ACE

Engineering Academy

Since 1995

Hyderabad | Delhi | Pune | Bhubaneswar | Bengaluru | Lucknow | Chennai | Vijayawada | Visakhapatnam | Tirupati | Kukatpally | Kolkata | Ahmedabad

Compiler Design

(Classroom Practice Questions)

2. Lexical Analysis

01. Comments are normally deleted from the source program during
(a) Lexical analysis
(b) Syntax analysis
(c) Semantic analysis
(d) Code generation

02. The macro–expansion or expansion of preprocessor statements take place during
(a) Lexical analysis
(b) Syntax Analysis
(c) Semantic Analysis
(d) None

03. Keywords are recognized in a compiler during
(a) Lexical analysis
(b) Syntax analysis
(c) Semantic analysis
(d) Code optimization

04. Type checking is done during
(a) Lexical analysis
(b) Syntax analysis
(c) Semantic analysis
(d) Code generation

05. Compiler can diagnose the following errors
(a) Grammatical errors
(b) Logical errors
(c) Runtime errors
(d) All of the above

06. A compiler that may run on one machine and produces target code for another machine is called
(a) Bootstrapping (b) Direct compiler
(c) Indirect compiler (d) Cross compiler

07. The object code which is obtained from Assembler is
(a) Assembly language code
(b) Relocatable object code
(c) Executable object code
(d) None

08. Choose the correct statement
(a) A compiler must compulsorily keep lexical analysis separate from the syntax analysis.
(b) The syntax analysis phase can be expanded to do lexical analysis also
(c) Macro expansion is done during syntax analysis.
(d) Macro expansion is done during semantic analysis

09. Choose the correct statement

(a) The symbol table is constructed in the lexical analysis phase

(b) The symbol table is constructed during the syntax analysis phase.

(c) The symbol table is not needed for semantic analysis

(d) The symbol table is not needed for code generation

10. Consider the following error recovery operations in lexical analysis.

1. Delete one character from the remaining input
2. Insert a missing character into the remaining input
3. Replace a character by another character
4. Transpose two adjacent characters
5. If parenthesis mismatch deletes the offending parenthesis

(a) 1, 2, 3, 4 are valid at the lexical analysis phase

(b) 4, 5 cannot be done at the lexical analysis phase

(c) 1, 2 are not valid at the lexical analysis phase

(d) Only 5 is not valid at the lexical analysis phase

11. for (i = 0; i < 10; i + +)

printf (“i = %d”, i);

The number of tokens in the above program segment is _______.

12. printf (“Total = % d”, sum);

The number of tokens in the above ‘C’ statement is _______.

13. Find number of tokens in the following program?

if (x >= y) {x = x + y;} else {x = x – y;};

(a) 23

(b) 24

(c) 25

(d) None

14. Which of the following is not a token of ‘C’ program?

(a) 1.02e+2

(b) MAX

(c) 123.33

(d) None

15. Match the following according to input (from the left column) to the compiler phase (in the right column) that processes it:

P: Syntax tree	i. Code generator
Q: Character stream	ii. Syntax analyzer
R: Intermediate representation	iii. Semantic analyzer
S: Token stream	iv. Lexical analyzer

(a) P → (ii), Q → (iii), R → (iv), S → (i)

(b) P → (ii), Q → (i), R → (iii), S → (iv)

(c) P → (iii), Q → (iv), R → (i), S → (ii)

(d) P → (i), Q → (iv), R → (ii), S → (iii)

16. for (i = 0; i < 10; i++)

printf (“welcome”);

The number of tokens in the above ‘C’ program code is_____.

17. Match the following:
P: Semantic analyzer
Q: Parser
R: Code optimizer
S: Target code generation

1. back end
2. Postfix notation
3. Eliminating redundant code
4. static type checking
5. syntax errors
6. correlating error messages

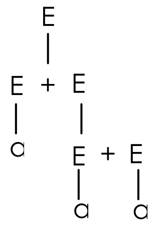
(a) P - 1, Q - 5, R - 3, S - 4
(b) P - 4, Q - 5, R - 3, S - 1
(c) P - 1, Q - 3, R - 6, S - 2
(d) P - 5, Q - 2, R - 3, S - 6

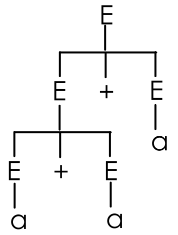
Key for Practice Questions

01. (a)	02. (a)	03. (a)	04. (c)	05. (a)
06. (d)	07. (b)	08. (b)	09. (a)	10. a& d
13. (b)	14. (d)	15. (c)	16. **18**	17. (b)

3. Parsing Techniques

01. Let + be a left associative operator then the syntax tree for a + a + a is in the grammar
E → E + E/a

(a)


(b)


(c) Either (a) or (b) (d) Neither(a) nor(b)

