Reg No.:_______________ Name:__________________________

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

B.Tech S6 (S,FE) (FT/WP/PT) Examination December 2025 (2019 Scheme)

Course Code: CST302

Course Name: COMPILER DESIGN

Max. Marks: 100 Duration: 3 Hours

PART A

Answer all questions, each carries 3 marks. Marks

1 What is the role of regular definition in lexical analysis? (3)

2 Find pattern for identifier and unsigned numbers. (3)

3 Check whether the given grammar is ambiguous or not. (3)

$S \rightarrow AA,\ A \rightarrow aA,\ A \rightarrow b$

4 Explain left factoring with an example. (3)

5 Illustrate operator grammar with an example. (3)

6 Which is the most efficient LR parser? Justify. (3)

7 Describe L attributed definitions. (3)

8 Identify three-address code for array access. (3)

9 Explain compile time evaluation. (3)

10 What is the role peephole optimization? (3)

PART B

Answer one full question from each module, each carries 14 marks.

Module I

11 a) Write regular definition and corresponding finite automata for identifier, unsigned numbers and arithmetic operators. (7)

b) Write code according to above design(Question no.11a.) for identifier, unsigned numbers and arithmetic operators. (7)

OR

12 a) Explain LEX program structure. (7)

b) Write LEX program for identifying tokens such as identifiers, arithmetic operators and relational operators. (7)

Module II

13 a) Explain limitations of top-down parsing. (7)

b) Illustrate left most and right most derivation with an example. (7)

OR

14 a) Write and explain methods for finding FIRST and FOLLOW. (5)

b) Construct predictive parsing table for given grammar. (9)

E→ E + T | E – T | T
T→ T * F | T / F | F
F→ (E) | id

Module III

15 a) Perform operator precedence parsing for input string "id_1 + id_2 * id_3" according the given grammar. (7)

S→ S + S | S * S | id

b) Explain item set construction method in SLR. (7)

OR

16 Construct LALR parsing table for the given grammar. (14)

S → L = R | R
L → * R | id
R → L

Module IV

17 a) Write translation scheme for the evaluation of arithmetic expressions. Also perform evaluation of a valid expression. (7)

b) Explain type checking of arithmetic operations, array access and function call. (7)

OR

18 a) Draw syntax tree representation for the given expression and write corresponding 3-address code. (5)

a * (a + b) + c

b) Construct quadruple, triple and indirect triple tables for the given expression. (9)

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

Module V

19 a) Differentiate machine dependent and machine independent optimizations. (5)

b) Perform common sub expression elimination globally for the given code segment. (9)

```
1) t1 = a + b
2) t2 = t1 * c
3) t3 = a + b
4) t4 = t2+ t3
5) x= t4
6) t5 = a+1
7) a= t5
8) t6= a+ b
9) y = t6
10) t7 = a + b
11) c = t7
12) If c < 100 goto 10
```

OR

20 a) Explain issues in the design of a code generator. (6)

b) Convert to optimized three-address code and write machine code for the given code segment. (8)

x = a* (b - c) + d / e

b = b + 1

y = (b – c) * d / e

****