

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**2007-2008, EVEN SEMESTER**  
**PRINCIPLES OF COMPILER DESIGN-CS1352**  
**TWO-MARK QUESTIONS**

1. What does translator mean?

A translator is a program that takes a input program on one programming language (source language) and produces output in another language (object language or target language).

2. What are the phases of a compiler?

- Lexical analysis phase or scanning phase
- Syntax analysis phase
- Intermediate code generation
- Code optimization
- Code generation

3. What is the role of lexical analysis phase?

Lexical analyzer reads the source program one character at a time, and grouped into a sequence of atomic units called tokens. Identifiers, keywords, constants, operators and punctuation symbols such as commas, parenthesis, are typical tokens.

4. Define lexeme?

The character sequence forming a token is called lexeme for the token.

5. What are the two functions of parser?

- ☞ It checks the tokens appearing in its input, which is output of the lexical analyzer.
- ☞ It involves grouping the tokens of source program into grammatical phrases that are used by the compiler to synthesize the output. Usually grammatical phrases of the source program are represented by tree like structure called parse tree.

6. Mention the role of semantic analysis?

- ☐ Semantic analysis checks the source program for semantic errors and gathers information for the subsequent code-generation phase.
- ☐ It uses hierarchical structure to identify the operators and operands of expressions and statements.
- ☐ An important component of semantic analysis is type checking .In type checking the compiler checks that each operator has operands that are permitted by the source language specification.

- In such cases, certain programming language supports operand coercion or type coercion also.
7. Name some variety of intermediate forms.
- Postfix notation or polish notation.
  - Syntax tree
  - Three address code
  - Quadruple
  - Triple

8. Write the syntax for three-address code statement, and mention its properties.

**Syntax: A= B op C**

- Three-address instruction has at most one operator in addition to the assignment symbol. The compiler has to decide the order in which operations are to be done.
- The compiler must generate a temporary name to hold the value computed by each instruction.
- Some three-address instructions can have less than three operands.

9. What are the techniques behind code optimization phase?

- ☰ Constant folding
- ☰ Loop constant code motion
- ☰ Induction variable elimination
- ☰ Common sub expression elimination
- ☰ Strength reduction
- ☰ Mathematical identities

10. Name minimum 4 compiler construction tools.

- ④ LEX
- ④ YACC-Parser generator
- ④ Syntax directed translation scheme.
- ④ Automatic code generator
- ④ Data flow engines

11. Define the rule to develop an unambiguous grammar from ambiguous grammar.

If  $S \rightarrow \alpha S \beta S \gamma / \alpha_1 / \alpha_2 / \alpha_3 / \alpha_4 / \dots / \alpha_n$  is an ambiguous grammar then

Unambiguous grammar becomes

$$S \rightarrow \alpha S \beta S^1 \gamma / S^1$$

$$S^1 \rightarrow \alpha_1 / \alpha_2 / \alpha_3 / \alpha_4 / \dots / \alpha_n$$

12. What is meant by recognizer?

It is a part of LEX analyzer that identifies the presence of a token on the input is a recognizer for the language defining that token. It is the program, which automatically recognize the tokens. A recognizer for a language L is a program that

takes an input string “x” and response “yes” if “x” is sentence of L and “no” otherwise.

13. What are the conditions to satisfied for NFA.

- ✚ NFA should have one start state.
- ✚ NFA may have one or more accepting states.
- ✚  $\epsilon$ -Transitions may present on NFA.
- ✚ There can be more than transitions, on same input symbol from any one state.
- ✚ In NFA, from any state “S”
  - i. There can be at most 2 outgoing  $\epsilon$ -transitions.
  - ii. There can be only one transition on real input symbol.
  - iii. On real input transition it should reach the new state only.

14. Define ambiguous grammar, and specify its demerits.

If a grammar produces more than one parse tree for the given input string then it is called ambiguous grammar. Its demerit is

- It is difficult to select or determine which parse tree is suitable for an input string.

15. What is DFA

- It has no  $\epsilon$ -transitions.
- For each state “S” and input symbol ‘a’, there is at most one edge labeled ‘a’ leaving from “S”.
- DFA is easy to simulate.

16. Write state program for the token, “id-identifier”.

```
State-0: C=GETCHAR ();
        If LETTER(C) then go to state-1
        Else FAIL ();
State-1: C=GETCHAR ();
        If LETTER(C) or DIGIT (C) then go to state-1
        Else if DELIMITER(C) then go to state-2
        Else FAIL ();
State-2: RETRACT ();
        Return (id, INSTRALL ());
```

17. Mention the properties of parse tree.

- The root is labeled by the start symbol.
- Each leaf is labeled by a token or by  $\epsilon$
- Each interior node is labeled by a non terminal
- If A is the Non terminal, labeling some interior node and  $x_1, x_2, x_3 \dots x_n$  are the labels of the children

18. List the rules for constructing regular expressions.

The rules are divided into two major classifications

- (i) Basic rules (ii) Induction rules

**Basic rules:**

i)  $\epsilon$  is a regular expression denoting  $\{\epsilon\}$  (i.e.) the language contains only an empty string.

ii) For each  $a \in \Sigma$ ,  $a$  is a regular expression denoting  $\{a\}$ , the language with only one string, which consists of a single symbol  $a$ .

**Induction rules:**

i) If  $R$  and  $S$  are regular expressions denoting languages  $L_R$  and  $L_S$  respectively then,

ii)  $(R)|(S)$  is a regular expression denoting  $L_R \cup L_S$

iii)  $(R).(S)$  is a regular expression denoting  $L_R . L_S$

iv)  $(R)^*$  is a regular expression denoting  $L_R^*$ .

