Branch : CSE & IT Batch : Hinglish

Compiler Design

Intermediate Code & Code Optimization

DPP

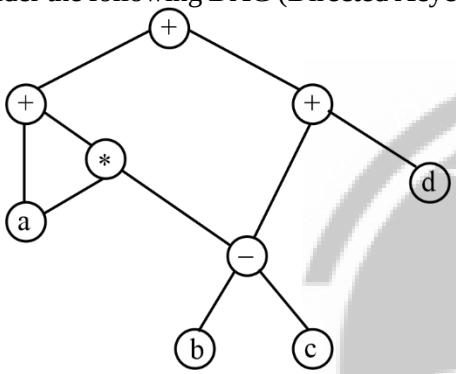
[MCQ]

1. The three addresses code involves____.

(a) At most 3 addresses.
(b) Exactly 3 addresses.
(c) At least 3 addresses.
(d) No unary operator.

[MCQ]

2. Consider the following DAG (Directed Acyclic graph):



Which of the following is the correct expression for given DAG?

(a) [a ∗ a + (b – c) ∗ (b – c) + d]
(b) [a + a ∗ (b – c) + (b – c) + d]
(c) [a + a ∗ (c – b) + (c – b) + d]
(d) [a ∗ a + (c – b) ∗ (c – b) + d]

[MCQ]

3. Type checking is performed by:

(a) Lexical analyses
(b) Syntax analyses
(c) Semantic analyses
(d) Intermediate code generator

[MCQ]

4. Which of the following are valid three addresses code (TAC)?

(a) if a > b goto C (b) return 0
(c) x[i] = y; (d) a = b∗c

[MCQ]

5. Which of the following is/are correct about abstract syntax tree (AST)?

(a) AST is also used in program analysis and program transformation systems.
(b) It is a tree representation of the abstract syntactic structure of source code written in a programming language.
(c) It has no impact on the final output of the compiler.
(d) It is the result of syntax analysis phase of a compiler.

[NAT]

6. Consider the following expression:

p + q ∗ r + s – t – p + q ∗ r

How many minimum number of temporary variable created in three addresses code?

Assume precedence order from lowest to highest is –, +, and ∗; and consider associativity of + & ∗ is not important but – is left associative.

[NAT]

7. Consider the following intermediate code:

1. loc = –1
2. i = 0
3. if (i < 100) goto 5
4. goto 13
5. $t_1 = 4 * i$
6. goto 3
7. $t_2 = A[t_1]$
8. if t_2 = x goto
9. goto 11
10. loc = i
11. $t_3 = i + 1$
12. $i = t_3$
13. goto 3

How many number of basic block from the given code?

[MCQ]

8. Consider the following expression.

((x ∗ x) ∗ (x ∗ x) ∗ ((x ∗ x) ∗ (x ∗ x)))

The total number of internal nodes in DAG representation are _____.

(a) 3 nodes and 5 edges
(b) 4 nodes and 7 edges
(c) 3 nodes and 6 edges
(d) 4 nodes and 6 edges

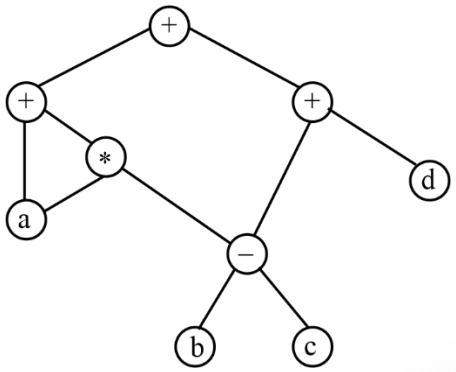
Answer Key

1. **(a)**
2. **(b)**
3. **(c)**
4. **(a, b, c, d)**
5. **(a, b, d)**
6. **(0)**
7. **(10)**
8. **(d)**

Hints & Solutions

1. (a)
A three address code has at most three address location to calculate the expression.
Hence, option A is the correct answer.

2. (b)



The correct expression is
a + a * (b – c) + d
So, option (b) is the correct answer.

