Q 1.

Consider the following context free grammar

$S \rightarrow X$

$X \rightarrow Yb \mid aa$

$Y \rightarrow a \mid bYa$

The given grammar is

a) Not LL(1)
b) LR(0) and SLR(1)
c) Not LR(0) but SLR(1)
d) Neither LR(0) nor SLR(1)

Q 2.

Consider the following context free grammar

$S \rightarrow A$

$A \rightarrow BC \mid DBC$

$B \rightarrow bB' \mid \varepsilon$

$B' \rightarrow bB' \mid \varepsilon$

$C \rightarrow c \mid \varepsilon$

$D \rightarrow a \mid d$

Number of symbols in the first of A are _____

Q 3.

Consider the following translation scheme:

$S \rightarrow ER$

$R \rightarrow * E\{print(" * ");\}R \mid \varepsilon$

$E \rightarrow F + E\ \{print(" + ");\} \mid F$

$F \rightarrow (S) \mid id\ \{print(id.\ value);\}$

What does this translation scheme print for the input: '6 + 7 * 9 + 2'?

a) 67+*92+
b)67+92+*
c)6+7+9+2*
d)6792+*

Q 4. If we merge states in LR(1) parser to form a LALR(1) parser, we may introduce

A. shift-reduce conflict
B. reduce-reduce conflict
C. no extra conflict
D. both shift-reduce as well as reduce-reduce

Q 5.

Suppose we have a rightmost derivation which proceeds as follows:

$$S \rightarrow Aabw$$
$$\rightarrow ABw$$

Which of the following is a possible handle for it?

A. $A \rightarrow ab$
B. $A \rightarrow a$
C. $S \rightarrow A$
D. $B \rightarrow ab$

Q 6.

Which of the following statements is FALSE?

A. Any DCFL has an equivalent grammar that can be parsed by a SLR(1) parser with end string delimiter
B. Languages of grammars parsed by LR(2) parsers is a strict super set of the languages of grammars parsed by LR(1) parsers
C. Languages of grammars parsed by LL(2) parsers is a strict super set of the languages of grammars parsed by LL(1) parsers
D. There is no DCFL which is not having a grammar that can be parsed by a LR(1) parser

Q 7.

Which of the following statements is FALSE?

A. In a SLR(1) parser, it is allowable for both shift and reduce items to be in the same state
B. In a SLR(1) parser, it is allowable for multiple reduce items to be in the same state
C. All SLR(1) grammars are LR(0)
D. All LR(0) grammars are SLR(1)

Q 8.

Which of the following statements regarding $LR(0)$ parser is FALSE?

A. A $LR(0)$ configurating set cannot have multiple reduce items
B. A $LR(0)$ configurating set cannot have both shift as well as reduce items
C. If a reduce item is present in a $LR(0)$ configurating set it cannot have any other item
D. A $LR(0)$ parser can parse any regular grammar

Q 9. Which of the following sentences is CORRECT?

A. A top-down parse produces a leftmost derivation of a sentence
B. A bottom-up parse produces a rightmost derivation of a sentence
C. A top-down parse produces a rightmost derivation of a sentence
D. A bottom-up parse produces a leftmost derivation of a sentence

Q 10. Which of the following is a requirement for an LL(1) grammar?

I. Unambiguity
II. No left recursion
III. If $A \rightarrow \alpha \mid \beta$ are two productions, then $FIRST(\alpha)$ and $FIRST(\beta)$ are disjoint

A. (i) and (ii)
B. (iii)
C. (i), (ii) and (iii)
D. (ii) and (iii)

Q 11. Which of the following sentences regarding Viable prefixes is/are CORRECT?

I. Viable prefixes is the set of prefixes of right-sentential forms that can appear on the stack of a shift-reduce parser
II. Viable prefixes is the set of prefixes of right-sentential forms that do not extend past the end of the right-most handle
III. Viable prefixes can be recognized using a DFA

A. Only (i)
B. Only (ii)
C. Only (i) and (ii)
D. (i), (ii) and (iii)

Q 12. For which of the following languages a LL(1) grammar does not exist?

A. $\{a^n o b^n \mid n \geq 1\} \cup \{a^n b^n \mid n \geq 1\}$

B. $\{a^n b^m \mid m, n \geq 0\}$

C. $\{a^i b^j \mid i \geq j\}$

D. $\{a^i b^j \mid i = j\}$

Q 13.

