Solution 1:

Correct Answer: (A), (C)

Language $A$ is nothing but the set of all strings $w \in \{0,1\}^*$ because every such string $w$ can be partition into $x$ and $y$ such that $w = xy$ and number of $0$'s in $x$ $\geq$ number of $1$'s in $x$. Take $x = \epsilon$. Hence $A$ is regular. Every regular language is also context free.

Solution 2:

Correct Answer: (D)
Consider the following grammar: $S \rightarrow SS \mid aSaSb \mid aSbSa \mid bSaSa \mid \epsilon$

It is easy to observe that all strings generated by the grammar have twice many $a$ as $b$. The crucial part is prove reverse direction that all strings which have twice many $a$ as $b$, can be generated by this grammar. Notice that length of the string with this property will be a multiple of 3.

Define $f(w)$ = number of $a$'s $-2 \times$ number of $b$'s, in string $w$.

We can prove our claim by induction on the length($n$) of the string.

**Base case:** $n = 0$, grammar generates $\epsilon$.

**Induction hypothesis:** Assume that grammar generates all strings of length $k \leq n$ such that $k$ and $n$ are multiple of 3 and string contain twice many $a$ as $b$. Now we will show that grammar can generate all string $w$ of length $n + 3$ with same property.

Case 1: If $w$ can be partition into non empty substrings $x$ and $y$ such that $w = xy$ and $f(x) = f(y) = 0$ then we are done, because $x$ and $y$ both can be generated by the grammar (According to induction hypothesis) and we can concatenate them using production rule $S \rightarrow SS$ to obtain $w$.

Case 2: Assume we can not find such partition then there following sub cases:

Case 2(i): $f(x) > 0$ for all $x$ as described above. In that case we can infer that string will start with $aa$ otherwise $f(x)$ will not be greater than 0 if we choose $x$ to some other string than $aa$. We can also infer that in this case string must end with $b$. Because if string end with $a$ we can write $w = xa$, we assumed that $f(x) > 0$, this gives $f(w) > 0$. But we know $f(w) = 0$. We can conclude that $w$ will be of form $aazb$. By induction hypothesis we know we can generate $z$, and we can generate $w$ using production rule $S \rightarrow aSaSb$ by replacing first $S$ with $\epsilon$ and second $S$ with $z$.

Case 2(ii): $f(x) < 0$ for all $x$ as describe above. we can use same argument as above to handle this case.

Case 2(iii): Here we consider the case when $f(x)$ switches its sign (from moving left to right) but never become 0 in between. Observe that $f(x)$ can only switch from positive to negative without being, 0 in between (if $f(x) = 1$ then $f(xb) = -1$). We infer that string will start with an $a$ and we will encounter a $b$ in between where $f(x)$ switches its sign and string ends with an $a$. So $w$ will have form $w = aubvz$. Notice that $f(au) > 0$ and $f(abu) < 0$, this implies $f(u) = 0$, and $f(v) = 0$. Now according to induction hypothesis we can generate $u$ and $v$ and string $w$ can be generated using production rule $S \rightarrow aSbSa$ by replacing first $S$ by $u$ and second $S$ by $v$.

Solution 3:

Correct Answer: (B)

We can easily prove that $A$ is not CFL using pumping lemma. Let $p$ be the pumping length given by pumping lemma. Choose string $w = a^p b^p b a^p \# a^p b^p b a^p$. We can write every string $w' \in A$ as $w' = l \# r$ where $l = r$. Now if in some partition of $w$ in $wuvxyz$, substring $vux$ contains part of only $l$ or only $r$. Then on pumping string $wv^iwx^iy$ for $i \geq 0$, we will only increase or decrease the length of one of $l$ or $r$. 

1
So the resultant string will not belong to $A$ because if the length of $l$ and $r$ is not same the they cannot be same.

Other case is when $vwx$ contain some part of $l$ and some part of $r$. One thing is clear that $u$ and $v$ cannot contain symbol #, because on pumping it will create more # symbol which is not allowed in any string in $A$. It is also necessary length of $l$ equals to length of $r$. Since $|vwx| \leq p$, we can say $v$ and $x$ contain only $a$'s. Any valid partition of this type will look as follow.

$$w = a^p b^p b^p a a ... a \# a a a a a a b^p b^p a^p$$

Now pumping the string $uv_iwx_iy$ for different $i$'s will generate string of type $a^p b^p b^p a^{p+k} \# a^{p+k} b^p b^p a^p$. Clearly this string does not belong to $A$ for any $k > 0$. Hence $A$ is not a CFL.