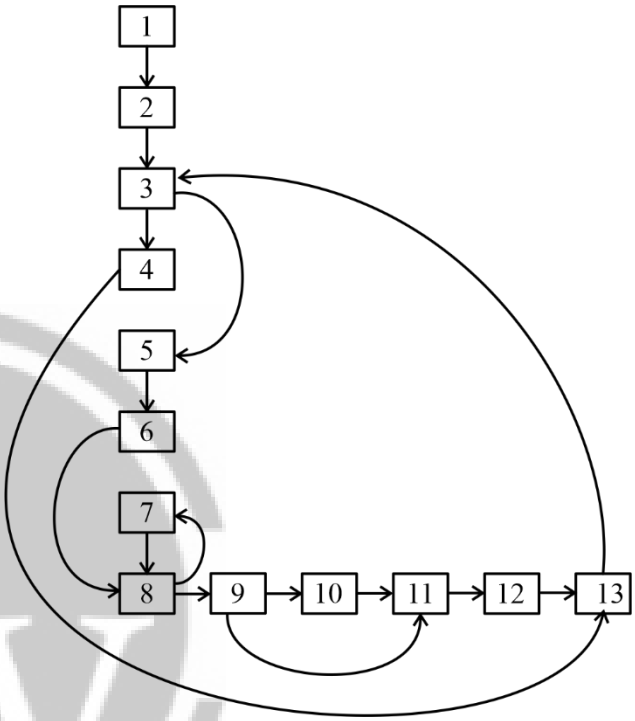
3. (c)
Semantic analysis mainly seals with the meaning of program and its execution. Type checking is an important aspect of semantic analysis where each operator should be compatible with its operands.

4. (a, b, c, d)
A three addresses code instruction can have at most three operand. So, all of the options are correct.

5. (a, b, d)
Abstract syntax tree has impact on the final output of the compiler. So, option (c) is incorrect else all options are correct.

6. (0)
Given,
(((p + (q * r) + s) – t) – (p + (q * r))
So, three-address code for the expression is as follows:
q = q * r
r = p + q
s = s – t
s = s – p
r = r + s
r = r + q
So, minimum 0 temporary variable are required.

7. (10)
Basic block: It is the collection of three address code statements from leader to next leader without including the next leader is known as the basic block.
In the given three-addresses code, there are 10 basic blocks:



8. (d)
DAG representation of given expression

4 nodes and 6 edges. So, option (d) is correct answer.

Branch : CSE & IT Batch : Hinglish

Compiler Design

Lexical Analysis & Syntax Analysis

DPP

[NAT]

1. Consider the following C program:

```
int main ( )
{
/*finding maximum element out of a & b*/
int a, b max;
a = 10; b = 20
if (a < b)
        max = b;
else
        max = a;
return (max);
}
```

Calculate the total number of tokens present in the program?

[MSQ]

2. Consider the following C-program:

```
1. int main )(
2. {
4. x = a + b* c;
5. y = x + a ;
6. char f = 'e' ;
7. in t g = 200;
8. ch/* comment ar = "gate";
9. }
```

Which of the following is correct regarding above program?

(a) The given program has 47 tokens.
(b) Given program produces compilation error
(c) Given program produces lexical error
(d) No error produced by program

[MCQ]

3. Compiler's first phase makes uses of following patterns for token (S_1,S_2,S_3) reorganization over the alphabet a,b,c.

S_1: b#(b|a)*c
S_2: c#(c|b)*a
S_3: a#(b|c)*b

Note: x# means 0 or 1 occurrence of the symbol x. The analyzer outputs the token that matched the longest possible prefix of the string. If abbbccccba is processed by first phase of compiler then which one of the following is the sequence of token of output.

(a) S_1, S_2, S_3 (b) S_1, S_2
(c) S_3, S_2 (d) S_3, S3

[NAT]

4. How many of the following strings are said to be tokens in C-language without looking at next input character?

(i) ; (ii) return
(iii) int (iv) (
(v) && (vi) >>

[NAT]

5. Consider the following C-program:

```
int main ( )
{
int x; /* comment */
x = = = y /*abcd***/*abcd*/;
int **p;
int b = 10, y;
x = *p ++ + ++ y;
}
```

How many tokens are present in the given program?