Which of the following is TRUE regarding LL(0) grammar?

A. We can have a LL(0) grammar for any regular language
B. We can have a LL(0) grammar for a regular language only if it does not contain empty string
C. We can have a LL(0) grammar for any regular language if and only if it has prefix property
D. We can have a LL(0) grammar for only single string languages

Q 14.

Which of the following is TRUE regarding the running time of a LR(1) parser?

A. It runs in linear time for all inputs
B. It runs in polynomial time but not necessarily $O(n^3)$ for all inputs
C. For some inputs it may take exponential time
D. It runs in $O(n^3)$ but not always $O(n^2)$

Q 15.

Which of the following statements regarding LR parsers is WRONG?

I. LR(1) does no guess work
II. LR parsers can handle a large range of grammars than predictive parsers
III. LR parsers can handle a large range of languages than predictive parsers
IV. LR parser is better at error reporting compared to LL ones

A. Only (i)
B. (i) and (iv)
C. Only (iv)
D. All are CORRECT

Q 16.

Match the following:

(i)	$LL(1)$	(A)	bottom-up
(ii)	Recursive Descent	(B)	Predictive
(iii)	Recursive Ascent	(C)	Top-down
(iv)	$LR(1)$	(D)	Deterministic CFL

A. i-b; ii-c; iii-a; iv-d

B. i-d; ii-a; iii-c; iv-d

C. i-c; ii-b; iii-d; iv-a

D. i-a; ii-c; iii-b; iv-d

Q 17.

Which of the below relations does hold TRUE regarding GRAMMARS?

A. $LL(1) \subset SLR(1) \subset LR(1)$

B. $SLR(1) \subset \epsilon - \text{free } LL(1) \subset LR(1)$

C. $\epsilon - \text{free } LL(1) \subset SLR(1) \subset LR(1)$

D. $LL(1) \subset SLR(1) = LR(1)$

Q 18.

Match the following:

a.	Canonical Parser	1.	No adjacent nonterminals
b.	SLR(1) Parser	2.	Follow sets must be disjoint
c.	LL(1) Parser	3.	Most powerful parser
d.	Operator Precedence Grammar	4.	Top-down parser

A. $a-2; b-3; c-1; d-4$
B. $a-3; b-4; c-2; d-1$
C. $a-2; b-1; c-4; d-3$
D. $a-3; b-2; c-4; d-1$

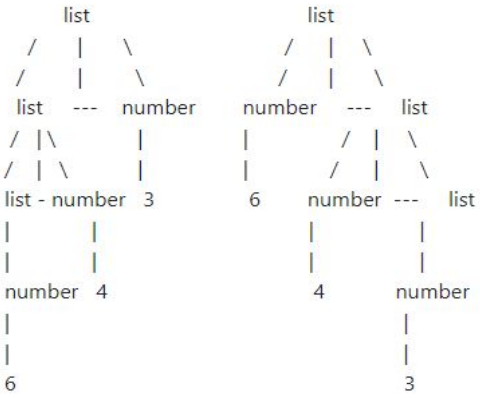
Q 19.

The default type of conflict resolution rule for the *Yacc parser generator* is _____.

A. in favor of shift
B. in favor of reduce
C. either shift or reduce
D. dependent on a situation

Q 20.

Consider the following two parse trees for the expression: $6 - 4 - 3$



(a) (b)

Next, consider the below statements:

1. The parse tree (a) represents right associative operator evaluation and the parse tree (b) represents left associative evaluation.
2. The grammar generating the sentence is not ambiguous.

Which of the above statements are FALSE?

A. Only 2
B. 1 and 2
C. Only 1
D. Both statements are correct

Q 21. A top-down parser generates _____.

A. right-most derivation
B. right-most derivation in reverse
C. left-most derivation
D. left-most derivation in reverse

Q 22.

_____ is the most powerful parsing method.

A. LL(1)
B. LALR
C. SLR
D. Canonical LR

Q 23.

Consider the following grammars:

$Z \rightarrow TZ'$

$Z' \rightarrow^* TZ' \mid \epsilon$

$T \rightarrow YT'$

$T' \rightarrow^* YT' \mid \epsilon$

$Y \rightarrow (Z) \mid id$

Which of the following is First(Z)?

