PART-A Questions

1. Is macro processing a phase in compilation? Justify your answer

2. A _____is a program that collects statistics on the behavior of the object program during execution

3. Determine whether the following regular expressions define the same language? (ab)* and a*b*

4. Why is buffering used in lexical analysis? What are the commonly used buffering methods?

5. _____ verifies whether the input string can be generated from the grammar of the source language.

6. Write the subset construction algorithm

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

8. List the properties of an operator grammar

9. What are the types of attributes in an attributed grammar?

10. What optimization can you propose for the following code

```
a := b*c;
x := b*c +5;
```

11. Define text formatters.
12. Write three address code for the expression r := 9*2+7.

13. Write shorthand notation for rr*.

14. Define finite automata.

15. What is LL(1) grammar?

16. What is YACC?

17. What are the two types of attributes that are associated with a grammar symbol?

18. State different storage-allocation strategies.

19. Give example for algebraic transformation on basic blocks?

20. Name tow descriptors used in code generation algorithm?

21. _____ is a program that reads a program written in one language and translates it into an equivalent program in another language.

22. What are the two classes of parsing methods?

23. Define Lexeme.

24. Write any two algebraic properties of regular expression.

25. _____ is a program consisting of a procedure for every non terminal.

26. What do you mean by ambiguous?

27. Define Annotated parse tree.

28. _____ keeps track of live procedure activations.

29. A loop that contains no other loops is called a _____.

30. Define optimizing compiler.

31. What is the purpose of semantic analysis in a compiler?

32. What are cousins of a compiler?

33. Draw NFA for regular expression (s/t).

34. What is meant by Lexeme?

35. List out bottom up parsing.

36. What is canonical derivation?

37. What is inherited attribute?

38. What is coercion?

39. What is constant folding?

40. What is a basic block?

41. List the functions of a preprocessor.

42. List the cousins of a compiler.

43. What language does the regular expression (0|1)*0(0|1)(0|1) generate?

44. Define ε-closure.

45. What language does the grammar S →aSa | bSb | c generate?

46. Define operator grammar.

47. What is meant by quadruple?

48. What is called annotated parse tree?
49. Define flow graph.
50. Translate the arithmetic expression a * - (b + c) into a syntax tree.
51. Define Lexeme.
52. List two compiler construction tools.
53. What is meant by Kleene Closure?
54. Define optimizing compilers.
55. What language does the grammar S → aSb | c generate?
56. Define ambiguity of a grammar.
57. What is synthesized attribute?
58. Give the postfix for a=b*-c+b*-c
59. Define constant folding.
60. What is data flow analysis?
61. Define Ambiguous grammar.
62. List out the cousins of compiler.
63. Define lexeme. Give example.
64. Compare DFA and NFA.
65. Define 'Handle Pruning' in bottom-up parsing.
66. What is meant by a Look ahead symbol?
67. What is three address code?
68. Define annotated parse tree.
69. Construct a DAG for the expression a=b*-c + b*-c
70. Define symbol table.
71. What is an interpreter?
72. List any two cousins of compiler.
73. Write a regular expression for an identifier.

74. List the various error recovery strategies for a lexical analysis.
75. Define a context free grammar.
76. What are the problems with top down parsing?
77. What are the various types of intermediate code representation?
78. What is back patching?
79. What is a flow graph?
80. What are the basic goals of code movement?
81. What is a byte code?
82. What is the use of an interpreter?
83. How is a character string recognized?
84. Define Lexeme.
85. List the common syntactic errors.
86. What do you mean by panic mode error recovery?
87. What is syntax directed definition?
88. What is the use of a dependency graph?
89. In a program, when the instruction HALT is given, where does the control return to?
90. Define semilattice.
91. What does the front end of a compiler do?
92. Write the components of context free grammar.
93. How is the C language identifier represented by a regular expression?
94. Write the purpose of minimizing the states of the DFA in a lexical analyzer.
95. List the properties of operator precedence grammar.
96. Mention the four values of LR parsing table.
97. Give an example of static checks.
98. Mention the strategies of storage allocation.
99. Write the various three address code form of intermediate code.