19. When a grammar is said to be ambiguous?

When a grammar produces more than one parse tree for a same sentence then the grammar is said to be ambiguous.

20. What do you mean by a syntax tree?

Syntax tree is a variant of a parse tree in which each leaf represents an operand and each interior node represents an operator.

21. Compare production with reduction.

The rules that define the ways in which the syntactic categories can be built are productions whereas the replacement of the string by a non-terminal according to a grammar production is called reduction.

22. Write the algebraic laws obeyed by the regular expression.

For any regular expressions  $R, S, T$  the following laws hold:

i.  $RS = SR$  (commutative)

ii.  $R(S|T) = (R|S)|T$  (associative)

iii.  $R.(S.T) = (R.S).T$  (associative)

iv.  $R(S|T) = RS|RT$  (distributive)

v.  $\epsilon R = R\epsilon = R$  ( $\epsilon$  is the identity for concatenation)

23. Define DFA.

A DFA is an acceptor for which any state and input character has at most one transition state that the acceptor changes to. If no transition state is specified the input string is rejected.

24. What is the function of a scanner?

The scanner scans the source program and separates the tokens.

25. Write down the sequence of auxiliary definitions for the following.

Decimal, Real, Exponential.

Digit =  $0|1|\dots|9$

Integer = Digit

Sign =  $+|-|\epsilon$

Signed-integer = Sign Integer

Decimal = Signed-integer. Integer| Signed-Integer| Sign. Integer  
Exponential = (Decimal| Signed-Integer) E Signed-Integer  
Real = Decimal Exponential

26. Write a short note on LEX.

A LEX source program is a specification of lexical analyzer consisting of set of regular expressions together with an action for each regular expression. The action is a piece of code, which is to be executed whenever a token specified by the corresponding regular expression is recognized. The output of a LEX is a lexical analyzer program constructed from the LEX source specification.

27. What is symbol table? What are its contents?

Symbol table is a data structure that contains all variables in the program and temporary storage and any information needed to reference or allocate storage for them.

28. What are the functions of parser?

- i. Representation of the tokens by codes (integers)
- ii. Grouping of tokens together into syntactic structure

29. Define handle pruning.

A technique to obtain the rightmost derivation in reverse (called canonical reduction sequence) is known as handle pruning (i.e.) starting with a string of terminals  $w$  to be parsed. If  $w$  is the sentence of the grammar then  $w = \alpha_n$  where  $\alpha_n$  is the  $n$ th right sentential form of unknown right most derivation.

30. Define  $\epsilon$ -closure?

$\epsilon$ -Closure is a set of states of an NFA with  $\epsilon$ -transitions, such that  $\epsilon$ -closure for some state includes all states attainable from that state by making state transitions with  $\epsilon$ -transitions. This is denoted by  $\epsilon$ -closure (state).

31. List various types of grammars.

- i. Phase sensitive grammar
- ii. Context sensitive grammar
- iii. Context-free grammar
- iv. Regular grammar

33. What are the demerits of SLR?

- ❖ It will not produce uniquely defined parsing action tables for all grammars.
- ❖ Shift-Reduce conflict.

34. Why LR parsing is good and attractive?

- ◆ LR parsers can be constructed for all programming language constructs for which CFG can be written.

- ◆ LR parsing is Non-backtracking Shift-Reduce parsing.
- ◆ Grammars parsed using LR parsers are super set of the class of grammar.
- ◆ LR parser can detect syntactic error as soon as possible, when left-to-right scan of the input.

- ◆
35. How will you change the given grammar to an augmented grammar?  
 If 'G' is a grammar with start symbol 'S' then  $G'$  is the augmented grammar for G, such that  $G'$  is G with a new start symbol  $S'$  and with production  $S' \rightarrow S$ .
36. Explain the need of augmentation.  
 To indicate to the parser that, when it should stop parsing and when to announce acceptance of the input. Acceptance occurs when the parser is about to reduce by  $S' \rightarrow S$ .
37. What are the rules for "Closure operation" in SLR parsing?  
 If 'I' is a set of items for grammar G then Closure (I) is the set of items constructed from I by the following 2 rules.
- ▶ Initially every item in I is added to Closure (I)
  - ▶ If  $A \rightarrow \alpha.B\beta$  is in Closure(I) and  $B \rightarrow \alpha$  to I, then add the item  $B \rightarrow \alpha$  to I, if it is not already there. Apply this until no more new items can be added to Closure (I).
38. Explain the rules for GOTO operation in LR parsing.  
 Goto (I, X) where I is a set of items and X is a grammar symbol. goto (I, X) is defined to be the closure of the set of all items  $[A \rightarrow \alpha X .\beta]$  such that  $[A \rightarrow \alpha . X\beta]$  is in I.
39. How to make an ACTION and GOTO entry in SLR parsing table?
- i. If  $[A \rightarrow \alpha .a\beta]$  is in  $I_i$  and  $\text{goto}(I_i, a) = I_j$  then set ACTION  $[I, a] = \text{Shift } j$ . Here 'a' must be a terminal.
  - ii. If  $[A \rightarrow \alpha.]$  is in  $I_i$  then set ACTION  $[I, a]$  to "reduce  $A \rightarrow \alpha$ " for all a in FOLLOW (A), here A should not be  $S'$ .
  - iii. If  $S' \rightarrow S$  is in  $I_i$  then set ACTION (i, \$) to "accept"
40. What are the rules to apply Closure function in CLR parser?  
 Let 'L' is a set of LR (1) items for grammar then 2 steps can compute Closure of I.
- Initially every item in I is added to Closure (I).
  - Consider,
    - $A \rightarrow X.BY, a$
    - $B \rightarrow Z$

are 2 productions and X, Y,Z are grammar symbols. Then add  $B \rightarrow .Z$ ,  $\text{FIRST}(Ya)$  as the new LR(1) item, if it is not already there. This rule has to be applied till no new items can be added to Closure (I).

