Compiler Design
Full Length Test-4

Question-1

Consider the following grammar over the alphabet {p,q,r,s}

A⟶BCD

B⟶pB|ϵ

C⟶Cq|q|Cr|s

D⟶AB|ϵ

Which of the following is True?

A. FOLLOW (C)={p,q,s,r,$} FIRST(D)={p,q,ϵ}

B. FOLLOW (B)={$,p,q,s} FIRST(A)={p,q,s}

C. FOLLOW (D)={p,s,$} FOLLOW (A)={p,s,$}

D. FOLLOW (B)={p,ϵ} FIRST(C)={q,s,p}

Question-2

Consider the following SDT.

$S \longrightarrow aaS$ {print("a") }

$S \longrightarrow bA$ {print(b) }

$S \longrightarrow b$ {print(c) }

$A \longrightarrow bcA$ {print(d)}

$A \longrightarrow cdS$ {print(e) }

If an LR parser carries out the translations on an input string “aabbccdaab”, what is the output?

A. cedbaa

B. Caedba

C. abdeac

D. aabdec

Question-3

Consider the following statements:

S1: Target code generator and code optimizer have to be modified depending on machine architecture.

S2: Intermediate code generator, Target Code generator and code optimizer needs to be modified depending on machine architecture.

Which of the above statement is/are true?

A. Only S1

B. Only S2

C. Both S1 and S2

D. Neither S1 nor S2

Question-4

Consider the grammar and the corresponding semantic actions.

A$\longrightarrow$XYZ {A.a=X.a+Y.a+Z.a}

{Y.b=A.b+X.b+Z.b}

A. a is synthesised and b is inherited.

B. a is inherited and b is synthesised

C. Both are inherited

D. Both are synthesised

Question-5

Which of the following tasks can be performed using semantic rules in an SDT?

(i) Type checking and parameter checking

(ii) Issuing error messages

(iii) Building syntax trees

(iv) Generating intermediate or target code

A. Only (i), (iii) and (iv)

B. Only (i) and (ii)

C. Only (iii) and (iv)

D. All the statements

Question-6

Consider the following SDT:

$S \longrightarrow AB$

$B \longrightarrow$ *AB|S|ϵ

$A \longrightarrow$ CA{print '$'}|ϵ

$C \longrightarrow$ (S)|id {print (num.val) }

id is a token that represents an integer and num.val represents the corresponding integer value. What does the SDT prints on an input “6*5 7”?

A. 6$57$$

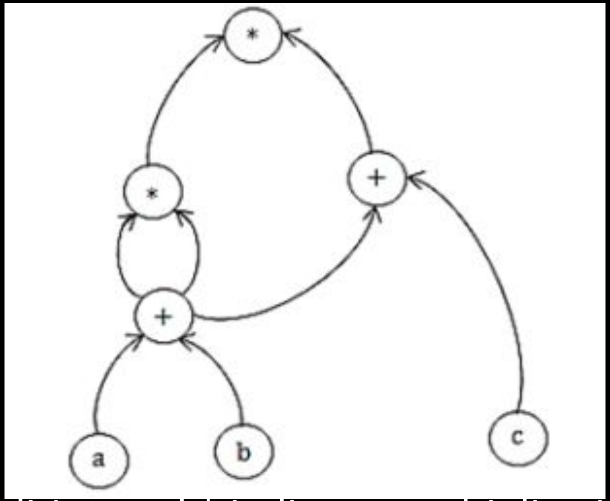
B. 6$5$7$

C. Both (a) and (b)

D. None of these

Question-7

Identify the correct expression which represents the DAG given below:



Assume + indicate addition and * indicates multiplication.

A. (a+b+c)2 (a+b)

B. (a*b*c)(a*b)(a+b)

C. (a+b)2 (a+b+c)

D. (a*b+c)2 (a*b)(a+b)

Question-8

The three address code for the following postfix notation is:

-a + b / c ↑ d ↑ e * f / g

A. t1 = a

t2 = t1 - b

t3 = c * t2

t4 = d ↑ e

t5 = t3 * t4

t6 = t5 * f

t7 = t6 / g

B. t1 = -a

t2 = t1 + b

t3 = t2 / e

t4 = t3 ↑ d

t5 = t4 ↑ e

t6 = t5 * f

t7 = t6 / g

C. t1 = -a

t2 = d ↑ e

t3 = c ↑ t2

t4 = b / t3

t5 = t4 * f

t6 = t5 / g

t7 = t1 + t6

D. None of the above

Question-9

What value will be returned by the below grammar for the following string : ‘(())((()))()’

S -> S1S2 {S.maxdepth = max (S1.mxdepth,S2.maxdepth)}

S -> (S1) {S.maxdepth = S1.maxdepth +1}

S ->() {S.maxdepth = 1}

A. 4

B. 5

C. 6

D. 3

Question-10

Which of the following SDT converts infix expressions to postfix?

A. E⟶E1+T {print("+");}
E⟶T { }
T⟶T1 *F {print(*);}
T⟶F { }
F⟶id {print(id.name);}

B. E⟶E1 *T {print("*");}
E⟶T { }
T⟶T1+F {print(+);}
T⟶F { }
F⟶id {print(id.name);})

C. Both (a) and (b) can convert infix to postfix

D. SDT cannot be used to convert infix to postfix

Question-11

Consider the following grammar :

P -> +{ print('(')}P1 {print(')+(')} P2{print(')') }

P -> - { print('(')} P1 {print(')-(')} P2 {print(')') }

P -> id {print('id.num')}

Using the above grammar,select the odd one out :

A. +1-23

B. +-123

C. +-231

D. -+123

Question-12

Consider the grammar with the following productions.

S⟶aaB/aaC

B⟶b

C⟶c

Which of the following option is true?

A. The grammar is LL(2)

B. It can't be LL(k) grammar for any k, as it contains left factoring.

C. The grammar is LL(1)

D. The grammar is LL(3)

Question-13

Which of the following statements is/are TRUE?

S1: The lexical analyzer detects an error when it discovers that an input prefix does not fit the specification of any token class.

S2: The parser can help the lexical analyzer improve its ability to recover from errors by making the list of legitimate tokens available to the error recovery routine.

A. Only S1 is TRUE

B. Only S2 is TRUE

C. Both S1 and S2 are TRUE

D. None of these

Question-14

Categorize each of the following errors into syntactic error/lexical error/static semantic error.

A. Redeclaration of an identifier.
B. B. Unbalanced parenthesis.
C. Array index out of bounds.
D. Divide by 0.
E. Mistype the constant integer “Count12” as “Count 12”.
F. Dereference of null pointer.
G. Use of uninitialized variable.

Let the number of Syntactic error from the above list be x, number of lexical errors from the above list be y and the number of static semantic errors be z. Then x-(y+z) = ____

Question-15

Which of the following statement is false?

A. L-attributed definition can be evaluated in the framework of bottom up parsing.

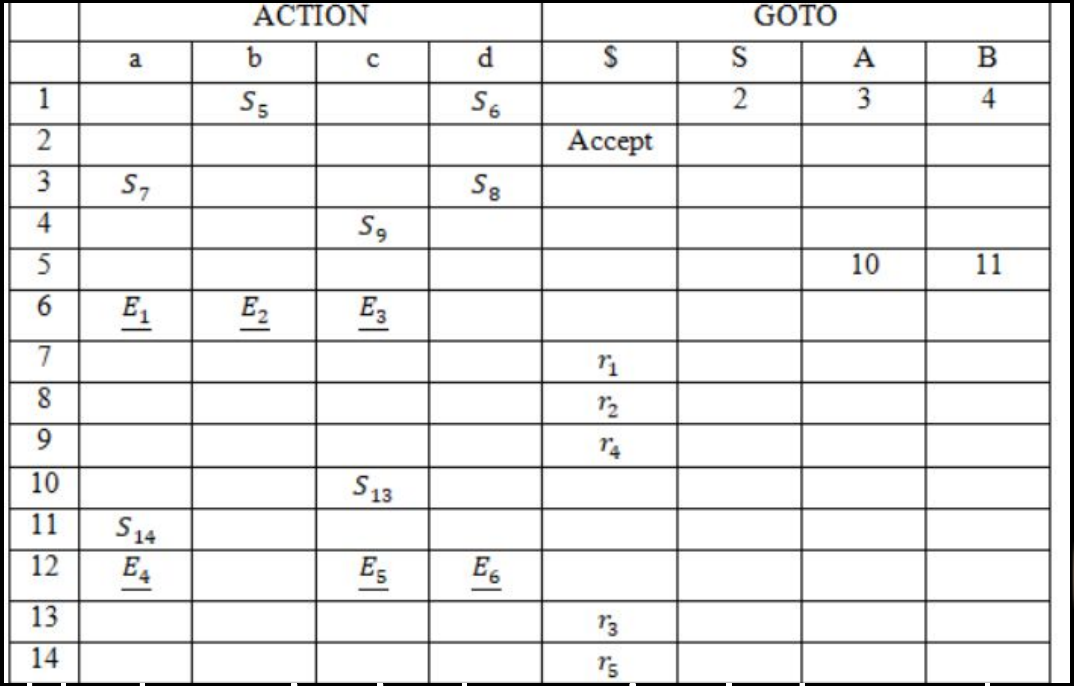
B. A programming language which allows recursion cannot be implemented with static storage allocation.

C. L- attribute SDD’s are proper subset of S-attributed SDD’s

D. LL-grammars are proper subset of LR grammars.

Question-16

What will be entries in E1,E2,E3,E4,E5,E6 respectively in CLR(1) parsing table



	ACTION				GOTO			
	a	b	c	d	$	S	A	B
1		S_5		S_6		2	3	4
2					Accept			
3	S_7			S_8				
4			S_9					
5							10	11
6	$\underline{E_1}$	$\underline{E_2}$	$\underline{E_3}$					
7					r_1			
8					r_2			
9					r_4			
10			S_{13}					
11	S_{14}							
12	$\underline{E_4}$		$\underline{E_5}$	$\underline{E_6}$				
13					r_3			
14					r_5			

Note: The blank entries in the option is denoted as "-".

A. r6,r7,r6,r7,r6,-
B. –,r6,r7,-,r7,r6
C. r7,r6,r7,-r6,r7,r6
D. r6,r7,-,r6,r7,-

Question-17

Consider the following SDD

T⟶FT1 { F.val=T1 .val }

T2 ⟶*AT3 { T2 .val=T3 .val×A.val}

Which of the following is true?

A. F.val is synthesized attribute and T2 .val is inherited attribute.

B. F.val is inherited attribute but T2 .val is not a synthesized attribute.

C. F.val is inherited attribute and T2 .val is synthesized attribute

D. F.val is synthesized attribute but T2 .val is not an inherited attribute.

Question-18

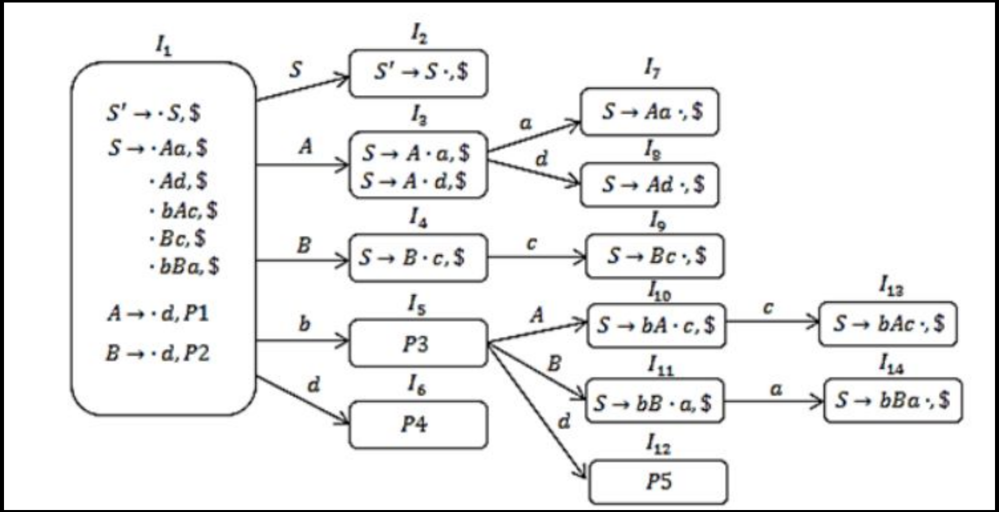
Consider the following grammar:

S⟶Aa/Ad/bAc/Bc/bBa (r1/r2/r3/r4/r5)

A ⟶d (r6)

B⟶d (r7)

Given below is a partial canonical collection of LR(1) items:



Find the values at missing positions P1,P2,P3,P4,P5.

A. P1: a, d

P2: c

P3:S⟶b.Ac, $

S⟶b.Ba, $

A⟶.d,

B⟶.d, a

P4:A⟶d., a

B⟶d., c

P5:A⟶d., c

B⟶d., a

B. P1: a

P2: d

P3:S⟶b.Ac, $

S⟶b.Ba, $

A⟶.d, d

B⟶.d, a

P4:A⟶d., a

B⟶d., d

P5:A⟶d., d

B⟶d., a

C. P1: a, d

P2: c

P3: $S \rightarrow b.Ac$, \$

$S \rightarrow b.Ba$, \$

$A \rightarrow .d$, c

$B \rightarrow .d$, a

P4: $A \rightarrow d.$, a, d

$B \rightarrow d.$, c

P5: $A \rightarrow d.$, c

$B \rightarrow d.$, a

D. None of these

Question-19

Consider the grammar given below and shift-reduce parse the input aaa+a*+ according to the grammar: S→SS*| SS+|a

Construct a shift reduce parsing table for the above grammar. Which of the following pair is NOT a stage in the parsing table? Parse the input expression in the form of ($stack, remaining input$).

