CHAPTER 1

Lexical Analysis

1. In a compiler the module that checks every character of the source text is called
(a) The code generator
(b) The code optimizer
(c) The lexical analyzer
(d) The syntax analyzer
[1988 :1 Mark]

2. Match the followings:
Group-I
A. Lexical analysis
B. Code optmization
C. Code generation
D. Abelian groups
Group-II
P. DAG's
Q. Syntax trees
R. Push down automation
S. Finite automaton
(a) A-S, B-P, C-Q, D-R
(b) A-R, B-P, C-Q, D-S
(c) A-S, B-Q, C-P, D-R
(d) A-S, B-P, C-R, D-Q
[1990 : 1 Mark]

3. Which of the foliowing strings can definitely be said to be token without looking at the next input character while compiling a Pascal program?
I. begin II. Program
III.<>
(a) I (b) II
(c) III (d) All of these
[1995 : 1 Mark]

4. In some programming languages an identifier is permitted to be a letter followed by any number of letters or digits. If L and D denotes the sets of letters and digits respectively. Which of the following expression defines an identifier?
(a) (L + D)+
(b) L(L + D)*
(c) (L.D)*
(d) L(L.D)*
[1995 : 1 Mark]

5. Type checking is normally done during
(a) Lexical analysis
(b) Syntax analysis
(c) Syntax directed translation
(d) Code optimization
[1998 : 1 Mark]

6. The number of tokens in the Fortran statement DO 10 I = 1.25 is
[1999 : 2 Marks]

7. The number of tokens in the following C statement
print f ("i = %d, &i = %x", i, &i); is
(a) 3 (b) 26
(c) 10 (d) 21 **[2000 : 1 Mark]**

8. Which of the following is NOT an advantage of using shared, dynamically linked libraries as opposed to using statically linked libraries?
(a) Smallersizesofexecutable
(b) Lesser overall page fault rate in the system
(c) Faster program startup
(d) Existing programs need not be re-linked to take advantage of newer versions of libraries
[2003 : 2 Marks]

9. Consider a program P that consists of two source modules M1 and M2 contained in two different files. If M1 contains a reference to a function defined in M2, the reference will be resolved at
(a) Edit-time (b) Compile-time
(c) Link-time (d) Load-time
[2004:1 Mark]

10. Consider line number 3 of the following C-program.
```
int main () {  /* Line 1 */
    int i, n;    /* Line 2 */
    fro (i = 0, i < n, i + +);   /* Line 3 */
}
```
Identify the compiler's response about this line while creating the object - module
(a) No compilation error
(b) Only a lexical error
(c) Only syntactic errors
(d) Both lexical and syntactic errors
[2005 : 2 Marks]

11. Which of the following statements are TRUE?

I. There exist parsing algorithms for some programming languages whose complexities are less than $O(n^3)$

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

III. No L-attributed definition can be evaluated in the framework of bottom-up parsing

IV. Code improving transformations can be performed at both source language and intermediate code level

(a) I and II (b) I and IV

(c) III and IV (d) I, III and IV

[2009 : 1 Mark]

12. Which data structure in a complier is used for managing information about variables and their attributes?

(a) Abstract syntax tree

(b) Symbol table

(c) Semantic stack

(d) Parse table

[2010 : 1 Mark]

13. In a compiler, keywords of a language are recognized during

(a) parsing of the program

(b) the code generation

(c) the lexical analysis of the program

(d) dataflow analysis

[2011 : 1 Mark]

14. Match the following

List-I

P: Lexical analysis

Q: Top down parsing

R: Semantic analysis

S: Runtime environments

List-II

(i) Leftmost derivation

(ii) Typechecking

(iii) Regular expressions

(iv) Activation records

Codes:

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

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

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

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

[2016 (Set-2) : 1 Mark]

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

List-I

(P) Syntax tree

(Q) Character stream

(R) Intermediate representation

(S) Token stream

List-II

(i) Code generator

(ii) Syntax analyzer

(iii) Semantic analyzer

(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)

[2017 (Set-2) : 1 Mark]

16. A lexical analyzer uses the followving patterns to recognize three tokens T_1, T_2 and T_3 over the alphabet {a, b, c}.

T_1 : a? (b|c)*a

T_2 : b? (a|c)*b

T_3 : 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) $T_1T_2T_3$ (b) $T_1T_1T_3$

(c) $T_2T_1T_3$ (d) T_3T_3

[2018 : 2 Marks]

ANSWERS

1. (c) 2. (a-S, b-P, c-R, d-Q) 3. (c) 4. (b) 5. (c) 6. (5) 7. (10) 8. (c) 9. (c)

10. (c) 11. (b) 12. (b) 13. (c) 14. (b) 15. (c) 16. (d)

EXPLANATIONS

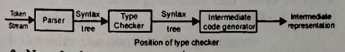
1. Lexical analyser phase read the source code character by character and generate tokens for a particular lexeme (sequence of character that satisfy some pattern).
2. Finite automata, regular expressions are used for lexical analysis DAG are used in code optimization phase to improve the efficiency of intermediate code.
3. <> used in expression for using the token.
4. Identifier–Language use identifiers as names of variables, arrays, functions. A gramar for an language often treats an identifier as a token. Each time an identifier appears in the input.

   Example : Count = Count + increment :

   id = id + id this token stream is used for parsing.
5. **Type checking.** A compiler must check that the source program follows both the syntactic and semantic conventions of the source language. This checking called static checking. Type checks is part of static checks. In type checking, a compiler should report an error if an operator is applied to an incompatible operand;

   e.g. An array variable and a function variable are added together.

   * A type checker verifies that the type of a construct mathes that expected by its content.
   * Type checker implements a type system. The type system are specified in a syntax-directed manner.



Position of type checker

6. No. of takens are ⇒ 5

   Do 10 I = 1.25

   ↓ ↓ ↓ ↓ ↓

   1 2 3 4 5

   total 5 tokens.
7. Print F "i = do d, & i = % x "

   1 3

   , i , 8 i) ;

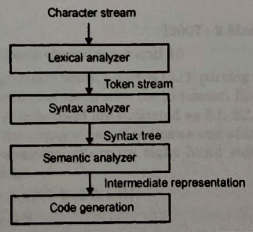
   ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

   4 5 0 7 8 9 10

   total 10 tokens
8. In Non-Shared (static) libraries, since library code is connected at compile time, the final executable has no dependencies on the library at run time i.e. no additional run-time loading costs, it means that you don't need to carry along a copy of the library that is being used and you have everything under your control and there is no dependency.
9. Linking between the different files during the compilation is done during the link time.
10. Consider 'fro' is also legel identifier, so loxical error is not generated.
11. Second option is false because recursion can not be implemented with static storage allocation.

    Third is false because L-attributed definition can be evaluated sin the framework of bottom-up parsing
12. Symbol table is a Data structure containing record for each identifier, with fields for the attribute of the identifier.

    The symbols entered into symbol table are nothing but identifier and operators. It contain information about variables and their attribute.
13. Any identifier is also a token so it is recognized in lexical Analysis
14. Lexical analysis uses Regular expression to recognize identifiers. Top down parsing uses Left Most Derivation to generate the string the of the language. Type checking is done at Semantic analysis phase of the compiler. Activation records of a function are loaded into stack memory at Run time.
15. According to the compiler phase process, we get the below structure:



16. Ans is T_3T_3 because from first T_3 bbaac is taken from second T_3 abc is taken longest possible prefix.

    Hence T_3T_3 token output.