A. $\{T,)\}$

B. $\{(, id\}$

C. $\{), \$\}$

D. None of the above

Q 24.

The number of proper prefixes for a string of length n are _____.

1. $n+1$
2. $n(n+1)/2$
3. $n-1$
4. $(n-1)/2$

Q 25.

Read the following grammar:

$$S \rightarrow Ka \mid bKc \mid dc \mid bda$$
$$K \rightarrow d$$

This grammar is NOT:

A. LALR(1)
B. SLR(1)
C. LR(1)
D. None of the above

Q 26.

_____ parsers build parse trees starting from the root node and work down to the leaves.

A. LR
B. LL
C. SLR
D. LALR

Q 27.

The grammar which has no epsilon transition or two adjacent nonterminals in the right side of any production is _____.

A. LL(1) grammar
B. Unambiguous grammar
C. Operator grammar
D. Context Sensitive grammar

Q 28.

In the construction of $LL(1)$ parsing table for the following grammar $M[S, (]\&M[W, \$]$ are respectively

- $S \rightarrow XY$
- $X \rightarrow (S) \mid \text{int } W$
- $Y \rightarrow +S \mid \epsilon$
- $W \rightarrow^* X \mid \epsilon$

A. $X \rightarrow (S),\ W \rightarrow \epsilon$
B. $S \rightarrow XY,\ W \rightarrow^* x$
C. $S \rightarrow XY,\ W \rightarrow \epsilon$
D. $S \rightarrow YX,\ W \rightarrow \epsilon$

Q 29.

Read the below mentioned grammar:

$S \rightarrow X$

$X \rightarrow YX \mid \epsilon$

$Y \rightarrow aY \mid b$

This grammar is NOT:

A. $LALR$
B. $LR(0)$
C. $LR(1)$
D. None of the above

Q 30.

Consider following Translation Scheme:

$S \rightarrow ER$

$R \rightarrow^{*} E\{print('*');\}R \mid \epsilon$

$E \rightarrow F + E\{print('+');\} \mid F$

$F \rightarrow S \mid id\{print(id, value);\}$

Here, id is a taken that represents an integer and id, $value$ represents the corresponding integer value.

What does this Translation Scheme print for an input " $1 * 7 + 9$ "?

A. $17 * +9$

B. $1 * 79+$

C. $179 * +$

D. $179 + *$

Q 31.

Consider the following grammar:

$E' \rightarrow E$

$E \rightarrow E + Y \mid Y$

$Y \rightarrow Y^{*} F \mid F$

$F \rightarrow id \mid (E)$

How many $LR(0)$ items are there in closure $(\{E' \rightarrow \cdot E\})$?

A. 1
B. 6
C. 7
D. 5

Q 32.

Which of these is NOT true about Abstract Syntax Tree (AST)?

A. An AST is usually the result of the syntax analysis phase of a compiler.
B. AST has no impact on the final output of the compiler.
C. AST is a tree representation of the abstract syntactic structure of source code written in a programming language.
D. AST is also used in program analysis and program transformation systems.

Q 33.

Consider the following grammars:

1. $S \rightarrow aS \mid Sa \mid \in$
2. $E \rightarrow E + E \mid E^* E \mid id$
3. $A \rightarrow AA \mid (A) \mid a$
4. $S \rightarrow SS \mid AB,\ A \rightarrow Aa \mid a,\ B \rightarrow Bb \mid b$

These grammars are:

A. Ambiguous
B. Unambiguous
C. Regular
D. Inherently Ambiguous

Q 34. Which one of the following statements is TRUE?

A. SLR parser is more powerful than LALR.
B. LALR parser is more powerful than Canonical LR.
C. Canonical LR is more powerful than LALR parser.
D. SLR, Canonical LR and LALR parsers have the same power.

Q 35.

Assume that the SLR(1) parser for a grammar has P1 states, and the CLR(1) parser for the same has P2 states. Which of the following clearly depicts the relationship between P1 and P2?

A. $P1 \leq P2$
B. $P1 = P2$
C. $P1 \geq P2$
D. It depends on the grammar.

Q 36.

