CLASS TEST

S.No. : 02SKCS_FGHI_14122023

MADE EASY
India's Best Institute for IES, GATE & PSUs

Delhi | Bhopal | Hyderabad | Jaipur | Pune | Bhubaneswar | Kolkata

Web: www.madeeasy.in | **E-mail:** info@madeeasy.in | **Ph:** 011-45124612

COMPILER DESIGN

COMPUTER SCIENCE & IT

Date of Test : 14/12/2023

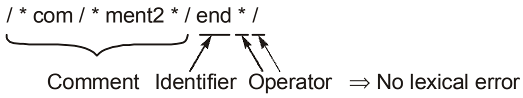
ANSWER KEY

1.	(d)	7.	(c)	13.	(c)	19.	(d)	25.	(a)
2.	(d)	8.	(d)	14.	(c)	20.	(c)	26.	(c)
3.	(b)	9.	(d)	15.	(b)	21.	(a)	27.	(c)
4.	(c)	10.	(d)	16.	(c)	22.	(b)	28.	(c)
5.	(c)	11.	(d)	17.	(a)	23.	(c)	29.	(c)
6.	(b)	12.	(b)	18.	(d)	24.	(a)	30.	(b)

DETAILED EXPLANATIONS

1. (d)

The given program contain no lexical error even through it contains syntax errors. In line number "5", comment started and searches for the first close comment pattern when it finds, it consider a comment. There is no start comment pattern (/*)but there is end comment at last in line 5, hence it is not lexical error but it is syntax error.



2. (d)

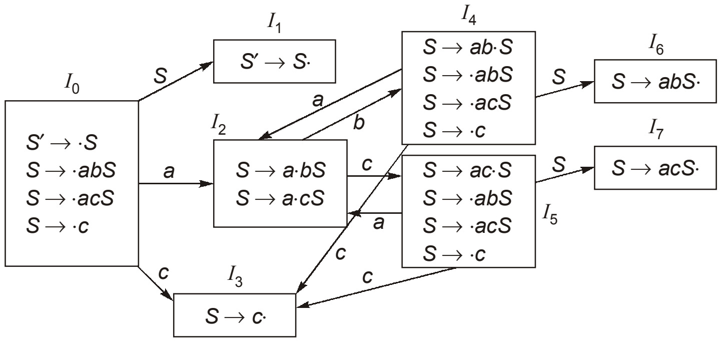
FIRST (X) = {s, e, ε}

FOLLOW (X) = {e, c, s, \$}

	c	s	e	\$
X	$X \to \varepsilon(4)$	$X \to sX(3)$ $X \to \varepsilon(4)$ $= E_1$	$X \to Yc(2)$ $X \to \varepsilon(4)$ $= E_2$	$X \to \varepsilon(4)$

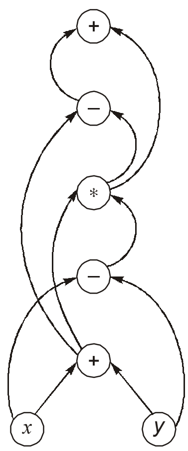
∴ $E_1 = \{3, 4\}$ and $E_2 = \{2, 4\}$

3. (b)



Total 8 states.

4. (c)



5. (c)

The drawback in quarduple representation is one extra field required to store the result.

In triple representation their is no need of extra field to store the result, So it require less space.

Both (a) and (b) are correct.

6. (b)

$S \to AA \to aA \to aa$

$S \to AA \to aA \to abA \to aba$

$S \to AA \to aA \to aAb \to aab$

$S \to AA \to Aa \to bAa \to baa$

$\therefore$ {*aa*, *aba*, *aab*, *baa*} can be generated within 4 steps.

7. (c)

If grammar contain left recursion, then recursive descent parser call itself every time and not reaching to terminal which leads it to an infinite loop.

Every LR parser is always unambiguous.

8. (d)

Lexical analyser uses symbol table to identity token and storing token into table.

Syntax analyser uses symbol table to generate parse tree.