2 CHAPTER

Parsing Techniques

PARSING

1. The pass numbers for each of the following activities
   (i) object code generation
   (ii) literals added to literal table
   (iii) listing printed
   (iv) address resolution of local symbols that occur in a two pass assembler respectively are
   (a) 1, 2, 1, 2 (b) 2, 1, 2, 1
   (c) 2, 1, l; 2 (d) 1, 2, 2, 2
   [1996 : 1 Mark]

2. The process of assigning load addresses to the various parts of the program and adjusting the code and data in the program to reflect the assigned addresses is called
   (a) Assembly
   (b) Parsing
   (c) Relocation
   (d) Symbol resolution **[2001 : 1 Mark]**

3. Which of the following statements is false?
   (a) An unambiguous grammar has same leftmost and rightmost derivation
   (b) An LL(1) parser is a top-down parser
   (c) LALR is more powerful than SLR
   (d) An ambiguous grammar can never be LR(k) for any k **[2001 : 1 Mark]**

4. Consider the context free grammar
   $E \rightarrow E + E$
   $E \rightarrow (E * E)$
   $E \rightarrow id$
   Where E is the starting symbol, the set of terminals is {id, (, + ,), *} and the set of non terminal is {E}.
   Which of the following terminal strings has more than one parse tree where parsed according to above grammar?
   (a) id + id + id + id
   (b) id + (id * (id * id))
   (c) (id + id * id)] + id
   (d) ((id * id + id) + id) **[2005 : 2 Marks]**

5. For the terminal string with more than one parse tree obtained as solution to in above Question, how many parse trees are possible?
   (a) 5 (b) 4
   (c) 3 (d) 2 **[2005 : 2 Marks]**

6. Find the grammar that generates the language $L = (a^i b^j \mid i \neq j)$. In that grammar what is the length of the derivation (number of steps starring from S) to generate the string $a^l b^m$ with $l \neq m$
   (a) max(l, m) +2
   (b) l + m + 2
   (c) l + m + 3
   (d) max (l, m) + 3 **[2006 : 2 Marks]**

Common Data for Q. 7 & Q. 8

Consider the CFG with {S, A, B} as the non-terminal alphabet, {a, b} as the terminal alphabet, S as the start symbol and the foliowing set of production rules

$S \rightarrow bA$	$S \rightarrow aB$
$A \rightarrow a$	$B \rightarrow b$
$A \rightarrow aS$	$B \rightarrow bS$
$A \rightarrow bAA$	$B \rightarrow aBB$

7. Which of the following strings is generated by the grammar?
   (a) aaaabb
   (b) aabbbb
   (c) aabbab
   (d) abbbba **[2007 : 2 Marks]**

8. For the correct answer string to above question how many derivation trees are there?
   (a) 1
   (b) 2
   (c) 3
   (d) 4 **[2007 : 2 Marks]**

Linked Data Questions 9 and 10

For the grammar below, a partial LL(1) parsing table is also presented along with the grammar. Entries that need to be filled are indicated as E1, E2, and E3. ε is the empty string, $ indicates end of input, and | separates alternate right hand sides of productions.

$S \rightarrow a\ A\ b\ B \mid b\ A\ a\ B \mid \varepsilon$
$A \rightarrow S$
$B \rightarrow S$

	a	b	$
S	E1		E2 $S \rightarrow \varepsilon$
A	$A \rightarrow S$	$A \rightarrow S$	error
B	$B \rightarrow S$	$B \rightarrow S$	E3

9. The FIRST and FOLLOW sets for the non-terminals A and B are

(a) FIRST (A) = {a, b, ε} = FIRST (B),
FOLLOW (A) = {a, b}
FOLLOW (B) = {a, b, $}

(b) FIRST (A) = {a, b, $}
FIRST (B) = {a, b, ε}
FOLLOW (A) = {a, b}
FOLLOW (B) = {$}

(c) FIRST (A) = {a, b, ε} = FIRST (B),
FOLLOW (A) = {a, b}
FOLLOW (B) = $

(d) FIRST (A) = {a, b} = FIRST (B)
FOLLOW (A) = {a, b}
FOLLOW (B) = {a, b}

[2012 : 2 Marks]

10. The appropriate entries for E1, E2 and E3 are

(a) E1 : S → aAbB, A → S
E2 : S → bAaB, B → S
E3 : B → S

(b) E1 : S → aAbB, S → ε
E2 : S → bAaB, B → ε
E3 : S → ε

(c) E1 : S → aAbB, S → ε
E2 : S → bAaB, S → E
E3 : B → S

(d) E1 : A → S, S → ε
E2 : B → S, S → ε
E3 : B → S

[2012 : 2 Marks]

11. Consider the grammar defined by the following production rules, with two operators * and +

S → T * P
T → U | T * U
P → Q + P | Q
Q → Id
U → Id

Which one of the following is TRUE?

(a) + is left associative, while * is right associative
(b) + is right associative, while * is left associative
(c) Both + and * are right associative
(d) Both + and * are left associative

[2014 (Set-2) : 1 Mark]

12. Match the following:

List-I

A. Lexical analysis
B. Parsing
C. Register allocation
D. Expression evaluation

List-II

1. Graph coloring
2. DFA minimization
3. Post-order traversal
4. Production tree

Codes:

	A	B	C	D
(a)	2	3	1	4
(b)	2	1	4	3
(c)	2	4	1	3
(d)	2	3	4	1

[2015 (Set-2) : 1 Mark]

13. Consider the following grammar:

- P → xQRx
- Q → yz|z
- R → w|ε
- S → y

What is FOLLOW (Q)?

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

[2017 (Set-1) : 1 Mark]

14. Consider the following expression grammar G:

E → E – T | T
T → T + F | F
F → (E) | id

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

(a) E → E-T | T
T → T + F | F
F → (E) | id

(b) E → TE'
E' → TE' | ∈
T → T + F | F
F → (E) | id

(c) E → TX
X → –TX | ∈
T → FY
Y → +FY | ∈
F → (E) | id

(d) E → TX | (TX)
X → TX | +TX | ∈
T → id

[2017 (Set-2) : 2 Marks]

15. Consider the following expression grammar. The semantic rules for expression calculation are stated next to each grammar production.

E → numberE.val = number.val
| E '+' E(1).val = E(2).val + E(3).val
| E '×' E(1).val = E(2).val × E(3).val;

Assume the conflicts in Part (a) of this question are resolved and an LALR (1) parser is generated for parsing arithmetic expressions as per the given grammar. Consider an expression 3 × 2 + 1. What precedence and associativity properties does the generated parser realize?