[MCQ]

6. Which of the following is equivalent unambiguous grammar for following rules?

Operator	Priority	Associativity
↑, #	3	Left to right
⊕, *	2	Right to left
–, =	5	Left to right
/, &	1	Right to left
+, $	4	Left to right

Note: 5 has the highest priority and 1 has the least priority.

(a) $A \rightarrow B \uparrow A \mid B\#A \mid B$
$B \rightarrow C \oplus B \mid C*B \mid C$
$C \rightarrow C - D \mid C = D \mid D$

D→ D+E| D $ E | E
E→ E–F| E=F | F
F→ id

(b) A→ A↑B |A # B| B
B→ C⊕B |B * C| C
C→ C–D |C = D| D
D→ D+E |D $ E| E
E→ F–E |F = E| F
F→ id

(c) A→ A/B |A & B| B
B→ B⊕C |B * C| C
C→ C↑D |C # D| D
D→ D+E |D $ E| E
E→ F–E |F = E| F
F→ id

(d) A→ B/A |B & A| B
B→ C⊕B |C * B| C
C→ C↑D |C # D| D
D→ D+E |D $ E| E
E→ E–F |E = F| F
F→ id

[MCQ]

7. What will be the equivalent grammar after removing left factoring from the given grammar?
S → a|ab| abc|abcd|e|f

(a) S→aS'
S→ b|c|d|e|f

(b) S→ e|f|S'
S'→ a|ab|abc|abcd

(c) S→ e|f|aS'|∈
S'→ bA'|∈
A'→ cB'|∈
B'→ d

(d) S→ e|f|aS'
S'→ bA'|∈
A'→ cB'|∈
B'→ d|∈

[MSQ]

8. Which of the following is correct regarding FIRST & FOLLOW of the given grammar.
E→TE'
E'→+TE' |∈
T→FT'
T'→*FT' |∈
F→id |(∈)

(a) FIRST (T)= {id, (}
FOLLOW(T') = {+, $,)}

(b) FIRST (E) = {id, (}
FOLLOW(E') = {$,)}

(c) FIRST (E) = {id, (}
FOLLOW(F) = {+, $,), *}

(d) FIRST (T') = {*, ∈}
FOLLOW (E) = {$, * ,)}

[NAT]

9. Consider the following grammar:
S→ BB
B→ aB|b
How many items are there in Closure (S' →.S, $)?

[NAT]

10. Consider the given grammar:
S→ (A) | a
A→ SA'
A' → ,SA' | ∈
Assume that initially stack has 2 symbols $S on stack. Then, what will be the maximum size of stack during LL(1) parsing for the input string (a, a)?

[NAT]

11. Consider the following
A→ B = C/C
C→ B
B→ *C | id
How many number of states are required for above grammar using CLR parser?
Note: A, B & C are non-terminal and *, =, id are terminal.

[MCQ]

12. How many conflicting entries are there in SLR(1) parse table of the following grammar?

S→EeEf

S→FfEe

E→x

f→x

(a) 1 (b) 2

(c) 3 (d) 4

[MSQ]

13. Consider following grammar G.

S→ Aa| Bb

A→ Ac| ∈

B→ Bc| ∈

Which of the following statement is true?

(a) It has shift-reduce conflict in the first state of the LR(0) machine.

(b) It has reduce-reduce conflict in the first state of the SLR(1) machine.

(c) It has reduce-reduce conflict on the first state on the LR(0) machine.

(d) It has shift-reduce conflict in the first state of the SLR(1) machine.

❑❑❑

[MSQ]

14. Consider the following grammar:

P→ Qx | yQz | tz | ytx

Q→ t

Which of the following is correct for above grammar?

(a) It is LR(1) (b) It is LL (1)

(c) It is LALR(1) (d) It is CLR (1)

[NAT]

15. The maximum number of reduce moves that can be taken during bottom-up evaluation of 21 token string by using a bottom-up parser. Assuming the grammar has no epsilon and unit production.