02. Consider the grammar given below
S → SS | a | ε
The sentence "aaa" has
(a) A unique derivation tree in the grammar
(b) Two leftmost derivations in the grammar
(c) Ten rightmost derivations in the grammar
(d) An infinite number of leftmost and rightmost derivations

03. A grammar that is both left and right recursive for a non-terminal is
(a) Ambiguous
(b) Unambiguous
(c) Information is not sufficient
(d) None

04. A grammar has the following productions:
S → a Sb/b/bSa
Which of the following sentences are in the language that is generated by this grammar

(a) aaaaabb (b) aabbaabb
(c) bbbaabbaa (d) abababb

05. Consider the following grammar

$E \rightarrow E + E\ /\ (E * E)/id$

The number of parse trees are possible for an input string w=id+id+id is ______.

06. How many parse trees exists for an input string w = abab in the CFG

$S \rightarrow aSbS/bSaS/\varepsilon$

(a) 4 (b) 3 (c) 2 (d) 5

07. Consider the left recursive grammar

$A \rightarrow Sa/b$

$S \rightarrow Sc/Ad$

Which of the following grammar is equivalent grammar with left recursion eliminated?

(a) $A \rightarrow Sa/b$
$S \rightarrow bdS^1$
$S \rightarrow cS^1/adS^1/\varepsilon$

(b) $A \rightarrow Sa/b$
$S \rightarrow bdS^1$
S^1cS^1/adS^1

(c) $S \rightarrow bdS'$
$S^1 \rightarrow cS^1/adS^1/\varepsilon$
$A \rightarrow ScaA^1/bA^1$
$A^1 \rightarrow daA^1/\varepsilon$

(d) None

08. Consider the following expression grammar G:

$E \rightarrow E - T \mid T$

$T \rightarrow T + F \mid F$

$F \rightarrow (E) \mid id$

Which of the following grammars is not left recursive, but is equivalent to G?

(a) $E \rightarrow E - T \mid T$
$T \rightarrow T + F \mid F$
$F \rightarrow (E) \mid id$

(b) $E \rightarrow TE'$
$E' \rightarrow -TE' \mid \varepsilon$
$T \rightarrow T + F \mid F$
$F \rightarrow (E)\ id$

(c) $E \rightarrow TX$
$X \rightarrow -TX \mid \varepsilon$
$T \rightarrow FY$
$Y \rightarrow +FY \mid \varepsilon$
$F \rightarrow (E) \mid id$

(d) $E \rightarrow TX \mid (TX)$
$X \rightarrow -TX + TX \mid \varepsilon$
$T \rightarrow id$

09. What is the precedence and associatively of operators in

$E \rightarrow E \uparrow T \mid T$

$T \rightarrow F + T \mid F$

$F \rightarrow i$

(a) ↑ has high precedence than +
↑ is right associative
+ is left associative

(b) ↑ has high precedence than +
↑ left associative
+ right associative

(c) ↑ has less precedence than +
↑ is right associative
+ is left associative

(d) ↑ has less precedence than +
↑ is left associative

10. Consider the following grammar

$E \rightarrow E \uparrow T / T$

$T \rightarrow F + T / F$

$F \rightarrow id$

(a) ↑ has higher precedence than +

(b) + has lower precedence than ↑

(c) ↑ has lower precedence than +

(d) id has lower precedence than ↑ and +

11. Given the following grammar

$E \rightarrow E*F/F+E/F$

$F \rightarrow F-F/id$

Which of the following is true?

(a) * > + (b) – > *

(c) + = – (d) + > *

12. Consider the following left-associative operators, in decreasing order of precedence –, \$ (exponent), *. The result of the expression 3 – 2 * 4 \$ 2 * 3 \$ 2 is _______.

13. Evaluate the following expression considering the given rules:

    3 – 2 * 4 $ 1 * 2 $ 3

    Where $ mean exponentiation

    (a) (–) highest followed by(*)and then followed by($)and all are left associative
    (b) (–) highest followed by(*)and then followed by($)and all are right associative
    (c) ($) highest followed by(–)and then followed by(*)and all are left associative

    (a) a<b<c (b) c<a<b
    (c) a<c<b (d) b<c<a

14. Choose the correct statement?
    (a) a bottom up parsing technique simulates a rightmost derivation
    (b) a top down parsing technique simulates the reverse of a leftmost derivation
    (c) a bottom up parsing technique building the derivation tree in bottom up and simulates a rightmost derivation in reverse
    (d) a top down parsing technique simulates building the derivation tree in top down and simulates a leftmost derivation in reverse

15. Which of the following is true?
    (a) Brute force technique is a SR parser.
    (b) Operator precedence parser is a Top down parser
    (c) Recursive descent parser is a Bottom up parser
    (d) Operator precedence parser is a SR parser