(a) Equal precedence and left associativity; expression is evaluated to 7

(b) Equal precedence and right associativity; expression is evaluated to 9

(c) Precedence of '×' is higher than that of '+', and both operators are left associative; expression is evaluated to 7

(d) Precedence of '+' is higher than that of '×', and both operators are left associative; expression is evaluaterd to 9

[2005 : 2 Marks]

16. 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) II only (d) None of I, II and III

[2020 : 1 Mark]

TOP DOWN PARSING

17. Which of the following suffices to convert an arbitrary CFG to an LL(1) grammar?

(a) Removing left recursion alone
(b) Factoring the grammar alone
(c) Removing left recursion and factoring the grammar
(d) None of the above

[2003 : 1 Mark]

18. The grammar A → AA | (A) | ε *is not suitable* for predictive-parsing because the grammar *is*

(a) ambiguous
(b) Left-recursive
(c) right-recursive
(d) an operator-grammar

[2005 : 1 Mark]

19. Which one of the following is a top-down parser?

(a) Recursive descent parser
(b) Operator precedence parser
(c) An LR(k) parser
(d) An LALR(k) parser

[2007 : 1 Mark]

20. Consider the grammar with non-terminals N = {S, C, S1}, terminals T = {a, b, i, t, e}, with S as the start symbol, and the following set of rules

S → iCtSS1 | a
S → eS | ε
C → b

The grammar is NOT LL(1) because:

(a) It is left recursive (b) It is right recursive
(c) It is ambiguous (d) It is not context-free

[2007 : 2 Marks]

21. Consider the following two statements:

P: Every regular grammar is LL (1)
Q: Every regular set has LR (1) grammar

Which of the following is TRUE?

(a) Both P and Q are true
(b) P is true and Q is false
(c) P is false and Q is true
(d) Both P and Q are false

[2007 : 2 Marks]

22. Consider the grammar given below:

S → Aa
A → BD
B → b | ∈
D → d | ∈

Let a, b, d, and $ be indexed as follows:

a	b	d	$
3	2	1	0

Compute the FOLLOW set of the non-terminal B and write the index values for the symbols in the FOLLOW set in the descendign order. (For example, if the FOLLOW set is {a, b, d, $}, then the answer should be 3210)

[2019 : 1 Mark]

23. Consider the following context-free grammar where the set of terminals is {a, b, c, d, f}.

$S \to da\,T \mid Rf$

$T \to aS \mid ba\,T \mid \epsilon$

$R \to ca\,T\,R \mid \epsilon$

The following is a partially-filled LL(1) parsing table.

	a	b	c	d	f	$
S			①	$S \to d\,a\,T$	②	
T	$T \to a\,S$	$T \to b\,a\,T$	③		$T \to \epsilon$	④
R			$R \to c\,a\,TR$		$R \to \epsilon$	

Which one of the following choices represents the correct combination for the numbered cells in the parsing table ("blank" denotes that the corresponding cell is empty) ?

(a) ① $S \to Rf$ ② $S \to Rf$ ③ $T \to \epsilon$ ④ $T \to \epsilon$

(b) ① blank ② $S \to Rf$ ③ $T \to \epsilon$ ④ $T \to \epsilon$

(c) ① $S \to Rf$ ② blank ③ blank ④ $T \to \epsilon$

(d) ① blank ② $S \to Rf$ ③ blank ④ blank

[2021 (Set-1) : 2 Marks]

24. Consider the following augmented grammar with {#, @, <, >, a, b, c} as the set of terminals.

$S' \to S$

$S \to S\,\#\,cS$

$S \to S\,S$

$S \to S@$

$S \to <S>$

$S \to a$

$S \to b$

$S \to c$

Let I_0 = CLOSURE ({$S' \to \bullet S$}). The number of items in the set GOTO (GOTO (I_0, <), <) is ____.

[2021 (Set-2) : 2 Marks]

BOTTOM UP PARSING

25. Consider the SLR(1) and LALR (1) parsing tables for a context free grammar. Which of the following statements is/are true?

(a) The goto part of both tables may be different.

(b) The shift entries are identical in both the tables.

(c) The reduce entries in the tables may be different.

(d) The error entries in the tables may be different.

[1992 : 2 Marks]

26. Which of the following statements is true?

(a) SLR parser is more powerful than LALR

(b) LALR parser is more powerful than Canonical LR parser

(c) Canonical LR parser is more powerful than LALR parser

(d) The parsers SLR, Canonical LR, and LALR have the same power

[1998 : 1 Mark]

27. Assume that the SLR parser for a grammar G has n_1 states and the LALR parser for G has n_2 states. The relationship between n_1 and n_2 is

(a) n_1 is necessarily less than n_2

(b) n_1 is necessarily equal to n_2

(c) n_1 is necessarily greater than n_2

(d) None of the above **[2003 : 1 Mark]**

28. Consider the grammar shown below

$S \to i\,E\,t\,S\,S' \mid \alpha$

$S' \to eS \mid \epsilon$

$E \to b$

In the predictive parse table M, of this grammar, the entries M[S', e] and M[S', $] respectively are

(a) {$S' \to e\,S$} and {$S' \to \epsilon$}

(b) {$S' \to e\,S$} and {}

(c) {$S' \to \epsilon$} and {$S' \to e$}

(d) {$S' \to e\,S$, $S' \to \epsilon$} and {$S' \to \epsilon$}

[2003 : 2 Marks]

29. Which of the following grammar rules violate the requirements of an operator grammar? P, Q, R are not terminals and r, s, t are terminals.

(i) $P \to Q\,R$ (ii) $P \to Q\,s\,R$

(iii) $P \to \epsilon$ (iv) $P \to Q\,t\,R\,r$

(a) (i) only (b) (i) and (iii) only

(c) (ii) and (iii) only (d) (iii) and (iv) only

[2004 : 1 Mark]

30. Consider the grammar

$S \to (S) \mid a$

Let the number of states in SLR (1), LR (1) and LALR(l) parsers for the grammar be n_1 n_2 and n_3 respectively. The following relationship holds good

(a) $n_1 < n_2 < n_3$

(b) $n_1 = n_3 < n_2$

(c) $n_1 = n_2 = n_3$

(d) $n_1 \geq n_3 \geq n_2$

[2005 : 2 Marks]

31. Consider the following grammar.

$S \to S \times E$

$S \to E$

$E \to F + E$

$E \to F$

$F \to id$

Consider the following LR(0) items corresponding to the grammar above.

(i) $S \to S \times .E$

(ii) $E \to F .+ E$

(iii) $E \to F + .E$

Given the items above, which two of them will appear in the same set in the canonical sets-of-items for the grammar?