41. Find the item  $I_0$  for the following grammar using CLR parsing method.

G:  $S \rightarrow AS$   
 $S \rightarrow b$   
 $A \rightarrow SA$   
 $A \rightarrow a$

$I_0$ :  $S' \rightarrow . S , \$$   
 $S \rightarrow .AS , \$$   
 $S \rightarrow .b , \$$   
 $A \rightarrow .SA , alb$   
 $A \rightarrow .a , alb$   
 $S \rightarrow .AS , alb$   
 $S \rightarrow .b , alb$

42. Specify the advantages of LALR.

Merging of states with common cores can never produce a shift/reduce conflict that was not present in any one of the original states. Because shift actions depends only one core, not the look ahead.

43. Mention the demerits of LALR parser.

- Merger will produce reduce / reduce conflict.
- On erroneous input, LALR parser may proceed to do some reductions after the LR parser has declared an error, but LALR parser never shift a symbol after the LR parser declares an error.

44. What is the syntax for YACC source specification program?

Declarations  
 $\% \%$   
 Translation rules  
 $\% \%$   
 Supporting C-routines

45. How YACC resolves parsing action conflicts?

- ✓ A reduce/reduce conflict is resolved by choosing the conflicting production listed first in the YACC specification.
- ✓ A shift/reduce conflict is resolved by shifting action or introduce associativity and precedence.

46. What is the new production added to the YACC on error recovery

When parser generated by YACC encounters errors, then YACC pops symbol from its stack until it finds the topmost state on its stack whose underlying set of items includes an item of the form  $A \rightarrow \cdot$  error  $\alpha$

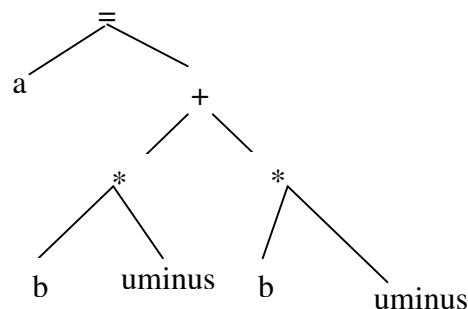
47. What are the two representations to express intermediate languages?

- Graphical representation.
- Linear representation.

48. What do you mean by DAG?

It is Directed Acyclic Graph. In this common sub expressions are eliminated. So it is a compact way of representation.

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



50. Explain postfix notation.

It is the linearized representation of syntax tree .It is a list of nodes of the syntax tree in which a node appears immediately after its children.

Eg, for the above syntax tree, a b c uminus \* b c uminus \* + =

51. What are the different types of three address statements?

- ❖ Assignment statements of the form  $x=y$  op z.
- ❖ Assignment statements of the form  $x=$  op y.
- ❖ Copy statements of the form  $x=y$ .
- ❖ An unconditional jump, goto L.
- ❖ Conditional jumps, if x relop y goto L.
- ❖ Procedure call statements, param x and call p , n.
- ❖ Indexed assignments of the form,  $x=y[i]$  ,  $x[i]=y$ .
- ❖ Address and pointer assignments.

51. In compiler how three address statements can be represented as records with fields for the operator and operands.

- Quadruples.
- Triples.
- Indirect triples.

52. Define quadruple and give one example.

A quadruple is a data structure with 4 fields like operator, argument-1, argument-2 and result.

Example:  $a=b*-c$

	operator	Argument-1	Argument-2	Result
(0)	uminus	c		T <sub>1</sub>
(1)	*	b	T <sub>1</sub>	T <sub>2</sub>
(2)	=	T <sub>2</sub>		a

53. Define triple and give one example.

A triple is a data structure with 3 fields like operator, argument-1, argument-2.

Example:  $a=b*-c$

	operator	Argument-1	Argument-2
(0)	uminus	c	
(1)	*	b	(0)
(2)	=	a	(1)

54. What is the merit of quadruples?

If we move a statement computing a variable, 'A' then the statements using 'A' requires no change. That is 'A' can be computed anywhere before the utilization of A.

55. What are the two methods of translation to find three-address statements in Boolean expression?

- Numerical method
- Control flow method

56. Generate three address code for "if A<B then 1 else 0", using numerical method.

- (1) if A<B goto (4)
- (2) T<sub>1</sub>=0
- (3) goto (5)
- (4) T<sub>1</sub>=1
- (5) ...

57. What is mean by syntax directed definition.

It is a generalization of a CFG in which each grammar symbol has an associated set of attributes like, synthesized attribute and inherited attribute.

58. How the value of synthesized attribute is computed?

It was computed from the values of attributes at the children of that node in the parse tree.

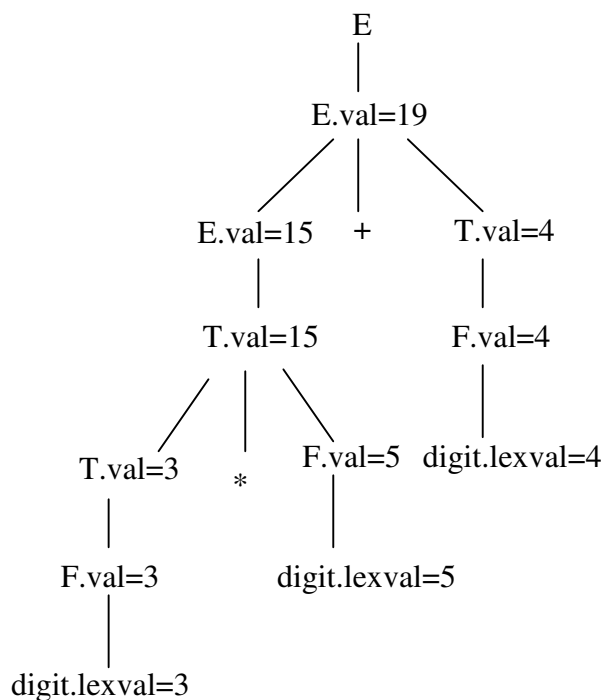
59. How the value of inherited attribute is computed?