Semantic analyser uses symbol table to identify the type of identifier or meaning to perform appropriate action.

9. (d)

Control link points to the activation record of the caller.

Access link points to the activation record associated with nearest enclosing scope of the subprogram definition.

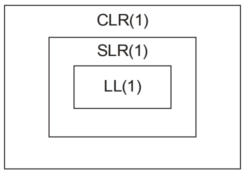
So, control link, access link and temporary variable are part of activation record.

10. (d)

LL(1) is CLR(1).

SLR(1) is also CLR(1).

CLR(1) need not be LL(1) or SLR(1).



11. (d)

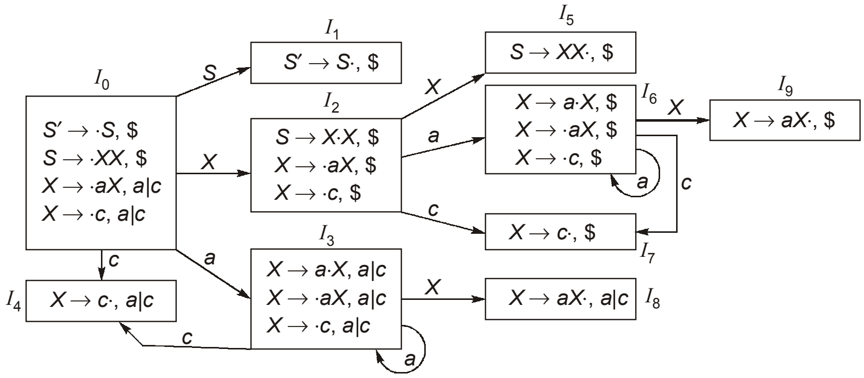
LR(1) item set is given below

$S' \to .S, \$$

$S \to .X X, \$$

$X \to .aX, a \mid c$

$X \to .c, a \mid c$

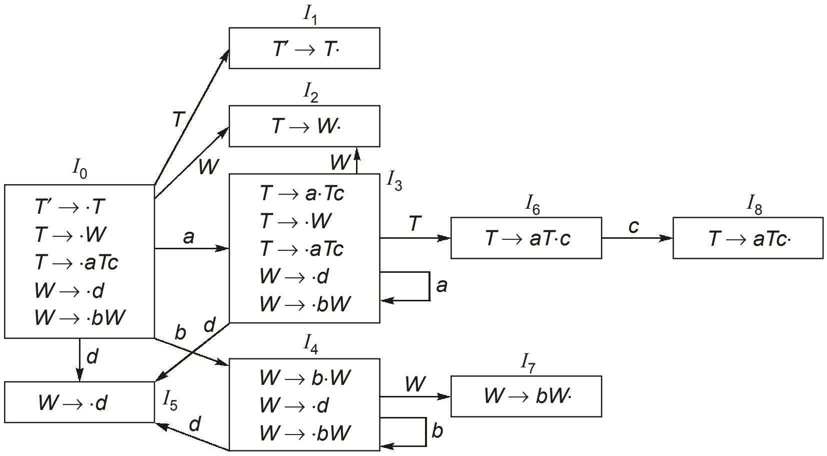


Total 10 states in CLR(1) parser.

Here, state (I_3, I_6), (I_4, I_7) and (I_8, I_9) have same transition item over *a* and *c* respectively which only differ in look ahead symbols. So to make LALR(1) combines $(I_3, I_6 = I_{36})$, $(I_4, I_7 = I_{47})$ and $(I_8, I_9 = I_{89})$.

So total number of states in LALR(1) is 7 and reduced states is 3.

