BTS–VI(SS)–07.19–0945 Reg. No. | | | | | | | | |

C

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

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

Time: 3 Hours Maximum Marks: 60

PART A
(Answer *ALL* questions)

(10 × 2 = 20)

I. (a) How tokens are specified in lexical analyzer? Discuss with an example.
(b) What are the reasons for separating the analysis phase of compiler in to lexical analyzer and syntax analyzer?
(c) What is the role of Parser in compiler design?
(d) What are the drawbacks of top down parsing? Discuss how it can be avoided.
(e) Explain the algorithm for operator precedence parsing.
(f) What is dependency graph? Draw the dependency graph for the string real id1,id2.
(g) Distinguish between synthesized attributes and inherited attributes with examples.
(h) Discuss about specification of simple type checker with an example.
(i) Distinguish between syntax tree and DAG with suitable examples.
(j) Discuss how declaration statements can be translated in to an intermediate form.

PART B

(4 × 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 are the different approaches for the implementation of lexical analyzer? (2)
(b) Explain about the different techniques for improving the speed of lexical analyzer. (8)

IV. (a) Write the nonrecusive predictive parsing algorithm. (4)
(b) What is LL(1) grammar? Check whether the following grammar is LL(1) grammar :- (6)

S --> iEtS / iEtSeS / a

E --> b

Also draw the predictive parsing table.

OR

V. (a) Explain LR Parsing algorithm. (4)
(b) Construct CLR parsing table for the grammar :- (6)

S --> AA

A --> aA / b

(P.T.O.)

VI.		Discuss how top-down translation of L-attributed definition takes place with suitable example.	(10)
		OR	
VII.		Discuss in detail about different storage allocation strategies.	(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)

***