Answer Key

1. **(41)**
2. **(b,c,)**
3. **(c)**
4. **(2)**
5. **(39)**
6. **(d)**
7. **(d)**
8. **(a,b,c)**
9. **(4)**
10. **(4)**
11. **(13)**
12. **(b)**
13. **(b,c)**
14. **(a,c,d)**
15. **(20)**

❑❑❑

Hints & Solutions

1. (41)

The total tokens in the program are:

int(1) main(2) (3))(4)
{(5)
/* finding maximum element out of a &b */
int(6) a(7) ,(8) b(9) ;(10) max(11) ;(12)
a(13) =(14) 10(15) ;(16) b(17) 20(19) ;(20)
if(21) (22) a(23) <(24) b(25))(26)
max(27) =(28) *b*(29) ;(30)
else(31)
max(32) =(33) a(34) ;(35)
return(36) (37) max(38))(39) ;(40)
}(41)

Total token in above program are: 41

2. (b, c)

The given program generates compilation and lexical error.

Therefore, option b, c are correct.

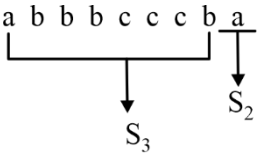
3. (c)

Minimum string of S_1: c

Minimum string of S_2: a

Minimum string of S_3: b

It is given that prefer longest matching. So,



Therefore, option (c) is correct answer.

4. (2)

Among all of the above ; , (are the only token for whom we need not to check next input character.

return can have next input character as returna which could be a variable name, similarly int..

&& can be **&&=** ·Similarly, >> could be **>>=**

5. (39)

The given program is:

int(1) main(2) (3))(4)
{(5)
int(6) x(7) ;(8) /*comment*/
x(9) =(10) =(11) =(12) y(13) /*abcd ***/ *(14) abcd(15) *(16) /(17) ;(18)
int(19) b(20) =(21) 10(22) ,(23) y(24) ;(25)
x(26) =(27) *(28) *p*(29) ++(30) +(31) ++(32) y(33) ;(34)
}(35)

There are total 39 tokens in program.

6. (d)

Given that

/, & has least priority and right to left associativity. So, it will be evaluated at last.

–, = has highest priority. It must be evaluated first.

The equivalent unambiguous grammar would be.

A→ B | A | B & A| B

B→ C⊕ B | C * B | C

C→ C ↑ D | C # D | D

D→ D + E | D $ E | E

E→ E – F | E = F | F

F→ id.

So, option (d) is the correct answer.

7. (d)

On eliminating left factoring from

S→ a |ab|abc|abcd|e|f

The equivalent grammar will be.

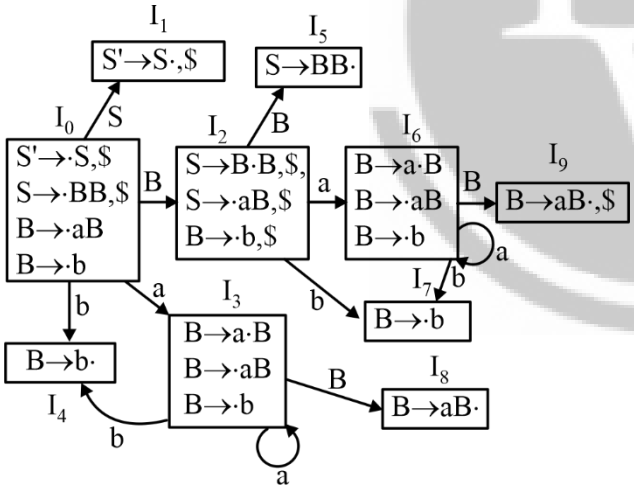
S→ e | f | aS’
S’→ bA’ |∈
A’→ cB’ |∈
B’→ d |∈
So, option (d) correct answer.

8. (a, b, c)
Given that

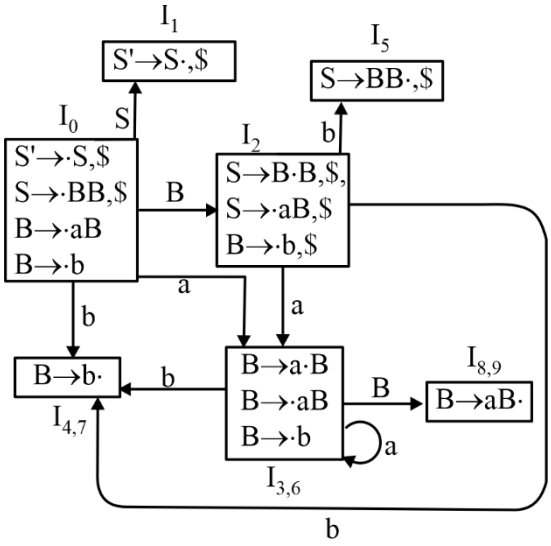
	Frist	**Follow**
E –	{(, id}	{$,)}
E’ –	{+, id}	{$,)}
T –	{id, (}	{+, $,) }
T’ –	{*, ∈}	{+ , $,)}
F –	{id, (}	{+, $,) , * }

