CLASS TEST S.No.:01SKCS_ABCDEFGHI_17625

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

COMPLIER DESIGN

COMPUTER SCIENCE & IT

Date of Test : 17/06/2025

ANSWER KEY

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

DETAILED EXPLANATIONS

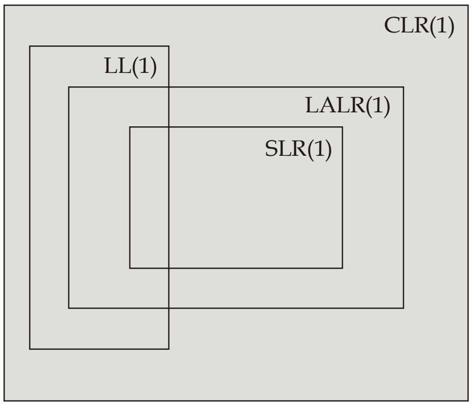
1. **(c)**

Simple two-pass assembler:

1. Allocates space for the literals.
2. Computers the total length of program (syntax analysis).
3. Builds the symbol table for the symbols and their values.

2. **(d)**

Relation between LL(1), SLR(1) and CLR(1) and LALR(1) given below:



S_1 is false, S_2 is true and S_3 is false.

3. **(b)**

In both stack and heap allocation, memory allocated at runtime.
Static allocation does not support recursion. However, in stack allocation, storage is organized as a stack and activation records are pushed and popped as activation begin and end respectively.

4. **(c)**

E and F both have left recursion rule.
So *, + both are left associative.

5. **(d)**

$$S \to Aa \mid b$$
$$A \to Ac \mid bd$$

Here, $A \to Ac$ has left recursion removing it we have

$$A \to bd\, A'$$
$$A' \to cA'$$

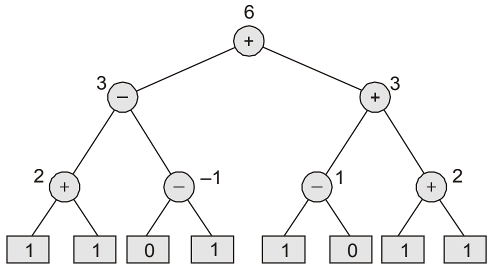
So, the resultant grammar is

$$S \to Aa \mid b$$
$$S \to bd\, A'$$
$$A' \to cA' \mid \in$$

6. **(d)**

$$\text{FIRST } \{B\} = \{b, \in\}$$
$$\text{FOLLOW } \{B\} = (\text{FIRST (C)} - \{\in\}) \cup \text{FOLLOW (A)} \cup \text{FIRST (b)}$$
$$= \{c\} \cup \{\$\} \cup \{b\}$$
$$= \{b, c, \$\}$$

7. (a)



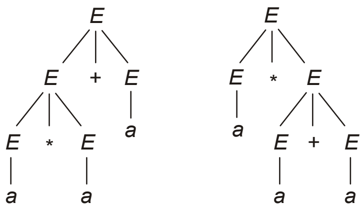
8. (c)

x is inherited.

y is synthesized.

9. (a)

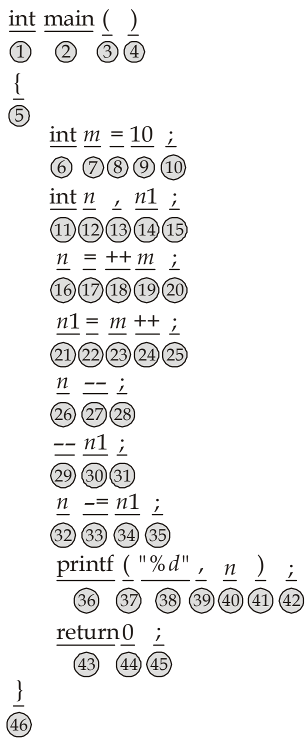
$a * a + a$



Since, two parse trees are possible.

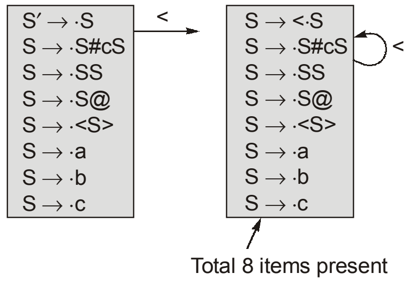
Hence, grammar is ambiguous.

10. (c)



Number of tokens are 46.

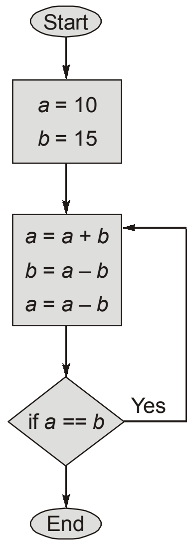
11. (b)



12 (c)

Stack contains only a set of viable prefixes.