(a) (i) and (ii)

(b) (ii) and (iii)

(c) (i) and (iii)

(d) None of these

[2006 : 1 Mark]

32. Consider the following grammar

$S \to FR$

$R \to S^* \mid \epsilon$

$F \to id$

In the predictive parser table, M, of the grammar the entries M[S, id] and M[R, $] respectively

(a) {$S \to FR$} and {$R \to \epsilon$}

(b) {$S \to FR$} and {}

(c) {$S \to FR$} and {$R \to *S$}

(d) {$F \to id$} and {$R \to \epsilon$}

[2006 : 2 Marks]

33. Which one of the following grammars generates the language $L = \{a^i b^j \mid i \neq j\}$?

(a) $S \to AC \mid CB$
$C \to aCb \mid a \mid b$
$A \to aA \mid \epsilon$
$B \to Bb \mid \epsilon$

(b) $S \to aS \mid Sb \mid a \mid b$

(c) $S \to AC \mid CB$
$C \to aCb \mid \epsilon$
$A \to aA \mid \epsilon$
$B \to Bb \mid \epsilon$

(d) $S \to ACICB$
$C \to aCb \mid \epsilon$
$A \to aA \mid a$
$B \to bB \mid b$ **[2006 : 2 Marks]**

34. 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

[2008 : 1 Mark]

35. An LALR(l) parser for a grammar G can have shift-reduce (S-R) conflicts if and only if

(a) the SLR(l) parser for G has S-R conflicts

(b) the LR(1) parser for G has S-R conflicts

(c) the LR(0) parser for G has S-R conflicts

(d) the LALR(l) parser for G has reduce-reduce conflicts **[2008 : 2 Marks]**

36. The grammar $S \rightarrow aSa \mid bS \mid c$ is

(a) LL(1) but not LR(1)

(b) LR(1) but not LL(1)

(c) Both LL(1) and LR (1)

(d) Neither LL(1) nor LR(1)

[2010 : 2 Marks]

37. What is the maximum number of reduces moves that can be taken by a bottom up parser for a grammar without epsilon and imit productions (i.e. of type $A \rightarrow \varepsilon$ and $A \rightarrow a$) to parse a string with n tokens?

(a) n/2 (b) n-1

(c) 2n-1 (d) 2^n

[2013 : 1 Mark]

38. Consider the following two sets of LR(1) items of LR(1) grammar.

$X \rightarrow c.X, c/dX \rightarrow c.X, \$$

$X \rightarrow .cX, c/dX \rightarrow .cX, \$$

$X \rightarrow .d, c/d \quad X \rightarrow .d, \$$

Which of the following statement related to merging of the two sets in the corresponding parser is/are FALSE?

1. Cannot be merged since look aheads are different.
2. Can be merged but will result in S-R conflict.
3. Can be merged but will result in R-R conflict.
4. Cannot be merged since goto on c will lead to two different sets.

(a) 1 only

(b) 2 only

(c) 1 and 4 only

(d) 1, 2, 3 and 4

[2013 : 2 Marks]

39. A canonical set of items is given below

$S \rightarrow L. > R$

$Q \rightarrow R.$

On input symbol > the set has

(a) a shift-reduce conflict and a reduce-reduce conflict.

(b) a shift-reduce conflict but not a reduce-reduce conflict.

(c) a reduce-reduce conflict but not a shift-reduce conflict.

(d) neither a shift-reduce nor a reduce-reduce conflict.

[2014 (Set-1) : 2 Marks]

40. Which one of the following is TRUE at any valid state in shift-reduce parsing?

(a) Viable prefixes appear only at the bottom of the stack and not inside

(b) Viable prefixes appear only at the top of the stack and not inside

(c) The stack contains only a set of viable prefixes

(d) The stack never contains viable prefixes

[2015 (Set-1) : 1 Mark]

41. Among simple LR (SLR), canonical LR, and look-ahead LR (LALR), which of the following pairs identify the method that is very easy to implement and the method that is the most powerful, in that order?

(a) SLR, LALR

(b) Canonical LR, LALR

(c) SLR, canonical LR

(d) LALR, canonical LR

[2015 (Set-3): 1 Mark]

42. Consider the following grammar G.

$S \rightarrow F \mid H$

$F \rightarrow p \mid c$

$H \rightarrow d \mid c$

Where S, F and H are non-terminal symbols, p, d and c are terminal symbols. Which of the following statement(s) is/are 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) Only S1 (b) Only S2

(c) Both S1 and S2 (d) Neither S1 and S2

[2015 (Set-3) : 2 Marks]

43. Which of the following statements about parser is/are CORRECT?

I. Canonical LR is more powerful than SLR.

II. SLR is more powerful than LALR.

III. SLR is more powerful than Canonical LR.

(a) I only

(b) II only

(c) III only

(d) II and III only

[2017 (Set-2) : 1 Mark]

44. Consider the following grammar.

$S \rightarrow aSB \mid d$

$B \rightarrow b$

The number of reduction steps taken by a bottom-up parser while accepting the string aaadbbb is _____.

45. Consider the following statements.

S_1: Every SLR (1) grammar is unambiguous but there are certain unambiguous grammars that are not SLR(1).

S_2: For any context-free grammar, there is a parser that takes at most $O(n^3)$ time to parse a string of length n.

Which one of the following options is correct ?

(a) S_1 is true and S_2 is false

(b) S_1 is false and S_2 is true

(c) S_1 is true and S_2 is true

(d) S_1 is false and S_2 is false

[2021 (Set-1) : 1 Mark]

46. Which one of the following statements is TRUE?

(a) The LALR (1) parser for a grammar G cannot have reduce-reduce conflict if the LR(1) parser for G does not have reduce-reduce conflict.

(b) Symbol table is accessed only during the lexical analysis phase.

(c) Data flow analysis is necessary for run-time memory management.

(d) LR(1) parsing is sufficient for deterministic context-free languages.

[2022 : 1 Mark]

47. Consider the augmented grammar with {+, *, (,), id} as the set of terminals.

$S' \rightarrow S$

$S \rightarrow S + R \mid R$

$R \rightarrow R * P \mid P$

$P \rightarrow (S) \mid id$

If I_0 is the set of two $LR(0)$ items $\{[S' \rightarrow S.], [S \rightarrow S. + R]\}$, then *goto* (*closure* (I_0), +) contains exactly _____ items.

[2022 : 1 Mark]

SHIFT REDUCE

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

$S \rightarrow xxW$ {print "1"}

$S \rightarrow y$ {print "2"}

$W \rightarrow Sz$ {print "3M}

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

(a) 23131

(b) 11233

(c) 11231

(d) 33211

[1995 : 2 Marks]

49. Which one of the following kinds of derivation is used by LR parsers?

(a) Rightmost in reverse

(b) Rightmost

(c) Leftmost in reverse

(d) Leftmost

[2019 : 1 Mark]

ANSWERS

1. (b)	2. (c)	3. (a)	4. (a)	5. (a)	6. (a)	7. (c)	8. (b)	9. (a)	10. (c)
11. (b)	12. (c)	13. (c)	14. (c)	15. (c)	16. (d)	17. (c)	18. (a,b)	19. (a)	20. (c)
21. (c)	22. (31)	23. (a)	24. (8 to 8)	25. (b,c,d)	26. (c)	27. (b)	28. (a)	29. (b)	30. (b)
31. (d)	32. (a)	33. (d)	34. (d)	35. (b)	36. (c)	37. (b)	38. (d)	39. (d)	40. (c)
41. (c)	42. (d)	43. (a)	44. (7)	45. (c)	46. (d)	47. 5 to 5	48. (a)	49. (a)	

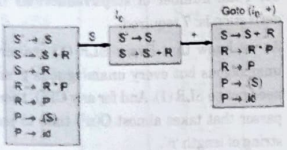
46. (a) False: Parsing is sufficient for deterministic CFL.

(b) False: Symbol table can be accessed from all the phases of a compiler

(c) False: Data flow analysis is done in control flow graph for code optimization phase.

(d) True: LR(1) parsing is sufficient for deterministic context free languages. (DCLF's)

47. According to the question;



Hence, Goto (closure (I_0) +) contains '5' items.