A. ($0S2S2S2+5,a*+$)
B. ($0S1S2S2*4,+$)
C. ($0S1S2+5,$)
D. ($0S1,aaa+a*+$)

Question-20

Consider the below given statements.

Statement I: If a grammar G is LL(1), then it must be SLR(1).

Statement II: If a grammar G is LL(1) then it must be CLR(1), but may not be LALR(1).

Statement III: For a grammar G the SLR(1) and LALR(1) parsing tables must have same GOTO part.

Statement IV: For a grammar G the SLR(1) and LALR(1) parsing tables must have identical shift entries.

Which of the following option is correct?

A. Statement I &IV are true only.

B. Statement II, III &IV are true only.

C. Statement II &III are true only.

D. Statement I,III &IV are true only.

Question-21

Consider the SDT shown below E⟶TE'' E''⟶+T{Print("+");} E'' |ϵ
T⟶num {print("num");}

Convert the grammar into the equivalent S-attributed definition

A. E⟶TE'‘

E''⟶+TME^'' |ϵ

M⟶ϵ {Print("\"+\"");}

T⟶num {print("num");

B. E⟶TE''

E''⟶+TE'' |ϵ {Print("+");}

T⟶num {print("num");}

C. E⟶TE''

E''⟶+TME'' |ϵ

M⟶E'' {Print("\"+\"");}

T⟶num {print("num");}

D. None of these

Question-22

Consider the following SDD

T⟶FT1 { F.val = T1 .val}

T2 ⟶*AT3 {T2 .val=T3 .val×A.val}

Which of the following is true?

A. None of the above.

B. This SDD follows both S-attributed as well as L-attributed definition.

C. This SDD follows L-attributed definition.

D. This SDD follows S-attributed definition.

Question-23

Eliminate left recursion from the following grammar:

A→Bd|Aff|CB

B→Bd|Bf|df

C→h|g

A.A→BdA' |CBA

B→dfB'

A'→ffA' |ϵ

B'→dB' |fB' |ϵ

C→h|g

B. A⟶ffA' |ϵ

A'⟶BDA' |CBA'

B⟶ dB' |fB' |ϵ

B'⟶dfB'

C⟶h|g

C. A⟶BdA'

B⟶dfB' |dB' |fB‘

B'⟶ϵ

A'⟶ffA' |ϵ

C⟶h|g

D. None of these

Question-24

Regular expression is used in Lexical analysis to model

A. The structure of lexemes with fix length identifier included.

B. The structure of tokens with variable length identifier included.

C. The structure of lexemes with variable length identifier included.

D. The structure of tokens with fix length identifier included.

Question-25

Consider the following statements:

S_1: Viable prefixes of grammar G are those

prefixes of right-sentenial form that can appear on top of stack of a shift reduce parsing.

S_2: Handle is sub-string that matches right side of production whose reduction to non terminal on left side of production represents one step along the reverse of leftmost derivation.

Which of the above statements is/are true?

A. S_1, is true and S_2 is false

B. S_2, is true and S_1, is false

C. Both are true

D. Both are false

Question-26

Consider the following grammar G.

$S \rightarrow eAB$

$A \rightarrow dC | c$

$B \rightarrow bA | a$

$C \rightarrow \epsilon | a$

Which of the following statement is correct?

S1: LL(1) can parse all strings that are generated using grammar G.

S2: LR(1) can parse all strings that are generated using grammar G.

A. Neither S1 nor S2

B. Only S2

C. Only S1

D. Both S1 and S2

Question-27

A canonical set of LR(1) items is given below, which is present in some state of DFA for CLR(1)parsing table construction.

B⟶b·C,$|c

C⟶c·,$|c A.

A reduce-reduce conflict, but not shift-reduce conflict.

B. It has a shift-reduce conflict and a reduce-reduce conflict.

C. It has a shift-reduce conflict, but not reduce-reduce conflict.

D. Neither a shift-reduce, nor a reduce-reduce conflict

Question-28

Consider the following grammar, which describes the list of nouns:S→noun|noun and noun|M,noun and noun

M→M,noun|noun

The terminals in the grammar are noun , and \$ (\$ is the end marker)
Identify the correct statement about the grammar

A. The grammar is LL(1)

B. All the productions for S contribute to the conflict

C. No production for M causes a conflict

D. None of these

Question-29

Which of the following are true ?

I. A programming language which allows recursion can be implemented with static storage allocation.

II. Activation Record contains information about formal parameters and return address along with some other information.

III. In stack storage allocation,memory can be allocated or deallocated any time. IV. Triple representation takes more space than Quadraple representation in IR code.

A. I and II

B. II and IV

C. I,II and IV

D. None of these

Question-30

stmt ->if expr then stmt

|if expr then stmt else stmt

|other

Consider the following statements for above grammar:

1. The grammar will cause shift reduce conflict.
2. The grammar is ambiguous.
3. The grammar is left recursive.

Which of the above statements is/are true?

A. 1 and 2 is true

B. 1, 2 and 3 is true

C. 2 and 3 is true

D. all are false

Question-31

Consider the following statements.

I. Symbol table is accessed only during lexical analysis and syntax analysis.

II. Compilers for programming languages that support recursion necessarily need heap storage for memory allocation in

the run-time environment.

III. Errors violating the condition 'any variable must be declared before its use' are detected during syntax analysis.

Which of the above statements is/are TRUE?

A. I only

B. I and III only

C. Il only

D. None of I, II, and III

Question-32

Consider the following statements:

S1:

```
int main (int i,int j)
{ a++-+--b;
printf("a+b=%d",c);
c>=d!=e;
return i>j?i:j; /*returning max(i,j)*/
}
```

has 40 tokens.

S2: Grammar

$S \longrightarrow AaB$

$A \longrightarrow aA|b$

$B \longrightarrow bB|\epsilon$

is an example of operator precedence grammar.

S3:

Top down parser regenerates string to be parsed beginning with the start symbol of the grammar and bottom up parser reduces the input string to the start symbol of the grammar.

A. S1 is false, S2, S3 are true

B. S1, S2 are true, S3 is false

C. S1, S3 is true, S2 is false

D. S1, S2, S3 are true

Question-33

Consider the following two grammars:

I.w⟶c|b

S⟶Ta|vTa

T⟶ST|w|a

v⟶b

II.S⟶BCd|λ

A⟶AaSb|SbC|λ

B⟶b|λ

C⟶c|B

Are these grammars LL(1)?

A. None of the two grammars is LL(1)

B. Both grammar I and II are LL(1)

C. Only grammar I is LL(1)

D. Only grammar II is LL(1)

Question-34

A programming language is to be designed to run on a machine that does not have a big memory. The language should

A. prefer a 2 pass compiler to a 1 pass compiler

B. prefer an interpreter to a compiler

C. not support recursion

D. all of the above

Question-35

For which of the following reasons, a compiler is preferable to an interpreter?

A. It can generate stand-alone programs that often take less time for execution.

B. It is much helpful in the initial stages of program development.

C. Debugging can be faster and easier.

D. If one changes a statement, only that statement needs recompilation

Question-36

Consider the grammar given below E? E+E | E*E | E-E | E/E | E^E | (E) | id Assume that + and ^ have the same but least precedence, * and / have the next higher precedence but the same precedence and finally ^ has the highest precedence. Assume + and ^ associate to the left like * and / and that ^ associates to the right. Choose the correct for the ordered pairs (^,^), (-,-), (+,+), (*,*) in the operator precedence table constructed for the grammar.

A. All <

B. All >

C. < >, =

D. < > > >

Question-37

Consider the following statements:

1. Lexical analysis removes white spaces and not the comments.
2. fro(i=0) generates lexical error in C
3. Lexeme is the string and not the pattern.
4. fi(a > b) generates no lexical error in C.

Identify the false statements:

A. Only 1 and 4

B. Only 1 and 2

C. Only 2

D. Only 4

Question-38

Consider line-3 of the following c-program.

```
 int main() {                / line1*/
int i, n;                    /*line2"/
Fro i = 0; i < n ;i++) ;  /* line 3*/
```

Identify the compiler response about the line 3 while creating the object module.

A. No compilation error

B. Only a lexical error

C. Only Syntactic error

D. Both lexical error and syntactic error

Question-39

Consider the following program.

```
main ()
char ch ='A' ;
 int x, y;
x = y = 20
x ++ ;
printf("%d %d",x, y);
```

The number of tokens in the above program are______

A. 32
B. 30
C. 33
D. 25

Question-40

Consider the grammar shown below:

S->iEtSS’| a

S’->es|∈

E->b

In predictive parsing table M, the entries M[S’,e] and M[S, $] respectively are

A. (S’->eS) and (S’-> ∈}

B. (S’->eS) and {}

C. (S’-> ∈) and (S'⟶ ∈)

D. (S’->eS, S’-> ∈} and {S’⟶ ∈}

Question-41

Which of the following statements is/are true?

(i) There are 4 parse trees for the input string “aaa” in S→Sa|aS|a

(ii) There are 3 parse trees for the input string “abab” in S→aSbS|bSaS|ϵ

(iii) The grammar S→αSβ|SS|ϵ is unambiguous

(iv) For every ambiguous grammar there is equivalent grammar for the same language which is unambiguous.

A. (i)-T,(ii)-F,(iii)-F,(iv)-T

B. (i)-F,(ii)-T,(iii)-F,(iv)-F

C. (i)-T,(ii)-F,(iii)-T,(iv)-T

D. (i)-T,(ii)-F,(iii)-F,(iv)-F

Question-42

The following grammar is given for predictive parsing table.

E->TE‘ T→FT’ F→(E)

E->+TE’ T’→ *FT’

E-> ∈ T’-> ∈

F->id

The start symbol E has entry in

A. +, $

B. *,id

C. id, (

D. (,$

Question-43

Consider the following grammar:

1. S -> S; S
2. S-> id:=E
3. S⟶ print (L)
4. E->id
5. E -> num
6. E -> E + E
7. E⟶ (S, E)
8. L -> E
9. L -> L, E

How many states are there in LR parsing table?

A. 15

B. 27

C. 23

D. None of these

Question-44

A grammar G is defined by:

G([a, b] . S, A, B), P, S)

When Pis the set

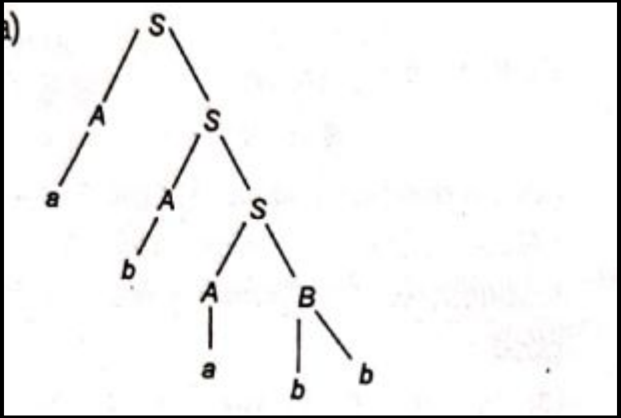
S-> AB |AS

A⟶ a| Aa

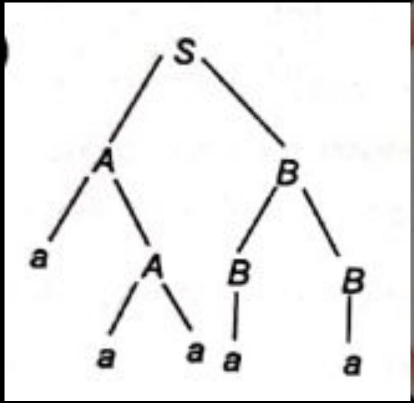
B->b|bb

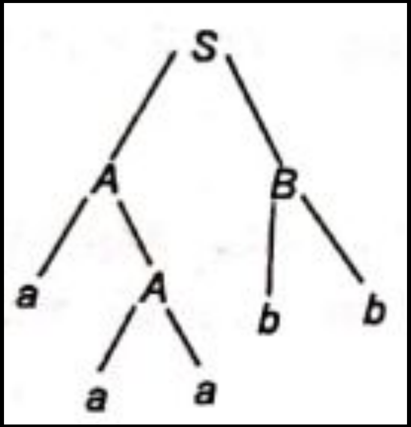
Consider the following trees which is are legitimate for the string "ababb"?

A.



B.



C. 

D.None of the above

Question-45

Consider the following grammar:

S⟶ ABDh

B->cC

C->bc|∈

D->EF

E->g| ∈

F->f| ∈

How many pair/pairs of elements has equivalent FOLLOW sets?

A. 0

B. 1

C. 2

D. 3

Question-46

Let G be a grammar and G has following set of productions.

$S \longrightarrow AA$

$A \longrightarrow AA \mid ba \mid Ab \mid a$

How many strings can be generated by grammar G with atmost four steps in the derivation. [initially start symbol is available without any step]

A. 3

B. 4

C. 5

D. 6

Question-47

Which of the following describes a handle (as applicable to LR-parsing) appropriately?

A. It is the position in a sentential form where the next shift or reduce operation will occur

B. It is a non-terminal whose production will be used for reduction in the next step

C. It is a production that may be used for reduction in a future step along with a position in the sentential form where the next shift or reduce operation will occur.

D. It is the production p that will be used for reduction in the next step along with a position in the sentential form where the right hand side of the production may be found

Question-48

Consider the following grammar:

S⟶MNzSc

M->a Ma|∈

N⟶bNb| ∈

Which of the following will represent the FOLLOW(S) and FIRST(S) respectively?