16. What is FIRST (S) in

    S → AC / Ca / Bb,
    A → d
    B → e / ε
    C → f / ε

    (a) {a, d, e, f} (b) {a, e, f, ε}
    (c) {a, b, d, e, f} (d) {a, b, d, e, f, ε}

17. Consider the following grammar for expressions:

    <expression>→atom<additional> | (<expression>)
    <additional>→+<expression> | *<expression> | ε

    The follow set for <additional> is

    (a) { atom, (} (b) { (, $}
    (c) {+, *, $} (d) { $,) }

18. S → ABC
    A → aA | c
    B → b | ε
    C → c

    FIRST (A) ∩ FOLLOW (A) is

    (a) {a} (b) {b}
    (c) {c} (d) None

19. FOLLOW (B) in the following grammar

    A → BCD
    B → w | Bx
    C→ yCz | m
    D → DB | a

    (a) {x, y, m, $} (b) {x, y, w, $}
    (c) {x, w, m, $} (d) {x, y, m, w, $}

20. S → [SX] | a
    X → ε | +SY | Yb
    Y → ε | –SXc

    Then FOLLOW (S) is

    (a) {$, +, –,], c, b} (b) {$, –,], c, b}
    (c) {$, +, –,], b} (d) {$,], c, b}

21. In the grammar S → TA,
    A → +TA/ε
    T → a/ε

    Follow (T) ∩ First (S) is

    (a) {+,ε} (b) {+, $}
    (c) {+} (d) {a,+,ε}

22. Consider the grammar below :

S → A B A
A → B c / dA / ε
B → eA

Find FOLLOW(A)

(a) { d, e, $} (b) {c, d, $}
(c) {c, d, e} (d) {c, d, e, $}

23. Consider the following grammar

S → AaC/Bd
A → BC
B → bB/C
C → accS

For which non terminals, its follow set contains terminal 'a'.

(a) {C} (b) {A}
(c) {A,B,C} (d) {A,B,C,S}

24. Consider the following grammar

<expression> → <factor> <rest>
<rest> → * <expression> | ε
<factor> → identifier

In the predictive parsing table M of the grammar the entries for M [<expression>, identifier] and M [<rest>,*] respectively are

(a) [<expression> → <factor> <rest>] and [<rest> → ε]
(b) [<expression>→ <factor> <rest>] and []
(c) [<expression> → <factor> <rest>] and [<rest> →* <expression>]
(d) [<factor> → identifier] and [<rest>→ ε]

25. Consider the grammar

S → a | ab | abc

Choose the correct statement from the following:

(a) The grammar is LL(1)
(b) The grammar is LL(2)
(c) The grammar is LL(3)
(d) None of the above

26. Consider the grammar

S → if expr then
| if expr then stmt
| if expr then stmt else
| if expr then stmt else stmt

The grammar is

(a) LL(1) (b) LL(4)
(c) LL(5) (d) LL(6)

27. Consider the grammar, G1

E → E +E|E*E|(E)|id

Consider the grammar G2,

E → E+T|T
T → T*F|F
F → (E)| id

Choose the incorrect statement?

(a) G2 generates the same language as G1
(b) G2 is unambiguous but not LL(1) as it is left recursive
(c) G2 gives a higher precedence to + over *
(d) None

28. Choose the true statement?

(a) If a grammar is left recursive it cannot be LL(1)
(b) If a grammar is right recursive it cannot be LL(1)
(c) An ambiguous grammar can sometimes be LL(1)
(d) If a grammar is not context free then it sometimes can be LL(1)

29. S → Aa

A → b/ε

If we construct a predictive parse table for the above grammar, the production A → ε is added in 'A' row and ________.

(a) '$' column (b) 'ε' column
(c) 'a' column (d) 'b' column

30. $S \to aSbS/bSaS/\varepsilon$

In the predictive parse table M of the above grammar M [S,a] = _______

(a) $S \to aSbS$ (b) $S \to bSaS$

(c) $S \to \varepsilon$ (d) $S \to aSbS$, $S \to \varepsilon$

31. $S \to Aa/Bb$

$A \to b$

$B \to b$ is

(a) LL(1) (b) LL(2)

(c) LR(0) (d) None

32. Consider a calculator modeled by the grammar

<accumulated_sum>→<accumulated_sum>+ number
| <accumulated_sum>* number
| number

For an input "number + number * number", the handles in the reverse of a rightmost derivation are

(a) number, <accumulated_sum> + number, <accumulated_sum>+ number * number

(b) number, <accumulated_sum> + number, <accumulated_sum>+ <accumulated_sum> * number

