1. Prove that, if ‘r’ is a regular expression, then there exists an NFA with e-moves for accepting the regular expression
2. Give the minimized DFA for the following expression (a/b)\*abb
3. Discuss in detail about the syntactic and semantic analysis phase of the compiler .
4. Draw the transition diagram for unsigned numbers
5. Explain lexical analysis and syntax analysis with examples.
6. Describe the structure of Lex programs.
7. Define regular expressions in brief.
8. Write the regular definitions for the following languages
9. All strings of lowercase letters that contain the five vowels in order.
10. All strings of digits with no repeated digits.
11. How to specify the tokens? Differentiate token, lexeme and pattern with suitable examples. And draw transition diagrams also
12. What is the role of regular expression in lexical analysis? Explain with example
13. What are the various phases of the compiler? Explain each phase in detail
14. State and prove the equivalence in between DFA and NFA
15. Prove that the regular sets are closed under union, concatenation and kleene closure
16. Explain the various phases of a compiler in detail. Also write down the output for the following expression after each phase a:=b\*c-d
17. Write a short note on the conversion of NFA and DFA
18. What is transition diagram? Write a transition diagram for relational operators
19. What are the various compiler construction tools? Explain in detail
20. What are the roles of various phases of a compiler? Explain in detail
21. Write a short note on Finite Automata and Regular Expressions

Unit 2

1. Explain in detail about the Operator precedence parser with suitable algorithms and example
2. Discuss in detail about predictive parsers with a suitable example
3. What are the error recovery strategies used in parser?
4. How the error recovery is performed in operator precedence parsing?Explain with an example.
5. Consider the context free grammar S SS+ | SS\* | a and the string aa+a\*

i) Give the leftmost derivation for the string.

ii) Give the rightmost derivation for the string.

1. Parse tree for the string.
2. Explain Recursive descent parsing technique with an example.
3. What is top down parsing? What are the problems in top down parsing? Explain each with suitable example
4. What is shift reduce parser? Consider the following grammar:

E→E+E,E→E\*E,E->(E),E→id

show the shift-reduce parser action for the following string id\*(id+id)

1. Construct the predictive parser for the following grammar:

S->(L)|1

L->L,S|S

1. Write the algorithm to construct LALR parsing table. Explain the steps carried out with an example
2. Explain how the stack implementation of shift reduce parsing is carried out. Write the steps involved with a grammar
3. Explain the Model of LR parser with neat diagram. Give the algorithm for LR parser
4. Construct the SLR parsing table for the given grammer

S →CC

C→cC/d

Unit 3

1. Write a short note on Quadruples and Triples with an illustration
2. Enumerate in detail about the role of postfix notation in three addresses code representation
3. Describe in detail about the syntax translation of assignment statements
4. Write a short note on the significance of indirect triples
5. How to construct canonical LR(1) parsing table? Construct canonical LR(1) parsing table for the grammar S CC CcC | d Show that the following grammar
6. S Aa | bAc | dc | bda A d is LALR(1) but not SLR(1).
7. Explain the model and algorithm for LR parser.
8. Write a Yacc program that tells whether its input is a palindrome.
9. Explain with a suitable example, the techniques used in YACC to resolve shift reduce and reduce-reduce conflicts
10. Show that the following grammar is not SLR(1)

S->AaAb|BbBa

A->ɛ

B-> ɛ

1. With neat sketch explain the structure of LR parser and the rules to compute LR item
2. Explain in detail about syntax directed translation with desk calculator as an example
3. Write a short note on the principle sources of code optimization
4. What do you mean by DAG? Explain the algorithm for constructing a DAG with an example 6
5. What do you mean by global data analysis? Explain
6. Write three-address code for the expression

(a+b)\*(c+d)+(a+b+c)

What are the different types of representation of three Address code? Explain in detail

1. Explain various types of code optimization techniques with example

Unit 4

1. Explain the concept of constant folding and copy propagation with an example
2. Discuss how compilers create code for procedure calls
3. Enumerate in detail about the machine model for a simple code generator
4. Write short notes on the register assignment during code generation
5. How to construct syntax trees for expressions using Syntax Directed Definitions (SDD)?
6. Write about S- attributed SDD and L- attributed SDD.
7. Define quadruples and triples. Translate the expression a=b\*-c+b\*-c into quadruples and triples.
8. What is back patching? How back patching is used to translate flow of control statements?
9. What is an intermediate code? Explain different types of intermediate codes forms and denote the subsequent statement in different forms:

W=(A+b) – (C+D) + (A+B+C)

1. What changes are to be made to the parser stack to enable implementation of L-attributed SDD during LL parsing? Explain using an example.
2. Write the syntax of Switch statement. Explain Syntax Directed Translation of Switch Statement
3. Using the SDD, translate and also draw the interpreted parse tree for the succeeding declaration z:x[i][j] + y[i][j]
4. Elaborate on the design issues that are considered in the design of a simple code generator
5. Describe in detail about Peephole optimization with an example
6. Discuss in detail about the generation of code from DAG
7. Write in detail about the role of register allocation and assignment in code generation
8. Discuss about Register allocation and assignment
9. Explain the code generation algorithm and generate the code

for the following:W=(A-B)+(A-C)+(A-C)

1. What are the different issues in the design of code generation?
2. Write about code generation machine model

Unit 5

1. Discuss in detail about the principles and merits and run time storage administration
2. Explain the basic concepts that lie behind the elimination of errors achieved based on symbol tables
3. Explain the following, i) Dead code elimination ii) Code Motion
4. Write about available expressions in brief.
5. Describe DAG (directed acyclic graph) representation of basic blocks. Construct the DAG for the following block,

d = b\*c

e = a+b

b = b+c

a = e-d

1. Explain the peephole optimization in brief.
2. Write a short note on Peephole optimization
3. Explain in brief i) Constant folding ii) Loop unrolling iii) Strength reduction
4. A)Explain available expressions with suitable example

B)Write a note on issues in code generation

1. Discuss in brief about the possible entries of a symbol table and state how it is created? Also mention the data structures that are used to implement.
2. What are the three entities that are needed to be managed at run time storage? Also explain the role of activation record in run time storage
3. Explain the organization of a symbol table
4. Explain runtime storage administration of symbol tables