A. FOLLOW (S) = (c), FIRST (S) = (a, b, z)

B. FOLLOW (S) = ($, c), FIRST (S) = (a, b, z}

C. FOLLOW (S) = (a, c), FIRST (S) = (a, b, z}

D. None of these

Question-49

Let G be a grammar with the following productions.

E->E+T|T

T⟶T* F|F

T-> (E)| T-F

F->id

If LR(1) Parser is used to construct the DFA using the above productions, then how many look-a-heads are present for an item T⟶ gT* Fin the initial state?________

Question-50

During the LR(0) construction of a grammar we get an intermediate state

S⟶T

T->F

T->T* F

T⟶ id

F⟶id |(T)

F⟶ id = T;

Then which of following statement is true

A. There is only one shift/reduce conflict in intermediate state

B. There is only one reduce/reduce conflict in intermediate state

C. There is both shift/reduce and reduce/reduce conflict in intermediate state

D. There is no such intermediate state.

Question-51

Which of the following statements is/are TRUE with respect to SLR(1) and LALR(1) parsing tables for a CFG.

I. The goto part of both tables may be different
II. The shift entries are identical in both thetables
III. The reduce entries in the entries the tables may be different
IV. The error entries in the table may be different

A. Statement I
B. Statement IV
C. Statements II and IV
D. Statements II, III and I

Question-52

Consider the following CFG:

S-ABC |a| ∈

A⟶BC |b| ∈

B⟶c| ∈

C⟶d| ∈

Compute the FOLLOW set of non-terminal A.

A. (c, $, d)

B. (a, b, c)

C. (d, c, ∈)

D. (a, b, ∈)

Question-53

Consider the following syntax directed definition

A⟶BC

C.i= c(A.i)…..1

B.i=b(A.s) ... 2

A.s = f(B.s)….3

i for inherited, s for synthesized attribute

Which of the following is true?

A. 1 is violating L attributed definition

B. 2 is violating L attributed definition

C. 3 is violating L attributed definition

D. None of the above

Question-54

Consider the following grammar:

S-ABC|aC

A ->bC|∈

B-> ∈

C⟶ e| ∈

Which of the following will represent the FOLLOW (A) and FIRST (S) set?

A. FOLLOW (A)= {e, $},FIRST (S) = (a, b, e, e)

B. FOLLOW (A) = (a, b, $),FIRST (S) (a, b, e, e}

C. FOLLOW (A) = {e, $) ,FIRST (S)=(a, b, e, $)

D. FOLLOW (A) = {b, $} ,FIRST (S) = (b, e, e}

Question-55

Let G be a grammar is used for arithmetic expressions. The grammar G is shown below with semantic actions and attribute "sign" can contain either 0 or 1.

E->E1+E2 (E.sign = E2 sign)

E->E1 x (E2) (Esign =E1. sign x E2.sign)

E->E1/E2, (if (E1.sign == 0) then

Esign= 1 else Esign = 0)

E->+E1 (E.sign = 0)

E->-E1 (Esign = 1}

E->id (E.sign=0)

[Note: E->E1+E2 is same as E⟶ E+E]

Find the attribute value at the root E for the given input: -id x (-id + id)

A. 0

B. 1

C. 2

D. None of these

Question-56

Consider the grammar G shown below:

$G: E \longrightarrow EzE \mid xyE \mid w$

Which of the following conflict is present in the DFA construction of SLR(1) parser for grammar G.

A. Reduce-Reduce conflict

B. Shift-Reduce conflict

C. Both (A) and (B)

D. No conflict present

Question-57

Consider the given below SDT.

P1: S⟶MN {S.val= M.val + N.val}

P2: M ⟶ PQ {M.val = P.val * Q.val and P.val =Q.val}

Select the correct option.

A. None of these

B. P1 is L attributed but P2 is not L-attributed. .

C. P1 is S attributed and P2 is L-attributed.

D. Both P1 and P2 are S attributed.

Question-58

A shift reduce parser carries out the actions specified with in braces immediately after reducing the corresponding rule of grammar.

A->bbB (print "+"}

A⟶ a {print "x"}

B⟶ Ac {print "-"}

What is the translation of bbbbacc using the syntax directed translation scheme described by the above rules?

A. * + - + *

B. * * - + -

C. + * - + *

D. * - + - +

Question-59

Consider the given below grammar.

S⟶ X&Z | Y*

X⟶ YZ

Y⟶ #Y | Z

Z⟶ &++S | ϵ

For which non terminal /non terminals, the “*” is a part of corresponding FOLLOW set.

A. {Y}

B. {S, Y, Z}

C. {S, X, Y, Z}

D. { Y, Z}

Question-60

A lexical analyzer uses the following patterns to recognize three tokens T1, T2, and T3 over the alphabet (a,b,c}.

T_1: a? (b|c)*a

T2: b? (ac)*b

T3: c? (b|a) *c

Note that 'x?' means 0 or 1 occurrence of the symbol x. Note also that the analyzer outputs the token that matches the longest possible prefix. If the string bbaacabc is processed by the analyzer, which one of the following is the sequence of tokens it outputs?

A. T_1T2T3

B. T_1T1T3

C. T_2T1T3

D. T3T3

Question-61

Consider the following production, along with the syntax directed definition.

A->QR,

Q.in=f.(As) || 1st rule

R.in=f.(Q.s) || 2nd rule

Where .in and .s have their regular meaning i.e., inherited and synthesized attributes respectively. Which of the following option is true?

A. Syntax directed definition are L-attributed.

B. Syntax directed definition is not L-attributed because of 2nd rule.

C. Syntax directed definition is not L-attributed because of 1st rule.

D. Syntax directed definition is not L-attributed because of both rule

Question-62

Consider the given below statements.

S1: Every regular language must have an LR(0) grammar.

S2: The language L={anb n | n >0} must have an LR(0) grammar. Select the correct option.

A. Both S1 and S2 are true.

B. Both S1 and S2 are false.

C. S1 is correct and S2 is false.

D. S1 is false and S2 is true

Question-63

Consider the grammar with given productions and SDT
S⟶ S $ S {print ('₹')}
S⟶ S ₹ S {print('$')}
S⟶ id {print('id.name')}
If a top down parser carries out the SDT for the string "a$b₹c" and considerthe given below outputs.
op1: "ab₹c$"
op2: "a₹b$c"
op3: "abc$₹"
Select the correct option.
A. op1 and op3, both are correct output for the given string
B. Only op1 is correct output for given string
C. op1 and op2, both are correct output for the given string
D. Only op2 is correct output for given string

Question-64

Consider the below given grammar ‘G’

S⟶-S

S⟶S-a

S⟶a

Choose the correct option.

A. G is CLR(1) grammar, but not LALR(1)

B. None of the above

C. G is LALR(1) grammar

D. G is an ambiguous grammar

Question-65

Consider the immediate code given below: (1) i=1

(2) j=1

(3) t1=5*i

(4) t2=t1+j

(5) t3=4*t2

(6) t4=t3

(7) a[t4]=-1

(8) j=j+1

(9) if j<=5 goto (3)

(10) i=i+1

(11) if i<5 goto(2)

The number of nodes and edges in the control flow graph constructed for the above code is respectively are:

A. 5 and 6

B. 4 and 5

C. 7 and 8

D. 6 and 7

SOLUTION

Solution-1

Answer: B

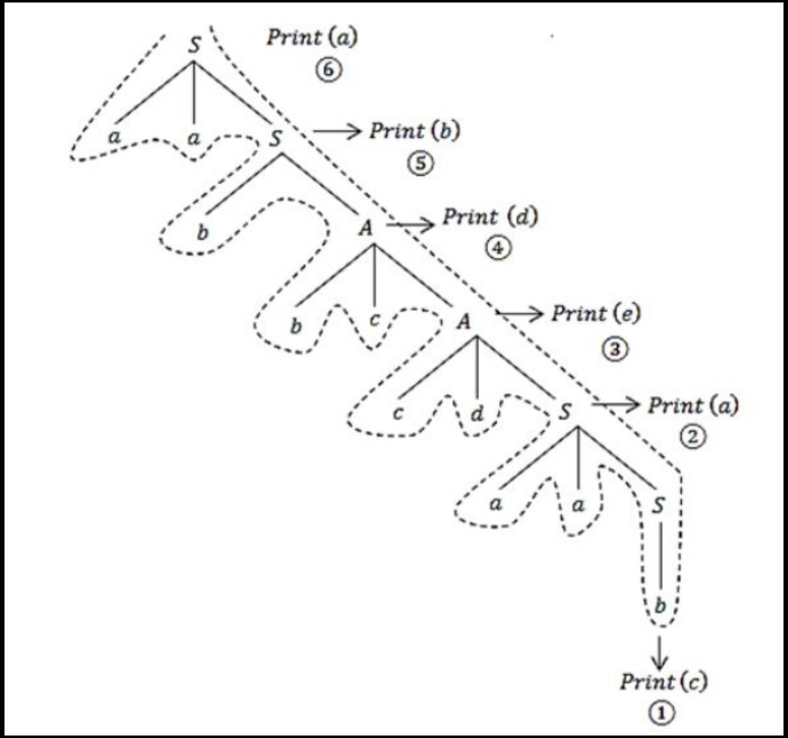
Explanation:

	FIRST()	FOLLOW()
A	{p,q,s}	{p,$}
B	{p, ϵ}	{p,s,$,p}
C	{q,s}	{p,q,s,r,$}
D	{p,q,s, ϵ}	{p,$}

Solution-2

Answer: B

Explanation:



So, the output is caedba

Solution-3

Answer A

Explanation: The entire frontend of a compiler i.e:

1. Lexical analyzer 2. Syntax analyzer 3. Semantic analyzer 4. Intermediate code generator is implemented together and does not depend on a specific computer architecture. But the back end which comprises of code optimization & Target Code Generation dependents on computer architecture & needs to be modified..

Solution-4

Answer:A

Explanation:

a is synthesised attribute because it is computed from the values of attributes at the children of A. b is inherited because it depends on the value of siblings X, Z and the parent A. Remember that Y can take values from left sibling or parent is the requirement of L-attribute grammar but an inherited attribute can take values from its parent or from any sibling.

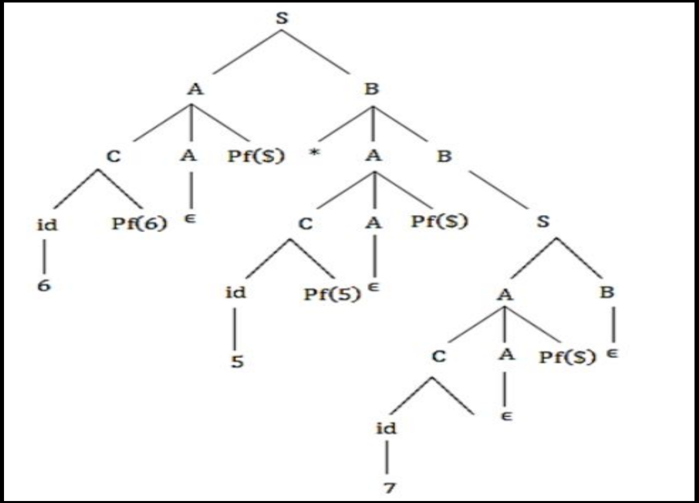
Solution-5

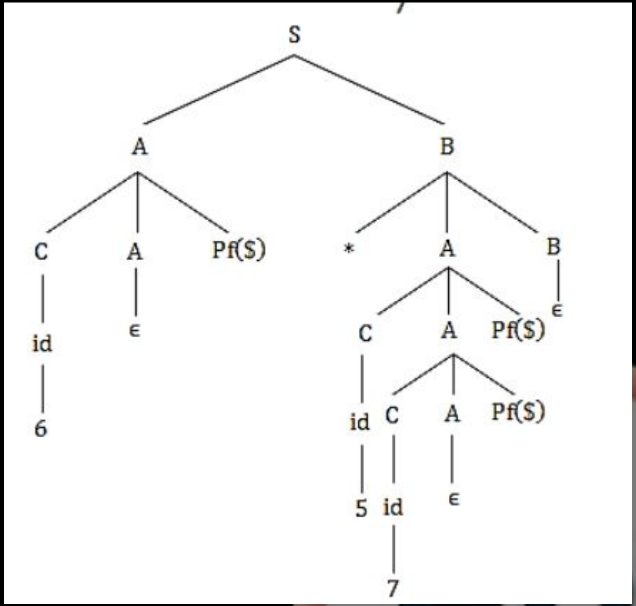
Answer: D

Explanation: All the statements can be performed by attaching semantic rules to the given grammar. We can also store/retrieve type information in symbol table

Solution-6

Answer: C

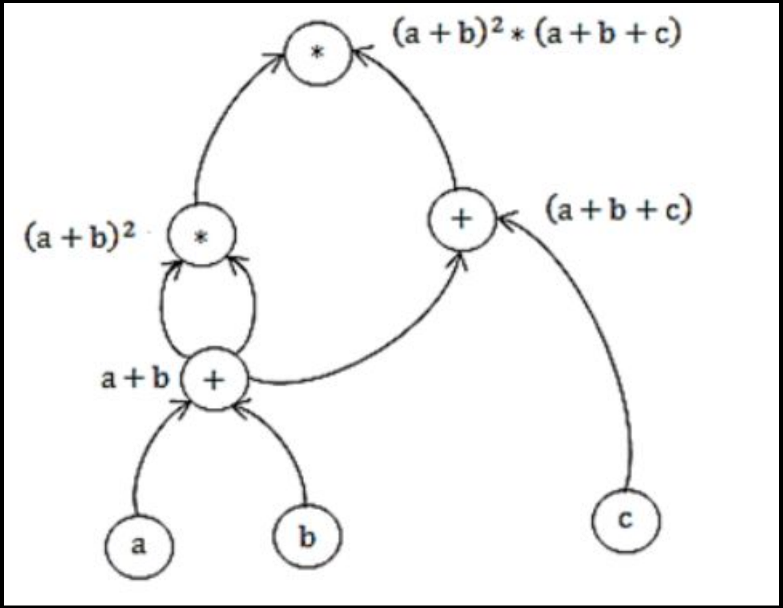




Solution-7

Answer:C

Explanation:



