COMPILER DESIGN IMPORTANT QUESTIONS

Short Questions

UNIT-I

1. Write regular expression over alphabet {a, b, c} containing at least one ‘a’ and at least one ‘b’
2. What is input buffering? How is input buffering implemented?
3. Define Boot strapping.
4. What are the differences between a compiler and an interpreter?
5. What is the key difference between lexical analysis and parsing?
6. Why lexical and syntax analyzers are separated?
7. List the various error recovery strategies for a lexical analysis.
8. What is the role of compiler in bootstrapping operation?
9. Give the types of a language processing system.
10. What is a preprocessor? Mention its objectives.
11. What happens in Analysis and Synthesis phases of compilation?
12. List any 4 compilers and 2 interpreters you know.
13. What is the key difference between lexical analysis and parsing?
14. What is the purpose of Loader/Linker in language processing?
15. How semantic analyzer processes imperative statements?
16. Differentiate between token, lexeme and pattern with examples.

UNIT-II

1. What are left-most and right-most derivations?
2. What is recursive decent parsing?
3. What are the problems in top-down parsers?
4. Name the three techniques for constructing LR parsing table.
5. What are the actions performed by Shift reduce parser?
6. Differentiate between top-down parser and bottom-up parser.
7. Give the usage of look ahead symbol in LALR parsing.
8. Give the rules to find the first function.
9. List the properties of LR parser
10. Write context free grammar for polish notation of arithmetic expressions.
11. Construct parse tree and syntax tree for 4-6/3*5+7

12. Mention the types of LR parser.
13. Define LR(0) items with examples.
14. Give the specification of the YACC parser generator.
15. Construct the LR(0) items for the “dangling-else” grammar.
16. What are the draw backs of predictive parsing?
17. What are the actions performed by Shift reduce parser?
18. Differentiate between SLR, LALR and CLR parsers.
19. What is Context free grammar?
20. What are the actions performed by Shift reduce parser?
21. Describe in brief about types of LR parsers?
22. Eliminate immediate left recursion for the following grammar:
23. E->E+T | T
    T->T* F | F
    F-> (E) | id [3]
    List the rules for computing FOLLOW SET.
24. Define CLOSURE (I).
25. What is operator precedence grammar? Give an example.
26. What is significance of lookahead operator in LR parsing?

UNIT-III

1. What is the S – attributes and L– attributes?
2. What is activation record?
3. What is a symbol table?
4. What does a semantic analysis do?
5. What is type expression?
6. Define Type Equivalence?
7. How to check structural equivalence of two type expressions?
8. Define and write the differences between synthesized attributes and inherited attributes.
9. What are the benefits of intermediate code generation?
10. Apply translation scheme to generate three-address code a<b or c<d
11. Write the applications of SDTs.
12. What is Static Checking? List out some examples of static checks.
13. Define inherited and synthesized attributes.
14. What is three-address code? Give an example.
15. Draw the typical structure of an activation record.
16. What is syntax-directed definition
17. Give three-address code for the statement: do i = i + 1 ; while (a [i] < v);
18. What is an activation link? Give an example.
19. What is an annotated parse tree? Give an example.
20. Give directed acyclic graph for the expression: a + a * (b - c) + (b - c) * d.

21. What are the basic functions of the memory manager?
22. Write the three-address code for a while-do statement.

UNIT-IV

1. Explain the rules to construct a flow graph.
2. What is dead code?
3. Mention the issues in design of code generation.
4. What are the advantages and disadvantages of heap storage allocation strategies for records?
5. What is peephole? What peephole optimizations can be performed on code?
6. Explain about hashing.
7. What is a basic block?
8. What are the applications of DAG?
9. What are the advantages of stack storage allocation strategy?
10. Define Basic block. What are the rules for defining a basic block?
11. Define Basic Block?
12. How can you identify the leader in a Basic block?
13. Define basic block in a flow graph.
14. What is a DAG? Mention its applications

UNIT-V

1. What is dead code elimination and reduction in strength?
2. Define loop unrolling. Give an example.
3. What is meant by register descriptor and address descriptor?
4. How to allocate registers to instruction?
5. Generate a object code for following statements a = b + c; d = a+e [2]
6. Mention the properties that a code generator should possess.
7. Which graph is used for identifying the common sub expression in an expression?
8. What is meant register allocation and assignment?
9. What is common sub expression elimination?
10. Define Dead code elimination?
11. What is register allocation? Give a brief description.
12. Write a short note on Flow graph.
13. Write an algorithm for constructing a basic block.
14. Define various possible outputs of the code generator.
15. Construct DAG for the following basic block:

