Roll No. .....

MCA (8-9)/D-14 **1041 3**

SYSTEM PROGRAMMING AND COMPILER CONSTRUCTION

Paper-MCA-504

Time Allowed 1 3 Hours] [Maximum Marks : 80

Note: Attempt five questions in all, selecting at least one question from each Unit. Q. No. 1 is compulsory.

Compulsory Question

1. (a) What is Dangling-Else Ambiguity?
(b) What is Reentrant code?
(c) Define System software.
(d) Discuss LEX in brief.
(e) Write a note on shift-reduce conflict in LR parsing.
(f) What is Pseudo 0p~code?
(g) What is common sub-expression elimination? Use suitable example.
(h) Explain the Conversion process of NFA to DFA. 8x3=24

UNIT-I

2. (a) What are the basic functions performed by a Loader? Discuss. 7
(b) The process of fixing up a few forward reference should involve less overhead than making a complete second pass of the Source Program. Why d0n't all assemblers use the one-pass technique for efficiency? Discuss. 7

3. (a) What do you understand by recursive and nested macro calls? Explain recursive macro expansion. Use suitable examples. 7
(b) What do you understand by Relocatable programs? What is self-relocating program? Write a node on relocating loader. 7

UNIT-II

4. (a) What is the difference between context-sensitive and context-free Grammar? Explain. 7
(b) What is an Ambiguous Grammar? Explain using suitable example. 7

5. (a) Write production for decimal constant with or without a Fractional part. 7
(b) What language does the following Grammar generate? Justify your answer.
S->SS+ |SS* 1 a. 7

UNIT—III

6. What is the difference between top-down and bottom- up parsing? Explain the operator precedence parsing using suitable examples. 14

7. What do you understand by recursive descent parsing? Construct recursive-descent parsers for the following Grammars : S ~> aSb | ab. 14

UNIT-IV

8. (a) What are the issues in the Design of a Code Generator? 7
(b) Discuss the use of control flow and Data flow analysis in Optimizations. 7

9. Construct the Directed A'cyclic Graph for the following basic block : 14
a[i] = b, *_p = c, d = afi], e = *p = a[i].
The assumptions are :
(a) p can point anywhere
(b) p can point only to b or d.