(c) number, number + number, number + number * number

(d) number, <accumulated_sum> + number, <accumulated_sum>* number

33. Which of the following is **not** an operator grammar?

I. $S \to AB$ II. $S \to AaB$

III. $S \to a$ IV. $S \to \varepsilon$

(a) only I (b) I and II

(c) I, III and IV (d) None

34. Which of the following is **not** an operator grammar

(a) $S \to AaB$
$A \to aA/b$
$B \to bB/a$

(b) $S \to AaB$
$A \to a/b$

(c) $S \to AaB$
$A \to aA/b$
$B \to bB/\varepsilon$

(d) None

35. What is equivalent operator grammar for the following grammar?

$S \to AB$, $A \to c/d$, $B \to aAB/d$

(a) $S \to Aa/Ab$
$A \to c/d$

(b) $S \to AaAB/Ad$
$A \to c/d$
$B \to aAB/d$

(c) $S \to AaS/Ab$
$A \to c/d$
$B \to aS/b$

(d) None

36. Consider the grammar given below

$E \to E+E \mid E*E \mid E–E \mid E/E \mid E\wedge E \mid (E) \mid id$

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 statement in the operator precedence table constructed for the grammar the relations for the ordered pairs (^,^),(- ,-) , (+,+), (*,*) are

(a) all < (b) all >

(c) <,>,=,< (d) <,>,>,>

37. $S \to AaBb$

$A \to Bc/d$

$B \to e$

In the operator grammar above lead (S) = _______

(a) {d} (b) {e,d}

(c) {c,d,e} (d) {a,c,d,e}

38. For the grammar

$E \rightarrow E+T/T$

$T \rightarrow T*F/F$

$F \rightarrow (E)/id$

Trailing symbols of E are

(a) *,+, (,id (b) *, +,), id

(c) *, (,id (d) *,), id

39. Consider the following operator grammar

$E \rightarrow E+T/T$

$T \rightarrow F \uparrow T/F$

$F \rightarrow id$

(a) $\uparrow > \uparrow$ (b) $\uparrow > +$

(c) $+ > \uparrow$ (d) $+ < +$

40. $S \rightarrow aSb/Ac$

$A \rightarrow d$

Which of the following operator precedence relations are incorrect?

(a) $a = b$ (b) $a < a$

(c) $d > c$ (d) $c < b$

41. Choose the correct statement

(a) A LL(1) grammar is compulsory SLR(1)

(b) An operator grammar can be ambiguous but a parser may be possible

(c) An operator grammar is recursive LR(1) always

(d) An grammar may be LALR(1) but not LR(1)

42. Choose the correct statement

(a) A grammar that is LR(0) may not be LL(1)

(b) A grammar that is LR(0) is necessarily LL(1)

(c) A SLR(1) grammar is always LR(0)

(d) None of the above

43. Choose the false statement

(a) An ambiguous grammar can never be LL(1)

(b) An ambiguous grammar can never be LR(0)

(c) An ambiguous grammar can never be SLR(1)

(d) An ambiguous grammar cannot have a shift reduce parser

44. Choose the false statement

(a) There exists a grammar that is LR(0) but not LL(1)

(b) There exists a grammar that is LL(1) but not LR(0)

(c) There exists a grammar that is LL(1) but not SLR(1)

(d) There exists an ambiguous grammar that is either LL(1) or LR(0)

45. Choose the false statement

(a) Top down parsing algorithms simulate a leftmost derivation

(b) Bottom up parsing algorithms simulate the reverse of a rightmost derivation

(c) An LL(1) parser is a top down parser

(d) An LR(1) parser may sometimes exist for some ambiguous grammars

46. Choose the correct statement

(a) A grammar that is LL(1) is necessarily LR(0)

(b) A grammar that is LL(2) is necessarily LR(1)

(c) A grammar that is LL(1) is necessarily LL(2)

(d) None of the above

47. Choose the false statement

(a) There exists a grammar that is SLR(1) but not LR(0)

(b) There exists a grammar that is LALR(1) but not SLR(1)

(c) There exists a grammar that is LR(1) but not LALR(1)

(d) There doesn't exist a grammar that is LL(2) and LR(0)

48. Which of the following is incorrect ?

(a) every LR(0) is CLR(1)

(b) every LL(1) is LR(0)

(c) every LR(1) is CFG

(d) every SLR(1) is CLR(1)

49. Consider the grammar shown below

<life> → <session> <session>

<session> → play <session> | rest

In the LR(0) machine for the grammar consider the items

<life> → <session>.<session>,

<session>→play.<session>,

<session>→play <session>.

Choose the correct statement?

(a) No two of the items occur in the same state

(b) All three of the items occur in same state

(c) At least two of them occur in multiple states

(d) None of the above

50. Consider the following grammar

<expression> → <factor> <rest>

<rest> → * <expression> | ε

<factor> → identifier

In the LR(0) machine the no two of the following LR(0) items appear in the same state

(i) <expression> →. <factor> <rest>

(ii) <expression> →. * <rest>

(iii) <factor> → . identifier

(a) (i) and (ii) (b) (i) and (iii)

(c) (ii) and (iii) (d) None of the above

51. S → SB / A

A → a

B → b

Find the closure set of S^1 →. S for the LR(0) machine is _____.