100. What is the various output form of the code generator?
101. What is meant by real time operating systems?
102. Compare loosely coupled and tightly coupled system.
103. What is meant by context switch?
104. What are the benefits of multithreaded programming?
105. What are the three requirements that a solution to the critical-section problem satisfy?
106. Define deadlock.
107. What is compaction?
108. How do you compute the effective access time for a demand-page system?
109. What is latency time?
110. What are streams?
111. What are the functions performed by analysis phase of a compiler?
112. Mention the components of context free grammar.
113. What are the two phases of lexical analyzer?
114. Write the regular expression for the language, the set of strings over {a,b,c} that contain exactly one b.
115. Define left factoring.
116. Mention the conflicts that occur in shift-reduce parser.
117. What are the functions used to create the nodes of the syntax trees for expressions with binary operators?
118. List out the methods available to represent the value of Boolean expression.
119. Name the techniques in loop optimization.
120. How are dummy blocks with no statements indicated in global data flow analysis?
121 . What are the two parts of compilation?
122. Define ambiguous grammar.
123. _____ represent strings of characters in the source program.

124. Write the regular expression for denoting the set containing the string 'a' and all strings consisting of zero or more 'a''s followed by a b.

125. Why is error detection and recovery centered around syntax analysis phase?

126. What is parsing?

127. Mention the restrictions that are invoked by translation routines during parsing?

128. Why do we need backpatching?

129. List the benefits of using a machine independent form.

130. Define basic block.

131. Define Compiler.

132. List any two cousins of a Compiler.

133. "baan" is a subsequence of the string "banana". (True/ False)?

134. Write the regular expression for a sequence of binary numbers.

135. The rightmost derivation is known as _____.

136. When is a grammar said to be left recursive?

137. In a syntax directed definition if all attributes are synthesized, what is the name of the syntax directed definition?

138. List the various static checks.

139. Give an example for a dangling reference.

140. Define induction variables.

141. What is parsing?

142. What do you mean by Cross-Compiler?

143. Define lexeme.

144. What is the role of lexical analysis phase?

145. 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

146 Context Free Grammar can be recognized by Push Down Automata. (True/False)

147. Name some variety of intermediate forms.

148. Draw syntax tree for the expression a=b*-c+b*-c.

149. Mention some of the major optimization techniques.

150. Define basic block.

151. List the functions performed by analysis phase of a compiler.

152. Mention the components of context free grammar.

153. Write the regular expression for the language, the set of strings over {a, b, c} that contain exactly one b.

154. Write the operations on NFA states.

155. Define left factoring.

156. List the various lexical errors.

157. Write the different storage allocation strategies.

158. List out the methods available to represent the value of Boolean expression.

159. List the two classes of transformations applied to basic blocks.

160. How to represent the dummy blocks with no statements indicated in global data flow analysis?

PART-B Questions

1. Differentiate between lexeme, token and pattern

2. Draw the parse tree for the following natural language grammar:

Sentence	→	<Noun Phrase><Verb Phrase>
Noun Phrase	→	<Article><Noun>
Verb Phrase	→	<verb><Noun Phrase>
Article	→	an | a
Noun	→	<Adjective> <Noun>
Adjective	→	huge
Noun	→	animal | elephant
Verb	→	is

3. Identify whether the following grammar is ambiguous:
   G={(S),{a,b},{S→SaS,S→b},S}

4. Is the following grammar left recursive? If so eliminate left recursion

   S → Aa | b
   A → Ac | Sd | e

5. What are the advantages of LALR parsing over SLR and CLR methods?

6. What do you meant by data-flow engine?

7. Draw the transition diagram for identifier.

8. Give example for reduction.

9. What is activation tree?

10. What is copy propagation? Give example.

11. Write a short note on functions of preprocessors.

12. What are the issues in lexical analysis?

13. Write a short note on left factoring.

14. Describe about type systems.

15. What are the different operations used to define semantic rules?

16. Draw a Parse tree for the assignment statement using appropriate grammar.

    position : = initial + rate* 60

17. Write a segment of code to move forward pointer of input buffer.

18. Consider the arithmetic expression grammar, show left most and right most derivation for id + id * id and draw their parse trees.