Following grammar generates language $B$:

$$S \rightarrow S_1 | S_2 | \epsilon$$

$$S_1 \rightarrow S_1' C$$
$$S_1' \rightarrow aS_1'b \mid \epsilon$$
$$C \rightarrow cC \mid \epsilon$$

$$S_2 \rightarrow AS_2'$$
$$S_2' \rightarrow bS_2'c \mid \epsilon$$
$$A \rightarrow aA \mid \epsilon$$

Hence $B$ is a CFL and $A$ is not.

**Solution 4:**

Correct Answer: (B)(C)

Grammar is ambiguous. String 001101 have two different parse tree as follow:

$S \rightarrow 0A \rightarrow 00AA \rightarrow 001A \rightarrow 0011S \rightarrow 00110A \rightarrow 001101$

Another parse tree is as follow:

$S \rightarrow 0A \rightarrow 00AA \rightarrow 001SA \rightarrow 0011BA \rightarrow 00110A \rightarrow 001101$

String 001110 can be generated as follow:

$S \rightarrow 0A \rightarrow 00AA \rightarrow 001A \rightarrow 0011S \rightarrow 00111B \rightarrow 001110$

**Solution 5:**

Correct Answer: (A)(B)

(ii) Unambiguous grammar for this language is

$S \rightarrow aaS \mid \epsilon$

(i) The ambiguous grammar for this language is:

$S \rightarrow SS \mid aSa \mid \epsilon$

(iii) Another unambiguous grammar for this language is:

$S \rightarrow Saa \mid \epsilon$

(iv) Another ambiguous grammar for this language is:

$S \rightarrow SS \mid aaS \mid \epsilon$

Hence option only (A) and (B) are correct.
Solution 6:
Correct Answer: (C)

This can be easily proved by induction on the length of string as in explanation of question 2.

Solution 7:
Correct Answer: (A)(C)

Notice that the grammar:

\[ S \rightarrow 1S \mid 0A0S \mid \epsilon \]
\[ A \rightarrow 1A \mid \epsilon \]

Generates all string which contain even number of 0’s.
Let say there is a string which contains even number of 0’s. We see how that string can be generated by this grammar. let \( s \) be such string. If \( s \) starts with 1 then we use production \( S \rightarrow 1S \) to generate any number of consecutive 1 as there in \( s \) before the first occurrence of a 0, and as we see first 0 we use the production \( S \rightarrow 0A0S \), because we know that string contain even number of 0’s so there will be atleast one more 0, which will be mapped by second 0 in the the production \( 0A0S \). Now if there are some 1 in between the two occurrence of 0 in \( s \) then we can use the production \( A \rightarrow 1A \) to generate same number of 1. Now the rest of the part of \( s \) can again be generated in the same way by \( S \) in the production \( 0A0S \).

Hence option (A) and (C) are correct.

Solution 8:
Correct Answer: (C)

If we start deriving the string from the root node(Start symbol), notice that at each step when we replace a variable by some production rule then actually we create an internal node. Since the string is derived in total \( m \) step, we will have \( m \) internal node in the parse tree. Also note that there is no \( \epsilon \) on the right hand side of any production rule, we can say that there will be \( n \) leaf in the parse tree(Because length of the the string is \( n \)). So there will be total \( m+n \) node in the derivation tree.

Note: A node is called internal if it has atleast one child.

Solution 9:
Correct Answer: (C)

We can write \( A \) as \( A = \{a^n b^n c^m \mid n, m \geq 0\} \)

Consider the following grammar \( G_1 \):
\[ S \rightarrow S_1S_2 \]
\[ S_1 \rightarrow aS_1b \mid \epsilon \]
\[ S_2 \rightarrow bS_2c \mid \epsilon \]

Consider another grammar \( G_2 \):
\[ S \rightarrow aSc \mid S_1 \]
\[ S_1 \rightarrow bS_1c \mid \epsilon \]

It is very easy to verify that \( G_1 \) and \( G_2 \) generate language \( A \) and \( B \) respectively.

Solution 10:
Correct Answer: (A)(C)

Instructor described the procedure in lecture 18 to convert any CFG in to Chomsky normal form.
We know that there are some languages called "Inherently ambiguous", which does not have any unambiguous grammar.
Yes, number of leaves can be more than the length of the string (If some leaf node are $\epsilon$).
We know that set of regular languages is the subset of set of context free language. Hence there does not exist any regular languages which not context free.