Consider the following grammar:

- $Z \rightarrow TZ'$
- $Z' \rightarrow +TZ' \mid \epsilon$
- $T \rightarrow YT'$
- $T' \rightarrow^* YT' \mid \epsilon$
- $Y \rightarrow (Z) \mid id$

Which of the following represents Follow(Y)?

A. $\{), id\}$
B. $\{), id, \$, *, +\}$
C. $\{^*, +,), \$\}$
D. $\{+,), \$\}$

Q 37.

The least number of temporary variables required to create a 3 address code sequence for the statement $L = P + R$ is _____.

Q 38.

Which one of the following can be handled by predictive parsers?

A. Left recursion
B. Left factors
C. Ambiguity
D. Non-determinism

Q 39.

The language which supports _____ needs heap allocation in the runtime environment.

A. Dynamic Data Structure
B. Static Scoping
C. Global Variable
D. Recursion

Q 40. Which one of the following statements is TRUE?

A. SLR parser has more states than LALR parser.
B. LALR parser has more states than Canonical LR.
C. Canonical LR has fewer states than SLR parser.
D. Both SLR and LALR parsers have the same number of states.

Q 41.

Consider the following grammar:

$E \rightarrow E + T \mid T$
$T \rightarrow T^{*} F \mid F$
$F \rightarrow (E) \mid id$

What are the productions for E, T and F after converting the above mentioned grammar to LL(1) grammar?

A. $E \rightarrow +TE',\ T \rightarrow^{*} FT',\ F \rightarrow (E) \mid id$
B. $E \rightarrow +TE' \mid \epsilon,\ T \rightarrow^{*} FT' \mid \epsilon$
C. $E \rightarrow T,\ T \rightarrow F,\ F \rightarrow (E) \mid id$
D. $E \rightarrow TE',\ T \rightarrow FT',\ F \rightarrow (E) \mid id$

Q 42. Consider the following grammar:

- $S \rightarrow aMd \mid bNd \mid aNe \mid bMe$
- $M \rightarrow c$
- $N \rightarrow c$

The grammar above is:

A. LR(1) but not LALR(1)
B. LALR(1) but not SLR(1)
C. SLR(1) but not LR(1)
D. LALR(1)

Q 43. In compilers, the type checking is done in:

A. Lexical Analysis
B. Semantic Analysis
C. Code Generation Phase
D. Parsing Phase

Q 44.

goto function of LR class of grammar is represented as:

A. Deterministic Finite Automata transitions
B. Non-deterministic Finite Automata transitions
C. PDA transitions
D. Parsing table

Q 45.

Consider the following syntax directed definition of any desk calculator:

1. $L \rightarrow En\{\text{print}(E.val)\}$
2. $E \rightarrow E1 + 1\{E.val = E1.val + Z.val\}$
3. $E \rightarrow ZE \cdot val = Z \cdot val$
4. $Z \rightarrow Z1^{*}F\{Z.val = Z1.val^{*}F.val\}$
5. $Z \rightarrow F\{Z \cdot val = F \cdot val\}$
6. $F \rightarrow (E)\{F \cdot val = E \cdot val\}$
7. $F \rightarrow digit\{F \cdot val = digit \cdot lexval\}$

How many internal nodes are there in annotated parse tree for input $7^{*}4 + 2n$?

A. 9
B. 6
C. 10
D. 11

Q 46.

Which grammar causes recursive-descent parser to go into infinite loop?

A. LL(1)
B. Left recursive grammar
C. Right recursive grammar
D. Grammar with left factors

Q 47.

Consider the following grammar:

$S \rightarrow L = P \mid P$

$L \rightarrow^* P \mid id$

$P \rightarrow L$

The above grammar is:

A. Ambiguous

B. SLR(1)

C. LALR(1)

D. None of the above

Q 48.

Given the following statement: 6 * (8-3)

Make the Annotated Parse Tree based on the translation scheme below.

S → B {A.i = B.val} A {S.val = A.s}

A → - B {A1.i = A.i - B.val} A1 {A.s = A1.s}

A → e {A.s = A.i}

B → D {C.i = D.val} C {B.val = C.s}

C → * D {C1.i = C.i * D.val} C1 {C.s = C1.s}

C → e {C.s = C.i}

D → (S) {D.val = S.val}

D → num {D.val = num.val}

Q 49.