It was computed from the value of attributes at the siblings and parent of that node.

60. Define unnoted parse tree and give one example.

A parse tree showing the values of attributes at each node is called an annotated parse tree.

Example:  $3*5+4$



61. Define backpatching.

To generate three address code, 2 passes are necessary. In first pass, labels are not specified. These statements are placed in a list. In second pass, these labels are properly filled, is called backpatching.

62. What are the three functions used for backpatching?

- Makelist (i)
- Merge (p1, p2)
- Backpatch (p, i)

63. Do left factoring in the following grammar.

$A \rightarrow aBcC \mid aBb \mid aB \mid a$

$B \rightarrow \epsilon$

$C \rightarrow \epsilon$

After applying left factoring,

$A \rightarrow A'$

$A' \rightarrow BcC \mid Bb \mid B$

$B \rightarrow \epsilon$

$C \rightarrow \epsilon$

64. When procedure call occurs, what are the steps to be taken?
- State of the calling procedure must be saved, so that it can resume after completion of procedure.
  - Return address is saved, in this location called routine must transfer after completion of procedure
65. What are the demerits in generating 3-address code for procedure calls?  
Consider,  $b = abc$  (I, J). It is very difficult to identify whether it is array reference or it is a call to the procedure  $abc$ .
66. Which information's are entered into the symbol table?
- ✦ The string of characters denoting the name.
  - ✦ Attributes of the name.
  - ✦ Parameters.
  - ✦ An offset describing the position in storage to be allocated for the name.
67. What is the demerit in uniform structure of symbol table?
- Length of the name should not exceed upper bound or limit of name field.
  - If length of name is small, then remaining space was wasted.
68. What are the different data structures used for symbol table?
- ◆ Lists
  - ◆ Self-organizing lists.
  - ◆ Search tree.
  - ◆ Hash table
69. What is the total work needed for search tree.  
To enter 'n' names, 'm' inquiries, total work is  $(n + m) \log n$ .
70. What are the demerits in self-organizing list?
- ❖ If more number of names is frequently used then it is difficult to maintain the list.
  - ❖ Since it organizes itself, sometimes problems may arrive on pointer movements.
  - ❖ For each record, one LINK field is used, it wastes memory space.
  - ❖ Time and space utilization is more.

71. What are the steps to search an entry in the hash table?
- To search, if there is an entry for string 'S' in the symbol table, apply hash function 'h' to 'S'.
  - If  $h(S)$  returns an integer between 0 to  $m-1$ .
  - If S is in Symbol table, then this integer is on the list numbered  $h(s)$ .

72. Specify one method of hash function.
- Multiply the old value of 'h' by a constant ' $\alpha$ ' .
  - $h_0=0$  ,  $h_i=\alpha h_{i-1} + C_i$   
     $C_i$ -Integer value for character.  
     $\alpha$ =constant.

73. Mention some of the major optimization techniques.
- ❖ Local optimization
  - ❖ Loop optimization
  - ❖ Data flow analysis
  - ❖ Function preserving transformations
  - ❖ Algorithm optimization.

74. What are the methods available in loop optimization?
- ◆ Code movement
  - ◆ Strength reduction
  - ◆ Loop test replacement
  - ◆ Induction variable elimination

75. What is the step takes place in peephole optimization?
- It improves the performance of the target program by examining a short sequence of target instructions. It is called peephole. Replace this instructions by a shorter or faster sequence whenever possible. It is very useful for intermediate representation.

76. What are the characteristics of peephole optimization?
- a. Redundant instruction elimination.
  - b. Flow of control optimization
  - c. Algebraic simplifications
  - d. Use of machine idioms

77. Define basic block.
- A basic block contains sequence of consecutive statements, which may be entered only at the beginning and when it is entered it is executed in sequence with out halt or possibility of branch.

78. What are the rules to find "leader" in basic block?
- ✓ It is the first statement in a basic block is a leader.

- ✓ Any statement which is the target of a conditional or unconditional goto is a leader.
- ✓ Any statement which immediately follows a conditional goto is a leader.

79. Define flow graph.

Relationships between basic blocks are represented by a directed graph called flow graph.

80. Specify some advantages of DAGs

- It automatically detects common sub expression.
- We can determine which identifiers have their values used in the block.
- We can determine which statements compute values, and which could be used outside the block.
- It reconstruct a simplified list of quadruples taking advantage of common sub expressions and not performs assignments of the form  $a=b$  unless necessary.

81. What is meant by registers and address descriptor?

Register descriptor: It contains information's about,

1. What registers are currently in use.
2. What registers are empty.

Address descriptor: It keeps track of the location where the current value of the name can be found at run time.

82. Explain heuristic ordering for DAG?

It is based on Node Listing algorithm.