48. S → XXW{Print + 1}

S → Y {Print + 2}

W → SZ {Print + 3}

Translation of XXXXYZZ

S → XXW	1
→ XXSZ	3
→ XXXXWZ	1
→ XXXXSZZ	3
→ XXXXYZZ	2

Syntax directed translation scheme described by 23131

49. LR parser is a bottom up parser and so it uses right most derivation in reverse order.

∴ option (b) is the correct.

CHAPTER 3

Syntax Directed Translation

INTERMEDIATE CODE

1. Generation of intermediate code based on an abstract machine model is useful in compilers because

(a) it makes implementation of lexical analysis and syntax analysis easier

(b) syntax-directed translations can be written for intermediate code generation

(c) it enhances the portability of the front end of the complier

(d) it is not possible to generate code for real machines directly from high level language programs

[1994 : 1 Mark]

2. A linker is given object modules for a set of programs that were compiled separately. What information need to be included in an object module?

(a) Object code

(b) Relocationbits

(c) Names and locations of ali external symbols defined in the object module

(d) Absolute addresses of internal symbols

[1995 : 1 Mark]

3. In a simplified computer the instructions are: OP R_j, R_i – Performs R_j OP R_i and stores the result in register R_i

OP m, R_i – Performs val OP R_i and stores the result in R_i, val denotes the content of memory location m.

MOV m, R_i – Moves the content of memory location m to register R_i.

MOV R_i, m – Moves the content of register R_i to memory location m.

The computer has only two registers, and OP is either ADD or SUB.

Consider the following basic block:

$t_1 = a + b$

$t_2 = c + d$

$t_3 = e - t_2$

$t_4 = t_1 - t_3$

Assume that all operands are initially in memory. The final value of the computation should be in memory. What is the minimum number of MOV instructions in the code generated for this basic block?

(a) 2 (b) 3

(c) 5 (d) 6

[2007 : 2 Marks]

4. One of the purposes 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 optimizer in other compilers.

(d) improve the register allocation.

[2014 (Set-3) : 1 Mark]

5. In the context of compilers, which of the following is/are NOT an intermediate representation of the source program ?

(a) Three address code

(b) Abstract Syntax Tree (AST)

(c) Control Flow Graph (CFG)

(d) Symbol table

[2021 (Set-2) : 1 Mark]

GRAMMAR

6. In the following grammar

X : : = X ⊕ Y/Y

Y : : = Z ⊙ Y/Z

Z : : = id

Which of the followving is true?

(a) '⊕' is left associative while '⊙' is right associative

(b) Both '⊕' and'⊙' is left associative

(c) '⊕' is the right associative while '⊙' is left associative

(d) Noneoftheabove

[1997 : 1 Mark]

7. Consider the grammar rule $E \rightarrow E_1 - E_2$ for arithmetic expressions. The code generated is targeted to a CPU having a single user register. The subtraction operation requires the first operand to be in the register. If E_1 and E_2 do not have any common subexpression, in order to get the shortest possible code

(a) E_1 should be evaluated first

(b) E_2 should be evaluated first

(c) Evaluation of E_1 and E_2 should necessarily be interleaved

(d) Order of evaluation of E_1 and E_2 is of no consequence **[2004 : 1 Mark]**

8. Consider the grammar with the following translation rules and E as the start symbol.

$E \rightarrow E_1$ # T{E.value = E_1.value * T.value}
| T {E.value = T.value}
$T \rightarrow T_1$ & F{T.value = T_1.value + F.value}
| F {T.value = F.value}
$F \rightarrow$ num{F.value = num.value}

Compute E. value for the root of the parse tree for the expression: 2 # 3 & 5 # 6 & 4.

(a) 200 (b) 180
(c) 160 (d) 40

[2004 : 2 Marks]

9. Consider the grammar E → E + n | Exn | n
For a sentence n + n x n, the handles in the right-sentential form of the reduction are

(a) n, E + n and E + n x n
(b) n, E + n and E + E x n
(c) n, n + n and n + n x n
(d) n, E + n and E x n **[2005 : 2 Marks]**

Data for Q. 10

Consider the following expression grammar. The semantic rules for expression calculation are stated next to each grammar production.

E → numberE.val = number.val
| E '+' EE(1).val = E(2).val + E(3).val
| E '×' EE(1).val = E(2).val × E(3).val;

10. The above grammar and the semantic rules are fed to a yacc tool (which is an LALR(1) parser generator) for parsing and evaluating arithmetic expressions. Which one of the following is true about the action of yacc for the given grammar?

(a) It detects recursion and eliminates recursion
(b) It detects reduce-reduce conflict, and resolves
(c) It detects shift-reduce conflict, and resolves the conflict in favor of a shift over a reduce action
(d) It detects shift-reduce conflict, and resolves the conflict in favor of a reduce over a shift action **[2005 : 2 Marks]**

11. Which one of the following statements is FALSE?

(a) Context-free grammar can be used to specify both lexical and syntax rules.
(b) Type checking is done before parsing.
(c) High-level language programs can be translated to different Intermediate Representations.
(d) Arguments to a function can be passed using the program stack.

[2018 : 1 Mark]

SYNTAX DIRECTED TRANSLATION

12. In a bottom-up evaluation of a syntax directed definition, inherited attributes can

(a) always be evaluated
(b) be evaluated only if the definition is L-attributed
(c) be evaluated only if the definition has synthesized attributes
(d) never be evaluated **[2003 : 1 Mark]**

13. Consider the translation scheme shown below:

S → TR
R → + T {print ('+');} R | ε
T → num {print (num.val);}

Here num is a token that represents an integer and num.val represents the corresponding integer value. For an input string '9 + 5 + 2', this translation scheme will print

(a) 9 + 5 + 2 (b) 95 + 2 +
(c) 9 5 2 + + (d) + + 9 5 2

[2003 : 2 Marks]

14. Consider the syntax directed definition shown below:

S → id : = E {gen (id.place = E.place;);}
E → E_1 + E_2 {t = newtemp ();
gen (t = E_1. place + E_2. place;);
E.place = t}
E → id {E.place = id.place;}

Here, gen is a function that generates the output code, and newtemp is a function that returns the name of a new temporary variable on every call. Assume that t_i's are the temporary variable names generated by newtemp. For the statement 'X: = Y + Z', the 3-address code sequence generated by this definition is

(a) X = Y + Z
(b) t_1 = Y + Z; X = t_1
(c) t_1 = Y; t_2 = t_1 + Z; X = t_2
(d) t_1 = Y; t_2 = Z; t_3 = t_1 + t_2; X = t_3

[2003 : 2 Marks]

15. Consider the following translation scheme.

S → ER
R → *E{print('*');R| ε
E → F + E {print ('+'); | F
F → (S) | id {print (id. value);}

Here id is a token that represents an integer and id.value represents the corresponding integer value. For an input '2*3 + 4', this translation scheme prints

(a) 2*3 + 4 (b) 2* + 34
(c) 2 3 * 4 + (d) 2 3 4 + *

[2006 : 2 Marks]

16. 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

[2016 (Set-1) : 2 Marks]

17. Consider the following grammar and the semantic actions to support the inherited type declaration attributes. Let X_1, X_2, X_3, X_4, X_5, and X_6 be the placeholders for the non-terminals D, T, L or L_1 in the following table:

Production rule	Semantic action
D → TL	X_1.type = X_2.type
T → int	T.type = int
T → float	T.type = float
$L \rightarrow L_1$,id	X_3.type = X_4.type add Type(id.entry, X_5.type)
L → id	add Type(id.entry, X_6.type)

Which one of the following are the appropriate choices for X_1, X_2, X_3 and X_4?