So, option a, b, c are correct.

9. (4)
Augmented grammar:
S→ ·S, $
S→ ·BB, $
B→ ·ab
B→ ·b·



The closure of (S’→.S, $) have 4 items.



LALR (1) parser has 7 states.

10. (4)
The given non-left recursive grammar is:
S→ (A) | a
A→ SA’
A’ → ,SA’ | ∈
Now, First (S) = { (, a) }
Follow (S) = { $, t }
First A = { , a }
Follow A = {) }
First A’ = {Follow (A’) = Follow (A’) ={) }
So, parsing table of LL(1) will be as follows.

	(	)	A	,	$
S	S→(A)		S→a		
A	S→SA’		A→SA’		
A’		A’→∈	A’→SA’		

For string (a,a)

Stack	Input	Output
$S	(a,a)$	S→(A)
$)A(	(a,a)$	
$)A	a,a)$	S→A’
$)A’S	a,a)$	S→a
$)A’a	a,a)$	
$)A’	, a)$	
$)A’S	, a)$	A’→SA’
$)A’S,	a)$	
$)A’a	a)$	S→a

$)A'	)$	
$)	)$	A'→∈
$	$	

So, maximum stack size is 4.

11. (13)

The augmented grammar for the given grammar is:

A' → ·A,$

A → ·B = C , $

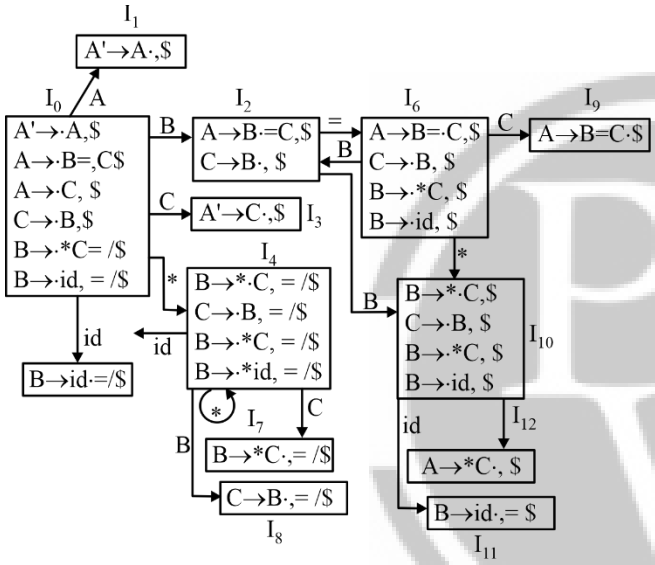
A → ·C , $

C → ·B , $

B → ·* C, = /$

B → ·id , = /$

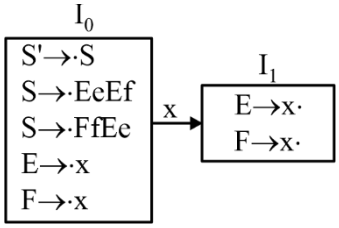
State diagram is as follows:



So, there are total 13 states needed for given grammar.

12. (2)

The LR(0) items of the grammar is:



R- R conflict (Reduce Reduce conflict)

In SLR(1) parse table on item I_1, E→ x & F→x are to be reduced in Follow (E) and Follow (F) respectively, which is both terminals e and f.

So, the number of conflicting entries in SLR(1) parse table is 2.

13. (b,c)

Here is the first state of the LR(0) machine.