Solution-8

Answer: C

Explanation: Postfix form step by step is given by:

-a + b / c ↑ d ↑ e * f / g

Or, (a-) + b / c ↑ (de↑) *f / g

Or, (a-) + b / (cde ↑↑) *f / g

Or, (a-) + (bcde ↑↑/) *f / g

Or, (a-) + (bcde ↑↑/f * g /)

Or, a – bcde ↑ ↑ / f * g / +

Postfix form is: a – bcde ↑ ↑ / f * g / +

Three address code: t1 = -a

t2 = d ↑ e

t3 = c ↑ t2

t4 = b / t3

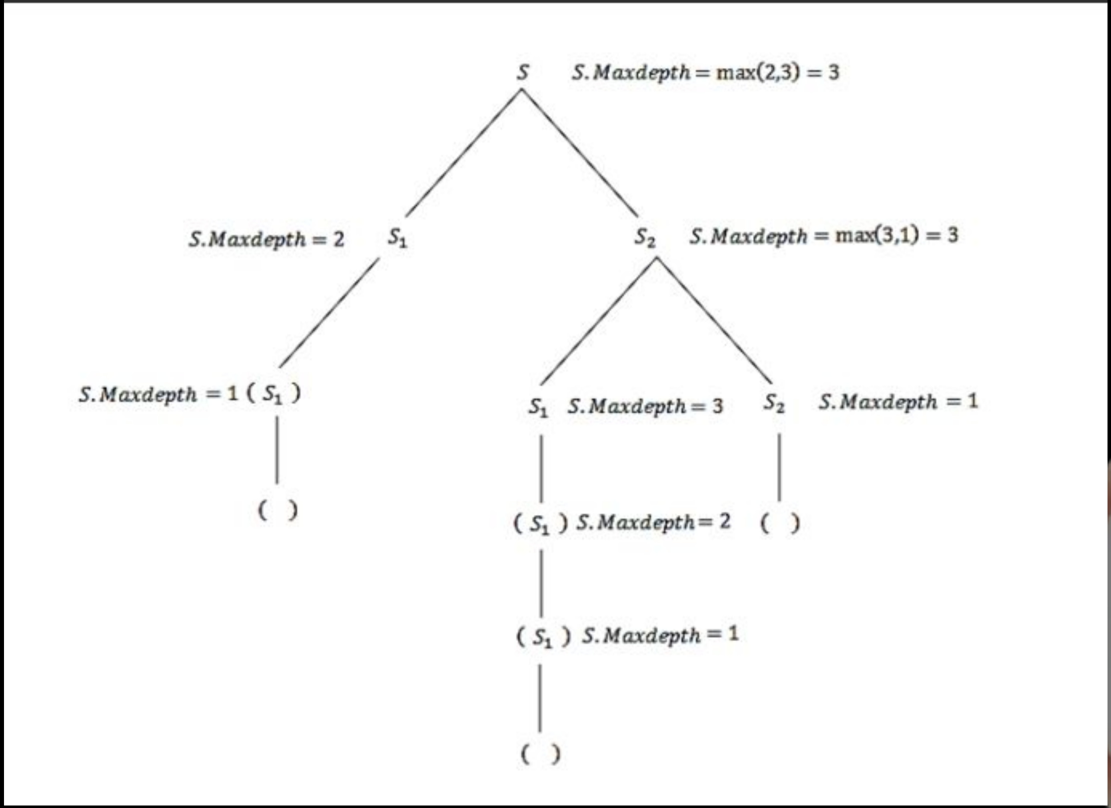
t5 = t4 * f

t6 = t5 / g

t7 = t1 + t6

Solution-9

Answer: D



S $S.Maxdepth = \max(2,3) = 3$
$S.Maxdepth = 2$ S_1
S_2 $S.Maxdepth = \max(3,1) = 3$
$S.Maxdepth = 1$ (S_1)
S_1 $S.Maxdepth = 3$
S_2 $S.Maxdepth = 1$
( )
(S_1) $S.Maxdepth = 2$
( )
(S_1) $S.Maxdepth = 1$
( )

Solution-10

Answer: A

Let us see how to write the SDT. The first step is define the grammar. As input is the same as the previous example, the same grammar can be used. Semantic actions are not the same because the expected output is different. Now take a simple string "a+b".

1. The parser first reads "a". It will be matched with the token "id". Now look at output, "ab+" , whatever is read should appear in output. So let the semantic action with reduce action F⟶id be {print("id·name");} , where "id·name" is the name of the identifier, whether it is "a" or "b" or "c".
2. Now look at the next step performed by the bottom-up parser T⟶F. Here during this reduction; do we have to propagate any additional information? No. As there is no additional information to be propagated, there is no need of assuming attributes to grammar symbols. So with reduce action T⟶F, there is no need of any semantic action. Hence, we define the translation as T⟶F{ }. Same thing happens with E⟶T also.

3. Next parser reads "+". Just with E+, it cannot perform any reduce action; hence reads "b". Now "b" is reduced to F, here "b" is printed out as F⟶id{print(id·name");} . Then it reduces, F to T, and there is no action.

4. The next reduction is "E⟶T"by E. Here already "ab" is printed and the only left out character to be printed is the operator "+". So whenever an expression is reduced, print operator. Add this as semantic action with the rule. The resulting SDT is shown below. E⟶E1+T {print("+");}

E⟶T { }

T⟶T1 *F {print(*);}

T⟶F { }

F⟶id {print(id.name);}

Solution-11

Answer: B

(A)

P -> + ‘(‘ P1 ‘)+(’ P2 ‘)’

-> + ‘(‘ 1 ‘)+(’ P2 ‘)’ -> + ‘(‘ 1 ‘)+(’ -‘(‘ P1 ‘)-(’ P2 ‘)’ ‘)’

-> + ‘(‘ 1 ‘)+(’ -‘(‘ 2 ‘)-(’ 3 ‘)’ ‘)’

So it prints (1)+((2)-(3)) which is equal to 0

(B)

P -> + ‘(‘ P1 ‘)+(’ P2 ‘)’

-> + ‘(‘ -‘(‘ P1 ‘)-(’ P2 ‘)’ ‘)+(’ P2 ‘)’

-> + ‘(‘ -‘(‘ 1 ‘)-(’ 2 ‘)’ ‘)+(’ 3 ‘)’

It prints ((1)-(2))+(3) which is equal to 2

Similarly Option (C) prints ((2)-(3))+(1) which is equal to 0

Option (D) prints ((1)+(2))-(3) which is equal to 0

Solution-12

Answer: D

Explanation:

With three look ahead symbols the top down parser is able to select the correct production. Hence it is LL(3)

Solution-13

Answer:C

Explanation: Statement 1: The tokens in C language include:

I. Keyword
II. Identifier
III. Constant
IV. String-literal
V. Punctuator
VI. Operators

If there is an invalid token which does not belong to the 6 types of tokens shown above we get a lex error.

Statement 2: Once the parser makes the list of legitimate tokens available to the error recovery routine then this routine can decide whether a remaining input's prefix matches one of these tokens closely enough to be treated as that token. This helps the lexical analyzer.

Solution-14

Answer 0

Static Semantic errors (z)

Redeclaration of an identifier

Lexical error (y)

Mistype the constant integer “Count12” as “Count 12”.

Syntactic error (x)

Unbalanced parenthesis

Use of uninitialized variable.

Runtime error

Array index out of bounds.

Preference of null pointer. Divide by 0.

$\Rightarrow$ Syntactic – (lexical + Static Semantic)

$\Rightarrow x - (y + z)$

$\Rightarrow 2 - (1 + 1) \Rightarrow 0$

Solution-15

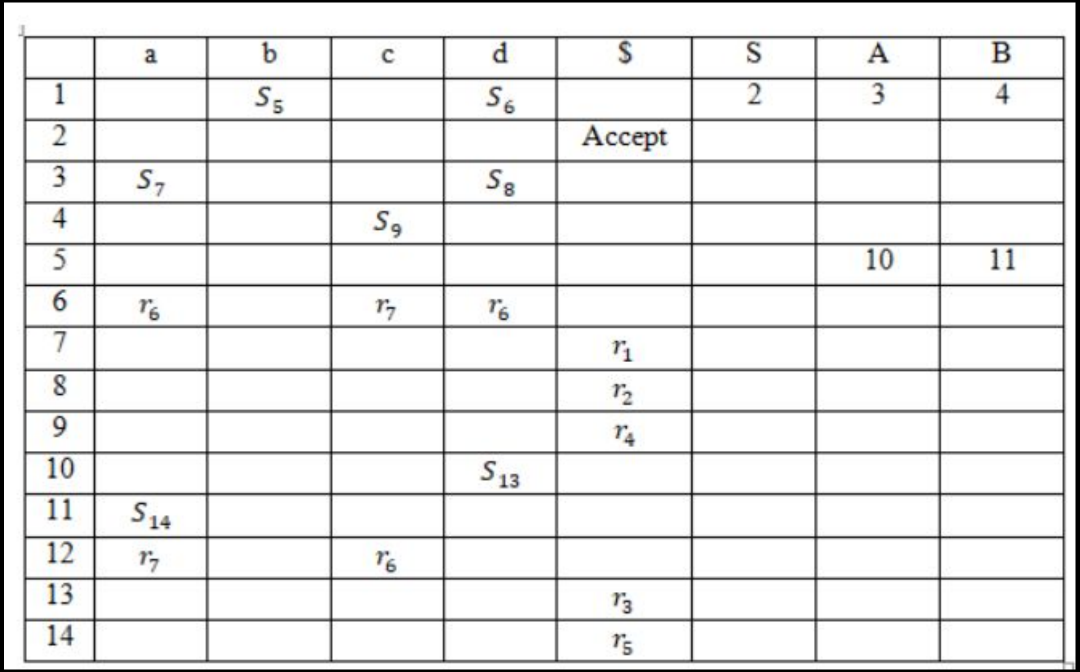
Answer: C
Explanation:

Option L-attributed definition can be evaluated in the framework of bottom up parsing is true. Although it is convenient to do top down parsing for L attributed definition framework, but still we can perform bottom up parsing for L-attributed definition. Also, one more way to think the given option is true, as if the L-attributed definition contains only synthesized attribute then also it can be evaluated in bottom up parsing.

Option A programming language which allows recursion cannot be implemented with static storage allocation is true. Recursion cannot be implemented with Static Storage Allocation. Static allocation means, compiler has to decide size for function calls. In case of recursion, it is not possible for compiler to decide as depth of recursion depends on recursion parameter which may be an input from user also.

Solution-16

Answer: A



	a	b	c	d	$	S	A	B
1		S_5		S_6		2	3	4
2					Accept			
3	S_7			S_8				
4			S_9					
5							10	11
6	r_6		r_7	r_6				
7					r_1			
8					r_2			
9					r_4			
10				S_{13}				
11	S_{14}							
12	r_7		r_6					
13					r_3			
14					r_5			

Solution-17

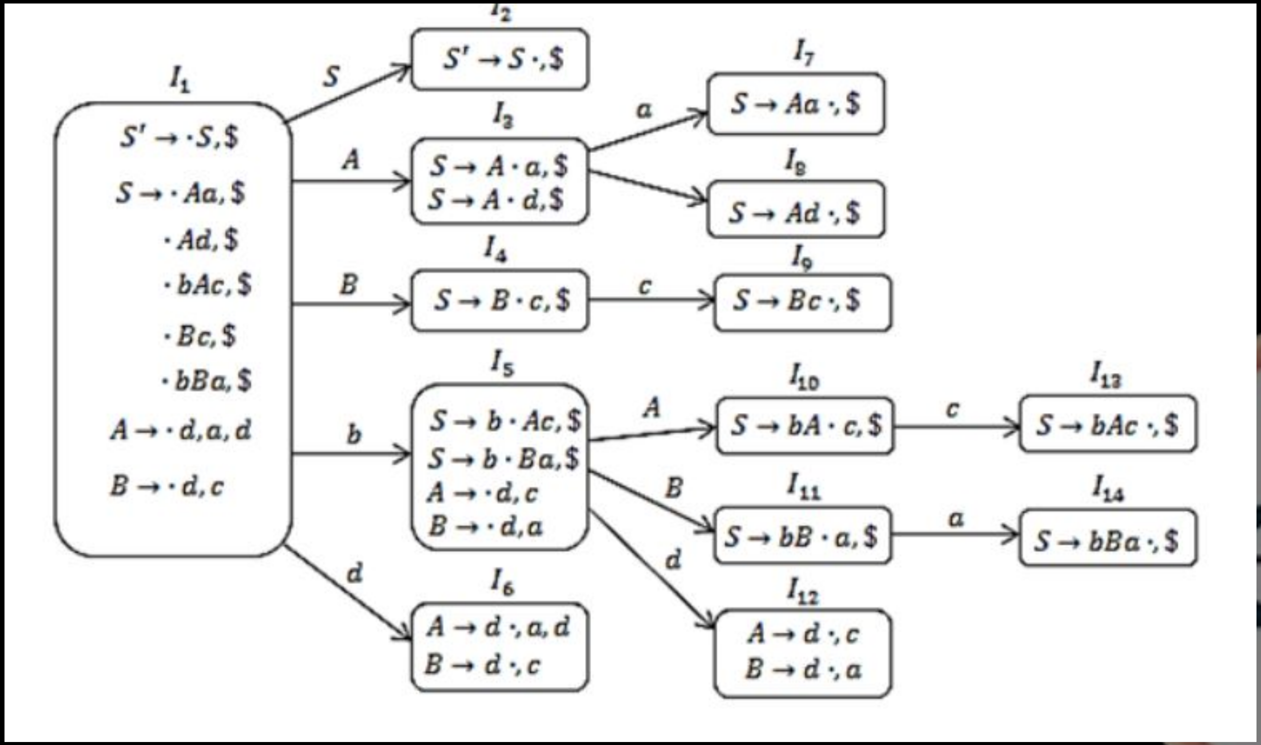
Answer: C

Explanation: Synthesized attribute: A synthesized attribute at node N is defined only in terms of attribute values at the children of N and at N itself. Inherited attribute: An inherited attribute at node N is defined only in terms of attribute values at N's parent, N itself, and N's siblings.

