Reg No.:_____ Name:_____

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

B.Tech Degree S6 (S, FE) / S6 (PT) (S) Examination January 2024 (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 Describe input buffering scheme in lexical analyser. (3)

2 Find regular definition for tokens in the generated strings of the following grammar. (3)

S→ ***if*** E ***then*** S | ***if*** E ***then*** S ***else*** S | ϵ

E → T ***relop*** T | T

T → ***id*** | ***num***

3 What is left recursion? Explain the rule to eliminate left recursion. (3)

4 Describe sets-of-items construction algorithm for SLR parser. (3)

5 Explain synthesized attributes and inherited attributes. (3)

6 Check whether the given translation scheme is L-attributed or not? Justify (3)

A → B { B.i = C.s} C { C.i = A.i}

7 What are the advantages of indirect triple compared to triple. (3)

8 Write three-address code for a ***while*** loop by choosing a suitable example. (3)

9 Differentiate local and global optimizations. (3)

10 Illustrate the role of register descriptor and address descriptor in code generation phase. (3)

PART B

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

Module I

11 a) Draw transition diagram for designing lexical analyser for the tokens such as arithmetical operator, relational operator, identifier and unsigned number. (7)

b) Write code for the lexical analyser for the above design. (7)

OR

12 a) Explain LEX program structure with a sample program. (7)

b) Explain YACC program structure with a sample program. (7)

Module II

13 a) Illustrate design of recursive descent parser with a suitable grammar. (7)

b) Explain any two drawbacks of recursive descent parser and its solutions. (7)

OR

14 a) Write and explain algorithm for constructing LL(1) predictive parsing table. (7)

b) Construct predictive parsing table for given grammar. (7)

$S \rightarrow SAaAb \mid SBbBa \mid \epsilon$
$A \rightarrow t$
$B \rightarrow c$

Module III

15 a) Illustrate actions according to operator precedence parser for the input id_1+id_2* id_3 based on the given grammar. (7)

$E \rightarrow E + E / E * E / (E) / id$

b) Explain how to construct an operator grammar without ambiguity. (7)

OR

16 Construct Canonical LR (1) parsing table and perform parsing actions for a valid input according to given grammar. (14)

$S \rightarrow A\ a \mid b\ A\ c \mid c\ A\ b$
$A \rightarrow a$

Module IV

17 a) Write syntax-directed definition for simple type declaration involving basic types such as int, float and char (assume syntax for C-programming language). (7)

b) Write SDD for generating syntax tree for arithmetic operations. (7)

OR

18 a) Draw DAG representation for the given statement. (5)

$s = (a + b) * (b + c) + (a + b)$

b) Construct quadruple, triple and indirect triple tables for above DAG representation. (9)

Module V

19 a) Explain any three code optimization transformation. (6)

b) Perform common sub-expression elimination for the following three-address code and represent it as a quadruple table. (8)

```
t1 = a + b
x  = t1
t2 = a + b
t3 = t2 + c
b  = t2
t4 = a + b
y  = t4
```

OR

20 a) Neatly explain code generation algorithm and *getreg* function. (7)

b) Convert to three-address code and write machine code for given statement. (7)

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

****