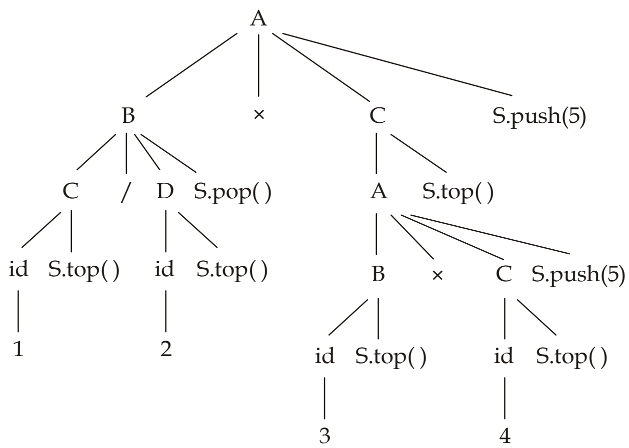
13. (d)

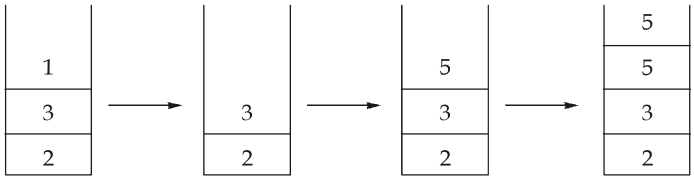


Here, $X = 5, Y = 5$

So, $X + Y = 10$

14. (a)





15. **(d)**

3 – 2 * 4 \$ 2 \$ 3

= (((3 – 2) * 4) \$ 2) \$ 3

= ((1 * 4) \$ 2) \$ 3

= (4 \$ 2) \$ 3

= $16^3 = 4096$

16. (b)

Checkit out using following code.

MOV	a, R_1	
opr	b, R_1	$t_1 = a + b$
MOV	d, R_2	
opr	c, R_2	$t_2 = c + d$
opr	e, R_2	$t_3 = e - t_2$
MOV	t_3, R_1	
opr	t_1, R_1	$t_4 = t_1 - t_3$

Minimum number of MOV instructions required = 3.

17 **(d)**

$$\text{IN} = \text{USE} \cup \{\text{OUT} - \text{DEF}\}$$
$$\text{OUT} = \cup\ \text{IN (successor)}$$

			FIRST GO		SECOND GO		THIRD GO	
Block	**USE**	**DEF**	**IN**	**OUT**	**IN**	**OUT**	**IN**	**OUT**
B1	{m, n}	{a, i, j}	{m, n}	{i, j}	{m, n}	{a, i, j}	{m, n}	{a, i, j}
B2	{i, j}	{i, j}	{i, j}	{a}	{a, i, j}	{a, j}	{a, i, j}	{a, j}
B3	ϕ	{a}	ϕ	{a}	ϕ	{a, j}	{j}	{a, j}
B4	{a}	{i}	{a}	{i, j}	{a, j}	{a, i, j}	{a, j}	{a, i, j}

∴ The variables that are live at exit (i.e. live out) of each basic block are

$$B1 = \{a, i, j\}, \quad B2 = \{a, j\}$$
$$B3 = \{a, j\}, \quad B4 = \{a, i, j\}$$

18. **(c)**

	FIRST	FOLLOW
S	{a, b, ∈}	{a, b, \$}
A	{a, b, ∈}	{a, b}
B	{a, b, ∈}	{a, b, \$}

LL(1) Parsing table:

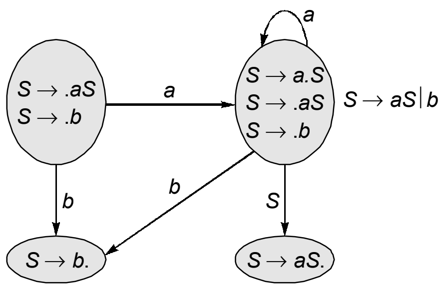
	a	b	$
S	$S \to aAbB$ $S \to \epsilon$	$S \to bAbB$ $S \to \epsilon$	$S \to \epsilon$
A	$A \to S$	$A \to S$	
B	$B \to S$	$B \to S$	$B \to S$

There are only 2 entries in which there are multiple productions.

19. (a)

- Statement I and IV is correct.
- Type checking is done at semantic analysis phase.
- Target code generation is dependent based on the machine.
- Symbol table is accessed during lexical, syntax and semantic analysis phase.

20. (a)



Some possible stack contents are

aaS, *ab*, *b*, etc.

21. (c)

Here × is highest and + is next highest.

Associativity does not matter.

Select the best way so that less number of temporary variables will be created.