19. Write an algorithm for construction of dependency graph from a parse tree.

20. Give quadruples and triples for an assignment statement a := b*-c + b*-c.

21. What are the different data structures used for symbol table?

22. Compare syntax tree and parse tree.

23. Do left factoring in the following grammar.

    A → aBcC | aBb | aB | a
    B → ε
    C → ε

24. Define *l*-value and r-value.
25. What is meant by induction variable elimination?
26. Define Preprocessor. What are its functions?
27. What are the different strategies employed by parsers to recover from a syntactic error?
28. Remove left recursion in the following grammar.

    S→AS | b
    A→SA | a
29. Compare quadruples and triples.
30. What are registers and address descriptors?
31. Write a short note on input buffering techniques.
32. What are the properties of Regular Expression?
33. Discuss about the 'panic mode' error recovery strategies of the parser.
34. Mention the different storage allocation strategies.
35. Write shortly about DAG (Directed Acyclic Graph) representation.
36. What are the phases that constitute the front end of a compiler?
37. Write the algorithm for FOLLOW.
38. What is shift – reduce conflict?
39. Give the syntax-directed definition for if-else statement.
40. What is a basic block? Give an example.
41 List the steps for a language processing system.
42. Differentiate lexical analysis and parsing.
43. Draw the parse tree for the arithmetic expression id + (id * id).
44. How is the syntax tree constructed?
45. What does data flow analysis frame work consist of?
46. What are the functions of structure editor?
47. Why are regular expressions used to define lexical syntax of a language?

48. What are the different types of errors a program can contain? List out the error handling strategies.

49. Write an algorithm to partition a sequence of three-address statements into basic blocks.

50. Write about type systems and checking of types.

51. List any three services provided by an operating system. Explain how each provides convenience to the users.

52. Compare short-term, medium-term, and long-term schedulers.

53. Summarize the Safety Algorithm.

54. What are the various scheduling criteria for CPU scheduling?

55. What is low-level formatting?

56. What are the functions performed when the preprocessors produce input to compilers?

57. Define NFA and DFA.

58. Show the techniques available to construct an LR parsing table for a grammar and features.

59. Draw the syntax tree and postfix notation for the expression a: =b*-c

60. Draw the DAG for the expression a: = b *-c + b*-c.

61. What is the function of Query interpreters?

62. List the issues involved in lexical analysis.

63. Write an algorithm to eliminate left recursion from a grammar.

64. What are the three storage allocation strategies?

65. What are the difficulties in instruction selection?

66. Explain about Assembler with an example.

62. E → E + E | E * E | (E) | id. Write the leftmost derivation and rightmost derivation for the sentence (id*id)+id.

63. Write the algorithm to find the FIRST of nono-terminals.

64. What is an Activation record?

65. Draw the DAG for the statement a=(a*b+c)-(a*b+c).

66. Write a note on syntax directed translation.

62. Write state program for the token, "id-identifier".

63. Define left recursion. Eliminate left recursion from the grammar. S → Aa/b, A → Ac/Sd/e?

64. What are the three kinds of intermediate representations? Explain them.

65. What is the step takes place in peephole optimization? What are the characteristics of peephole optimization?

66. Write the benefits of intermediate code generation.

67. Differentiate NFA and DFA.

68. Write about the role of the parser.

69. Draw the syntax tree and postfix notation for the expression a: =b*-c.

70. Write about type systems and checking of types.

PART –C Questions

1. Explain in detail the process of compilation. Illustrate the output of each phase of compilation for the input "a = (b+c)* (b+c) * 2"

2. a. Define the flowing terms Compiler, Interpreter, Translator and differentiate between them.

   b. Why is it necessary to study the theory behind the design of compiler.