- While unlisted interior nodes remain do  
begin
  - Select an unlisted node  $n$ , all of whose parents have been listed;
  - list  $n$ ;
  - While the leftmost child ' $m$ ' of ' $n$ ' has no unlisted parents and is not a leaf do

```

do
  {
    /*since n was just listed, surely m is not yet listed */
  }
begin
  list m
  n=m;
end
end

```

83. Write labeling algorithm.

- if  $n$  is a leaf then

- ✦ if n is the leftmost child of its parent then
- ✦ LABEL(n)=1
- ✦ else
- ✦ LABEL(n)=0
- ✦ else
- ✦ begin
- ✦ let n1 ,n2 ,n3,,nk be the children of n ordered by LABEL , so  
LABEL(n1)>=LABEL(n2)>=.....LABEL(nk)
- ✦ LABEL(n)=max(LABEL(ni)+i -1)
- ✦ end

84. Explain multiregister operation.

If the number of registers used for an operation is more than two then the formula can be modified as,

$\begin{aligned} \text{LABEL}(n) &= \max(2, l_1, l_2) \text{ if } l_1 \neq l_2 \\ &= l_1 + 1 \quad \text{if } l_1 = l_2 \end{aligned}$
--

85. What is meant by U-D chaining?

It is Use-Definition chaining. It is the process of gathering information about how global data flow analysis can be used id called as use-definition (ud) chaining.

86. What do you mean by induction variable elimination?

It is the process of eliminating all the induction variables , except one when there are two or more induction variables available in a loop is called induction variable elimination.

87. What are dominators?

A node of flow graph is said to be a dominator, i.e one node dominates the other node if every path from the initial node of the flow graph to that node goes through the first node.(d Dom n).when d-node dominates n-node.

88. What is meant by constant folding?

Constant folding is the process of replacing expressions by their value if the value can be computed at complex time.

89. What are the principle sources of optimization?

The principle sources of optimization are,

- ✦ Optimization consists of detecting patterns in the program and replacing these patterns by equivalent but more efficient constructs.
- ✦ The richest source of optimization is the efficient utilization of the registers and instruction set of a machine.

90. List any two structure preserving transformations adopted by the optimizer?

The structure preserving transformations adopted by the optimizer are,

- ✦ Basic blocks.
- ✦ Flow graphs.

91. What are the various types of optimization?

The various type of optimization is,

- ✦ Local optimization
- ✦ Loop optimization
- ✦ Data flow analysis
- ✦ Global optimization

92. List the criteria for selecting a code optimization technique.

The criteria for selecting a good code optimization technique are,

- ◆ It should capture most of the potential improvement without an unreasonable amount of effort.
- ◆ It should preserve the meaning of the program
- ◆ It should reduce the time or space taken by the object program.

93. State the problems in code generation.

Three main problems in code generation are,

- Deciding what machine instructions to generate.
- Deciding in what order the computations should be done and
- Deciding which registers to use.

94. What are static and dynamic errors?

Static error: It can be detected at compile time. Eg: Undeclared identifiers.

Dynamic errors: It can be detected at run time. Eg: Type checking

95. What are the different kinds of errors

- ❖ Specification errors
- ❖ Algorithmic errors
- ❖ Coding errors
- ❖ Transcription errors
- ❖ Compiler errors.

96. When does a parser detect an error?

A parser detects an error when it has no legal move from its current configuration, which is determined by its state, its stack contents and the current input symbol. To recover from an error a parser should ideally locate the position of the error, correct the error, revise its current configuration, and resume parsing.

97. What are semantic errors? Give example.

The errors detectable by the compiler is termed as semantic error. These errors are detected both at compile time and runtime.

- ❖ Semantic errors that can be detected at compile time errors of declaration and scope.

- ❖ Semantic errors that can be detected at runtime are range-checking for certain values, particularly array subscripts and case statement selectors.

98. Write down the postfix notation for the following statement.

If  $c*d+e$  then if  $a-b$  then  $a+c$  else  $a+c*d$  else  $c*d+e$ .

Answer:  $cd*e+ab-ac+acd*+cd*e+$

99. What is data flow analysis?

The data flow analysis is the transmission of useful relationships from all parts of the program to the places where the information can be of use.

100. Define code motion and loop-variant computation.

Code motion: It is the process of taking a computation that yields the same result independent of the number of times through the loops and placing it before the loop.

Loop-variant computation: It is eliminating all the induction variables, except one when there are two or more induction variables available in a loop

101. What is the main difference between phase and pass of a compiler?

A phase is a sub process of the compilation process whereas combination of one or more phases into a module is called pass.

102. Define Compiler.

Compiler is a program that reads a program written in one language –the source language- and translates it into an equivalent program in another language- the target language. In this translation process, the compiler reports to its user the presence of the errors in the source program.

103. Give the analysis –synthesis model of compilation.

The analysis part breaks up the source program into constituent pieces and creates an intermediate representation of the source program.

The synthesis part constructs the desired target program from the intermediate representation.

104. Define interpreter.

Interpreter is a language processor program that translates and executes source code directly, without compiling it to machine code.

105. Define linker.

Linker is a program that combines (multiple) object files to make an executable. It converts names of variables and functions to numbers (machine addresses).

106. Define editor.

Editors may operate on plain text, or they may be wired into the rest of the compiler, highlighting syntax errors as you go, or allowing you to insert or delete entire syntax constructed at a time.

107. Define profiler

Profiler is a program to help you see where your program is spending its time, so you can tell where you need to speed it up.

108. What is literal table?

Literal table is a table that stores “literal” values, mainly strings and variables names that may occur in many places. Only one copy of each unique string and name needs to be allocated in memory.

Let  $L = \{A, B, \dots, Z, a, b, \dots, z\}$ ,  $D = \{0, 1, \dots, 9\}$ . Determine the languages represented by  $L(LUD)^*$  and  $L^*$ .

$L(LUD)^*$  is the set of all strings of letters and digits beginning with a letter.

$L^+$  is the set of all strings of letters including the empty string.

109. What do you mean by context free grammar?

Context free grammar is a notation for specifying the syntax of a language.

A context free grammar has four components:

- i. A set of tokens, known as terminal symbols.
- ii. A set of non-terminals.
- iii. A set of productions where each production consists of a non-terminal, called the left side of the production, an arrow, and a sequence of tokens and/or non-terminals, called the right side of the production.
- iv. A designation of one of the non-terminals as the start symbol.