Synthesized attribute can easily be simulated by an

A. LL grammar
B. ambiguous grammar
C. LR grammar
D. none of the above

Q 50.

Syntax directed translation scheme is desirable because

A. It is based on the syntax
B. It is easy to modify
C. Its description is independent of any implementation
D. All of these

Q 51.

Consider the following SDD

$T \rightarrow FT1$ { F.val = T1.val}

$T2 \rightarrow *AT3${T2.val=T3.val×A.val}

Which of the following is true?

1This SDD follows L-attributed definition

2This SDD follows S-attributed definition.

3None of the above.

4This SDD follows both S-attributed as well as L-attributed definition.

Q 52.

Consider the following productions along with their semantic rules

Production	Semantic Rules
$T \rightarrow FT'$	$T'.i = F.val$ $T.val = T'.s$
$T' \rightarrow *FT_1'$	$T_1'.i = T'.i + 2 * F.val$ $T'.s = T_1'.s$
$T' \rightarrow \epsilon$	$T'.s = T'.i$
$F \rightarrow id$	$F.val = id.lexval$

Here $\cdot\, i$ and $\cdot\, s$ corresponds to inherited and synthesized attributes respectively. The value for expression 3 $*$ 4 _____.

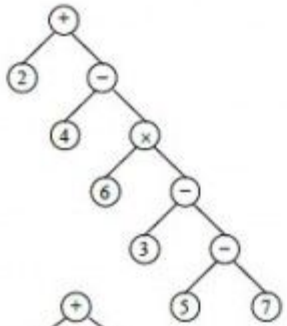
Q 53.

Consider the following grammar:

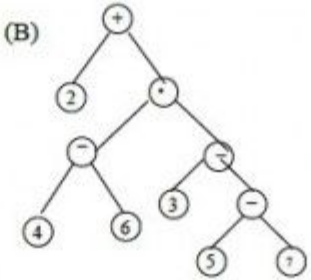
$S \rightarrow$	$S_1 + E$	$\{S.val = S_1.val + E.val\}$
	$\mid E$	$\{S.val = E.val\}$
$E \rightarrow$	$F - E_1$	$\{E.val = F.val - E_1.val\}$
	$\mid F$	$\{E.val = F.val\}$
$F \rightarrow$	$F_1 * T$	$\{F.val = F_1.val * T_1.val\}$
	$\mid T$	$\{F.val = T.val\}$
$T \rightarrow$	num	$\{T.val = num.val\}$

What is the parse tree for the expression $2 + 4 - 6 * 3 - 5 - 7$

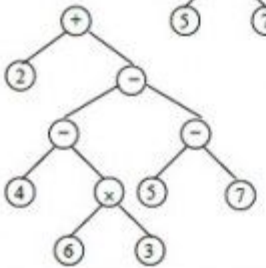
(A)



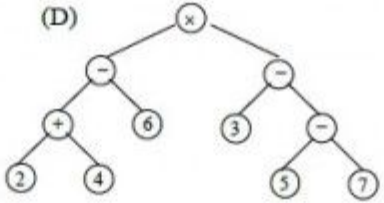
(B)



(C)



(D)



Q 54.

$Q_1 \Rightarrow$ For synthesized attribute, value is only computed when symbol is on the _____ side of production.
For inherited attribute, value is computed in productions where symbol is on the _____ side.

1. Left, Right **2.** Right, Right **3.** Right, Left **4.** Left, Left

---

$Q_2 \Rightarrow$ In static scoping undefined variable

$a)$. Search in upper block
$b)$. Search where the function is called
$c)$. Refers to global variable
$d)$. None of these

Q 55.

Consider the $SDTS$ for the ambiguous grammar

$E \rightarrow E + E \qquad out(''1 + 2'')$

$E \rightarrow E^{*}E \qquad out(''2^{*}3'')$

$E \rightarrow num \qquad out(num.val)$

Assume a shift reduce parser. The output is treated as an arithmetic expression in C & evaluated. The input is $1^{*}1 + 1$.

The value obtained is _____.

Q 56.