3. Construct the NFA, DFA and minimized DFA for the regular expression have single line comments with characters from the alphabet {a,b}.

4. a. What is the role of the lexical analyzer? Explain
   b. Identify the token and, lexemes in the following function:
   function gcd (m, n: integer): integer;

```
begin
        if n = 0 then gcd := m
        else gcd := gcd (n, m mod n)
end; ( * of gcd*)
```

5. a. Write the algorithm to carry our operator precedence parsing action and explain

b. Construct the operator precedence parse table for the following grammar and show its shift-reduce actions for the input string " abab".

$S \rightarrow aSbS \mid bSaS \mid \varepsilon$

6. Consider the following grammar

$S \rightarrow AS \mid b$

$A \rightarrow SA \mid a$

Construct the SLR parse table for the grammar. Show the actions of the parser for the input string "abab".

7. a. Compare the different implementations of three address codes with examples

b. Are the attributes in the following CFG synthesized or inherited? Give reasons:

Var → IntConstant
$\{\$0.val = \$1.lexval;\}$

Expr → Var
$\{\$0.val = \$1.val;\}$

Expr → Expr B-op Expr
$\{\$0.val = \$2.val(\$1.val, \$3.val);\}$

B-op → +
$\{\$0.val = PLUS;\}$

B-op → *
$\{\$0.val = TIMES;\}$

8. Describe the syntax directed translation procedure for assignment statements with integers and mixed types and explain.

9. Construct the TAC and draw the flow graph for the following procedure.

```
void quicksort(int m, int n)
{
   int i, j;
   int v, x;
   if (n <= m) return;
   i = m-1; j = n; v = a[n];
   while(1) {
            do i = i+ 1; while (a[i] < v);
            do j = j+ 1; while (a[j] > v);
            if( i >= j) break;
            x = a[i]; a[i] = a[j]; a]j] = x;
   }
   x = a[i]; a[i] = a[n]; a]n] = x;
```

quicksort(m, j); quicksort(i+1, n);
}
Identify and eliminate induction variables in the same.

10. Discuss the issues in code generation with examples.

11. Explain the various phases of compiler in detail with suitable example.

12. a. Explain program development environment. b. What are the two parts of compilation?

13. Obtain the minimized state DFA for the regular expression (a/b)*a using subset construction method.

14. Write the algorithm to convert from NFA to DFA.

15. Construct the predictive parser for the following grammar

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

16. a. Describe the different error recovery techniques.
b. What is left recursive grammar? How will you eliminate? Give example.

17. Explain the Syntax directed definition for construction of syntax tree with example.

18. Explain different type expressions with example.

19. Explain the principal sources of optimization with example.

20. Explain the issues of code generator.

21. Explain different phases of compiler in detail.

22. Write short notes on the following:

a. Compiler – construction tools

b. Precedence of operators

23. Explain how to incorporate a symbol table.

24. Describe about Nondeterministic and Deterministic finite automata.

25. Explain non recursive predictive parsing in detail.

26. Describe LR parsing algorithm.

27. Describe the following

    a. Type exression

    b. Synthesized and inherited attributes.

28. What is different storage – allocation strategies? Explain any two allocation strategies in detail.

29. Explain different issues in the design of a code generator

30. Describe the following:

    a Structure – preserving transformation

    b. Basic Blocks

31. Explain syntax directed definition for a simple translating expression. Give an example.

32. What are compiler construction tools? Write note on each Compiler-Construction tool.

33. a. List out the Algebraic properties of regular expressions in a table.
    b. Construct NFA and then find out DFA for regular expression (a/b)*abb.

34. a. Write note on data structures for Symbol table implementation.

    b. Consider a simple programming language. Write regular expressions for tokens and draw atransition diagram for recognition of these tokens.

35.. Construct Operator Precedence Relation function for the following Grammar.

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

36. Explain in detail the various phases of compilers with an example.

37. a. Explain in detail about the compiler construction tools.
    b. List the rules for constructing Regular expressions.