S' →·S

S →·Aa

S →·Bb

A →·Ac

A →·∈

B →·Bc

B →·∈

Follow(A) = {a,c} & Follow (B) = {b,c}. Here it has reduce- reduce conflict between production (A→∈) and production (B→∈).

14. (a,c,d)

Augmented grammar:

P' →.P, $

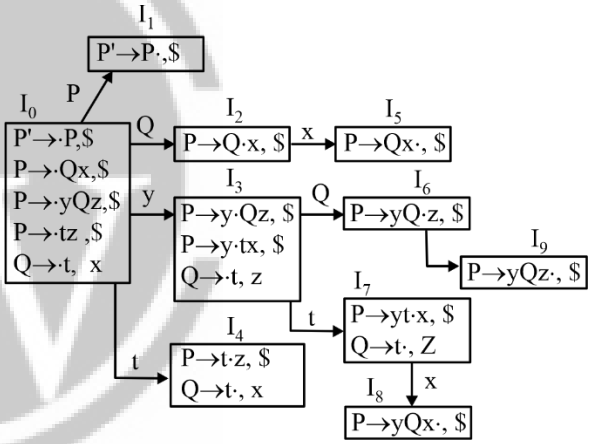
P →·Qx, $

P →·yQz, $

P →·tz, $

P →·ytx, $

Qr →·t, x



There are no conflict present so it will LALR(1) and no state is different in just look ahead symbol so it will also be LALR(1).

First (P)= First (Qx) ∩ First (yQz) ∩ First (tz) ∩ First (ytx)

= t ∩ y ∩ t ∩y

First (P) ≠ ϕ

So, it is not LL(1).

15. (20)

Maximum number of reduce moves for n tokens = n – 1.

So, for 21 token, 20 reduce moves are required.

❑❑❑

Any issue with DPP, please report by clicking here:- https://forms.gle/t2SzQVvQcs638c4r5

For more questions, kindly visit the library section: Link for web: https://smart.link/sdfez8ejd80if

PW Mobile APP: https://smart.link/7wwosivoicgd4

Branch : CSE & IT Batch : Hinglish

Compiler Design

Syntax Directed Translation

DPP

[MCQ]

1. Synthesized attributes can be easily simulated using
(a) LL grammar
(b) LR grammar
(c) ambiguous grammar
(d) None of these

[MCQ]

2. Consider the following translation rules for the grammar G:

S → a{print "A"} A

A → b {print "C"}B

A → ∈ {print "C"}

B → e {print "B"}A

B → ∈ {print "C"}

C → c {print "A"}

What will be the output for the input string abesebe you top-down parser?
(a) ACBCCBAC
(b) ACCBCCBC
(c) ACBCCBCC
(d) ACBCBCBC

[MCQ]

3. Consider the following attribute grammar:

A → BA'	A'· b = a · a
	A · a = A' · b
A_1' → +BA_2'	A_2' = A_2'b + B · a
	A_1'a = A_2'a

Which of the following is true?
(a) Both a and b are inherited attributed.
(b) Both a and b are synthesized attributed.
(c) a is inherited, b is synthesized
(d) b is inherited, a is synthesized

[MCQ]

4. Consider the following grammar:

E → E + T | T

T → T/F | F

F → F ∗ A | A

A → id

Which one of the following is true?
(a) / have higher precedence than ∗
(b) ∗have higher precedence than +
(c) + have lower precedence than/
(d) ∗, +, / all have some precedence.

[MCQ]

5. A shift reduce parser perform action specified within process immediately after reduction to the corresponding rule of grammar.

S → abv {print '11'}

S → cc {print '2'}

V → Sd {print '33'}

What is the translation of ababccdd using the SDT scheme described by above rules/
(a) 2233113311 (b) 1133113322
(c) 2211331122 (d) 1122113322

[MCQ]

6. Consider the following transition rules:

A → BC

C → +BC | A + | ∈

B → DB {print '+'} | ∈

D → (A) id {print number value}

If input is given "2 + 34" then his translation scheme will generate output.
(a) 2 + 3 + 4 + (b) + 2 + 3 + 4
(c) + + 2 + 34 (d) 2 + 34 + +

[MCQ]

7. ____is performed by attaching rules or algorithms to production in a grammar.

(a) Lexical analysis
(b) Execution
(c) syntax directed translation
(d) None of these.