12. (b)

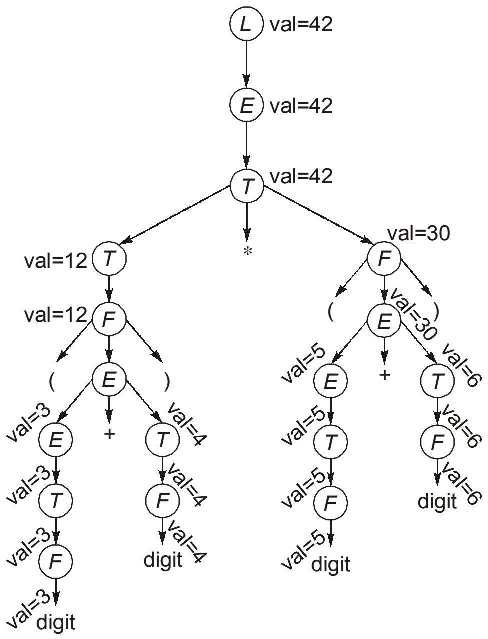


13. (c)

```
for (i = 1; i ≤ n ; i ++)                 i = 1;
{                                         if i ≤ n then
  a = b + c;                   ⟹              t1 = b + c;
  a = a * d;                                  a = t1;
}                                             a = t1 * d;
                                              i = i + 1 goto (2)
                                          else
                                              end
```

Intermediate code represent option (c).

14. (c)



15. (b)

The given grammar generate two derivation trees for the string '*abc*'.

1st derivation tree   2nd derivation tree

Hence, given grammar is ambiguous.

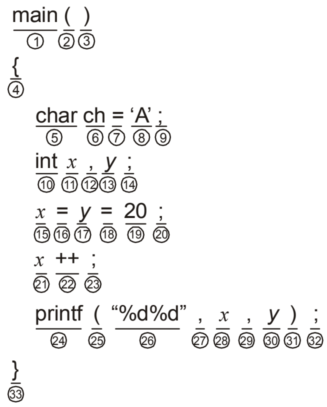
16. (c)

The loader performs relocation where address of data and address of instruction can be changed.

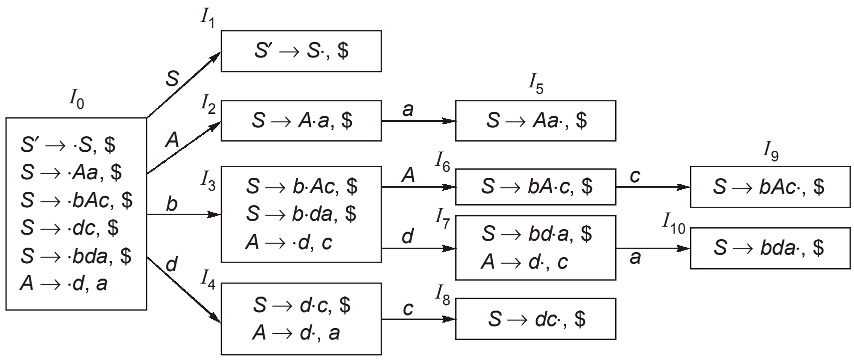
17. (a)

Regular expression is used in lexical analysis to identify the tokens.

18. (d)

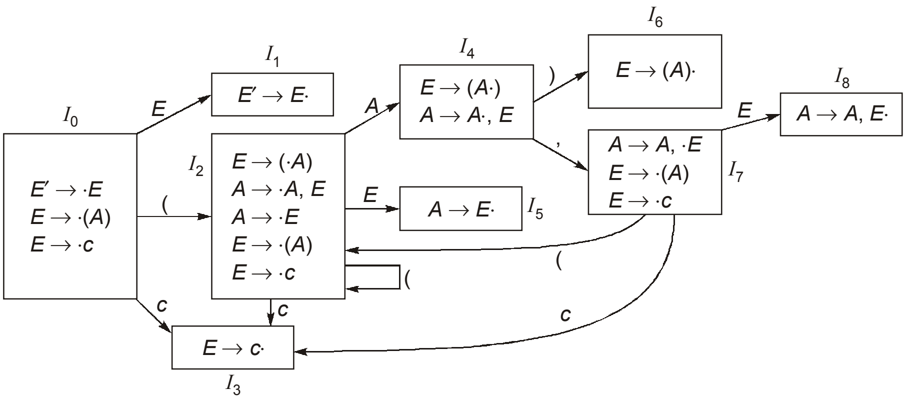


19. (d)