(a) S^1 → . S
S → . SB
S → .A
A → .a
B → . b

(b) S^1 → . S
S → . SB
S → . A
A → .a

(c) S^1 → .S
S → .SB
S → .A

(d) None

52. S → AA

A →aA/b

In the LR(0) machine for the above grammar, the number of states are______.

53. The grammar E → E + T/T; T → a is

(a) not LR(0)

(b) not SLR(1) but CLR (1)

(c) CLR(1)

(d) not LALR(1)

54. The LR(0) parser for the grammar

E → E+T | T

T → T*F | F

F → id

Contains _________ number of conflicts.

55. Consider the grammar

<statement> → a <statement> | b

<statement> b | c

Choose the false statement

(a) Not LR(0) (b) LALR(1)
(c) SLR(1) (d) LR(1)

56. Consider the grammar

$$A \rightarrow A + A \mid i$$

Choose the proper statement regarding the property of the above grammar.

(a) The grammar is LR(1) but not LALR(1)
(b) The grammar is LALR(1) but not LL(1)
(c) The grammar is SLR(1) but not LR(0)
(d) None of the above

57. Find the number of conflicting entries in SLR(1) parse table of the following grammar is _______.

S → AaAb
S → BbBa
A → d
B → d

58. Consider the grammar

S^1→S, S→ aAd | bBd | aBe | bAe, A → c, B → c

The grammar is

(a) LR(1) but not LALR(1)
(b) LALR(1) but not LL(1)
(c) SLR(1) but not LR(0)
(d) None of the above

59. Consider the grammar

S → Aa | bAc | Bc | bBa, A → d, B→ d

(a) The grammar is LR(1) but not LALR(1)
(b) LALR(1) but not LL(1)
(c) SLR(1) but not LR(0)
(d) None of the above

60. What is the conflict in LR(1) construction for the grammar

A → aA | Ab | d

(a) RR conflict
(b) SR conflict
(c) Both RR and SR conflicts
(d) No conflict

61. Consider the grammar

S → Aa | bAc | dc | bda
A → d

(a) The grammar is LL(1)
(b) The grammar is LR(0)
(c) The grammar is SLR(1)
(d) The grammar is LALR(1)

62. What is the closure of S→. A, $ for the grammar S→A, A→AB/ε, B→b

(a) S→. A, $/b
A→. AB $/b
A→. , $/b

(b) S →. A, $
A →.AB, $
A →. , $
B →. b, $

(c) S →.A, $
A →. AB, $/b
A →. , $/b

(d) None

63. Consider the grammar given below

S → aS | Sa | a

Find number of conflicting states in DFA with LR(1) items

(a) 0 (b) 3
(c) 4 (d) 2

64. $S \rightarrow L = R/R$
$L \rightarrow *R/id$
$R \rightarrow L$

In the LR (1) items of the above grammar, the closure set of items in $S^1 \rightarrow .S, \$$ contains

(a) $L \rightarrow .*R,=.$ (b) $L \rightarrow .*R,\$$
(c) $L \rightarrow .*R, = /\$$ (d) $L \rightarrow .id,=$

65. Consider the grammar :
$S \rightarrow a A b$
$A \rightarrow a A b \mid a$
then which of the following is true ?
(a) the grammar is LL(1)
(b) the grammar is LR(0)
(c) the grammar is LR(1)
(d) the grammar is LR(0) but not LR(1)

66. The grammar
$S \rightarrow (S) / \varepsilon$ is
(a) LL (1)
(b) LR (0)
(c) CLR (1)
(d) both LL(1) and LR (1)

67. The LALAR (1) parsers for a grammar G can have SR conflict if and only if
(a) the SLR (1) Parser for G has SR conflict
(b) the LR (1) Parser for G has SR conflicts
(c) the LR (0) Parser for G has SR-conflicts
(d) the LALR (1) Parser for G has RR Conflicts.

68. Consider the grammar

<statement> :: =[<statement>] | assignment | null

Let the Number of states in the SLR(1), LR(1) and LALR(1) parsers for the grammar be N1, N3 and N2 respectively. The following relationship holds good
(a) N1<N2<N3 (b) N1 = N2 < N3
(c) N1 = N2 > N3 (d) N1>= N3 >= N2