[MCQ]

8. Consider a translation scheme is given as:

$S \rightarrow S_1 + S_2$ {S · val = S_1· val + S_2 · val}

$S \rightarrow S_1 * S_2$ {S · val = S_1· val * S_2 · val}

S → id {S · val = id}

What will be the output for $5 * 6 + 7$?

(a) 18
(b) 37
(c) 65
(d) Cannot be identified because it is ambiguous grammar.

[MCQ]

9. Consider the given translation rules.

If the expression 8 # 12 & 4 # 16 & 12 # 4 & 2 is evaluated to 512, then which of the following is correctly representing x?

E → E # T	{E·val = E1· val * T · val}
IT	{E · val = T · val}
T → T & F	x
IF	{T · val = F · val}
F → id	{F · val = id}

(a) T · val = T_1 · val * f · val
(b) T · val = T_1 · val + f · val
(c) T · val = T_1 · val – f· val
(d) T · val = T · val ÷ f · val

[NAT]

10. Consider the following SDT :

S → E	{S · val = E · val}
E → E + T	{E · val} = E_1 · val + T · val}
E → T	{E · val = T · val}
T → T F	{T ·v al = T_1 · val * f· val}
T → F	{T · val = f · val}
F → (E)	{F · val = E · val}
F → a	{f · val = a}

What will be the output of the expression "20 + 8 × 6"

❑❑❑

Answer Key

1. **(b)**
2. **(d)**
3. **(d)**
4. **(b, c)**
5. **(a)**
6. **(d)**
7. **(c)**
8. **(d)**
9. **(c)**
10. **(68)**

❑❑❑

Hints & Solutions

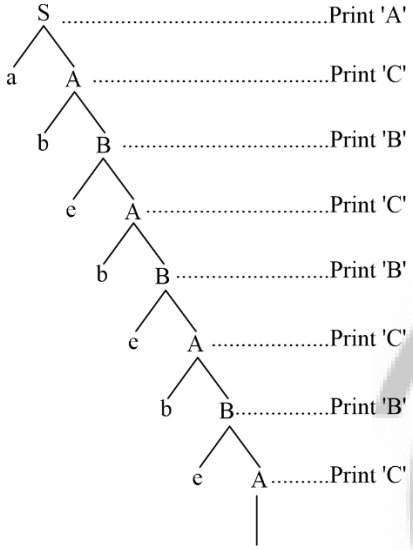
1. (b)

Synthesized attributes can be easily simulated by LR grammar.

As Synthesized attribute are evaluated in bottom-up manner and LR grammar is also evaluated in bottom-up manner.

2. (d)

Evaluating given grammar using top-down parser.



So, the output will be ACBCBCBC.

Therefore, option (d) correct.

3. (d)

For the first production a, b are attributes, such that a ∈A, and b ∈ A'.

Now if you look into the translation closely,

A' · b = A · a (b attribute is taking value from its parent attribute)

A'· a = A' · b (a attribute is taking value from one of its child attributes)

Similarly for second production, b is taking values from its parent as will as sibling which left to it (it is ∠ -attributed as well)

So, (b) follows inherited attribute definition while a follows synthesized attribute definition.

4. (b, c)

Here, ∗ have highest precedence, then/have precedence and + have least precedence

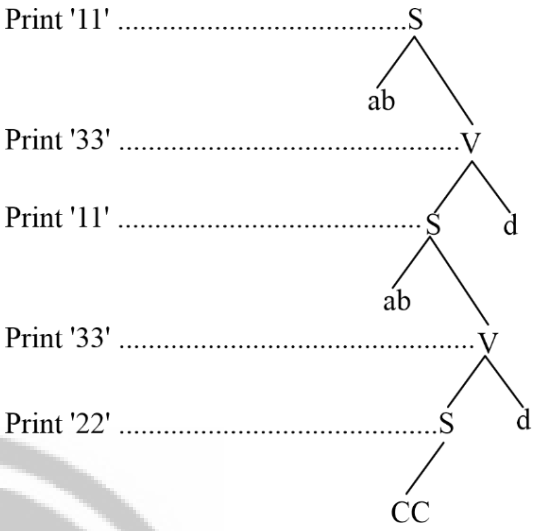
> / > +

As, ∗ will be evaluated first, so it has highest precedence.