    T1=A+B

    T2=C+D

    T3=E – T2

    T4=T1– T3

16. Discuss about common sub expression elimination.
17. How do you calculate the cost of an instruction?
18. List out the common issues in the design of code generator
19. Write in detail about the sub-division of run-time memory.
20. Copy propagation leads to dead-code elimination, justify this with example
21. Define abstract Syntax tree.
22. Write about the sub-division of run-time memory.
23. List the characteristics of peephole optimization.
24. What is Static Checking? List out some examples of static checks.
25. Mention the issues in design of code generation.
26. Write the criteria for achieving machine independent code optimization
27. What are the advantages and disadvantages of heap storage allocation strategies for records?
28. Write the factors that affects the target code generation.
29. What is dead code?
30. Define a global common sub expression.
31. Write short notes on flow-of-control optimization

Long Questions

UNIT-I

1. What are program translators? Explain.
2. Explain the procedure to convert regular expression to Finite automata.
3. Explain various phases in the construction of compiler with a neat sketch. Explain each phase in detail with example x=(a+b)*(c+d)
4. What is the functionality of preprocessing and input buffering?
5. Explain compiler construction tools.
6. Explain and Construct from how convert from NFA to DFA with examples
7. Define Regular Expression? Explain about the Properties of Regular Expressions.
8. Describe the languages denoted by the following regular expressions:
   (i) (a|b)*a(a|b)(a|b.
   (ii) a*ba*ba*ba*
9. Design grammars for the following languages:
   (i) The set of all strings of 0s and 1s, such that every 0 is immediately followed by at least one 1.
   (ii) The set of all strings of 0s and 1s that are palindromes.
10. Define Compiler? Explain in brief about the syntax and semantic analysis of a compiler with an example?

11. Explain the concept of bootstrapping with example.
    Explain various error recovery strategies in lexical analysis
12. Consider the following Conditional statement:
    if (x > 3) then y = 5
    else y = 10;
    How does lexical analyzer help the above statement in process of compilation?
13. Describe the functionality of Scanner? Design a simple scanner for the postfix notation algorithm.
14. Explain basic functions of language translator, and describe various building blocks used to design a language translator?
15. What are the difference between pass and Phase of a compiler
16. What are the difference between Compiler and Interpreter
17. Explain in details the Role of the Lexical Analyzer with neat diagram
18. Write regular expressions for the following languages:
    i) All strings of lowercase letters that contain the five vowels in order.
    ii) All strings of lowercase letters in which the letters are in ascending lexicographic order.
    iii) All strings of a's and b's with an even number of a's and an odd number of b's.

UNIT-II

1. Define Context Free Grammar. Explain how it is suitable for parsing?
2. Explain problems in top down parsing Techniques
3. Describe difference between top-down and Bottom-up parsing Techniques
4. Compute FIRST and FOLLOW for the grammar: S -> S + S , S-> S * S , S-> a
5. Write an algorithm to find LR(0) items and give an example
6. Explain problems in top down parsing techniques and eliminate for the following grammar
   E->E+T | T
   T->T* F | F
   F-> (E) | id
   And construct predictive parsing table
7. What is the use of sentential forms in bottom-up parsing especially in shift-reduce operations? Explain with an example.

8. Find the SLR parsing table for the given grammar:
   E->E+E | E*E | (E) | id.
   And parse the sentence (a+b)*c
9. Construct CLR Parsing table for the grammar S ->L=R/R, L-> *R/id, R->L
10. Define Ambiguous Grammar? Check whether the grammar S->aAB , A->bC/cd, C->cd,

B->c/d Is Ambiguous or not?

11. Design LALR(1) parser for the following grammar:
    S -> aAd | bBd | aBc |bAc
    A -> e
    B -> e where a, b, c, d, e are terminals
12. Write the limitations of recursive descent parser with an example of grammar.
    Discuss the following:
    i) Structure of LR(0) parser
    ii) Action and Goto operations
    iii) Error handling in syntax analysis.
13. Construct LALR Parsing table for the grammar
    S→L=R/R
    L→*R/id
    R→L
14. Define Ambiguous Grammar? Check whether the grammar
    S→aAB,
    A→bC/cd,
    C→cd,
    B→c/d Is Ambiguous or not?
15. Construct SLR Parsing table for the grammar
    S→ (L)|a
    L→L,S|S
16. Discuss in brief about model of LR parser
17. Construct the predictive parser for the following grammar:
    S->(L)/a
    L->L,S/S
18. Find the SLR parsing table for the given grammar:
    E->E+E | E*E | (E) | id.
    And parse the sentence (a+b)*c.
19. Construct an LALR Parsing table for the following grammar:
    E-> E+T |T
    T-> T*F|F
    F->id
20. Find the LR (0) set of items for the following grammar. Describe state diagram and construct parse table of that
    S → CC
    C → cC |d
21. Write a procedure to construct LALR parsing table
22. Differentiate between LR(1), Canonical-LR and LALR parsing methods.
23. Describe about YACC.