Solution-18

Answer: C

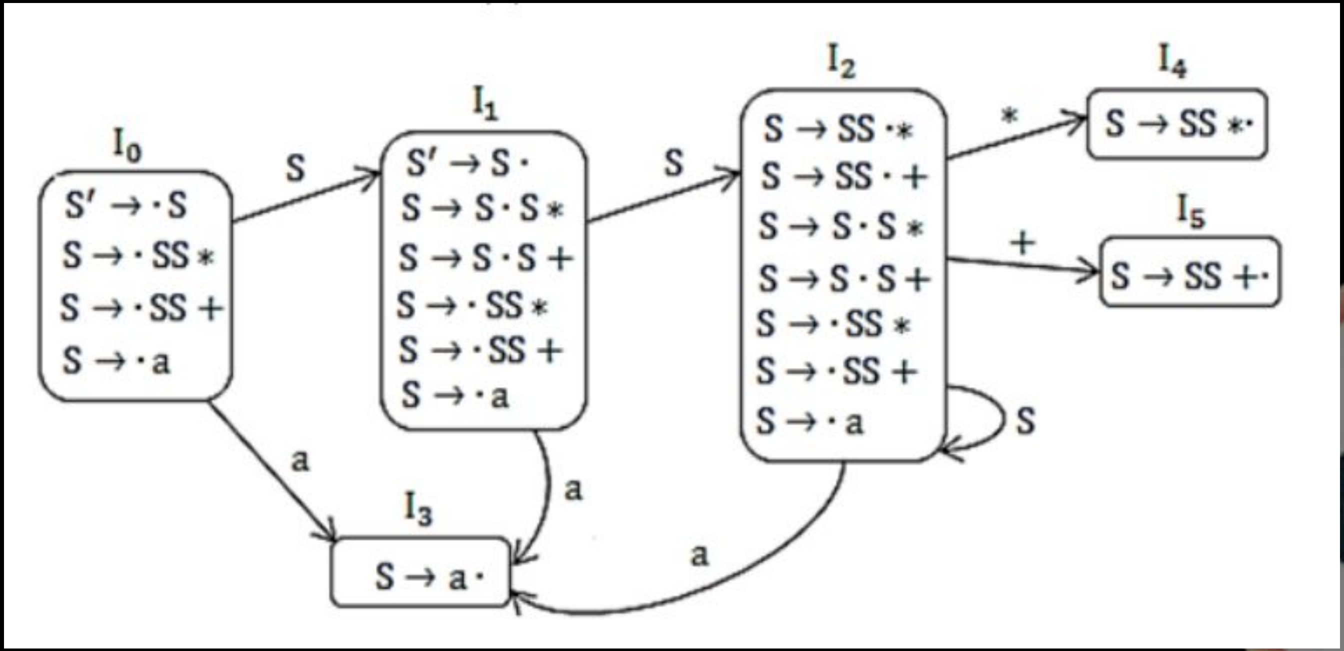
Explanation:



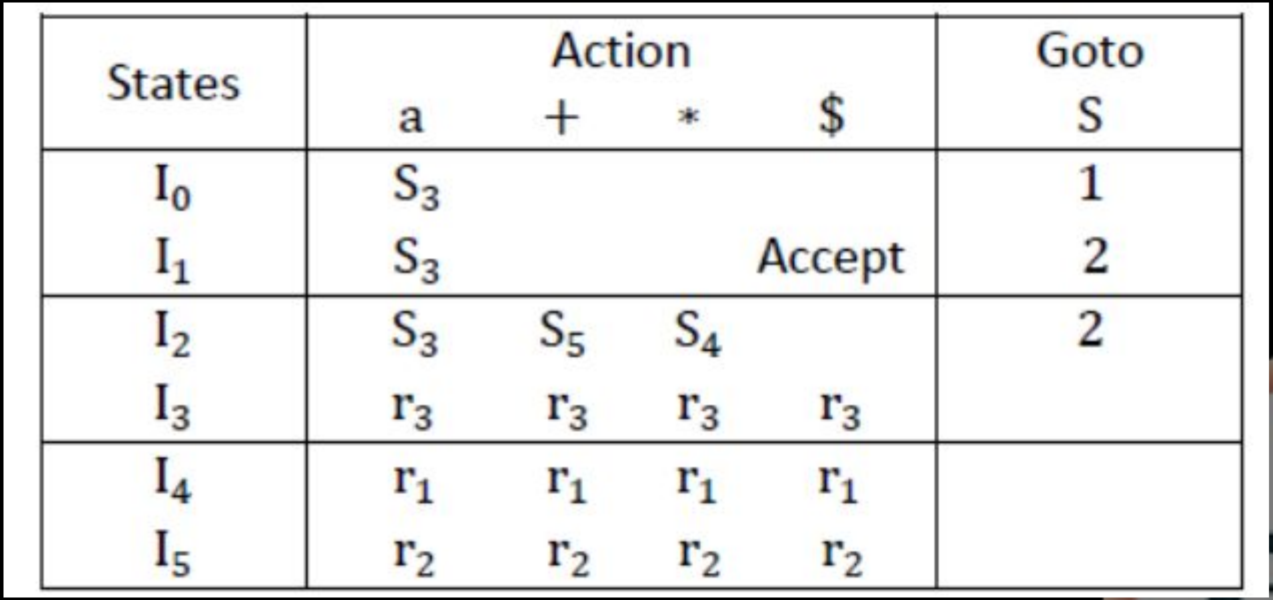
Solution-19

Answer: D

Explanation: The canonical collection of LR(0) items are:



The corresponding shift reduce parsing table is: .



States	Action				Goto
	a	+	*	$	S
I_0	S_3				1
I_1	S_3			Accept	2
I_2	S_3	S_5	S_4		2
I_3	r_3	r_3	r_3	r_3	
I_4	r_1	r_1	r_1	r_1	
I_5	r_2	r_2	r_2	r_2	

Algorithm to parse the input: Repeat for ever

1. If action[X,a]=Si , then (Shift action) Push “a” then state I onto stack. Advance the input pointer to the next symbol.

2. If action [X,a]=ri , then (Reduce action) {

Take ri , that is, the ith production from the grammar. Let ri is A→β.

Pop 2*|β| symbols from stack and replace them by nonterminal A.

If Xi is the state below the nonterminal A, then push goto[Xi,A] onto the stack. Output the production A→β }

Now continue parsing.

3. If action [X,a]=accept, then (Successful completion)

All the LR parsers use the same parsing algorithm.

Stack	Input	Action
\$	aaa + a * +\$	Push state 0
\$0	aaa + a * +\$	Shift ‘a’ and then 3,
\$0a3	aa + a * +\$	Reduce S → a and Goto[0, 5]=1
\$0S1	aa + a * +\$	Shift ‘a’ and then 3
\$0S1a3	a + a * +\$	Reduce S → a and Goto[1, 5]=2
\$0S1S2	a + a * +\$	Shift ‘a’ and then 3
\$0S1S2a3	+a * +\$	Reduce S → a and Goto[2, 5]=2
\$0S1S2S2	+a * +\$	Shift ‘+’ and then 5
\$0S1S2S2 + 5	a * +\$	Reduce S → SS+ and Goto[1, 5]=2
\$0S1S2	a * +\$	Shift ‘a’ and then 3
\$0S1S2a3	* +\$	Reduce S → a and Goto[2, 5]=2
\$0S1S2S2	* +\$	Shift ‘*’ and then 4
\$0S1S2S2 * 4	+\$	Reduce S → SS* and Goto[1, 5]=2
\$0S1S2	+\$	Shift ‘+’ and then 5
\$0S1S2 + 5	\$	Reduce S → SS+ and Goto[0, 5]=1

Solution-20

Answer: B

Explanation: If a grammar G is LL(1) then it must be CLR(1) . Hence it may or may not be SLR(1). So statement I is false and II is true. The number of states in SLR(1) and LALR(1) is equal hence there GOTO part and shift entries are same. So statement II and IV are true.

Solution-21

Answer: A

Explanation: The given grammar is L-attributed with semantic actions in between the grammar symbols. Suppose we have an L-attributed as follows:

S⟶A{ }B

How to convert it to an equivalent S-attributed definition? It’s very simple!! Replace actions by nonterminal as follows:

S⟶AMB

M⟶ϵ { }

So the productions of E'' can be written as

E''⟶+TME'' |ϵ

M⟶ϵ {Print("+");}

∴The equivalent S-attributed definition is

E⟶TE'' E''⟶+TME'' |ϵ M⟶ϵ {Print("\"+\"");} T⟶num {print("num");}))

Solution-22

Answer: A

Explanation: It is not S-attributed as F.val is inherited attribute. It is not L-attributed as F.val is depending on T1.val, since in L-attributed definition is: Each attribute must be either

1. Synthesized
2. Inherited, but with the rules limited as follows. Suppose that there is a production A⟶ X1 X2…Xn and that there is an inherited attributeXi.a computed by a rule associated with this production. Then the rule may use only:

a. Inherited attributes associated with the head A.

b. Either inherited or synthesized attributes associated with the occurrences of symbols X1, X2,. . . , Xi-1 located to the left of Xi.

Solution-23

Answer: A

Left recursion is in the productions: A⟶Aff,B⟶Bd and B⟶Bf

The productions of A are of the form X⟶Xα|β1 |β2

To eliminate left recursion, rewrite the productions as X⟶β1 X' |β2 X' X'⟶αX' |ϵ

∴The productions of A can be replaced by the productions

A⟶BdA' |CBA'

A'⟶ffA' |ϵ

The productions of B are of the form X⟶Xα1 |Xα2 |β

To eliminate left recursion, rewrite the productions as X⟶βXX'

X'⟶α1 |X' | α2 X' |ϵ

∴The productions of B can be replaced by the productions B⟶dfB'

B'⟶dB' |fB' |ϵ

∴The grammar after eliminating left recursion is A⟶BdA' |CBA'

A'⟶ffA' |ϵ

B⟶dfB'

B'⟶dB' |fB' |ϵ

C⟶h|g

Solution-24

Answer:

Explanation: A token is a pair consisting of a token name and an optional attribute value.

A lexeme is a sequence of characters in the source program that matches the pattern for a token and is identified by the lexical analyzer as an instance of that token.

Also identifier may have variable length.

For example:int value=3;

Lexeme is “value”

Token is <id,3>

Solution-25

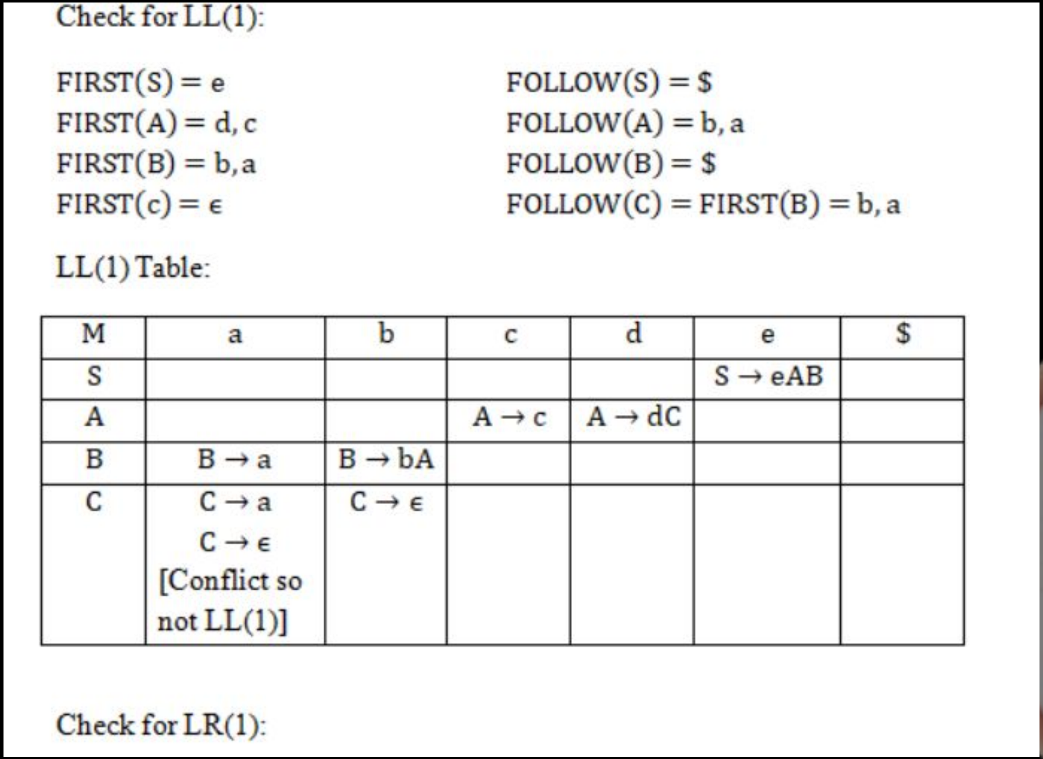
Answer: A

"Handle" is a substring that matches the right side of a production, and whose reduction to the nonterminal on the left side of the production represents one step along the reverse of a right most derivation.