110. Define Assembler.

Assembler is a translator from human readable (ASCII text) files of machine instructions into the actual binary code (object files) of a machine.

111. Define Loader.

Loader is a program that performs the functions of loading and linkage editing. The process of loading consists of taking re-locatable machine code, altering the re-locatable address and placing the altered instruction and data in memory at the proper location.

112. Define Preprocessor. What are its functions?

Program that processes the source code before the compiler sees it. Usually, it implements macro expansion, but it can do much more.

The functions of a preprocessor are:

- Macro processing.
- File inclusion.
- Rational preprocessors.
- Language extensions

#### 113. Define Debugger.

Debugger is a program to help you see what is going on when your program runs. It can print the values of variables, show what procedure called what procedure to get where you are, run up to a particular line, run until a particular variables gets a special value etc.

#### 114. What is front-end and back-end of the compiler?

Often the phases of a compiler are collected into a front-end and back-end. Front-end consists of those phases that depend primarily on the source program and largely independent of the target machine. Back-end consists of those phases that depend on the target machine language and generally those portions do not depend on the source language, just the intermediate language. In back end we use aspects of code optimization, code generation, along with error handling and symbol table operations.

#### 115. Define Passes.

In an implementation of a compiler, portion of one or more phases are combined into a module called pass. A pass reads the source program or the output of the previous pass, makes the transformation specified by its phases and writes output into an intermediate file, which is read by subsequent pass.

#### 116. Define syntax and semantics.

The rules and regulations used to form a language are known as syntax. The meaning given to a programming construct is known as semantics.

#### 117. What are the classifications of a compiler?

The classifications of compiler are:

- Single-pass compiler.
- Multi-pass compiler.
- Load and go compiler.
- Debugging compiler.

- Optimizing compiler.

118. What are the attributes of an identifier?

The attributes of an identifier are:

- Storage allocated.
- Type of the identifier.
- Scope of the identifier.

119. What is linear analysis?

The stream of characters making up the source program is read from left-to-right and grouped into tokens that are sequence of characters having a collective meaning.

120. What are the cousins of the compiler?

- Preprocessors.
- Assemblers.
- Two pass assembly.
- Loader and link editors.

121. What is a silicon compiler?

A silicon compiler has a source language that is similar or identical to a conventional programming language. However, the variables of the language represent, not locations in memory, but logical signals (0 or 1) or group of signals in a switching circuit. The output is a circuit design in an appropriate language.

122. What is meant by token?

- ✓ A classification for a common set of strings.
- ✓ Example: include<Identifier>, <number>, etc.

123. What is meant by pattern?

- ✓ The rules, which characterize the set of strings for a token.
- ✓ Recall Files and OS Wildcards ([A-Z] \* . \*).

124. What are the possible error recovery actions in a lexical analyzer?

- i. Panic mode recovery.

- ✓ Delete successive characters from the remaining input until the analyzer can find a well-formed token.
- ✓ May confuse the parser.
- ii. Possible error recovery actions.
  - ✓ Deleting or Inserting input characters.
  - ✓ Replacing or Transposing characters.

125. What are the drawbacks of using buffer pairs?

- i. This buffering scheme works quite well most of the time but with it amount of lookahead is limited.
- ii. Limited lookahead makes it impossible to recognize tokens in situations where the distance, forward pointer must travel is more than the length of buffer.

126. What are the advantages of Grammar?

- Easy to understand.
- Imparts a structure to a programming language.
- Acquiring new constructs.

127. What is the role of the error handler in a parser?

- It should report the presence of errors clearly and accurately.
- It should recover from each error quickly.
- It should not significantly slow down the processing of correct programs.

128. What are the different strategies that a parser can recover from a syntactic error?

- Panic mode.
- Phrase level.
- Error productions.
- Global correction.

129. Define terminal.

Terminals are the basic symbols from which the strings are formed.

130. Define Nonterminal

Nonterminal are syntactic variables that denote set of strings.

131. What is meant by left most derivation?

In derivation, only the leftmost nonterminal in any sentential form is replaced at each step. Such derivations are termed leftmost derivation.

132. Define left factoring.

Left factoring is a grammar transformation that is useful for producing a grammar suitable for predictive parsing.

133. Define predictive parsing.

It is a program based on a transition diagram attempts to match the terminal symbols against the input.

134. What you mean by nonrecursive predictive parser?

A nonrecursive predictive parser is a program that matches the terminal symbols against the input by maintaining a stack rather than using recursive calls.

135. What is meant by shift-reduce parsing?

Shift reduce parsing constructs a parse tree for an input string beginning at the leaves and working up towards the root i.e., “reducing” a string  $w$  to the start symbol of a grammar.

136. Define Handle.

A handle of a string is a substring that matches the right side of a production and whose reduction to the nonterminal on the left side of the production represents one step along the reverse of a rightmost derivation.

137. What are the possible actions of a shift reduce parser?

- Shift.
- Reduce.
- Accept.
- Error.

138. Define viable prefixes.

The set of prefixes of right sentential forms that can appear on the stack of a shift reduce parser are called viable prefixes.

139. List down the conflicts during shift-reduce parsing.

- Shift/reduce conflict.  
Parser cannot decide whether to shift or reduce.
- Reduce/reduce conflict.  
Parser cannot decide which of the several reductions to make.

140. Differentiate Kernel and non-Kernel items.

Kernel items, which include the initial item,  $S' \rightarrow .S$  and all items whose dots are not at the left end. Whereas the nonkernel items have their dots at the left end.

141. What are the components of LR parser?

- An input.
- An output.
- A stack.
- A driver program.
- A parsing table.