Fill in the blanks

Consider the $SDTS$ below

$E1 \rightarrow E\ out(*2)$

$E \rightarrow +T\ out('1')$

$E \rightarrow T\ out(10^*)$

$E \rightarrow T * F\ out('*')$

$T \rightarrow F\ out('100 +')$

$F \rightarrow num\ out(num\ val)$

The input is $1 + 1 * 1$ and the output generated is evaluated as an arithmetic expression . The value obtained is _____.

Q 57.

The dangling else problem in the construct If (E) sales S | is (E) S | a can be resolved IN SDTS by

(A) Using the associative & precedence of operating & the 'exe' munch principle
(B) By change the grammar to an unambiguous one
(C) Cannot be removed as it Is undecidable
(D) None of the above

Q 58.

The identification of common sub-expression and replacement of run time computations by compile-time computations is:

A. Local optimization
B. Constant folding
C. Loop Optimization
D. Data flow analysis

Q 59.

Which of the following statement(s) regarding a linker software is/are true ?

I. A function of a linker is to combine several object modules into a single load module.

II. A function of a linker is to replace absolute references in an object module by symbolic references to locations in other modules.

A. Only I

B. Only II

C. Both I and II

D. Neither I nor II

Q 60.

The task of adjusting programs so that they may be placed in arbitrary core locations is called relocation. This task is often performed by the relocating loaders. Given below are the tasks performed by the relocating loaders. Identify the correct sequence in which the tasks need to be performed.

1. Resolve symbolic references between object decks
2. Allocate space in memory for the program
3. Physically place the machine instructions and data into the memory
4. Adjust all address dependent locations

A. 1 2 3 4
B. 2 1 3 4
C. 3 1 4 2
D. 2 1 4 3

Q 61. Which of the Following is True ?

A. Symbol table Construction is during the analysis part of the Compiler.

B. Type checking is Done during Syntax Analysis phase

C. SDD with only synthesized attribute have an order of evaluation

D. Both A and C

Q 62. Match the following with respect to activation record fields:

Group A	Group B
1. Control link	A. Points to activation record of calling procedure
2. Access link	B. Refers to non local data in other activation record
	C. Dynamic link
	D. Static link field

A 1 → A, D; 2 → B, C

B 1 → A, C; 2 → B, D

C 1 → B, C; 2 → A, D

D 1 → B, D; 2 → A, C

Q 63.

Consider the following table :

A.	Activation record	**p.**	Linking loader
B.	Location counter	**q.**	Garbage collection
C.	Reference counts	**r.**	Subroutine call
D.	Address relocation	**s.**	Assembler

A. $a - p, b - q, c - r, d - s$
B. $a - q, b - r, c - s, d - p$
C. $a - r, b - s, c - q, d - p$
D. $a - r, b - s, c - p, d - q$

Q 64.

Consider the following statements:

S1 **:** Static allocation can not support recursive function.

S2 **:** Stack allocation can support pointers but can not deallocate storage at run-time.

S3 **:** Heap allocation can support pointers and it can allocate or deallocate storage at run-time.

Which of the above statements are true?

a *S*1 and *S*2

b *S*2 and *S*3

c *S*3 and *S*1

d *S*1, *S*2 and *S*3

Q 65.

In an absolute loading scheme which loader function is accomplished by assembler ?

(A) re-allocation

(B) allocation

(C) linking

(D) loading

Ans 1.	Ans 2.
Ans 3.	Ans 4. B
Ans 5.D	Ans 6.B
Ans 7.C	Ans 8.D
Ans 9.A	Ans 10.C
Ans 11.D	Ans 12.C
Ans 13.D	Ans 14.A
Ans 15.D	Ans 16.A
Ans 17.C	Ans 18.D
Ans 19.D	Ans 20.B
Ans 21.C	Ans 22.D
Ans 23.B	Ans 24.C

Ans 25.B	Ans 26.B
Ans 27.C	Ans 28.C
Ans 29.B	Ans 30.D
Ans 31.C	Ans 32.B
Ans 33.A	Ans 34.C
Ans 35.A	Ans 36.C
Ans 37.1	Ans 38.B
Ans 39.A	Ans 40.D
Ans 41.D	Ans 42.A
Ans 43.B	Ans 44.A
Ans 45.A	Ans 46.B
Ans 47.C	Ans 48.30

Ans 49.C	Ans 50.D
Ans 51.C(3)	Ans 52.11
Ans 53.C	Ans 54.D
Ans 55.34474	Ans 56.12102
Ans 57.B	Ans 58.A
Ans 59.A	Ans 60.D
Ans 61.B	Ans 62.B
Ans 63.C	Ans 64.D
Ans 65.A	

Solutions to difficult problems

Ans 5. D

Handle is part of the string in sentential form that will be reduced to non-terminal i.e left hand side of a production

In the above derivation, sentential form Aabw is reduced to ABw so has to be a production with B --> ab and that is the handle at this point of derivation.