UNIT-III

1. Give syntax directed translation scheme for simple desk circulator?
2. Describe in detail the syntax directed translation of case statements?
3. What is symbol table? Discuss various ways to organizing symbol table?
4. Describe in details Syntax-Directed Translation Schemes?
5. Write an SDT to convert infix to postfix expression.
6. Syntax directed translation scheme
   if(a>b)
   x=a+b
   else
   x=a-b
   Where **a** and **x** are of **real** and **b** of **int** type data.
7. Explain about how to construct the syntax tree in details?
8. Consider the following grammar
   $E \rightarrow E_1 + T$
   $E \rightarrow E_1 - T$
   $E \rightarrow T$
   $T \rightarrow T_1 * F$
   $T \rightarrow F$
   $F \rightarrow (E)$
   $F \rightarrow$ **id**
   $F \rightarrow$ num
   Construct L-Attributes for expression (3+4)*(5+6).
9. Explain the construction of syntax tree, Dependency Graph and Evaluation order for expressions **a-4+c**
10. Consider the following grammar
    S -> T L
    T -> int
    T -> float
    T-> real
    T -> double
    L -> L_1, id
    L -> id
    Construct L-Attributes for real id1, id2, id3
11. Explain the construction of syntax tree , Dependency Graph and Evaluation order for expressions int a, b
12. Construct annotated parse tree of the given input string int a,c
13. Construct annotated parse tree of the given input string 5-6*10
14. Given below the syntax-directed definition (SDD), construct the annotated parse tree for the input expression: “int a, b”.

15. Given below the syntax-directed definition (SDD), construct the annotated parse tree for the input expression: 5+3*4
16. For the grammar below construction of syntax tree, Dependency Graph and Evaluation order for 5+6.9
    E → E + T | T
    T → num . num | num
    Give an SDD to determine the type of each term T and expression E.
17. Below grammar generates binary numbers with a "decimal" point:
    S → L . L | L
    L → LB | B
    B → 0 | 1
18. Design an L-attributed SDD to compute S.val, the decimal-number value of an input string.
19. Describe in detail the syntax directed translation of case statements.
20. What is the role of type system in type checker? Write the syntax directed definition for type checker.
21. Applications of Syntax-Directed Translation
22. L->En, E->E+T/T, T-> (T/F)/F, F-> (E)/num for the given desktop calculator generate syntax directed translator scheme
23. Explain in details about Variants of Syntax Trees?
24. What is symbol table? Discuss various ways to organizing symbol table.
25. Generate intermediate code for the following code segment along with the required
    if(a>b)
     x=a+b
     else
     x=a-b Where a and x are of real and b of int type data
26. Translate the following expression:
    (a + b) * (c + d) + (a + b + c) into
    a) Quadruples b) Triples c) Indirect triples
27. Explain in brief about equivalence of type expressions.
28. Explain the role of type checking in error detection and recovery.
29. Write various semantic routines used to construct abstract syntax tree with an Example.
30. What is a three address code? What are its types? How it is implemented?
31. Give Three-Address Code and it's quadruple representation for the assignment:
    a = b * - c + b * - c ;
32. Translate the arithmetic expression a[i] = b*c - b*d into a syntax tree, quadruples and triples.
33. Give three-address code for the statement: do i = i + 1 ; while (a [i] < v);
34. Generate three-address code for the grammar below: (B is a Boolean expressing and S is a statement)
    S → if (B) S1
    S → if (B) S1 else S2

S → while (S) S1

35. Explain the translation scheme for Boolean expression using the back patching technique.
36. Explain the translation scheme for case statement with example
37. Give directed acyclic graph for the expression: a + a * (b - c) + (b - c) * d.