38. Convert the given regular expression (a | b)*abb (a | b)* into NFA using Thompson construction and then convert to minimized DFA.

39. Construct the minimum state DFA for the regular expression a (b | c)*

40. Construct a SLR parsing table for the grammar

E → E + T | T
T → TF | F
F → F * | a | b

41. Explain in detail the various phases of compilers with an example.

42. Discuss in detail the cousins of compilers.

43. Construct a DFA directly from an augmented regular expression ((ε | a) b*)*

44. Convert the given regular expression (a | b)*abb(a | b)* into NFA using Thompson construction and then to minimized DFA.

45. Construct SLR parsing table for the grammar and parse the string ( )( )$.

S → S (S)
S → ε

46. Construct LALR(1) parsers for the following grammar.

S → L = R
S → R
L → * R
L → id
R → L

47. Explain syntax directed translation to draw syntax tree. Draw syntax tree and DAG for a + a * (b – c) + (b – c) * d.

48. a. Explain about activation records.
b. Explain stack and heap allocation.

49. Elaborate on the peephole optimization.

50. Discuss in detail about run time storage management.

51. Construct predictive parsing table for the grammar and parse the string (0 or 1)$

E → E or E | E and E | not E | (E) | 0 | 1

52. Explain syntax directed definition of a simple desk calculator. Using that draw annotated parse tree for 3 * 5 + 4n.

53. Discuss about the storage allocation strategies.

54. Explain in detail about loop optimization techniques.

55. a. Explain the various issues in the design of code generation.
b. Construct the DAG for the following basic block

d : = b * c
e : = a +b
b : = b * c

a : = e – d

56. Explain the various phases of a compiler in detail. Also write down the output for the following expression after each phase, position: = initial + rate*60.

57. a. Explain briefly about the compiler construction tools.
b. Write a note on front end and back end of the compiler.

58. Construct the NFA for the regular expression 011(0 | 1)* using Thompson method and convert it into minimized DFA.

59. Explain briefly about the design of Lexical Analyzer generator with its suitable diagram.

60. Construct the SLR parsing table for the following grammar.

S → CC
C → cC
C → d

61. a. Explain the various compiler construction tools in detail.
b. What is an ambiguous grammar? Is the following grammar ambiguous? Prove. E → E + E | E * E | (E) | id

62. a. Discuss the issues involved in designing Lexical Analyzer.
b. Draw NFA for the regular expression ab* / ab.

63. Write an algorithm to convert NFA to DFA and minimize DFA. Give an example.

64. Construct the predictive parsing table for the following grammar and parse any one of the strings.

be -> be or bt / bt

bt -> bt and bf / bf

bf -> not bf / (bf) / true / false

65. Write an algorithm to construct SLR parsing table with all sub routines.

66. Write code to generate intermediate code for Boolean expression with back patching.

67. a. Write down the implementation of 3-address code with an example.
b. Write the syntax directed translation for declarations.

68. a. Explain briefly about the storage allocation strategies.
b. Write down the issues involved in code generation.

69. Explain with an example about the optimization of basic blocks.

70. a. Describe briefly about YACC parser generator.

b. Consider a grammar

$S \rightarrow (L) \mid a$

$L \rightarrow L, S \mid S$

i) What are the terminals, non-terminals and start symbol?

ii) Construct leftmost and rightmost derivations of the corresponding parse trees for the following sentence (a, (a, a)).

71. Explain briefly about Intermediate Languages and its implementation.

72. Describe briefly about the construction of syntax tree using syntax directed definitions.

73. Discuss briefly about the issues in the design of code generator.

74. Explain briefly about the principal sources of optimization.

75. Explain the phases of compiler.

76. Define parsing. Explain the types of parsing with example.

77. Define Token. Explain with suitable example about how the tokens are specified?

78. What is a finite automata? Explain DFA and NFA with examples.

79. What is a grammar? Explain with example about the ways to eliminate ambiguity.

80. State the reasons for LR parsers. Explain LR parsing algorithm.

81. Explain syntax directed translation schemes.

82. What is a three address code? List the common three address instruction forms.

83. Explain the issues in the design of code generator.

84. What is data flow analysis? Explain data flow abstraction with examples

85. Write in detail about syntax-directed translation with an example.

86. Convert NFA for (a | b)* for DFAs. Show the sequence of moves made by each in processing the input string ababba.

87. Construct minimum state DFA for the following regular expression

(a ı b)* a (a ı b).