(a) X_1 = T, X_2 = L, X_3 = L_1, X_4 = T
(b) X_1 = T, X_2 = L, X_3 = T, X_4 = L_1
(c) X_1 = L, X_2 = L, X_3 = L_1, X_4 = T
(d) X_1 = L, X_2 = T, X_3 = L_1, X_4 = L

[2019 : 2 Marks]

18. Consider the productions A → PQ and A → XY. Each of the five non-terminals A, P, Q, X and Y has two attributes: s is a synthesized attribute, and *i* is an inherited attribute. Consider the following rules.

Rule 1: P.i = A.i + 2, Q.i = P.i + A.i and A.s = P.s + Q.s

Rule 2: X.i = A.i + Y.s and Y.i = X.s + A.i

Which one of the following is TRUE?

(a) Both Rule 1 and Rule 2 are L-attributed.
(b) Only Rule 1 is L-attributed.
(c) Only Rule 2 is L-attributed.
(d) Neither Rule 1 nor Rule 2 is L-attributed.

19. Consider the following grammar (that admits a series of declarations, followed by expressions) and the associated syntax directed translation (SDT) actions, given as pseudo-code :

P → D* E*
D → int ID {record that ID.lexeme is of type int}
D → bool ID {record that ID.lexeme is of type bool}
E → E_1 + E_2 {check that E_1.type = E_2.type = int;set E.type := int}
E → !E_1 {check that E_1.type = bool; set E.type : = bool}
E → ID {set E.type := int}

With respect to the above grammar, which one of the following choices is correct ?

(a) The actions can be used to correctly type-check any syntactically correct program.
(b) The actions can be used to type-check syntactically correct integer variable declarations and integer expressions.
(c) The actions can be used to type-check syntactically correct boolean variable declarations and boolean expressions.
(d) The actions will lead to an infinite loop.

[2021 (Set-1) : 2 Marks]

20. Consider the following grammar along with translation rules.

$S \rightarrow S_1 \# T$	$\{S.val = S_1.val * T.val\}$
$S \rightarrow T$	$\{S.val = T.val\}$
$T \rightarrow T_1 \% R$	$\{T.val = T_1.val + R.val\}$
$T \rightarrow R$	$\{T.val = R.val\}$
$R \rightarrow id$	$\{R.val = id.val\}$

Here # and % are operators and id is a token that represents an integer and id_{val} represents the corresponding integer value. The set of non-terminals is {S, T, R, P} and a subscripted non-terminal indicates an instance of the non-terminal.

Using this translation scheme, the computed value of S_{val} for root of the parse tree for the expression 20#10%5#8%2%2 is _____.

[2022 : 2 Marks]

21. Consider the syntax directed translation given by the following grammar and semantic rules. Here N, I, F and B are non-terminals. N is the starting non-terminal, and #, 0 and 1 are lexical tokens corresponding to input letters "#", "0" and "1", respectively. X.val denotes the synthesized attribute (a numeric value) associated with a non-terminal X. I_1 and F_1 denote occurrences of I and F on the right hand side of a production, respectively. For the tokens 0 and 1, 0.val = 0 and 1.val = 1.

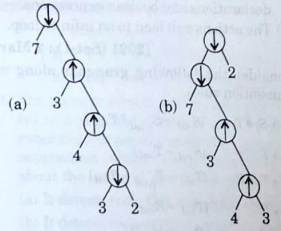
$N \rightarrow I \# F$	N.val = I.val + F.val
$I \rightarrow I_1 B$	I.val = (2 I_1.val) + B.val
$I \rightarrow B$	I.val = B.val
$F \rightarrow BF_1$	$F.val = \frac{1}{2}(B.val + F_1.val)$
$F \rightarrow B$	F.val = 2 B.val
$B \rightarrow 0$	B.val = a.val
$B \rightarrow 1$	B.val = 1.val

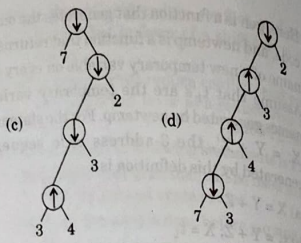
The value computed by the translation scheme for the input string 10#011 is _____.
(Rounded off to three decimal places)

[2023 : 2 Marks]

PARSER TREE & SYNTAX TREE

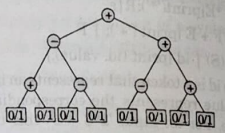
22. Consider two binary operators '↑' and '↓' with the precedence of operator ↓ being lower than that of the ↑ operator. Operator ↑ is right associative while operator ↓ is left associative. Which one of the following represents the parse tree for expression (7 ↓ 3 ↑ 4 ↑ 3 ↓ 2)?





[2011 : 2 Marks]

23. Consider the expression tree shown. Each leaf represents a numerical value, which can either be 0 or 1. Over ali possible choices of the values at the leaves, the maximum possible value of the expression represented by the tree is ____.



[2014 (Set-2) : 2 Marks]

24. Consider the following ANSI C program :

```
int main () {
    Integer x ;
    return 0 ;
}
```

Which one of the following phases in a seven-phase C compiler will throw an error ?

(a) Lexical analyzer

(b) Syntax analyzer

(c) Semantic analyzer

(d) Machine dependent optimizer

[2021 (Set-2) : 1 Mark]

POSTFIX NOTATION

25. The attributes of three arithmetic operators in some programming language are given below.

Operator	Precedence	Associativity	Arity
+	High	Left	Binary
–	Medium	Right	Binary
*	Low	Left	Binary

The value of the expression 2 – 5 + 1 – 7*3 in this language is _____.

[2016 (Set-1): 2 Marks]

ANSWERS

1. (a)	2. (d)	3. (b)	4. (c)	5. (d)	6. (a)	7. (b)	8. (c)	9. (d)	10. (c)
11. (b)	12. (c)	13. (b)	14. (b)	15. (d)	16. (c)	17. (*d*)	18. (b)	19. (b)	20. 80 to 80
21. (2.375)	22. (b)	23. (6)	24. (c)	25. (9)					

EXPLANATIONS

1. Generation of intermediate code based on an abstract machine model is useful in compilers because, it makes implementation of lexical analysis and syntax analysis easier

2. Linker is a program in a system which helps to link a object modules of program into a single object file. It performs the process of linking. Linker are also called link editors. Linking is process of collecting and maintaining piece of code and data into a single file. Linker also link a particular module into system library. It takes object modules from assembler as input and forms an executable file as output for loader.

3. Instructions will be following :

MOV	b,	R_1
ADD	a,	R_1
MOV	d,	R_2
ADD	c,	R_2
SUB	e,	R_2
SUB	R_1,	R_2
MOV	R_2,	t_4

It takes 3 MOV instructions.