Solution-26

Answer: B

Explanation:



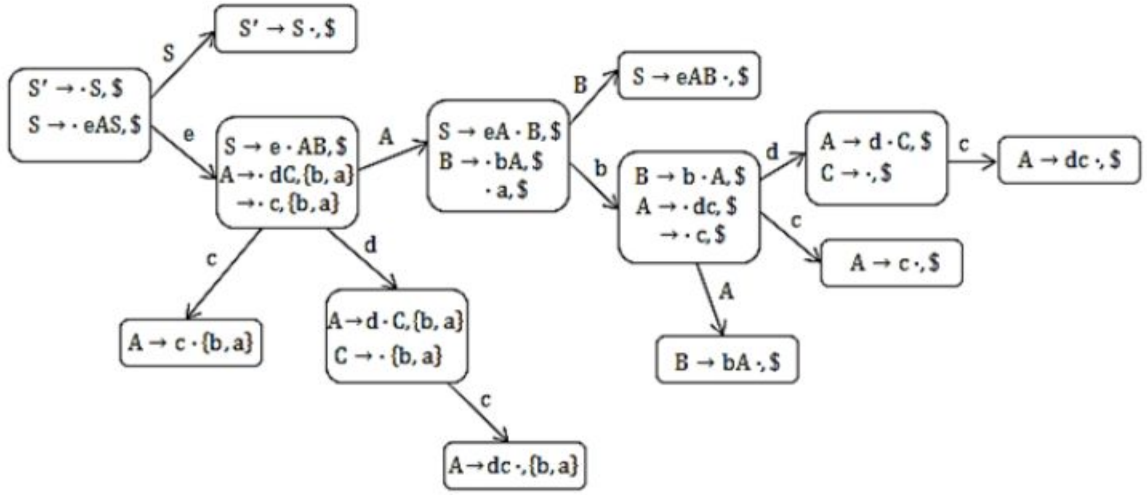
Check for LL(1):

FIRST(S) = e	FOLLOW(S) = \$
FIRST(A) = d, c	FOLLOW(A) = b, a
FIRST(B) = b,a	FOLLOW(B) = \$
FIRST(c) = ϵ	FOLLOW(C) = FIRST(B) = b, a

LL(1) Table:

M	a	b	c	d	e	\$
S					$S \rightarrow eAB$	
A			$A \rightarrow c$	$A \rightarrow dC$		
B	$B \rightarrow a$	$B \rightarrow bA$				
C	$C \rightarrow a$ $C \rightarrow \epsilon$ [Conflict so not LL(1)]	$C \rightarrow \epsilon$				

Check for LR(1):



No conflict in any state. So LR(1).

Solution-27

Answer: D

reduce entry will be present in terminal symbol (small) c, while shift entry is due to non-terminal (capital) C

Solution-28

Answer: B

Explanation: The grammar is not LL(1) because all the three productions for S conflict with one another, because they all start (directly/ indirectly) with the terminal symbol ‘noun’. This gives a three-way First/ First conflict. The two productions for M also give a First/ First conflict, because the grammar is left-recursive.

Solution-29

Answer: D

Explanation: Programming language which allow recursion can not be implemented by static storage allocation.Ex : C supports recursion so require stack storage allocation.

Activation record contains following parameters : Local variable,globalvariable,temporary variable,non localfunction,returnaddress,actualparameter,machine status.

In heap storage allocation,memory can be allocated or deallocated at any time by using malloc() and free(). Triple representation requires less space than quadraple.

Solution-30

Answer: A

The grammar

```
stmt ->if expr then stmt
        |if expr then stmt else stmt
        |other
```

When we construct parsing table it will cause shift reduce conflict because at else we will not be able to decide whether shift or reduce and also the given grammar is ambiguous.

Solution-31

Answer: D

Explanation: Statement I: FALSE

Symbol table is the data structure, which is used in all phase, that is, from lexical analysis till code generation and optimization.

Statement II: FALSE

Recursion mandatorily requires stack memory during the runtime environment, not heap memory.

Statement III: FALSE

Error such as any variable must be declared before its use is semantic error and thus cannot be detected during syntax analysis.

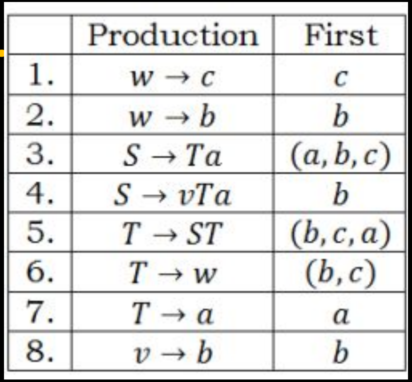
Solution-32

Answer: C
Explanation:
S1:
```
int| main |(| int| i|,| int| j|)|
{|
a|++|-|+|--|b|;|
printf|(|"a+b=%d"|,|c|)|;|
c|>=|d|!=|e|;|
return| i|>|j|?|i|:|j|;| /*return something*/
}|
```
40 tokens
S2:
In an operator precedence grammar, two consecutive non-terminals or ϵ is not allowed and given grammar contains ϵ.
∴not operator precedence grammar.
S3:
Top down parser generates string top to bottom, left to right. Ex.: S⟶L,S|a,L⟶S S⟶L,S ⟶S,S ⟶a,S ⟶a,a
Bottom up parser generates string bottom to top in reverse order as

Solution-33

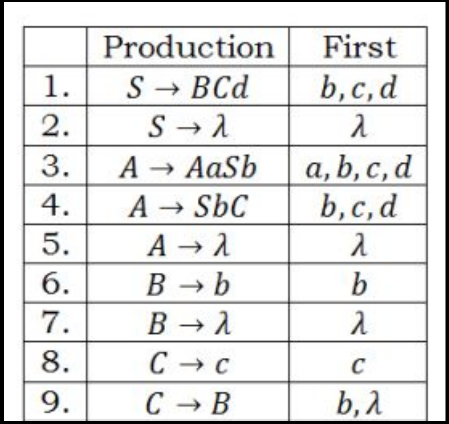
Answer: A

I.



	Production	First
1.	$w \to c$	c
2.	$w \to b$	b
3.	$S \to Ta$	(a, b, c)
4.	$S \to vTa$	b
5.	$T \to ST$	(b, c, a)
6.	$T \to w$	(b, c)
7.	$T \to a$	a
8.	$v \to b$	b

II.



	Production	First
1.	$S \to BCd$	b, c, d
2.	$S \to \lambda$	λ
3.	$A \to AaSb$	a, b, c, d
4.	$A \to SbC$	b, c, d
5.	$A \to \lambda$	λ
6.	$B \to b$	b
7.	$B \to \lambda$	λ
8.	$C \to c$	c
9.	$C \to B$	b, λ

$\because 3 \cap 4 \neq \phi, 5 \cap 6 \neq \phi$

$\therefore$ not LL(1)

$\because 3 \cap 4 \neq \phi$ $\therefore$ not LL(1)

Solution-34

Answer:D

Since the programming language is to be designed to run on a machine that does not have a big memory. Therefore pass-2 (multiphase), which uses less space is preferred also instead of using a compiler we use an interpreter, which scans the program line by line hence uses less memory at compiler time for each instance. Also it does not support recursion.

Solution-35

Answer:A

Explanation:

A compiler compiles the program in one time while the interpreter compiles it taking line by line. Hence, security checking is more in interpreter while compiler is fast in compilation. Hence, it can generate stand-alone programs that often take less time for execution

Solution-36

Answer: D

Explanation: This relation is established of basis of the precedence of operators.

Solution-37

Answer:B

Explanation:Lexical analysis removes comments as well. (i=0) will not generate lexical error. It will tokenise as fro, (, i,=,0,). No error is generated.

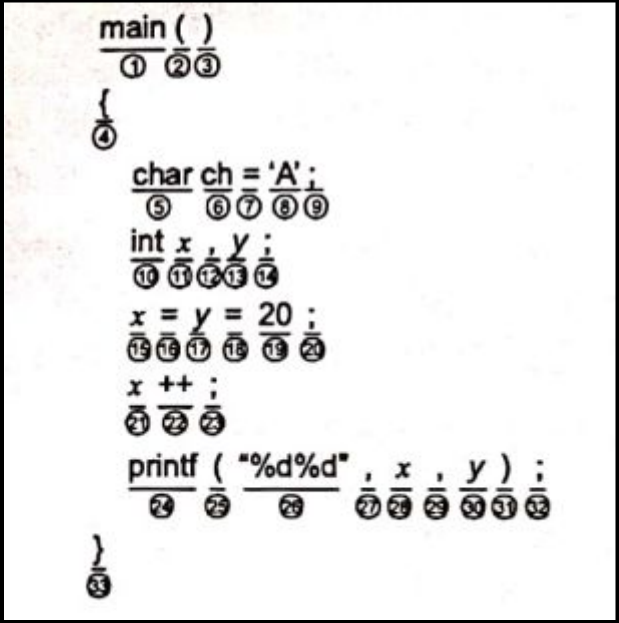
Solution-38

Answer: C
The C-code contains syntax error, where 'fro' is written in place of 'for'. The lexical analyzer will not declare it as an error, whether 'fro' is a misspelling of the keyword 'for' or an undeclared function identifier. Since 'fro' is a valid lexeme for the token id, the lexeme analyzer must return the token id to the parser and let some other phase of the compiler (parser in this case) handle the error due to transposition of the letters

Solution-39

Answer: C

Explanation: TOTAL-33

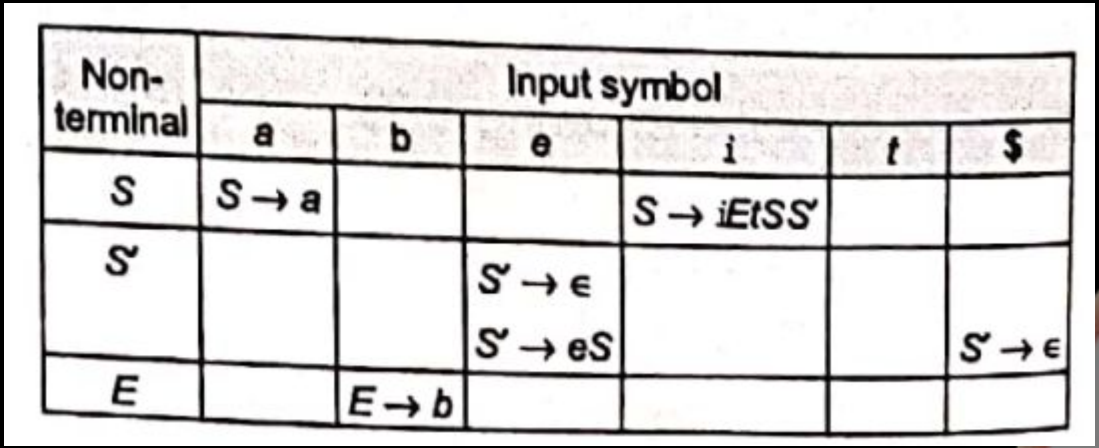


```
main ( )
{
   char ch = 'A' ;
   int x , y ;
   x = y = 20 ;
   x ++ ;
   printf ( "%d%d" , x , y ) ;
}
```

Solution-40

Answer: D

Explanation:

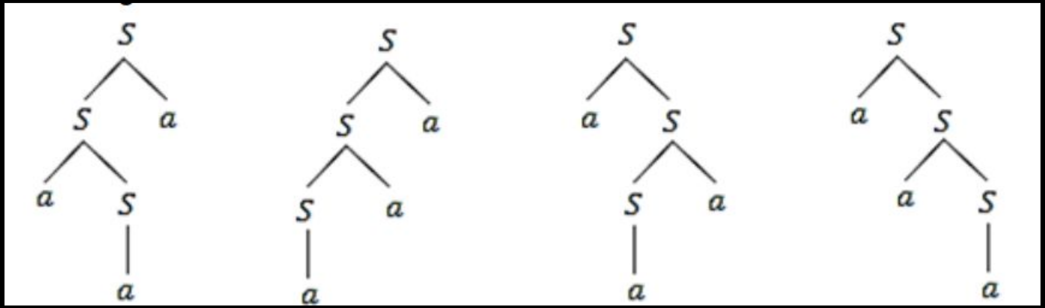


Non-terminal	Input symbol					
	a	b	e	i	t	\$
S	$S \rightarrow a$			$S \rightarrow iEtSS'$		
S'			$S' \rightarrow \epsilon$ $S' \rightarrow eS$			$S' \rightarrow \epsilon$
E		$E \rightarrow b$				

Solution-41

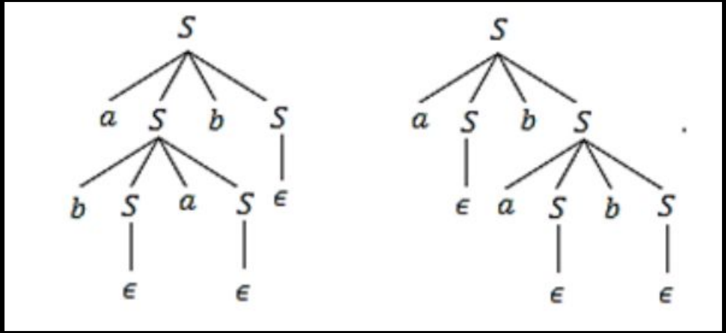
Answer:D

(i) $S \longrightarrow Sa \mid aS \mid a$ for string w=aaa



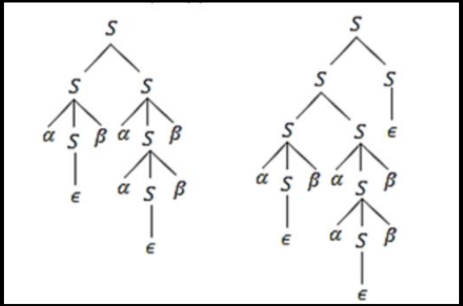
(ii) $S \longrightarrow aSbS \mid bSaS \mid \epsilon$

w=abab



(iii) S⟶αSβ|SS|ϵ

To prove this asunambiguous we need to prove that for every string that generates this grammar, there should be exactly one parse tree. So, we prove by contradiction. We try to find a string which has more than one parse tree to disprove the statement. Consider w=αβααββ



So the given grammar is ambiguous.