88. Construct an LR parsing table for the following grammar

a. E -> E + T b. E -> T c. T -> T*F

d. T -> F e. F -> (E) f. F-> id

89. Explain the non-recursive implementation of predictive parsers with the help of the grammar

E-> E+T | T

T->T*F | F

F->(E) | id

90. Write short notes on the following:

a. Syntax directed definitions

b. Storage allocation strategies

91. Write and discuss the specification of simple type checker for statements, expressions and functions.

92. Discuss about basic blocks with an example.

93. Write about runtime storage management.

94. Explain how protection is provided for the hardware resources by the operating system.

95. Explain about the various systems calls.

96. Explain in detail about interprocess communication.

97. Consider the following set of processes, with the length of the CPU-burst time given in milliseconds:

Process	Burst time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

Processes are arrived in P1, P2, P3, P4, P5 order of all at time 0. Draw Gantt charts to show execution using FCFS, SJF, non-preemptive priority (lower number implies higher priority) and RR (time quantum = 1) scheduling. Also calculate waiting and turnaround time of each process for each one of the above scheduling algorithms.

98. Explain about the Banker's algorithm for deadlock avoidance.

99. Explain about the basic concepts of paging.

100. Consider the following page-reference string

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.

How many page faults would occur for the following replacement algorithms, assuming three frames? Remember all frames are initially empty.

a. FIFO replacement b. LRU replacement c. Optimal replacement

101. a. Briefly explain about the single and two level directory structures.

b. Write notes about the protection strategies provided for files.

102. Explain SSTF, SCAN, C-SCAN and LOOK disk scheduling techniques with examples.

103. Discuss about the kernel I/O subsystem.

104. Discuss the phases of compiler with the help of the statement

p: = initial + rate × 60.

105. Explain in detail about compiler construction tools and cousins of the compiler.

106. Construct the DFA's for the regular expression (a* ıb*)* abb (aıb)*.

107. Construct minimum state DFA for the regular expression (a ı b)* a(a ı b).

108. Construct a predictive parser for the following grammar

bexpr → bexpr or bterm ı bterm

bterm→ bterm and bfactor ı bfactor

bfactor→not bfactor (bexpr)ıtrueıfalse

109. Check whether the following grammar is LL(1) or not a grammar.

S → iEtSıiEtSeSıa

E→b

110. Consider the following grammar and construct the SLR parsing table

E → E +T ı T

T → TF ı F

F →F * ı a ı b

111. Explain in detail about intermediate languages with example.

112. Discuss in detail about function preserving transformation technique with suitable example.

113. Write an algorithm for construction of DAG and construct the DAG with the following three address code of dot program.

a. t1:=4*i

b. t2:a[t1]

c. t3:=4*i

d. t4:=b[t3]

e. t5:=t2*t4

f. t6:=prod + t5

g. prod := t6

h. t7 := i+1

i. i:= t7

j. if i<=20 go to (1)

114. With a neat diagram, explain the various phases of a compiler.

115. Briefly list and state the various compiler construction tools.

116. Can you minimize the number of states of DFA? Write an algorithm to minimize and justify your answer with examples.

117. With examples, explain the method of converting a NFA into a DFA.

118. Give an overview of shift reduce parsing and write the issues in it.

119. Explain the two types of bottom up parsing with examples.

120. Define a Directed Acyclic Graph. Construct a DAG and write the sequence of instructions for the expression a + a * (b – c) + (b – c) * d .

121. a. Briefly explain the types of three address code with implementation.
b. Write a short note on the syntax directed translation for assignment statements.