69. Let there are '10' states for a grammar which is SLR(1), then the number of states in LALR(1) parser is ______.

70. The parser generator tool YACC uses____ parsing table
(a) LR(0) (b) SLR(1)
(c) LALR(1) (d) LR(1)

Key for Practice Questions

01.(b)	02.(d)	03.(a)	04.(d)	05. **2**
06.(c)	07.(d)	08.(c)	09.(d)	10.(c)
11.(b)	12.**144**	13.(b)	14.(c)	15.(d)
16.(c)	17.(d)	18.(c)	19.(d)	20.(a)
21.(c)	22. (d)	23.(d)	24.(c)	25.(c)
26.(c)	27.(c)	28.(a)	29.(c)	30.(d)
31.(b)	32.(d)	33.(d)	34.(c)	35.(c)
36.(d)	37.(d)	38.(b)	39.(b)	40.(d)
41.(b)	42.(a)	43.(d)	44.(d)	45.(d)
46.(c)	47.(d)	48.(b)	49.(a)	50.(b)
51.(b)	52.**7**	53.(c)	54.**2**	55.(a)
56.(d)	57. **2**	58.(a)	59.(a)	60.(b)
61.(d)	62.(c)	63.(d)	64.(c)	65.(c)
66.(d)	67.(b)	68.(b)	69.**10**	70.(c)

4. Syntax Directed Translation Scheme

01. Syntax Directed Translation is part of
    (a) Lexical Analysis
    (b) Syntax Analysis
    (c) Semantic Analysis
    (d) Intermediate code generation

02. The following SDT is

$E \rightarrow E_1 + T$ $\{E.\ val = E_1.\ val + T.\ val\}$
$E \rightarrow T$ $\{E.\ val = T.\ val\}$
$T \rightarrow id$ $\{T.\ val = id\}$

(a) S- attributed (b) L- attributed
(c) Both (a) and (b) (d) None

03. The following SDT checks

$S \rightarrow aSb$ $\{S.\ count = S.\ count + 2\}$
$S \rightarrow bSa$ $\{S.\ count = S.\ count + 2\}$
$S \rightarrow \varepsilon$ $\{S.\ count = 0\}$

(a) Equal number of a's and b's.
(b) Number of a's or number of b's in given string
(c) Number of a's and b's in a given string
(d) None

04. Consider the following Syntax Direct Translation scheme

$E \rightarrow E + T$ { E.value = '+' | | E.value | | T.value}
$E \rightarrow E - T$ { E.value = '–' | | E.value | | T.value}
$E \rightarrow T$ { E.value = T.value}
$T \rightarrow id$ {T.value = id}
(Assume | | is string concatenation)

The above SDTS specifies ________.
(a) Expression Evaluation
(b) Infix to Postfix conversion
(c) Infix to Prefix conversion
(d) None of the Above

05. Consider the Syntax Directed Definition below

$S \rightarrow S_1 S_2 c$ $\{S.\ t = S_1.\ t - S_2.\ t\}$
$S \rightarrow a$ $\{S.\ t = 2\}$
$S \rightarrow b$ $\{S.\ t = 6\}$

If the Top down parser carries out the translations for an input string "babcc", output is _____.
(a) 0 (b) 6
(c) 10 (d) 5

06. Consider the SDT given below.

$S \rightarrow S_1 S_2\ c$ $\{S.val = S_1.\ val * S_2\ .val - 4\}$
$S \rightarrow a$ $\{S.val = 6\}$
$S \rightarrow b$ $\{S.\ val = 2\}$

If the SDT gives the input string abc, then the output is
(a) 5 (b) 8
(c) 0 (d) 2

07. Consider the following grammar and associated semantic actions.

$G \rightarrow F$ {G.p = F.p}
$F \rightarrow F1 \wedge F2$ { F.p = And(F1.p, F2.p)}
$F \rightarrow F1 \vee F2$ {F.p = Or(F1.p, F2.p)}
$F \rightarrow \neg F1$ { F.p = Neg(F1.p)}
$F \rightarrow F1 \Rightarrow F2$ { F.p = Or(Not(F1.p), F2.p)}
$F \rightarrow (F1)$ {F.p = F1.p}
$F \rightarrow id$ { F.p = id.lexeme}

Give the value of the attributes of G after parsing $\neg(A \wedge (A \Rightarrow B))$.
(a) Neg(And(A,Or(Not(B),A))).
(b) Neg(And(A,Or((A),notB))).
(c) Neg(And(A,Or(Not(A),B))).
(d) Neg(And(A,Or((A),B))).