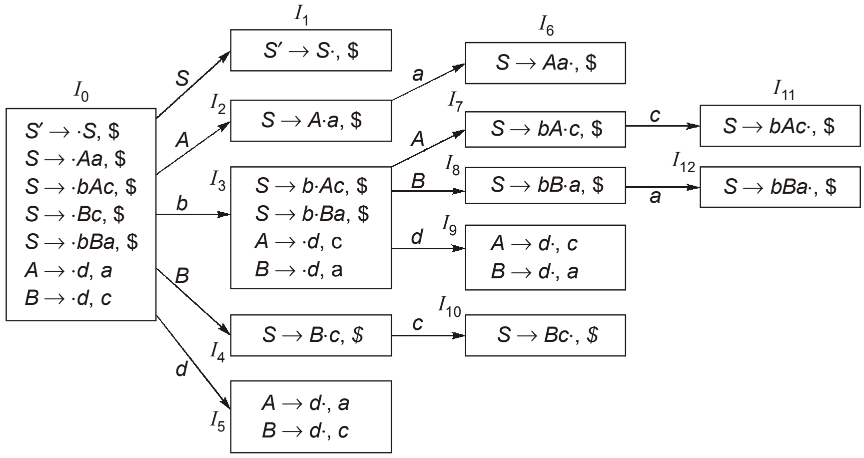
The number of states presents in LALR(1) parser is 11.

20. (c)



Since $A \to A\cdot, E$ and $E \to (A\cdot)$ present in I_4 but $E \to c\cdot$ not present with $E \to (A\cdot)$ or $A \to A\cdot, E$.

21. (a)



Since their is no conflict in any state in parsing table. So given grammar is LR(1) but when we merge I_5 and I_9 the resulting state will be

$$I_{5+9} = A \rightarrow d\cdot, a|c$$
$$B \rightarrow d\cdot, a|c \text{ creates reduce-reduce conflict.}$$

So given grammar is not LALR(1). Therefore given grammar is LR(1) but not LALR(1).

22. (b)

$$\text{FOLLOW (S)} = \{c, \$\}$$
$$\text{FIRST (S)} = \text{FIRST } (M\,N\,z\,S\,c) = \{a, b, z\}$$

23. (c)

Static storage allocation does not support recursion because memory will be allocated at compile time itself and at compile time we don't know how much memory is required. So it is the drawback.

In stack allocation when one function complete it execution then it will be poped out from stack. If in near future again that function called it will be evaluated again. So it consume lots of time to evaluate same function again and again. So it is the drawback.

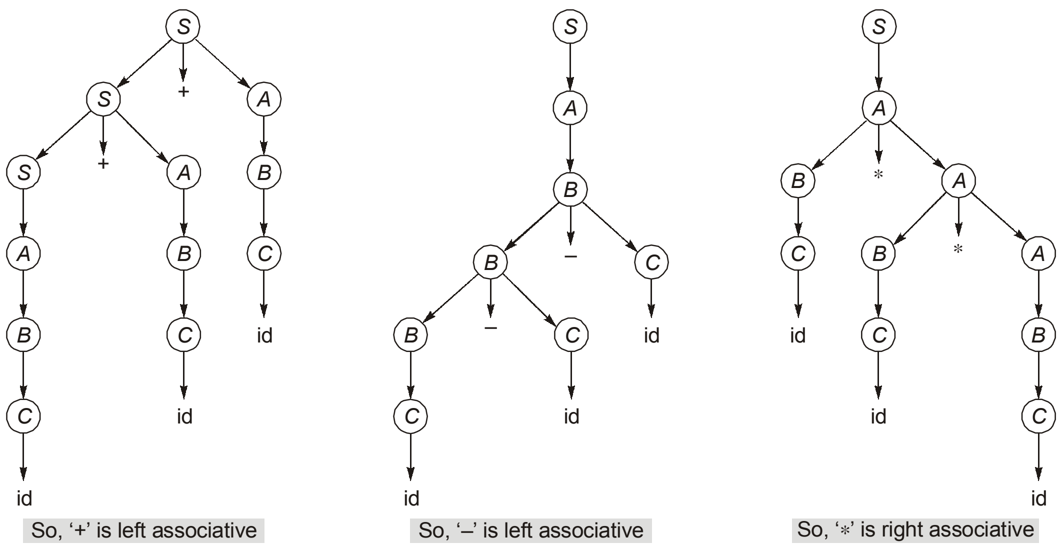
24. (a)

Option (b) contain two consecutive variables so not operator grammar.

