Total Pages—3

PG/IIS/MCA/204/25

MCA 2nd Semester Examination, 2025

MCA

(*Compiler Design*)

PAPER – MCA-204

Full Marks : 100

Time : 3 hours

Answer **all** questions

The figures in the right hand margin indicate marks

Candidates are required to give their answers in their own words as far as practicable

GROUP – A

Answer any **five** questions : 2 × 5

1. Define Ambiguous grammar. Give one example.

2. Differentiate between Top Down and Bottom up Parsing.

3. Write the function of code optimization phase of a compiler.

4. Write down the function of Symbol Table.

5. What is LR(1) grammar ?

6. Explain the model of a non-recursive predictive parser with a diagram.

7. What are the functions of error handler ?

8. Define annotated parse tree.

GROUP – B

Answer any **four** questions : 15 × 4

9. (*a*) Let *G* be a Context Free Grammar given below :

S->aB bA, A->a|aS bAA, B->b|bS|aBB

Drive the string aaabbabbba using the above grammar (using Left Most Derivation and Right most Derivation). 3 + 3

(*b*) Remove left recursion of the following Grammar : 5
A-> ABd|Aa|a, B-> Be|b

(*c*) Remove left factoring in the following grammar :
A-> ad | a | ab | abc | b 4

10. (*a*) Calculate FIRST and FOLLOW for the following grammar.
A->aCDq | aBg | ε
D->d ε
B->e | ε
C-> Ct | p | ε | BD rAb 3 + 3

(*b*) Construct predictive parsing table for the given grammar
D-> L:T
L-> L, id | id
T-> int 6

(*c*) "No left recursive grammar can be LL(1)"–Justify. 3

11. Consider the following grammar :
S-> cA|ccB
A->cA | a
B-> ccB | b

(*a*) Compute item sets for SLR (1) parsing table.

(*b*) Construct SLR (1) parsing table for the grammar.

(*c*) Show how the SLR parser parses the string 'ccccb' 5 + 5 + 5

12. (*a*) What is intermediate code and write the two benefits of intermediate code generation.

(*b*) Draw the syntax tree and DAG for the following expression :
(a*b) + (c-d)*(a*b)+b.

(*c*) Write quardruples, triples and indirect triples for the expression :
-(a*b)+(c+d)-(a+b+c+d) (2−2)−(2+3)+(2+2−2)

(*Turn Over*)

13. (*a*) What are basic blocks ? Write the algorithm for partitioning into Blocks.

(*b*) What is flow graph ? Give one example.

(*c*) What is common sub-expression and how to eliminate it ? Explain with example.

(*d*) Define Dead-code elimination with example. (2 + 3)+(2 + 2) + 3 + 3

14. (*a*) What is LL(1) Grammar ? How we can check that any grammar can be LL(1) ?

(*b*) Differences between SLR, CLR, LALR parsers.

(*c*) Construct CLR Parsing table for the given grammar
S-> (L) | a
L-> L,S | S
5 + 5 + 5

15. Construct LALR table for the following Gramar.
S→P
P→aAd/bBd/aBc/bAc, A → ∈, B → ∈
and parse the sentence be. 10 + 5

16. Write short note (any *three*) : 5 + 5 + 5

(*a*) Bootstrapping

(*b*) Handle pruning

(*c*) Dependency graph

(*d*) Symbol Table

(*e*) Phases of a compiler.

[Internal Assessment — 30 Marks]