(iv) There are some ambiguous grammars which cannot be defined with an equivalent unambiguous grammar. These are called inherently ambiguous. For example, L={a^i b^j c^k| i=j or j=k}

The grammar for the above language is S⟶S1 |S2

S1⟶S1 c|A S2⟶aS2 |B

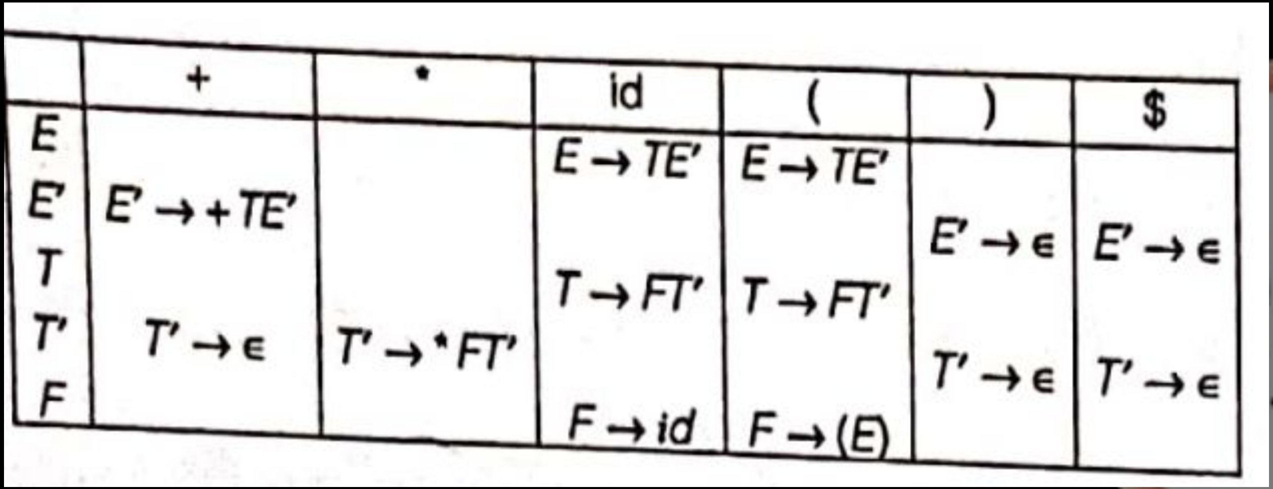
A⟶aAb|ϵ

B⟶bBc|ϵ This cannot be converted into an equivalent unambiguous gramma

Solution-42

Answer: C

Explanation: Predictive parsing table for the given grammar are created by calculating FIRST and FOLLOW for each nonterminal first. Then make entries according to the algorithm of predictive parsing table.

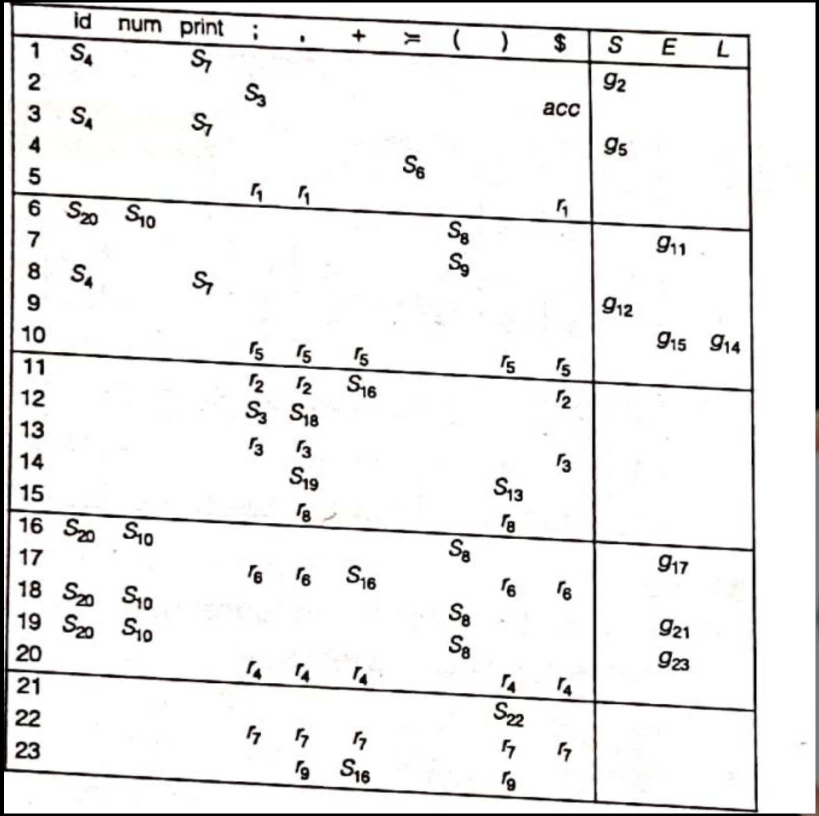


	+	*	id	(	)	$
E			$E \rightarrow TE'$	$E \rightarrow TE'$		
E'	$E' \rightarrow +TE'$				$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T			$T \rightarrow FT'$	$T \rightarrow FT'$		
T'	$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$			$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F			$F \rightarrow id$	$F \rightarrow (E)$		

Solution-43

Answer: C

Explanation:



	id	num	print	;	,	+	:=	(	)	$	S	E	L
1	S_4		S_7								g_2		
2				S_3						acc			
3	S_4		S_7								g_5		
4							S_6						
5				r_1	r_1					r_1			
6	S_{20}	S_{10}						S_8				g_{11}	
7								S_9					
8	S_4		S_7								g_{12}		
9												g_{15}	g_{14}
10				r_5	r_5	r_5			r_5	r_5			
11				r_2	r_2	S_{16}				r_2			
12				S_3	S_{18}								
13				r_3	r_3					r_3			
14					S_{19}				S_{13}				
15					r_8				r_8				
16	S_{20}	S_{10}						S_8				g_{17}	
17				r_6	r_6	S_{16}			r_6	r_6			
18	S_{20}	S_{10}						S_8				g_{21}	
19	S_{20}	S_{10}						S_8				g_{23}	
20				r_4	r_4	r_4			r_4	r_4			
21									S_{22}				
22				r_7	r_7	r_7			r_7	r_7			
23					r_9	S_{16}			r_9				

Solution-44

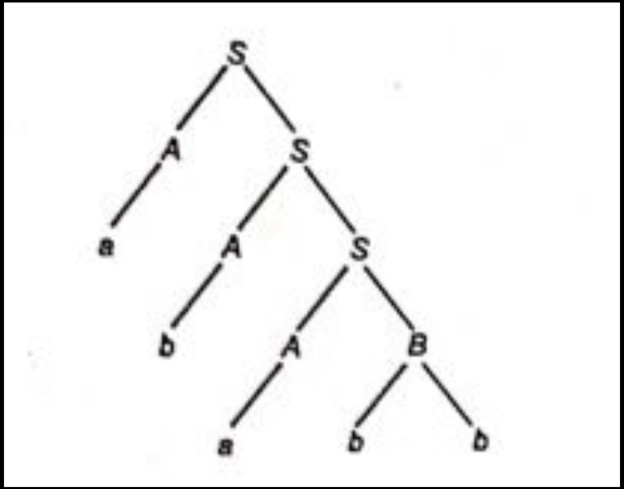
Answer: A

S ->AB |AS

A-> a| Aa

B->b|bb

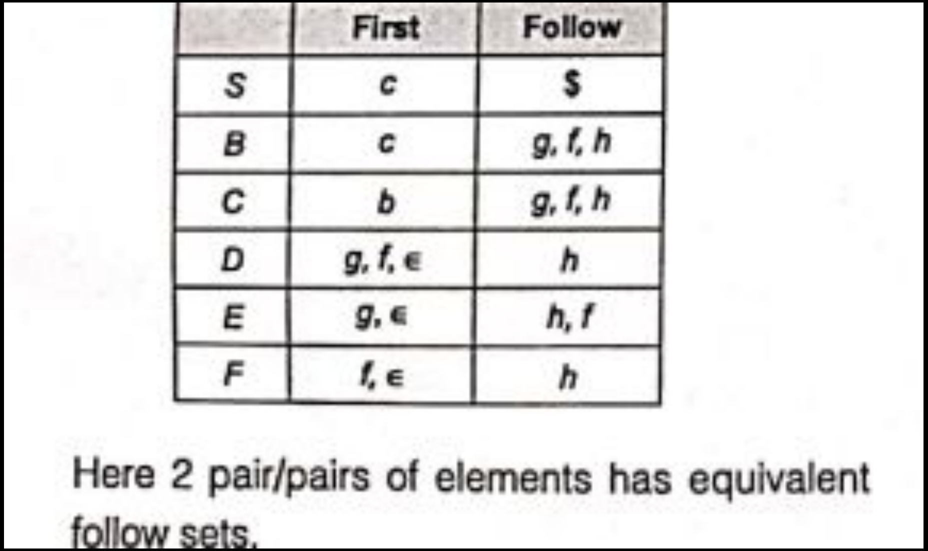
will generate the parse tree



Solution-45

Answer: c

Explanation:



	First	Follow
S	c	$\$$
B	c	g, f, h
C	b	g, f, h
D	$g, f, \in$	h
E	$g, \in$	h, f
F	$f, \in$	h

Here 2 pair/pairs of elements has equivalent follow sets.

Solution-46

Answer: B

Explanation:

S->AA-> aA⟶ aa

S->AA-> aA-> abA⟶ aba

S⟶AA⟶ Aa->aAb⟶ aab

S->AA-> Aa->bAa⟶ baa

(aa, aba, aab, baa) can be generated within 4 steps.

Solution-47

Answer: D

Explanation: Handles are the part of sentential form, & they are identified as the right side of any given production which will be used for reduction in the next step.

Solution-48

Answer: B

Explanation:

FOLLOW (S)= {c, $)

FIRST (S) FIRST (MNZSc)

= (a, b, z).

Solution-49

Answer: .4

E’->.E,$ ➡1 Look-a-heads

E->.E+T, {$,+} ➡ 2 Look-a-heads

E->.T, {$,+} ➡2 Look-a-heads

T->.T*F, {*,-, $, +} ➡ 4 Look-a-heads

T->.F (*,-,$, +) ➡ 4 Look-a-heads

T->.(E), (*,-, $, +} ➡ 4 Look-a-heads

T->.T-F (*,-,$,+] ➡ 4 Look-a-heads

T->.id, (*,-, $, +) ➡ 4 Look-a-heads

Solution-50

Answer:C

Explanation:

F->id.

F->id=T

Shift-Reduce conflict

F->id.

T->id.

Reduce-Reduce conflict

Solution-51

Answer: A

Explanation:

In the given grammar for derivation of any string 'b' and 'c' comes first then atlast 'a' come. So a has less precedence than b and c but c don't have less precedence than a.

