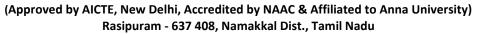


MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)





DEPARTMENT OF INFORMATION TECHNOLOGY

QUESTION BANK- ODD SEMESTER

Course Code / Title: 16ITD08 / Principles of Compiler Design

Year / Semester: III / V

UNIT I – INTRODUCTION TO AUTOMATA AND COMPILER PART-A (2 MARKS)

- **1.** What is a Complier?
- **2.** Write the cousins of compiler.
- 3. Define lexeme, Token and Pattern
- **4.** Differentiate Compiler, Assembler and Interpreter.
- **5.** State some compiler construction tools?
- **6.** What is the need for separating the analysis phase into lexical analysis and parsing?
- 7. Define Finite Automata.
- **8.** What is a sentinel? What is its usage?
- 9. Differentiate NFA and DFA.
- 10. Write Regular expression for identifier

PART-B QUESTIONS (16MARKS)

- **1.** Explain the phases of a compiler in detail with example.
- 2. (i)Briefly explain Compiler construction tools.(8)(ii)Explain the role Lexical Analyzer and Issues of Lexical Analyzer.(8)
- 3. Explain Briefly about Input buffering techniques.
- **4.** Give the minimized DFA for the following expression (a / b)*a (a / b).
- **5.** Write an algorithm for constructing a DFA from a regular expression. Describe with the RE= (a/b)*abb by using subset construction method.

UNIT II LEXICAL ANALYSIS PART-A (2 MARKS)

- **1.** What is the output of syntax analysis phase? What are the three general types of parsers for grammars?
- 2. List some different strategies that a parser can employ to recover from a syntactic error?
- **3.** Differentiate parse tree and Syntax tree?
- **4.** What is an ambiguous grammar? Give an example.

- **5.** When will you call a grammar as the left recursive one?
- **6.** Define left factoring. Give Left factor the following grammar: $S \rightarrow iEtS \mid iEtSeS \mid a$
- 7. What are the goals of error handler in a parser?
- **8.** Define Panic mode.
- **9.** What is yield or frontier?
- **10.** Write a regular definition to represent date in the following format: JAN-5th2014.

PART-B QUESTIONS (16 MARKS)

- 1. Give a neat sketch of role of parser and explain the error recovery strategies on parsing
- 2. Evaluate the parse tree for the input string w=cad using topdown parser $S \rightarrow cAd A \rightarrow ab \mid a$
- 3.Evaluate predictive parsing table for the grammar and find moves made by predictive parser on input id+id*id and find FIRST and FOLLOW $E \rightarrow E+T \mid T$; $T \rightarrow T*F \mid F$; $F \rightarrow (E) \mid id$
- 4.Write down the algorithm to eliminate left-recursion and left-factoring and apply both to the following grammar $E \to E + T \mid E T \mid T$; $T \to a \mid b \mid$ (E) Construct predictive parsing table for the above grammar and parse(a+b)-a
- 5. Check whether the following grammar is a LL (1) grammar S->iEtS / iEtSeS /a, E->b

UNIT III SYNTAX ANALYSIS PART-A (2 MARKS)

- 1. Differentiate Top down and Bottom up parsing.
- 2. What is Shift-Reduce parsing?
- 3. Define handle. What do you mean by handle pruning?
- 4. Define LR (0) items.
- 5. What do you mean by viable prefixes?
- 6. What is meant by an operator grammar? Give an example
- 7. List the disadvantages of operator precedence parsing?
- 8. State error recovery in operator-Precedence Parsing.
- 9. Why LR parsing is attractive one grammar
- 10. What are kernel and non kernel items?

PART-B QUESTIONS (16 MARKS)

```
1. Show SLR parsing table for the following grammar
```

 $S \rightarrow Aa \mid bAc \mid Bc \mid bBa$

 $A \rightarrow d$

 $B \rightarrow d$

And parse the sentence "bdc" and "dd".

2. Find the LALR for the given grammar and parse the sentence (a + b) * c

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

3. Give an LALR parser for the following grammar and parse the input id = id

 $S \rightarrow L = R \mid R$

 $L \rightarrow R \mid id$

 $R \rightarrow L$

4. Show a predictive parser for the following grammar

 $S \rightarrow (L) \mid a$

 $L \rightarrow L,S \mid S$

5. Give SLR parsing table for the following grammar

 $E \rightarrow E + T \mid T$ $T \rightarrow T F \mid F$

 $F \rightarrow F * |a| b$

UNIT IV INTERMEDIATE CODE GENERATION PART-A (2 MARKS)

- **1.** What are the benefits of using machine-independent intermediate form?
- **2.** List the three kinds of intermediate representation.
- **3.** How can you generate three-address code?
- **4.** What is a syntax tree? Draw the syntax tree for the assignment statement a := b * -c + b * -c.
- **5.** Define three-address code.
- **6.** What is called an abstract or syntax tree?
- 7. Draw the DAG for a := b * -c + b * -c
- **8.** Define back patching.
- **9.** List the following functions:
 - i) makelist(i) ii) merge(p1,p2) iii) backpatch(p,i)
- 10. Define back patching

PART-B QUESTIONS (16 MARKS)

- 1. What is three address codes. Mention its types. How would you implement the three
- 2. Briefly explain about Boolean expression.
- 3. Describe in detail the syntax directed translation of case statements
- 4. How Back patching can be used the generate code for Boolean expressions and flow of control statements.
- 5. Explain in detail the translation of assignment statements.

UNIT -V CODE OPTIMIZATION AND GENERATION PART-A (2 MARKS)

- 1. What are basic blocks?
- 2. What is a flow graph?
- 3. Mention the applications of DAGs.
- 4. What are the advantages and disadvantages of register allocation and assignments?
- 5. List the types of addressing modes.
- 6. What is input to code generator?
- 7. What are the primary structure preserving transformations on basic blocks?
- 8. Define DAG.
- 9. What are the issues in the design of code generators?
- 10. What is meant by Dead Code?

PART-B QUESTIONS (16 MARKS)

- 1. Explain Peephole optimization.
- 2. Describe in detail about optimization of basic blocks with example.
- 3. Explain the Principal sources of optimization.
- 4. Describe issues in design of code generator.
- 5. Wrtie a simple code generator algorithm.