Ans 8. No grammar with empty production can be LR(0). But empty rules are allowed in regular grammar as the only condition for a grammar to be regular is to be either left linear or right linear.

Ans 11. Answer is D.
I : Definition of viable prefix -

> *Viable prefixes are the prefixes of right sentential forms that do not extend beyond the end of its handle.*

i.e. a viable prefix either has no handle or just one possible handle on the extreme RIGHT which can be reduced.

II : Definition of viable prefix -

> *viable prefixes are the set of prefixes of right sentential forms that can appear on the stack of a shift reduce parser*

III : The set of all viable prefixes of the right sentential forms of a grammar is a REGULAR LANGUAGE. i.e., viable prefixes can be recognized by using a FINITE AUTOMATA.

So I, II & III are true.

Ans 13. LL(0) grammars have no lookahead. And since they follow Leftmost derivation, at each step the parser has to derive the string by seeing 0 symbols. => Parser sees nothing.

So whenever we have multiple choices for any Variable in the grammar, LL(0) fails.

Hence, LL(0) parser can only parse grammars that strictly generate one single string.

Option D

Ans 14.

- Parsing is the process of constructing a parse tree for a sentence generated by a given grammar
- If there are no restrictions on the language and the form of grammar used, parsers for context-free languages require $O(n^3)$ time (n being the length of the string parsed)
  - Cocke-Younger-Kasami's algorithm
  - Earley's algorithm
- Subsets of context-free languages typically require $O(n)$ time
  - Predictive parsing using $LL(1)$ grammars (top-down parsing method)
  - Shift-Reduce parsing using $LR(1)$ grammars (bottom-up parsing method)

Answer17 : C

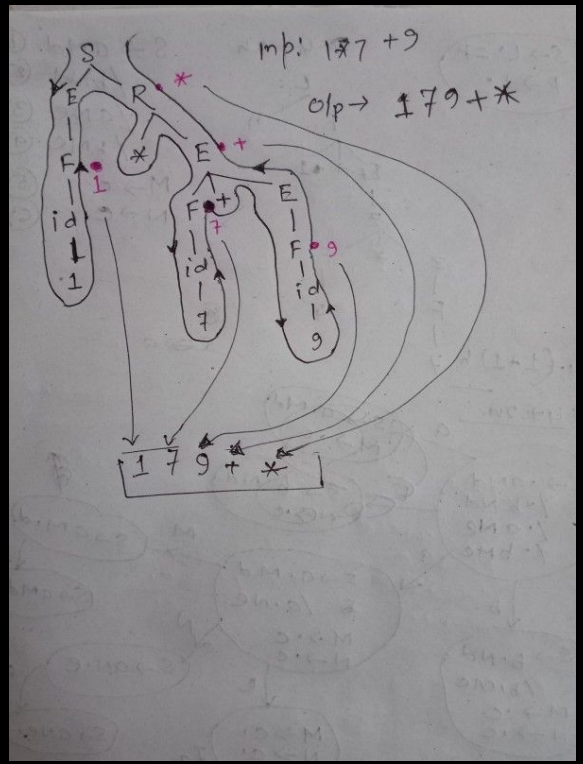
$\epsilon-$free LL(1) $\subset$ SLR(1) $\subset$ LR(1)

because every $\epsilon-$free LL(1) are SLR(1) and every SLR(1) are LR(1).

