BTS-VI(SS)-01.19-0009 Reg. No. | | | | | | | | |

A

B.Tech. Degree VI Semester Special Supplementary Examination January 2019

CS/IT 15-1602 COMPILER CONSTRUCTION
(2015 Scheme)

Time: 3 Hours Maximum Marks: 60

PART A
(Answer *ALL* questions)

($10 \times 2 = 20$)

I. (a) What is the role of transition diagram in the construction of lexical analyzer? Draw and explain the transition diagram for an identifier.
(b) Distinguish between tokens, patterns and lexemes using examples.
(c) What is the role of parser in compiler design?
(d) Explain the process of handle pruning with an example.
(e) Distinguish between left recursion and left factoring with examples.
(f) What is dependency graph? Draw the dependency graph for the string int id1, id2, id3.
(g) Distinguish between S-attributed and L-attributed definitions with examples.
(h) What are the data structures used for the implementation of a symbol table? Explain.
(i) What are directed acyclic graphs? Give an example.
(j) Briefly discuss any four issues in the design of a code generator.

PART B

($4 \times 10 = 40$)

II. (a) Explain the different phases of a compiler with a block diagram. (7)
(b) Briefly write about lex tool in lexical analyzer design. (3)

OR

III. (a) What is the main concern related to lexical analyzer in compiler design? (2)
(b) Also discuss in detail how this problem is solved in lexical analyzer using different methods. (8)

IV. (a) What is operator grammar? Discuss operator precedence parsing algorithm. (6)
(b) Explain how validity of the string id+id*id is verified using operator precedence parsing algorithm. (4)

OR

V. (a) Explain LR parsing algorithm. (4)
(b) Write the canonical collection of sets of LR (0) items and also draw the DFA for the following grammar: (6)

$E --> E + T / T$

$T --> T * F / F$

$F --> (E) / id$

VI. Discuss how bottom-up evaluation of L-attributed definition takes place with suitable example. (10)

OR

VII. Discuss in detail static allocation and stack allocation. (10)

VIII. (a) What is the advantage of generating intermediate code? (2)
(b) Explain different methods to represent intermediate code for the expression a = b * -c + b * -c. (8)

OR

IX. What are the principal sources of optimization in a code? Illustrate with suitable examples. (10)

***