142. List the different techniques to construct an LR parsing table?

- Simple LR(SLR).
- Canonical LR.
- Lookahead LR (LALR).

143. Define LR (0) item.

An LR(0) item of a grammar  $G$  is a production of  $G$  with a dot at some position of the right side.

Eg:  $A \rightarrow .XYZ$

$A \rightarrow X.YZ$

$A \rightarrow XY.Z$

$A \rightarrow XYZ.$

144. Define three-address code.

Three address code is a sequence of statements of the general form.

$x := y \text{ op } z$

where  $x, y, z$  – names, constants or compiler generated temporaries.

$\text{op}$  – operator i.e., fixed or floating point arithmetic operator or logical operator.

145. Define Boolean expression.

Boolean expressions are composed of the boolean operators (and, or, not) applied to elements. That are boolean variables or relational expressions.

146. Define short-circuit code.

It translates a boolean expression into three-address code without generating code for any of the boolean operators and without having the node necessarily evaluate the entire expression. This type of evaluation is called short circuit or jumping code.

147. Define code generation.

The code generation is the final phase of the compiler. It takes an intermediate representation of the source program as the input and produces an equivalent target program as the output.

148. Define Target machine.

The target computer is byte-addressable machine with four bytes to a word and  $n$ -general purpose registers.  $R_0, R_1, \dots, R_{n-1}$ . It has two address instructions of the form Op, source, destination in which Op is an op-code, and source and destination are data fields.

149. How do you calculate the cost of an instruction?

We take the cost of an instruction to be one plus the costs associated with the source and destination address modes. This cost corresponds to the length (in words) of the instruction. Address modes involving registers have cost zero, while those with a memory location or literal in them have cost one, because such operands have to be stored with the instruction.

150. What is meant by optimization?

It is a program transformation that made the code produced by compiling algorithms run faster or takes less space.

151. Define optimizing compilers.

Compilers that apply code-improving transformations are called optimizing compilers.

152. When do you say a transformation of a program is local?

A transformation of a program is called local, if it can be performed by looking only at the statement in a basic block.

153. Write a note on function preserving transformation.

A compiler can improve a program by transformation without changing the function it compiles.

154. List the function-preserving transformation.

- Common subexpression elimination.
- Copy propagation.
- Dead code elimination.
- Constant folding.

155. Define common subexpression.

An occurrence of an expression E is called a common subexpression if E was previously computed and the values of variables in E have not changed since the previous computation.

156. Define live variable.

A variable is live at a point in a program if its value can be used subsequently.

157. What is meant by loop optimization?

The running time of a program may be improved if we decrease the number of instructions in an inner loop even if we increase the amount of code outside that loop.

158. Define formal parameters.

The identifiers appearing in procedure definitions are special and are called formal parameters.

159. What do you mean by data flow equations?

A typical equation has the form  
$$\text{out}[s] = \text{gen}[s] \cup (\text{in}[s] - \text{kill}[s])$$

It can be read as information at the end of a statement is either generated within the statement or enters at the beginning and is not killed as control flows through the statement.

160. State the meaning of in[s], out[s], kill[s], gen[s].

in[s]-The set of definitions reaching the beginning of S.

out[s]-End of S.

gen [s]-The set of definitions generated by S.

kill[s]-The set of definitions that never reach the end of S.

161. Define actual parameters

The arguments passed to a called procedure are known as actual parameters.

162. What is meant by an activation of the procedure?

An execution of a procedure is referred to as an activation of the procedure.

163. Define activation tree.

An activation tree depicts the way control enters and leaves activations.

164. What is the use of control stack?

It keeps track of live procedure activations. Push the node for activation onto the control stack as the activation begins and to pop the node when the activation ends.

165. What is meant by scope of declaration?

The portion of the program to which a declaration applies is called the scope of that declaration. An occurrence of a name in a procedure is said to be local to the procedure if it is in the scope of declaration within the procedure; otherwise the occurrence is said to be nonlocal.

166. What does the runtime storage hold?

Runtime storage holds

- 1) The generated target code
- 2) Data objects
- 3) A counter part of the control stack to keep track of procedure activations.

167. Define activation record

Information needed by a single execution of a procedure is managed using a contiguous block of storage called an activation record.

168. Define access link

It refers to nonlocal data held in other activation records.

169. What are the different storage allocation strategies?

- 1) Static allocation.  
It lays out storage for all data objects at compile time.
- 2) Stack allocation.

It manages the runtime storage as a stack.

3) Heap allocation.

It allocates and deallocates storage as needed at runtime from a data area.

170. What do you mean by dangling reference?

A dangling reference occurs when there is a reference to storage that has been deallocated.

171. What is meant by static scope rule?

It determines the declaration that applies to a name by examining the program text alone.

172. What do you mean by dynamic scope rule?

It determines the declaration applicable to a name at run time by considering the current activations.

173. Define block.

A block is a statement containing its own local data declarations.

174. Define *l*-value and *r*-value.

The *l*-value refers to the storage represented by an expression and *r*-value refers to the value contained in the storage.

175. What are the different parameters passing methods?

- Call by value.
- Call by reference.
- Copy restores.
- Call by name

1. For the regular expression ,  $(a / b)^* abb$  · Draw the NFA. Obtain DFA from NFA. Minimize DFA using  $\Pi_{new}$  construction.