08. Consider the SDTS (using synthesized attributes)

$E_1 \rightarrow E_2 * E_3$ [E_1.sem = E_2.sem × E_3.sem]

$E_4 \rightarrow E_5 + E_6$ [E_4.sem = E_5.sem + E_6.sem]

E → id [E..sem = 1]

The output for the input a+b+c+d+e is

(a) 3 (b) 4 (c) 5 (d) 6

09. Consider the SDTS (using synthesized attributes)

S → aS out ("1")

S → a out ("2")

The output for the input aaaa is

(a) "1111" (b) "2222"
(c) "2111" (d) "1112"

10. Consider the SDTS (using synthesized attributes)

S → aB {out "cat"}

S → bA {out "dog"}

B → aBB {out "pig"}

A → bAA {out "cow"}

B → b {out "deer"}

A → a {out "lion"}

The output for aabb is

(a) deer deer pig cat (b) cat dog pig lion
(c) cat cow deer lion (d) cat deer dog pig

11. Consider the following Syntax Directed Translation Scheme (SDTS), with non-terminals {S, A} and terminals {a,b}.

S → aA {print 1}

S → a {print 2}

A → Sb {print 3}

Using the above SDTS, the output printed by a bottom-up parser, for the input **aab** is:

(a) 1 3 2 (b) 2 2 3
(c) 2 3 1 (d) syntax error

12. Here is a postfix SDT:

S → aS {print "x"}

S → bS {print "y"}

S → a {print "z"}

S → b {print "z"}

Suppose we execute this SDT in connection with a bottom-up parser. What will be printed in response to the input " ababb"

(a) zxyxy (b) zyxyy
(c) zyxyx (d) zyyxx

13. The output for the SDT

A → A + A {print 1}

A → a { print 2}

For a + a + a is

(a) 22121 (b) 21221
(c) 21212 (d) 12122

14. Consider the SDTS given below

E1 → E printf("$")

E → E + E printf("+")

E → E * E printf("*")

E → (E)

E → id printf(id.value)

If we assume a bottom up shift reduce parser, the output for the input string a+b* c is

(a) abc * +
(b) ab + c*
(c) abc * + or ab+c* depending on the compiler
(d) There will be an error and no parsing can take place.

15. Consider the following translation scheme

S → AB

B → * A {print *} B / ε

A → C + A {print +} / C

C → S / id {print id}

Here id represents an integer. For input '5 * 6 + 7' this translation prints ______.

(a) 3 7 (b) 5 * 6 + 7

(c) 5 6 * 7 + (d) 5 6 7 + *

16. Consider the SDT given below

E → E ↑ E {print ↑}
E → E * E {print *}
E → id {print id}

If the shift reduce parser constructs a parse tree for the input sentence a * b ↑ c, the above translation prints.

(a) a b * c ↑ (b) a b c ↑ *

(c) Both (d) None

Key for Practice Questions

01. (c)	02. (c)	03. (c)	04. (c)	05. (c)
06. (b)	07. (c)	08. (c)	09. (c)	10. (a)
11. (c)	12. (c)	13. (a)	14. (c)	15. (d)
16. (a)				

5. Intermediate Code Generation

01. One of the purpose of using intermediate code in compilers is to

(a) Make parsing and semantic analysis simpler

(b) Improve error recovery and error reporting

(c) Increase the chances of reusing the machine independent code optimizes in other compilers

(d) Improve the register allocation

02. The following are **not** standard intermediate forms of the source program

(a) Quadruples

(b) Triples

(c) Postfix

(d) Machine language code

03. Which of the following is **not** a Three Address Code(TAC)

(a) return 0 (b) if a > b goto c

(c) a[i] = b; (d) none

04. Which of the following is not a form of Three Address Code(TAC) implementation?

(a) Quadruples (b) Triples

(c) Indirect triples (d) Syntax tree

05. The quadruple form of a+b*c–d is

(a) (1) (+,a, b, T_1)
(2) (*, T_1, c, T_2)
(3) (–, T_2, d, T_3)

(b) (1) (*, b, c, T_1)
(2) (+, a, T_1, T_2)
(3) (–, T_2, d, T_3)

(c) (1) (+, a, b, T_1)
(2) (–, c, d, T_2)
(3) (*, T_1, T_2, T_3)

(d) None

06. Which of the following is in the quadruple form for the expression a or b and c or d (Assume 'and' has higher precedence than 'or'). Assume each quadruple is numbered from 1.

(a) (1) (or, a, b, T_1) (b) (2) (and, b, c, T_1)

(c) (2) (or, a, T_1, T_2) (d) None

07. The triples of a*–(b + c) is

(a) 1. (+, b, c)
2. (NEG, (1),)
3. (*, a, (2))

(b) 1. (+ , b, c)
2. (*, a, (1))
3. (NEG, (2), –)

(c) 1. (*, a, b)
2. (+, b, c)
3. (NEG, (1), (2))

(d) 1. (+, b, c, T1)
2 . (*, a, T1, T2)
3. (NEG, T2, T3, -)

08. Consider the following code segment.

```
x = u – t;
y = x * v;
x = y + w;
y = t – z;
y = x * y;
```

The minimum number of total variables required to convert the above code segment to static single assignment form is ______.

09. Consider the following intermediate program in three address code

```
p = a – b
q = p * c
p = u * v
q = p + q
```

Which one of the following corresponds to a static single assignment form of the above code?