$a + b \times c + d - e - a + b \times c$

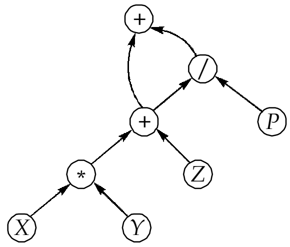
$$= ((a + (b \times c)) + d) - e - (a + (b \times c))$$
$$= (((a + (b \times c)) + d) - e) - (a + (b \times c))$$

Equivalent 3-address code is:

$$t_1 = b \times c$$
$$t_2 = a + t_1$$
$$t_1 = t_2 + d$$
$$t_1 = t_1 - e$$
$$t_1 = t_1 - t_2$$

∴ Only two temporary variables are used.

22. **(d)**



Total 8 edges

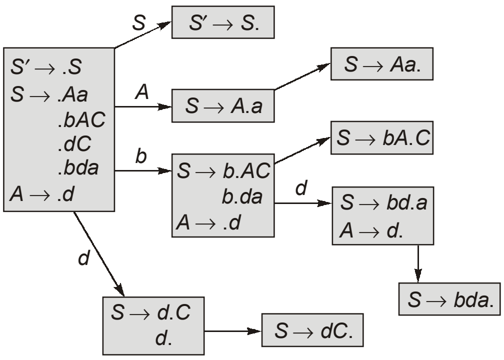
23. **(c)**

$$t_1 = a * b$$
$$t_2 = -t_1$$
$$t_3 = c + d$$
$$t_4 = -(a * b) + (c + d)$$
$$t_1 = a + b$$
$$t_2 = t_1 + t_3$$
$$t_5 = -(a * b) + (c + d) - (a + b + c + d)$$

24. **(d)**

- Statement S_1 and S_2 are correct.
- Statement S_3 is incorrect. Heap and stack both are present in main memory.

25. **(c)**



Given grammar is not SLR (1), but LAR (1).

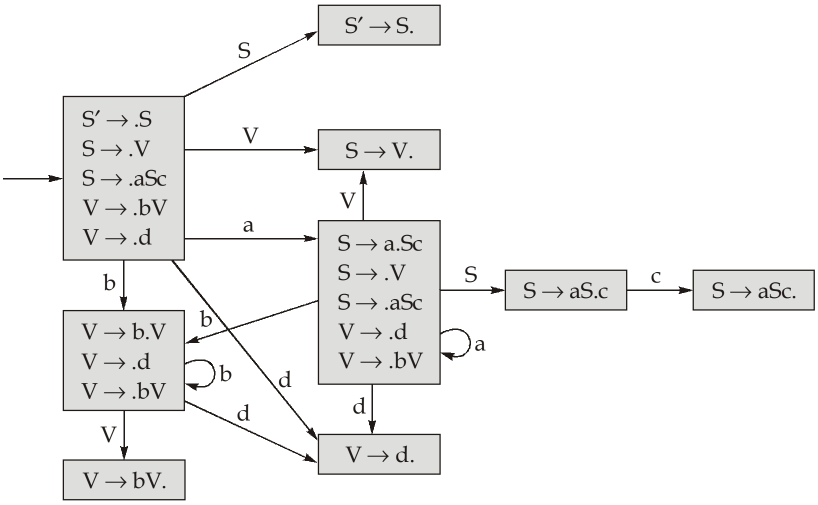
26. **(c)**

Static scoping means that *x* refers to the *x* declared innermost scope of declaration. Since '*h*' is declared inside the global scope, the innermost *x* is the one in the global scope (it has no access to the *x*'s in '*f*' and '*g*', since it was not declared inside them), so the program prints 23 twice.

Dynamic scoping means that *x* refers to the *x* declared in the most recent frame of the call stack. '*h*' will use the *x* from either '*f*' or '*g*', whichever one that called it so the program would print 22 and 45.

27. (d)

SLR Parser:



Zero inadequate states since no SR conflict or RR conflict is present.

28. (c)

Since in grammar every # per for multiplication between two operands. So, $ much represents substraction to get 512.

$8 \times 12 - 4 \times 16 = 12 \times 4 - 2$

$8 \times 8 \times 4 \times 2 = 2^3 \times 2^3 \times 2^2 \times 2^1$

$= 2^9$

$= 512$

29. (d)

Recursion can not be implemented using static allocation.

Recursion can be implemented using dynamic allocation.

30. (d)

(a) i%2 is inner loop invariant, it can be moved before inner loop.

(b) 4*j is common sub-expression appeared in two statements.

(c) 4*j can be reduced to j<<2 by strength reduction.

(d) There is no dead code in given code segment. So there is no scope of dead code elimination in this code. Hence only option (d) is FALSE.

■■■■