Answer 20.Observe the parse tree (a), it grows toward left, while (b) grows to-wards right.
The left one (a) represent the following sequence of operations:

list --> list − digit
--> list − digit − digit
--> digit − digit − digit
--> 6 − digit − digit
--> 6 − 4 − digit
--> 6 − 4 – 3
This shows that the operators are evaluated in left associative manner that is from left to right. That (a) represent.The second parse tree evaluates in right associative manner that is from right to left.The grammar is obviously ambiguous since it produces two different parse trees for the same expression.
The reality is:
1. The parse tree (a) represent left associative evaluation while that of (b) right associative evaluation.
2. The grammar is ambiguous.
So both the given statements are false.

Answer 30.



Answer 35. Number of states all three parser will be same but there might be increase coming to CLR(1)

A is Correct.

Answer 33. 1. $S \rightarrow aS|Sa|\in$

2. $E \rightarrow E+E|E*E|id$

3. $A \rightarrow AA|(A)|a$

4. $S \rightarrow SS|AB$, $A \rightarrow Aa|a$, $B \rightarrow Bb|b$

Let's look at first grammar, you can generate "aa" by two parse trees so its ambiguous. If you are in hurry, you can choose this option because other grammar's are ambiguous too. Otherwise, check for all options...

Option B says Unambiguous, which can't be possible because grammar 1 is ambiguous.

Option C says regular, but we can generate

$a^n b^n$

So can't be regular grammar.

Choice D is also wrong because, to be an Inherently Ambiguous language we need all grammar generated by this Lang. To be ambiguous, which is not take grammar 3 can we generate a by two parse trees?

Answer 50.Option D as SDT is Syntax based, scheme should be independent of implementation and hence option D.

Answer 51.Because of attribute F taking value from its Right attribute (i.e T1) it is not L attributed SDD, and of course it is not S attributed SDD so NONE.

Answer 53.only Option c can be answer since by looking at parse tree we see 1st * will performed same as in grammar in which * has more precedence than +,-. But - is right associative so actual parse tree look like this. so Nothing is correct answer

Answer 56.1100+10*1100+1*1*2
which is equal to 12102 and is the correct answer.

Answer 57
The most appropriate answer is (A) [It can also be B, if its possible to reduce into unambiguous].

Let me explain...

'Dangling Else' problem is like deciding with WHICH 'if' some 'else' can go with. Say for following...

ex1) X --> S | b

S --> if E X else a | if E X

ex2) S --> if E S else a | if E S | b

Both of the above cases suffers through Dangling Else problems. Why? Both can derive, " if E if E b else a" . (E stand for some expression in Code.) Now, with which 'if' should we configure our 'else'. Compilers (LALR(1) in our day to day life) uses the approach of choosing 'nearest if' . But this is just ONE of the way to resolve this so called ambiguity. Some other methods are...

1) Associate dangling else with nearest if by using some matching/unmatching statemnets

2) By using something like 'endif' for marking the end of of conditional structure.

3) By using opening ({) and closing (}) curly braces around statements.

4) By using different precedence rules to associate the dangling else with nearest if. (By using lower precedence for THEN and higher precedence for ELSE. Try it.) **And by doing this and other techniques you are HANDLING ambiguity in grammer.**

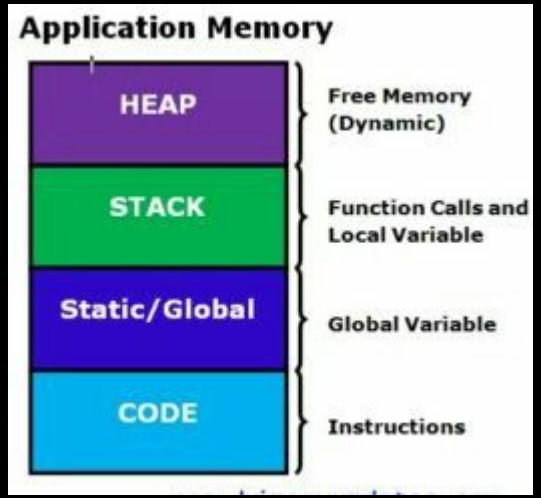
5) By insisting space indentation

Answer 59. A linker is a computer program that takes one or more object files generated by a compiler and combines them into one, executable program.

Computer programs are usually made up of multiple modules that span separate object files, each being a compiled computer program. The program as a whole refers to these separately compiled object files using symbols. The linker combines these separate files into a single, unified program, resolving the symbolic references as it goes along.

so only statement 1 is true

Answer 64.




ept


ept


ept


ept