UNIT – IV

1. Explain about stack allocation space
2. The following C program computes Fibonacci numbers:

```
int f (int n) {
int t,s;
if (n < 2) return 1;
s = f(n-1);
t = f(n-2);
return s+t;
}
```

   Suppose that the activation record for f includes the following elements in order: return value, argument n, local s, and local t. Show the complete activation tree for the call f(5).
3. Describe about Access to Nonlocal Data on the Stack
4. How to access non-local data? Explain implication details with example
5. Describe about Heap Management allocation
6. Discuss the advantages and disadvantages of heap storage allocation strategy.
7. Introduction to Garbage Collection and Trace-Based Collection
8. What is code optimization? Explain about various levels and types of optimizations.
9. Describe the application of peephole? What kinds of peephole techniques can be used to perform machine-dependent optimizations?
10. Explain the following peephole optimization techniques:
    a) Elimination of Redundant Code
    b) Elimination of Unreachable Code
11. List out Issues in the Design of a Code Generator and explain in details
12. Explain various storage allocation strategies with its merits and demerits.
13. Explain how data flow equations are set up and solved for improving code.
14. Explain ad construct Basic Blocks and Flow Graphs of the given example:

```
prod=0;
i=1;
do
{
        prod=prod+a[i]*b[i];
        i=i+1;
```

```
}
 while (i<=10);
```

15. Explain ad construct Basic Blocks and Flow Graphs of the given example:

```
sum=0;
While(i<=10)
{
        sum=sum+a[i];
        i=i+1;
}
```

16. Explain ad construct Basic Blocks and Flow Graphs of the given example:

```
t1=a+b
t2= c-d
t3=t1*t2
x=t3
```

17. Define Flow Graph? Explain how a given program can be converted in to flow graph.
18. Construct flow graph for the three-address code equivalent of the below code:

```
for (i=0; i<n; i++)
      for (j=0; j<n; j++)
            c[i] [j] = 0.0;
for (i=0; i<n; i++)
      for (j=0; j<n; j++)
            for (k=0; k<n; k++)
                  c[i][j] = c[i][j] + a[i][k]*b[k][j];
```

19. Explain Optimization of Basic Blocks with suitable example with each
20. Explain in brief about function preserving transformations on basic blocks.
21. Explain in brief about Induction variable elimination.
22. llustrate loop optimization with suitable example.
23. Discuss about various transformations that are characteristic of peephole optimizations.
24. Explain A Simple Code Generator with suitable example
25. Describe Register Allocation and Assignment
26. Write about Dynamic Programming Code-Generation
27. Explain in details code generation for DAG
28. Explain in details **Gen**eric code generation algorithm
29. Generate the code for the following expression: x = (a + b) – ((c + d) – e). Also Compute its cost
30. Optimize the following code using various optimization techniques:

```
i=1; s=0;
for (i=1; i<=3; i++)
for (j=1;j<=3;j++)
```

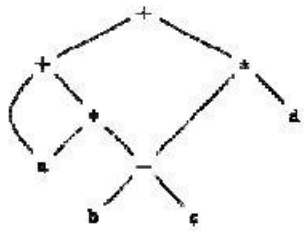
c[i][j]=c[i][j] + a[i][j] + b[i][j]

31. Give an example to show how DAG is used for register allocation.
32. Optimize the code given below, by eliminating common subexpressions, performing reduction in strength on induction variables, and eliminating all the induction variables.

```
dp = 0.
i = 0
L: tl = i*8
t2 = A[tl]
t3 = i*8
t4 = B[t3]
t5 = t2*t4
dp = dp+t5
i = i+1
if i<n goto L
```

UNIT-V

1. Distinguish between machine dependent and machine independent optimization.
2. What are the object code forms? Explain the issues in code generation.
3. Explain about machine dependent code optimization.
4. Describe the Data-Flow Analysis
5. Describe the Data-Flow equations
6. Explain in details about Loops in Flow Graphs with suitable examples
7. Explain the following with an example:
   a) Redundant sub expression elimination
   b) Frequency reduction
   c) Copy propagation.
8. Explain in brief about different Principal sources of optimization techniques with suitable examples.
9. Explain in brief about the DAG based local optimization
10. Explain the algebraic transformations of local machine independent optimization.
11. Discuss about the following:
    a) Copy Propagation
    b) Dead code Elimination and
    c) Code motion.
12. What is DAG? Construct DAG for the following Basic block.
    D: = B*C; E:= A+B; B:=B+C; A:=E-D;
13. Discuss and analyze all the allocation strategies in a run-time storage environment
14. Compute three-address code for the DAG below:



15. What does heap and stack areas of run-time memory store?
16. Explain how copy propagation can be done using data flow equation.
17. Explain in detail the procedure that eliminates global common sub expression.
18. Write the code generation for the d: =(a-b)+(a-c)+(a-c)
19. Write a code generation algorithm. Explain about the descriptor and function getreg(). Give an example.
20. Explain the following two classes of local machine independent transformations
    i) Structure preserving transformations
    ii) Algebraic transformations.
21. Explain the procedures for elimination of unreachable code and algebraic simplifications in Peephole Optimization
22. Explain about the method of computing transfer equations for reaching definitions
23. Construct an algorithm that will perform redundant-instruction elimination in a sliding peephole on target machine code.