4. Intermediate code is machine independent code which makes it easy to retarget the compiler to generate code for newer and different processors.

5. Three major categories of Intermediate representation called structural, linear and hybrid.

Structural Intermediate representation or Graphical IR (parse tree, abstract syntax trees, AST DAG. . .)

Linear Intermediate representation or postfix expression: (i.e. , non graphical) and three Address Code (TAC) instructions of the form "result=op1operator op2"

Hybrid Intermediate representation or Static single assignment (SSA) form: each variable is assigned once.

6. "+" is left associative while '.' is right associative because in the productin X : : = X + Y there is a left recursion and in the production Y : : = Z . Y there is right recursion

7. To optimise the solution, evaluate the expression E_2. Then E_1 can calculated and finally E_1 will be one of operands that will be in register and subtraction can be performed directly. But if opposite is followed, then make move store operations.

8. The parse tree will be

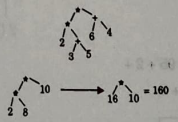
is multication

& is addition

so converting

This parse tree to multiplication and addition.

This will evaluated as



9. Derive the sentence

$E \rightarrow E * n \rightarrow E + n * n \rightarrow n + n * n$

Then handle is always substring, where reduction gives us previous right-sentential form.

$u + n * n$,

Here, handle is underlined

If we reduce, we get $\underline{E + n} * n$

(consider this is same as previous right-sentential form)

$\underline{E * n}$

E

n , $E + n$, $E * n$

CHAPTER 4 Code Generation & Optimization

1. Consider the following C code segment.

```
for (i = 0; i < n; i ++)
{
    for (j = 0; j < n; j ++)
    {
        if (i % 2)
    {   x+=(4 * j + 5*i);
        y+=(7 + 4 *j);
    }
    }
}
```

which one of the following is false?

(a) The code contains loop invariant computation

(b) There is scope of common sub-expression elimination in this code

(c) There is scope strength reduction in this code

(d) There is scope of dead code elimination in this code **[2006 : 2 Marks]**

2. Some code optimizations are carried out on the intermediate code because

(a) They enhance the portability of the complier to other target processors

(b) Program analysis is more accurate on intermediate code than on machine code

(c) The information from data flow analysis cannot otherwise be used for optimization

(d) The information from the front end cannot otherwise be used for optimization **[2008 : 1 Mark]**

3. Which languages necessarily need heap allocation in the runtime environment?

(a) Those that support recursion

(b) Those that use dynamic scoping

(c) Those that allow dynamic data structure

(d) Those that use global variables **[2010 : 1 Mark]**

Common Data for Questions 4 and 5

The following code segment is executed on a processor which allows only register operands in its instructions. Each instruction can have almost two source operands and one destinations operand. Assume that all variables are dead after this code segment.

```
c = a + b;
d = c * a;
e = c + a;
x = c * c;
if (x > a) {
    y= a * a;
}
else {
    d = d * d;
    e = e * e;
}
```

4. Suppose the instruction set architecture of the processor has only two registers. The only allowed complier optimization is code motion, which moves statements from one place to another while preserving correctness. What is the minimum number of spills to memory in the compiled code?

(a) 0 (b) 1

(c) 2 (d) 3 **[2013 : 2 Marks]**

5. What is the minimum number of registers needed in the instruction set architecture of the processor to compile this code segment without any spill to memory? Do not apply any optimization other than optimizing register allocation?

(a) 3 (b) 4

(c) 5 (d) 6 **[2013 : 2 Marks]**

6. Which one of the following is FALSE?

(a) A basic block is a sequence of instructions where control enters the sequence at the beginning and exits at the end.

(b) Available expression analysis can be used for common subexpression elimination.

(c) Live variable analysis can be used for dead code elimination.

(d) $x = 4 \times 5 \Rightarrow 20$ is an example of common subexpression elimination. **[2014 (Set-1) : 1 Mark]**

7. Which one of the following is NOT performed during compilation?

(a) Dynamic memory allocation

(b) Type checking

(c) Symbol table management

(d) Inline expansion **[2014 (Set-2) : 1 Mark]**

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

[2014 (Set-2) : 2 Marks]

9. Which of the following statements are CORRECT?

(1) Static allocation of ali data areas by a compiler makes it impossible to implement recursion.
(2) Automatic garbage collection is essential to implement recursion.
(3) Dynamic allocation of activation records is essential to implement recursion.
(4) Both heap and stack are essential to implement recursion.

(a) 1 and 2 only (b) 2 and 3 only
(c) 3 and 4 only (d) 1 and 3 only

[2014 (Set-3) : 1 Mark]

10. Consider the basic block given below.

```
a = b + c
c = a + d
d = b + c
e = d – b
a = e + b
```

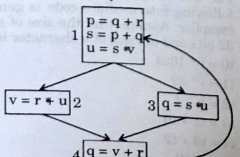
The minimum number of nodes and edges present in the DAG representation of the above basic block respectively are

(a) 6 and 6 (b) 8 and 10
(c) 9 and 12 (d) 4 and 4

[2014 (Set-3) : 2 Marks]

11. A variable x is said to be live at a statement S- in a program if the follovving three conditions hold simultaneously:

1. There exists a statement S_j that uses x
2. There is a path from S_i to S_j in the flow graph corresponding to the program
3. The path has no intervening assignment to x including at S_i and S_j



The variables which are live both at the statement in basic block 2 and at the statement in basic block 3 of the above control flow graph are

(a) p, s, u (b) r, s, u
(c) r, u (d) q, v

[2015 (Set-1): 2 Marks]

12. The least number of temporary variables required to create a three -address code in static single assignment form for the expression q+r/3 + s–t* 5+u * v/w is _____.

[2015 (Set-1): 2 Marks]

13. In the context of abstract-syntax-tree (AST) and control-flow-graph (CFG), which one of the follovving is True?

(a) In both AST and CFG, let node N_2 be the successor of node N_1. In the input program, the code corresponding to N_2 is present after the code corresponding to N_1
(b) For any input program, neither AST nor CFG will contain a cycle
(c) The maximum number of successors of a node in an AST and a CFG depends on the input program
(d) Each node in AST and CFG corresponds to at most one statement in the input program

[2015 (Set-2) : 1 Mark]

14. Consider the intermediate 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, respectively, are

(a) 5 and 7 (b) 6 and 7
(c) 5 and 5 (d) 7 and 8

[2015 (Set-2) : 2 Marks]

15. Consider the following code segment.

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

The minimum number oftotal variables required to convert the above code segment to static single assignment form is _____.

[2016 (Set-1) : 1 Mark]

16. A student wrote two context-free grammars G1 and G2 for generating a single C-like array declaration. The dimension of the array is at least one. For example,

```
int a [10] [3];
```

The grammars use D as the start symbol, and use six terminal symbols int; id [ ] num.

Grammar G1 | Grammar G2

$D \rightarrow$ int L; $D \rightarrow$ int L;
$L \rightarrow$ id [E $L \rightarrow$ idE
$E \rightarrow$ num] $E \rightarrow$ E[num]
$E \rightarrow$ num] [E $E \rightarrow$ [num]

Which of the grammars correctly generate the declaration mentioned above?

