $S \rightarrow SS$ and second $S \rightarrow T$. In both ways we can derive the string and both ways leads to two different parse trees.

Option D is also correct as string belongs to $L(G)$ and there are five parse trees. We are not giving parse trees here but one can try all combination like above option to get five of them.

**Solution 8:**

**Correct Answer: (A,C)**

In all the PDAs at $q_1$ we are non-deterministically deciding that we have seen half of the string and we have pushed the same number of $X$ on stack as the number of symbols we have seen. Then in $q_2$ and $q_3$ we pop $X$ everytime we see a new symbol.

In option A we move from $q_2$ to $q_3$ when we see the very first 1 in second half and then we move to $q_4$ when we see $\#$.

In option B we move from $q_2$ to $q_3$ when we see the very first 1 in second half and then we move to $q_4$ when we see $\#$. But the problem is that it is not reading anyother 1s at $q_2$ or $q_3$, so basically it is accepting strings whose second half contains exactly one 1.

In option C we move from $q_2$ to $q_3$ when we see the last 1 in second half and then we move to $q_4$ when we see $\#$.

In option D we are reading only 1s in second half at $q_2$ and $q_3$ so this PDA is also not correct.

**Solution 9:**

**Correct Answer: (C)**

The description of DPDA for $L_1$ is as follows. We first add $\#$ to stack before reading any symbol then we start reading all the symbols and also push them on the stack in the same order. When we see $\epsilon$ we do not push it on stack but move to some other state where we now read the rest of the string and while reading we pop a symbol from stack if the symbol read and on top of stack is same. Finally on seeing $\#$ we move to the final state.

$L_2$ is a tricky one, since there is no restriction on the size of $w$ so we can always take $w = \epsilon$, that means that every string $s$ can be broken into $xwx^\epsilon$ where $x = s$ and $w = \epsilon$. Thus $L_2$ is the set of all binary strings and thus regular and DCFL both.
Solution 10:

Correct Answer: (A, D)

A PDA can store infinite information in its stack so A is correct and B is incorrect. C is incorrect because DPDA can accept all and only DCFL which is a proper subset of CFL.

To prove that D is true we need to show that if a CFL L is accepted by a general PDA then it can also be accepted by a PDA in which we either push or pop in every step but not both. Let M be the general PDA for L, then for every transition \((p, a, X) = (q, Y)\) of the type which pops and push both we add a dummy state \(r\) and replace the transition \((p, a, X) = (q, Y)\) with two transitions \((p, a, X) = (r, \epsilon)\) and \((r, \epsilon, \epsilon) = (q, Y)\). And we are done.