This question paper contains 4 printed pages.

6134 *Your Roll No. . . . . .*

MCA / IV Sem.

J

MCA 401 - COMPILER DESIGN
(Admissions of 2007 and onwards)

Time 3 hours *Maximum Marks 60*

(Write your Roll No on the top immediately on receipt of this question paper)

Attempt all questions.
Parts of a question must be answered together.

1. a) Write a regular expression that generates the same language as the following grammer

$A \rightarrow aA \mid B \mid \in$

$B \rightarrow bB \mid A$ *02*

b) Write a context free grammar that generates the same language as the following regular expression

$(a \mid c \mid ba \mid bc)^* b$ *02*

c) What is meant by peehole optimization? *02*

2 Consider the grammar

lexp-seq → lexp-seq lexp \ lexp

lexp → **atom** \ list

atom → **number** \ identifier

list → (lex-seq)

a) Remove left recursion

P.T.O

b) Construct FIRST and FOLLOW sets for the non - terminals of the resulting grammar.

c) Construct the LL(1) passing table for the resulting grammar

d) Show the actions of the corresponding LL(1) parser, given the input string

(b (2)) *12*

3 Consider the grammar

$S' \rightarrow S$

$S \rightarrow (S)\ S/E$

a) Construct the DFA to be used in SLR parsing

b) What are the valid states for the viable prefix '((' ? Justify.

c) Construct the initial state for the LR (1) parsing.

d) Determine the valid states for the viable prefix '((' *12*

4 a) Consider the attribute grammar :

Grammar Rule	**Semantic Rule**
decl → type var-list	var-list dtype = type.dtype
type → **int**	type.dtype = integer
type → **float**	type dtype = real
varlist$_1$ → **id**, var-list$_2$	id.dtype = var-list$_1$ dtype
	var-list$_2$ dtype = var-list$_1$ dtype
var-list → **id**	id dtype = var-list.dtype

Show that if the attribute type dtype is kept on the value stack during an LR parser, than this value cannot be found at a fixed position in the stack when reductions by the rule var-list → id occur *04*

b) Show that the grammar given below is not LR(1) Can it be LR (K) for some suitable k. Justify.

$B \rightarrow ABb / a$

$A \rightarrow \in$ *04*

5 Consider the following grammar for type expression

var-decls → var-decls; vardecl / var-decl

var-decls → **id**, type-exp

type-exp → simple-type / structured - type

simple-type → **int** / **bool** / **real** / **char** / void

structured - type → **array** [**num**] **of** type - exp /
record var - decls **end** /
union var - decls **end**

Write pseudocode for checking type equivalence. *06*

6 Write pseudocode for code generation for control statements described by the grammar :

stmt → if - stmt / while - stmt / break / other

if - stmt → **if** - (exp) stmt / if (exp) stmt **else** stmt

while - stmt → **while** (exp) stmt

exp → **true** \ false *06*

7 a) Consider the following grammar rule

$S \rightarrow$ **while** (C) S_1

Outline the implementation of synthesize attributes as well as inherited attributes for the above grammar in LR parsing. You may make necessary modification to grammar without changing the language. *04*

b) How does yacc resolve reduce / reduce and shift / reduce conflicts. *02*

P.T.O.

c) Consider the following grammar along with semantic rules:

Production	Semantic Rule
$D \rightarrow TL$	L.int = T.type
$T \rightarrow$ **int**	T type = integer
$T \rightarrow$ float	T type = float
$L \rightarrow L_1$, id	L_1. int = L.int
	addtype (id entry, L.int)
$L \rightarrow$ id	addtype (id.entry, L.int)

Give a annotated parse tree for the following expression
int a, b,c. *04*