Therefore, option (b) and (c) are correct.

5. (a)

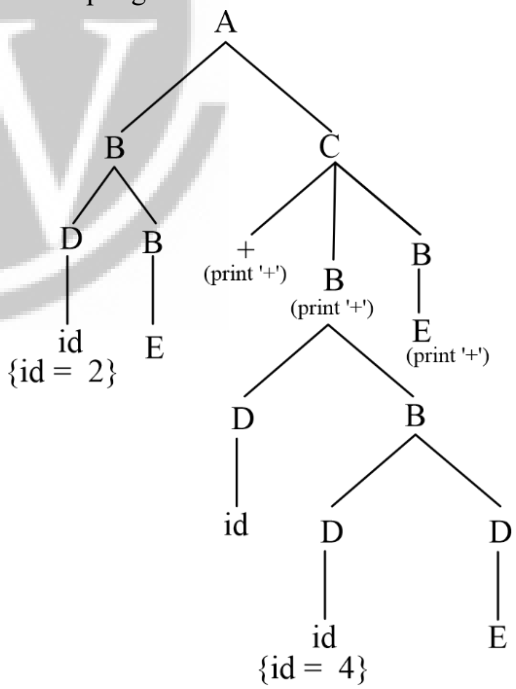
Given input string ababccdd



So, output will be 2233113311. Therefore option (a) is correct.

6. (d)

The input given is "2 + 34"



On scanning them left to right.

Output = 2 + 34 ++

So, option (d) is correct.

7. (c)

SDT is done by attaching augmented rules to the analysis. SDT rules used the following:

1. lexical value of node
2. Constants
3. Attributes associated to the non-terminal in their definition.

8. (d)

The given grammar is ambiguous grammar

S → S1 + S2 | S1 * S2 | id an ambiguous grammar.

Because for expression 5 * 6 + 7 two output are possible.

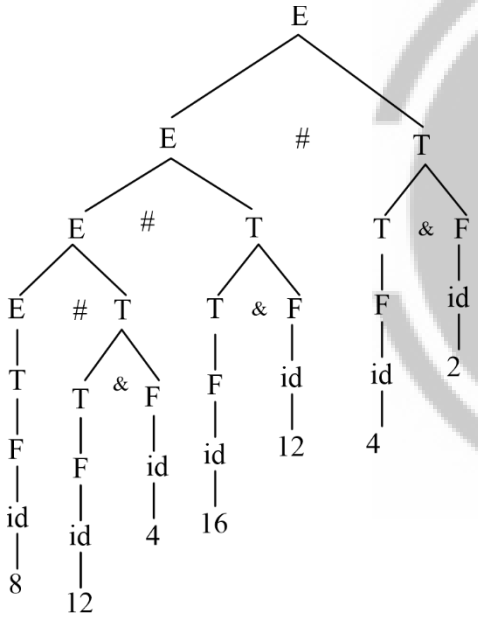
Output 1 : 37

Output 2 : 65

So option (d) is correct answer.

9. (c)

8 # 12 & 4 # 16 & 12 # 4 & 2 = 512



8 # 12 & 4 # 16 & 12 # 4 & 2

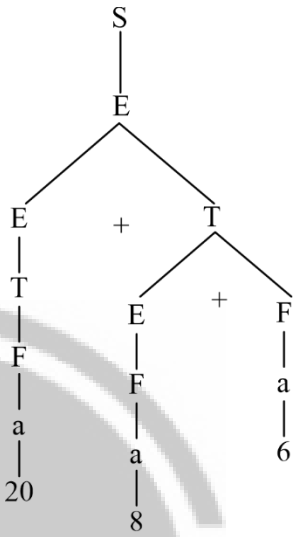
⇒ (8 # (12 & 4)) * (16 & 12) * (4 & 2)

If & = –

Then, ((8 ∗(12 – 4) ∗ (16 – 12) ∗ (4 – 2) = 8 ∗ 8 ∗ 4 ∗ 2 = 512

So, option (c) is correct answer.

10. (68)



20 + 8 × 6 = 68

❑❑❑