Solution-52

Answer .A

Explanation:

$$\begin{tikzcd}
\text{FOLLOW}(A) = \text{FIRST}(BC) \arrow[d] \\
c, \text{FOLLOW}(B) \arrow[d] \\
\text{FIRST}(C) \arrow[d] \\
d, \text{FOLLOW}(C) \arrow[d] \\
\text{FOLLOW}(S) \arrow[d] \\
\$
\end{tikzcd}$$

FOLLOW (A) is (c, d, $)

Solution-53

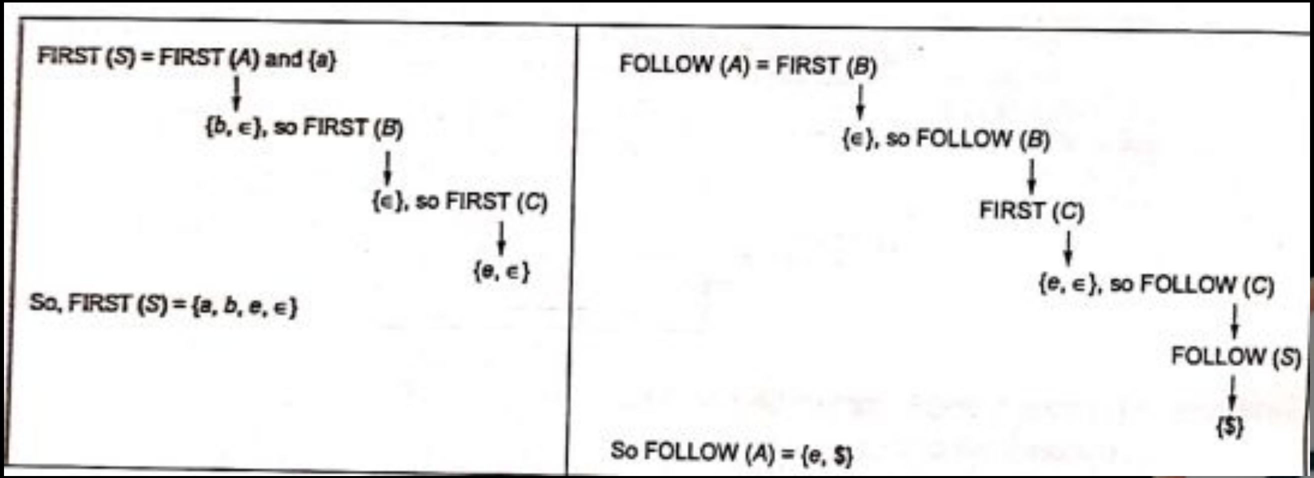
Answer .B

$$\begin{tikzcd}
 & A \arrow[dl] \arrow[dr] & \\
B & & C
\end{tikzcd}$$

According to L attributed definition, B's attributes can depend on inherited attributes of A but not on synthesized attributes of A

Solution-54

Answer . A



FIRST (S) = FIRST (A) and $\{a\}$

$$\begin{tikzcd}
\text{FIRST}(A) \arrow[d] \\
\{b, \epsilon\}\text{, so FIRST}(B) \arrow[d] \\
\{\epsilon\}\text{, so FIRST}(C) \arrow[d] \\
\{e, \epsilon\}
\end{tikzcd}$$

So, FIRST $(S) = \{a, b, e, \epsilon\}$

FOLLOW (A) = FIRST (B)

$$\begin{tikzcd}
\text{FIRST}(B) \arrow[d] \\
\{\epsilon\}\text{, so FOLLOW}(B) \arrow[d] \\
\text{FIRST}(C) \arrow[d] \\
\{e, \epsilon\}\text{, so FOLLOW}(C) \arrow[d] \\
\text{FOLLOW}(S) \arrow[d] \\
\{\$\}
\end{tikzcd}$$

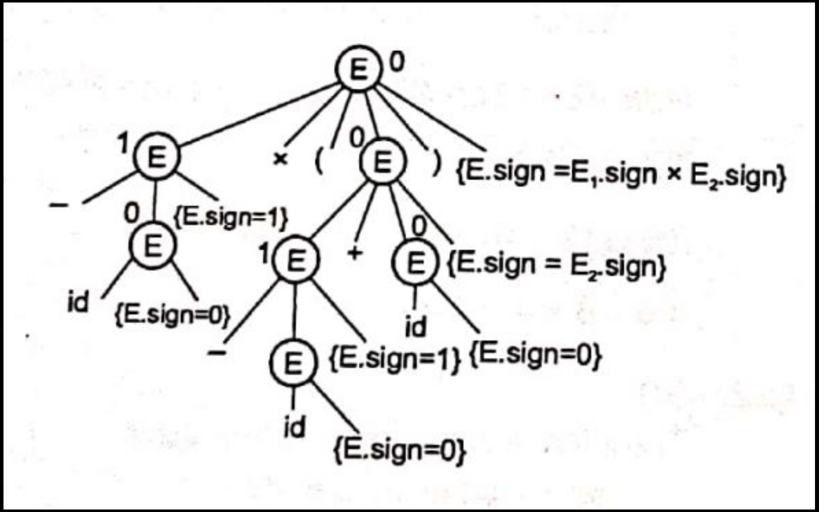
So FOLLOW $(A) = \{e, \$\}$

Solution-55

Answer .A

Concept:

Input: -id x (-id + id)



The value computed at root is "0".

Solution-56

Answer B

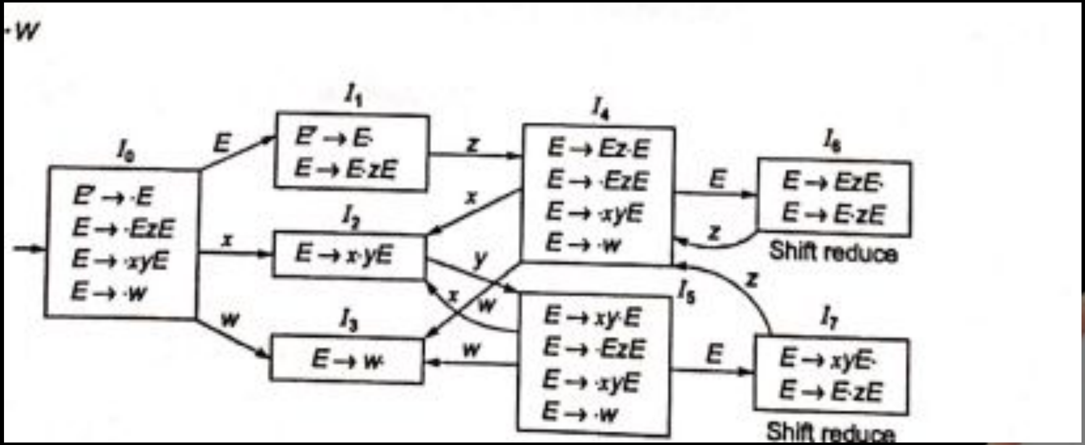
Augmented grammar G:

E'->.E

E- >.EzE

E->.xyE

E'->.w



In state, and I, shift reduce conflict is present. The element to be shifted is reduced so they will come in same column

Solution-57

Answer B

Explanation: In P1, S is a synthesized attribute and in L-attribute definitionsynthesized is allowed. So, P1 follows L-attributed definition. But P2 doesn’tfollow L-attributed definition as P is depending on Q which is RHS to it.

Solution-58

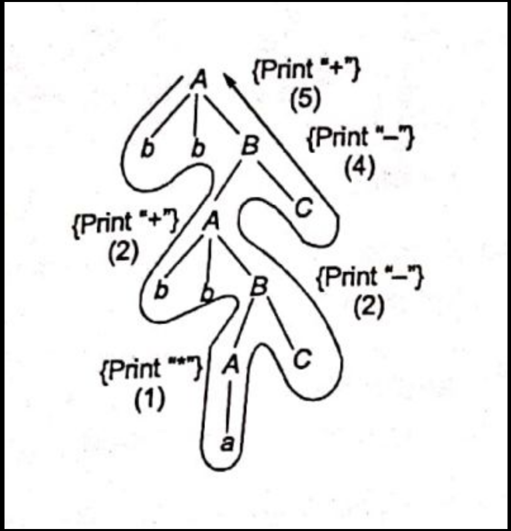
Answer: D

A⟶ bbB (Print "+"}

A⟶ a {Print "*"}

B⟶ Ac {Print "-"}

INPUT:bbbbacc



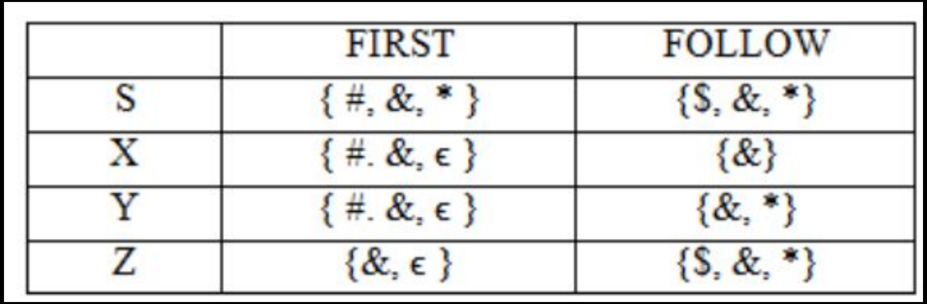
The above syntax tree prints *-+-+

Hence, (D) is correct option.

Solution-59

Answer: B

Explanation:



	FIRST	FOLLOW
S	{ #, &, * }	{$, &, *}
X	{ #. &, ϵ }	{&}
Y	{ #. &, ϵ }	{&, *}
Z	{&, ϵ }	{$, &, *}

Solution-60

Answer: D

Take relational algebra which generates the longest subsequence.

With T3 we get a subsequence of 5, and T_1 we get a subsequence of only 4, T_2 we get a subsequence of only 3.

Hence, T3 preferred over T2 and T1

Explanation

String = bbaacabc

(bla)* = (a + b)*

c? = (e + c)

T3: c? (ba)+c

T3 = bbaac (longest prefix match)

T3= abc

T3T3 = bbaac abc

Therefore option d is correct

Solution-61

Answer: C

SDD is L-attributed iff:

The inherited attributes of Y, in the production.

$X \longrightarrow Y_1\ Y2 \ldots\ Yi\ \ Yi + 1\ \ldots\ Y\ p$

- Should only be derived from only attributes

    Y_1 to Y+1

-Should only be derived from inherited attribute of X.

Hence in first production rule Q is being derived from synthesized attribute of A. Hence SDD is not L-attribute because of 1's rule

Solution-62

Answer: D

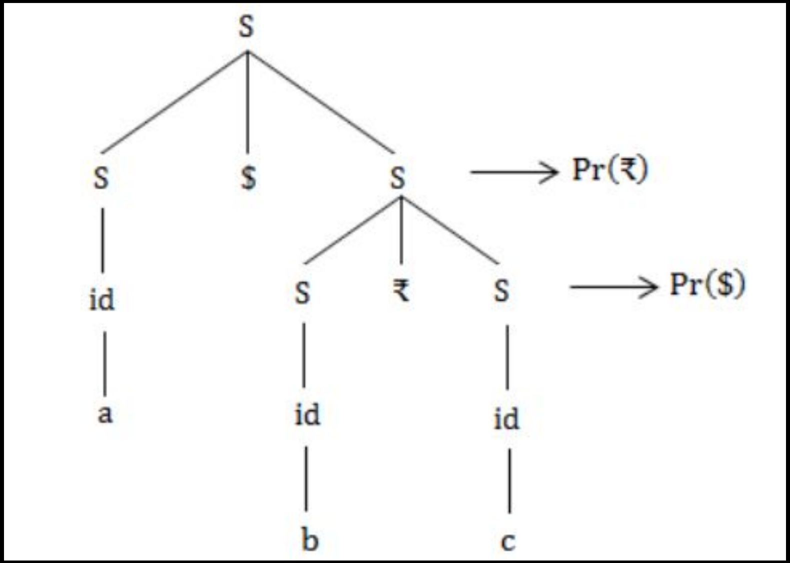
Explanation: A language L has an LR(0) grammar if and only if L is a DCFL with prefix property. This is the statement of a theorem.

L is said to have a prefix property if, whenever “w” is in L, no proper prefix of “w” is in L.

Clearly language in S2 is a DCFL and has prefix property, so S2 is true. While the language L=a*b* is a regular language and it doesn’t follow the prefix property. So S1 is false statement

Solution-63

Answer: A
Explanation:
a$b₹c



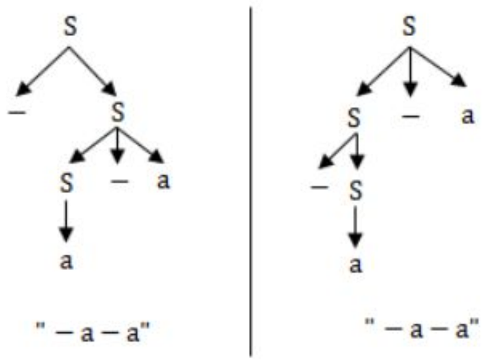
Output-abc$₹

Solution-64

Answer: D

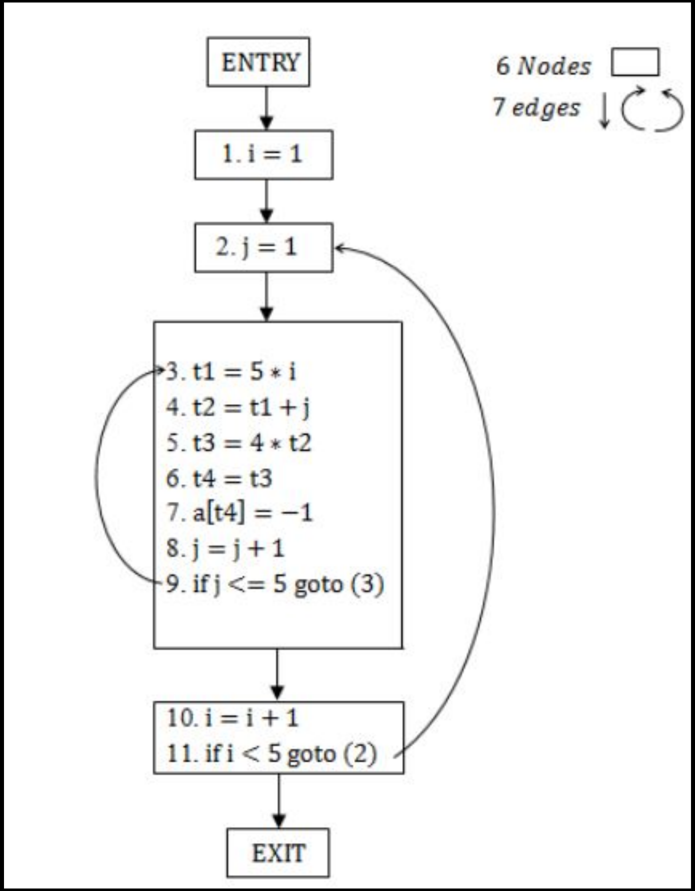
Explanation

The grammar "G" is ambiguous, as it has two different parse tree for string "$- a - a$".



Solution-65

Answer. D


ENTRY
1. i = 1
2. j = 1
3. t1 = 5 * i
4. t2 = t1 + j
5. t3 = 4 * t2
6. t4 = t3
7. a[t4] = −1
8. j = j + 1
9. if j <= 5 goto (3)
10. i = i + 1
11. if i < 5 goto (2)
EXIT
6 Nodes
7 edges