Option (a) is operator grammar because it does not contain two consecutive variables and null production.

25. (a)

Consider 3 strings id + id + id, id – id – id and id ∗ id ∗ id.



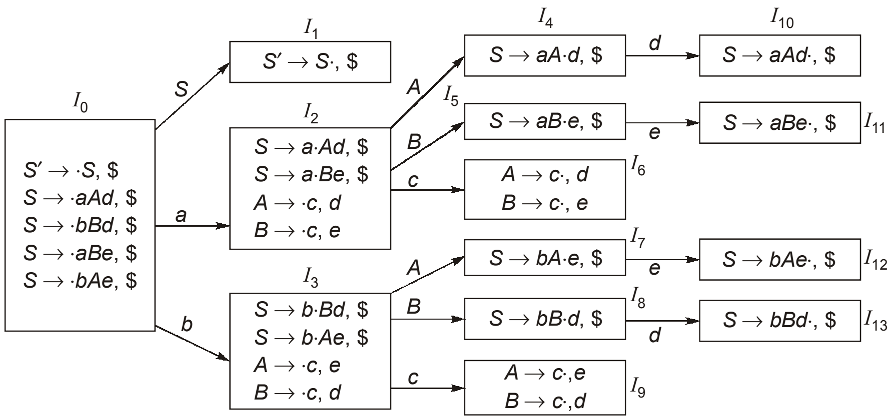
26. (c)
In static single assignment, every variable assigned only once and that variable can be used any number of times without assignment.
Expression : $a + b/9 + c - d * 4 + e$
$t_1 = b/9$;
$t_2 = a + b/9$;
$t_3 = t_2 + c$
$t_4 = d * 4$
$t_5 = t_3 - t_4$
$t_6 = t_5 + e$
$\therefore$ 6 temporary variables are required.

27. (c)
$G_1 : S \rightarrow \underline{A}\, a$
$B\, \underline{C}\, a$
$\underline{B}\, S\, a\, a$
$\in S\, a\, a$
Since it contain production $S \Rightarrow S\, a\, a$ in which S call itself. So left recursion present.
$G_2 : A \rightarrow \underline{B}\, C$
$\in \underline{C}$
$A\, D$
Here grammar contain production $A \Rightarrow AD$ i.e. A call itself so left recursion is present.
$\therefore$ Both G_1 and G_2 contain left recursion.

28. (c)
For grammar $S \rightarrow Sa \mid d \mid Sb \mid e$
Non-left recursive grammar is
$$S \rightarrow dS' \mid eS'$$
$$S' \rightarrow aS' \mid bS' \mid \in$$
By removing null production from above non-left recursive grammar resulted grammar is
$$S \rightarrow eS' \mid dS' \mid e \mid d$$
$$S' \rightarrow bS' \mid aS' \mid b \mid a$$
So both (a) and (b) are non-left recursive for given left recursive grammar.

29. (c)
$$S \rightarrow aS \mid AB$$
$$A \rightarrow bA \mid B$$
$$B \rightarrow cB \mid d$$
The above grammar is LL(1) because
FIRST $(aS) \cap$ FIRST (AB) = $\{a\} \cap \{b, c, d\} = \phi$ and
FIRST $(bA) \cap$ FIRST (B) = $\{b\} \cap \{cd\} = \phi$ and
FIRST $(cB) \cap$ FIRST(d) = $\{c\} \cap \{d\} = \phi$
So it is LL(1), also LR(1) because LL(1) grammar is always LR(1) grammar.

30. (b)



There are 14 states in LR(1).

■■■■