122. Explain basic blocks and their optimization.

123. Explain the generic issues in the design of code generators.

124. Describe the various phases of a compiler in detail. trace the output of each phase for the program segment, position = initial + rate *60, where rate is a real data type.

125. Explain the Compiler construction tools.

126. Convert the regular expression (a | b)*abb into NFA. Then convert the respective NFA to DFA and minimize it.

126. Write the algorithm to convert a regular expression to a DFA and explain this with an example.

127. Construct the predictive parsing table for the grammar and parse the statement:

not (true or false) Productions:

E → E or T | T

T → T and F | F

F → not F | (E) | true | false

128. Write an algorithm to construct an SLR parsing table with all subroutines.

129. Translate a=b*-c+b*-c into three address statements, quadruples, triples and postfix form.

130. Explain the concept of Type Systems in detail.

131. Explain about the various storage allocation strategies.

132. Discuss with an example about the optimization of basic blocks.

133. Discuss the phases of compiler with the help of the statement

p: = initial + rate × 60.

134. a. Write short notes on compiler construction tool.

b. Elucidate about the cousins of the compiler. Discuss about context free grammar and give an example for leftmost and rightmost derivations, reduction and ambiguity.

135. Construct minimum state DFA for the regular expression. (a ı b)* a(a ı b)

136. Construct the DFA's for the following regular expression. (a* ıb*)* abb (aıb)*

137. Construct a predictive parser for the following grammar

bexpr → bexpr or bterm ı bterm

bterm→ bterm and bfactor | bfactor

bfactor→not bfactor (bexpr)|true|false

138. Consider the following grammar and construct the SLR parsing table

E → E +T | T

T → TF | F

F →F * | a | b

139. Discuss about the various intermediate languages.

140. Explain the Specification of simple type checker for statements, expressions and functions.

141. Explain about runtime storage management.

142. Write an algorithm for construction of DAG and construct the DAG with the following three-address code of dot program.

a.	t1:=4*I	b.	t2:a[t1]	c.	t3:=4*I
d.	t4:=b[t3]	e.	t5:=t2*t4	f.	t6:=prod + t5
g.	prod := t6	h.	t7 := i+1	i.	i:= t7 j.

if i<=20 go to (1)

143. a. Define regular expression.
b. Give the precedence of regular expression operator.
c. Give the rules in regular expression.
d. Give the algebraic properties of regular expression.

144. a. Define Compiler. What are the phases of the Compiler? Explain with a neat diagram.

b. What are Compiler construction tools? Explain its specifications in detail.

144. a. What is DFA?
b. What are the conditions to be satisfied for NFA?
c. What is meant by recognizer?

145. a. Construct a finite automata that will accept a string of zeros and ones that contains an odd number of zeros and an even number of ones.

b. Write a short note on Token Patterns and Lexemes.

146. a. Define LL (1) grammar.
b. What are the possibilities of non-recursive predictive parsing?

c. Define LR (0) items.

d. What are the three techniques for constructing LR parsing table?

147. a. Find the item I_0 for the following grammar using CLR parsing method.

G: S →AS

S→b

A→SA

A→a

b. Why LR parsing is good and attractive?

148. What are the different storage allocation strategies? Explain.

149. Explain the various source language issues.

150. Generate the Three-address code for

```
While(i<10)
{
x = 0;
i = i+1;
}
```

151. Construct a predictive parsing table for the following Grammar

S → iEtSS' | a

S→ eS/ε

E→ b and show the parsing of input" ibtaea".

152. Write syntax directed definition to implement of a desk calculator with an LR parser and show the evaluation of expression '95*4 +5'.

153. Explain the sequence of stack allocation process for a function call.

154. Discuss about the principle sources of code optimization.

155. Write note on Runtime storage Management.

156. Describe the various phases of a compiler in detail. Trace the output of each phase for the program segment position: = initial+ rate*60 where rate is real data type.

157. a. Write short notes on conventional compilers.

b. Discuss about the process, which supports for compilation.