(a) Both G1 and G2 (b) Only G1
(c) Only G2 (d) Neither G1 nor G2

[2016 (Set-2) : 2 Marks]

17. 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) $p_1 = a - b$ (b) $p_3 = a - b$
$q_1 = p_1 * c$ $q_4 = p_3 * c$
$p_1 = u * v$ $p_4 = u * v$
$q_1 = p_1 + q_1$ $q_5 = p_4 + q_4$
(c) $p_1 = a - b$ (d) $p_1 = a - b$
$q_1 = p_2 * c$ $q_1 = p * c$
$p_3 = u * v$ $p_2 = u * v$
$q_2 = p_4 + q_3$ $q_2 = p + q$

[2017 (Set-1) : 1 Mark]

18. Consider the expression (a –1)* (((b + c) /3) + d)). Let X be the minimum number of registers required by an optimal code generation (without any register spill) algorithm for a load/store architecture, in which (i) only load and store instructions can have memory operands and (ii) arithmetic instructions can have only register or immediate operands. The value of X is ____.

[2017 (Set-1): 2 Marks]

19. Consider the following grammar:

stmt → if expr then expr else expr; stmt | 0 expr
→ term relop term | term
term → id | number
id → a | b | c
number → [0-9]

where relop is a relational operator (e.g., <, >, ...), 0 refers to the empty statement, and if, then, else are terminals.

Consider a program P following the above grammar containing ten if terminals. The number of control flow paths in P is _____.

For example, the program

if e_1 then e_2 else e_3

has 2 control flow paths, $e_1 \rightarrow e_2$ and $e_1 \rightarrow e_3$.

[2017 (Set-1) : 2 Marks]

20. Consider the following C code segment :

```
a = b + c;
e = a + 1;
d = b + c;
f = d + 1;
g = e + f;
```

In a compiler, this code segment is represented internally as a directed acyclic graph (DAG). The number of nodes in the DAG is _____.

[2021 (Set-1) : 2 Marks]

21. Consider the following ANSI C code segment :

```
z = x + 3 + y->f1 + y->f2;
for (i = 0; i < 200; i = i + 2) {
 if (z > i) {
   p = p + x + 3;
   q = q + y->f1;
 } else {
   p = p + y->f2;
   q = q + x + 3;
 }
}
```

Assume that the variable y points to a struct (allocated on the heap) containing two fields f1 and f2, and the local variables x, y, z, p, q, and i are allotted registers. Common sub-expression elimination (CSE) optimization is applied on the code. The number of addition and dereference operations (of the form y–> f1 or y–> f2) in the optimized code, respectively, are :

(a) 403 and 102 (b) 203 and 2
(c) 303 and 102 (d) 303 and 2

[2021 (Set-2) : 2 Marks]

22. For a statement S in a program, in the context of liveness analysis, the following sets are defined:

USE (S) : the set of variables used in S

IN (S) : the set of variables that are live at the entry of S

OUT (S) : the set of variables that are live at the exit of S

Consider a basic block that consists of two statements, S_1 followed by S_2. Which one of the following statements is correct ?

(a) OUT (S_1) = IN (S_2)

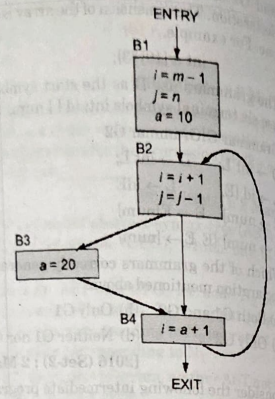
(b) OUT (S_1) = IN (S_1) ∪ USE (S_1)

(c) OUT (S_1) = IN (S_2) ∪ OUT (S_2)

(d) OUT (S_1) = USE (S_1) ∪ IN (S_2)

[2021 (Set-2) : 2 Marks]

23. Consider the control flow graph shown:



Which one of the following choices correctly lists the set of live variables at the exit point of each basic block?

(a) B1: { }, B2: {a}, B3: {a}, B4: {a}

(b) B1: {i, j}, B2: {a}, B3: {a}, B4: {i}

(c) B1: {a, i, j}, B2: {a, i, j}, B3: {a, i}, B4: {a}

(d) B1: {a, i, j}, B2: {a, j}, B3: {a, j}, B4: {a, i, j}

[2023 : 1 Mark]

ANSWERS

1. (d)	2. (a)	3. (c)	4. (b)	5. (b)	6. (d)	7. (a)	8. (a)	9. (d)	10. (a)
11. (c)	12. (8)	13. (c)	14. (b)	15. (d)	16. (a)	17. (b)	18. (2)	19. (1024)	20. (6 to 6)
21. (d)	22. (a)	23. (d)							

EXPLANATIONS

1. 4 * j is comman subexpression so we can eleminate it. B is true.

   5 * i can be moved out of inner loop so can be i % 2.

   Means, A is true as we have loop invariant computation.

   Now, 4 * j as well as 5*i can be replaced with a = – 4; before j loop then a = a + 4; where 4 * j is computed, likewise for 5 * i. So option C is true as there is scope of strength reduction.

2. Code optimizations are carried out on the intermediate code so that same source code can be converted to machine language of the target machine depending upon the back end tools. It also increases the probability of the compiler, i.e. compiler can generate machine code of different machine codes.

3. When dynamic data structure is used, heap should be used for runtime environment.

4. R1 R2

   a b c = a + b

   a c x = c * c

   a x but we will have to store c in memory as we don't know if x > a or not

   y x y = a * a

   The best case of x > a , min spills = 1

5. 4 register Requierd

Statement	Registers used
c = a + b;	R2 ← R1 + R2
d = c * a;	R3 ← R2 * R1
e = c + a;	R4 ← R2 + R1
x = c * c;	R2 ← R2 * R2
if(x > a){	Here Wd Compare R2 and R1
y = a * a;	R1 ← R1 * R1
}	
else{	
d = d * d;	R3 ← R3 * R3
e = e * e;	R4 ← R4 * R4
}	

6. X = 4 * 5 ⇒ x = 20 is an example of Constant folding.

7. At compilation time dynamic memory allocation is not feasible.

8. The expression correspoint to t5 to calculate address of array is

   t5 = X[t4]

   = X[t3 + t2]

   = X[t1 + t0 + t2]

   = X[i * 1024 + j * 32 + k * 4]

   = X + i * 1024 + j * 32 + k * 4

   So, X is declared as "int X[32][32][8]".

9. Statement 1 and 3 is true.

10. The given basic block can be rewritten as

$a = b + c$	$a = b + c$
$c = a + d$	$c = b + c + d$
$d = b + c \Rightarrow$	$d = b + b + c + d = 2b + c + d$
$e = d - b$	$e = b + b + c + d - b = b + c + d$
$a = e + b$	$a = b + b + c + d = 2b + c + d$

From above simplification it is visible that e is same as c and final value of a is same as d. So the final basic block can be written as follows:

$a = b + c$

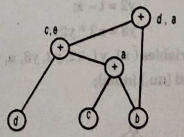
$c = a + d$

$d = 2b + c + d$

$e = c$

$a = d$

The DAG generated for the above basic block in as



Maximum number of modes and edges in above DAG is (6,6)

11. According to question, a variable is live if it holds a value that may be needed in the future i.e. it is used in future before any new assignment.

    So r and u is live variable.