(a) $p_1 = a - b$
$q_1 = p_1 * c$
$p_1 = u * v$
$q_1 = p_1 + q_1$

(b) $p_3 = a - b$
$q_4 = p_3 * c$
$p_4 = u * v$
$q_5 = p_4 + q_4$

(c) $p_1 = a - b$
$q_1 = p_2 * c$
$p_3 = u * v$
$q_2 = p_4 + q_3$

(d) $p_1 = a - b$
$q_1 = p * c$
$p_2 = u * v$
$q_2 = p + q$

10. Choose the incorrect statement

(a) DAG representation of a basic block enables elimination of common subexpression.

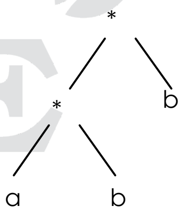
(b) DAG representation of a basic block enables dead code elimination.

(c) DAG representation allows optimal register usage by specifying & controlling the lifetime of variables in registers.

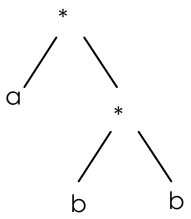
(d) none

11. The DAG for a*b*b is

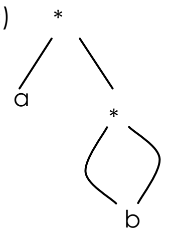
(a)



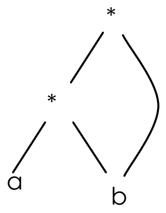
(b)



(c)



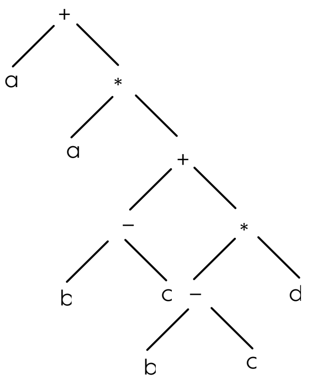
(d)



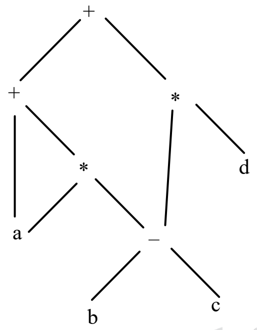
12. The DAG for the expression

a + a * (b – c) + (b – c)* d

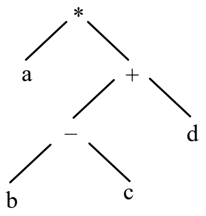
(a)



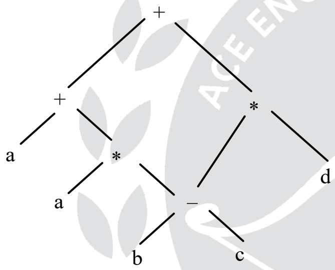
(b)



(c)



(d)



13. Consider the following block of three address statements

a = c + d

e = a + d

d = c + d

Then the number of nodes in the DAG constructed for the above block of statement is _____.

14. Consider the basic block given below.

a = b + c

c = a + d

d = b + c

e = d – b

a = e + b

The minimum number of nodes and edges present in the DAG representation of the above basic block respectively are

(a) 6 and 6 (b) 8 and 10

(c) 9 and 12 (d) 4 and 4

15. For a C program accessing X[i] [j] [k], the following intermediate code is generated by a complier. Assume that the size of an integer is 32 bits and the size of character is 8 bits.

t0 = i * 1024

t1 = j * 32

t2 = k * 4

t3 = t1 + t0

t4 = t3 + t2

t5 = X [t4]

Which one of the following statements about the source code for the C program is CORRECT?

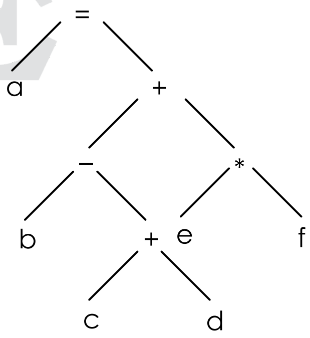
(a) X is declared as "int X[32] [32] [8]"

(b) X is declared as "int X [4] [1024] [32]"

(c) X is declared as "char X[4] [32] [8] "

(d) X is declared as "char X[32] [16] [2]"

16. The following syntax tree:



Represents the assignment

a = (b – (c+d)) + e*f;

Convert this tree to (op, arg1, arg2) triples, following these rules:

1. Evaluate the right subtree of a node before the left subtree.
2. Number the instructions (1), (2).

Then, identify from the list below, the one triple, with its instruction number, that would appear in your translation.

(a) (1) [+, e, f] (b) (3) [*, e, f]

(c) (3) [–, b, (2)] (d) (4) [–, b, (1)]

Key for Practice Questions

01. (c) 02. (d) 03. (d) 04. (d) 05. (b)

06. (c) 07. (a) 08. **3** 09. (b) 10. (d)

11. (d) 12. (b) 13. **4** 14. (b) 15. (a)

16. (b)