**Hints:**

- NFA
- $\epsilon$ -closure computation
- DFA diagram
- Minimization steps
- Minimized DFA

2. Explain the various phases of a compiler in detail. Also write the output for the following expression after each phase  $a=b*c-d$

**Hints:**

- ❖ Lexical analysis
- ❖ Syntax analysis and semantic analysis
- ❖ Intermediate code generation
- ❖ Code optimization
- ❖ Code generation

3. Explain in detail about compiler construction tools.

**Hints:**

- LEX
- YACC
- Syntax Directed Translation
- Code generator

4. i. Find the language from

(4)

$$S \rightarrow 0S1 / 0A1$$

$$A \rightarrow 1A0 / 10$$

- ii. Define Parse tree , Regular Expression , Left most derivation , Right most (4)

derivation , and write example for each.

iii. Write algorithm to convert NFA from Regular expression. (4)

iv. Find the language from (4)

$$S \rightarrow 0S1 / 0A / 01B / 1 \quad A \rightarrow 0A / 0 \quad B \rightarrow 1B / 1$$

**Hints:**

i. Refer book for rules.

ii. Definition of parse tree, an regular expression and LMD, RMD with examples

iii. Algorithm –refer book

iv. Refer book for algorithm

5. i. Prove the grammar is ambiguous.

(4)

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

ii. Specify the demerits of ambiguous grammar.

(2)

iii. What are the rules to convert an unambiguous grammar from ambiguous

(6)

grammar . Write necessary steps for the above ambiguous grammar.

iv. Using unambiguous grammar, write Leftmost derivation , draw parse tree

(4)

for the string  $id * id * id + id * id$

**Hints:**

- Draw more than parse tree for any expression.
- It is difficult to find which parse tree is correct for evaluation
- Refer book for rules.
- Apply rules for conversion.

6. Explain in detail about error recovery.

**Hints:**

- ✓ Lexical phase error recovery
- ✓ Syntactic phase error recovery

- ✓ Intermediate code generation error recovery.
- ✓ Code optimization phase error recovery.
- ✓ Code generation error recovery.

7. Write in detail about

i. Recursive descent parsing with algorithm.

ii. Top down parsing with algorithm.

**Hints:**

- Definition of recursive descent and algorithm.
- Definition of top down parser, and algorithm for top down parsing table, parsing method of top down parser.

8. Explain in detail about lexical analyzer generator.

**Hints:**

- Definition
- Typical program style
- Transition diagram.

9. Construct predictive parsing table and parse the string  $id+id*id$ .

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid id$

**Hints:**

- ✚ Find FIRST and FOLLOW
- ✚ Construct predictive parsing table
- ✚ Parse the string.

10. Construct predictive parsing table and parse the string NOT(true OR false)

$bexpr \rightarrow bexpr \text{ OR } bterm \mid bterm$

$bterm \rightarrow bterm \text{ AND } bfactor \mid bfactor$

$bfactor \rightarrow \text{NOT } bfactor \mid (bexpr) \mid \text{true} \mid \text{false}$

**Hints:**

- Find FIRST and FOLLOW and construct table and parse the string.

11. Construct CLR parsing table to parse the sentence  $id=id*id$  for the following grammar.

$S \rightarrow L=R \mid R$        $L \rightarrow *R \mid id$        $R \rightarrow L$

**Hints:**

- ❖ Find LR(1) items;
- ❖ Construct CLR table.
- ❖ Parse the string.

12. Construct operator precedence parsing table and parse the string.

$S \rightarrow a \mid \uparrow \mid ( T )$

$T \rightarrow T , S \mid S$

**Hints:**

- Compute LEADING and TRAILING
- Construct operator precedence table.
- **Parse the string.**

13. Parse the string (a,a) using SLR parsing table.

$S \rightarrow (L) \mid a$

$L \rightarrow L , S \mid S$

**Hints:**

- ◆ Find LR(0) items.
- ◆ Construct SLR parsing table.
- ◆ Parse the string.

14. Construct LALR parsing table for the grammar.

$E \rightarrow E+T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid id$

**Hints:**

- ▶ Find LR(1) items.
- ▶ Find same core items different second component .them merge it
- ▶ After merging construct LALR parsing table.

15. Write algorithms for SLR and CLR string parsing algorithm.

**Hints:**

- ▶ Brief introduction about SLR and CLR parsers.
- ▶ Algorithm –refer book

16. Explain different data structures available for symbol table.

**Hints:**

- ⊕ Lists
- ⊕ Self organizing list
- ⊕ Search tree.
- ⊕ Hash table.

17. Construct quadruples and triples and indirect triples for any 3-address statement.

**Hints:**

- Construct quadruples table
- Develop triples table
- Construct indirect triple table.

18. Write syntax directed translation scheme for generating three address code for Boolean expression using back patch method.

**Hints:**

- ◆ Write syntax directed translation schemes
- ◆ Explain with an example.

19. Explain syntax directed translation scheme of flow of control statements and give one example.

**Hints:**

- Write semantic rules for if, while, if...else statements
- Give an example.

20. Explain syntax directed translation scheme of procedure call statements and give one example

**Hints:**

- Write semantic rules for procedure call statements and give one example.

21. Explain in detail about loop optimization techniques.

**Hints:**

- ❖ Loop test replacement
- ❖ Dead code elimination
- ❖ Code motion
- ❖ Code movement

22. Explain about peep hole optimization.

**Hints:**

- ✦ Redundant instruction elimination
- ✦ Flow of control optimization
- ✦ Algebraic simplification
- ✦ Use of machine idioms.

23. Explain in detail about run time storage management.

**Hints:**

- ✖ Definition of run time storage management
- ✖ Static allocation
- ✖ Stack allocation

24. Write about labeling algorithm, heuristic ordering algorithm in code generator.

Hints:

- Labeling algorithm
- Heuristic ordering algorithm
- Code generation using the above algorithm
- Find evaluation order, and number of registers used using the same algorithms.

25. Explain in detail about register allocation and assignment.

**Hints:**

- Global register allocation
- Register assignment
- Number of registers selection based on register descriptor and address descriptor.