Nirma University

Institute of Technology

Semester End Examination (IR), December - 2016
B. Tech. in Computer Engineering, Semester-VII
IT794 Compiler Construction

RollNo/ Exam No | Supervisor's Initial with date

Time: 3 Hours | Max Marks: 100

Instructions: 1. Attempt all questions. Figures to right indicate full marks.
2. Section wise separate answer book to be used.
3. Assume additional information if required.

SECTION – I

Q-1 Answer the following questions: **[18]**

A) Explain Why can't left LL(1) grammar be left recursive? **[3]**

B) New languages 'H' and 'HS' is designed for distinct HPC systems (HPC1, HPC2, HPC3, HPC4). Compilers are to be designed for these languages. Advisor 1 suggest to assign one compiler design to one developer team. Advisor 2 proposes to divide compiler tasks into 2 pass: front-end phases and back-end phases, and distribute each pass to distinct developer teams. Which proposal do you prefer? Justify. **[3]**

C) Write Recursive descent parser for the following parse table **[4]**

	(	[	)	]	$
S	TS	[S]S	)S	ε	ε
T	(X)				
X	TX	[X]X	ε	ε	

D) Write a regular expression to identify email address. Construct DFA directly from its augmented regular expression. **[4]**

E) Is Error handling required at each phase of compiler? Justify. **[4]**

Q-2 Answer the following questions: **[16]**

A) Is the following grammar LL(1)? Justify using LL(1) parsing table. Show derivation of the string "class classA implements interfaceB". Lexer recognize classA and interfaceB as identifier. **[8]**

$C \rightarrow P\ F$ *class identifier* $X\ Y$
$P \rightarrow$ *public* $\mid \varepsilon$
$F \rightarrow$ *final* $\mid \varepsilon$
$X \rightarrow$ *extends identifier* $\mid \varepsilon$
$Y \rightarrow$ *implements* $I \mid \varepsilon$
$I \rightarrow$ *identifier*, $I \mid$ *identifier*

OR

A) Construct operator precedence matrix, and operator precedence function matrix for operators "+ - * / ( )", token ID, and delimiter $. Give comparison of these matrices for operator precedence parsing. **[8]**

B) Eliminate Left-Recursion (if exist) from the following grammar: **[4]**

$S \rightarrow Aa \mid b$
$A \rightarrow Ac \mid Sd \mid Bf$
$B \rightarrow Sa \mid Ab \mid d$

C) Analyze syntax of "ID + ID * ID" using operator precedence parsing for grammar : $E \rightarrow E + E \mid E * E \mid ID$ **[4]**

Q-3 Answer the following questions: **[16]**

A) Show that the following grammar is LALR(1) but not SLR(1). **[8]**

$S \rightarrow Aa \mid bAc \mid dc \mid bda$
$A \rightarrow d$

B) Trace LR parsing for input string "(( ) ( ))" using given parsing table for a grammar containing production rules 1) $S \rightarrow (S)S$ 2) $S \rightarrow null$ **[4]**

	S	(	)	$
0	1	s2	r2	r2
1				acc
2	3	s2	r2	r2
3			s4	
4	5	s2	r1	r1

OR

B) LR parsing table need more space than LL parsing table. Describe an approach to reduce its space complexity. **[4]**

C) Give an example of grammar, which is LL(1) but not LR(0) **[4]**

SECTION – II

Q-4 Answer the following questions: **[18]**

A) Rewrite following Grammar G to avoid ambiguity considering given precedence, and associatively of symbols. **[4]**

$G = \{S \rightarrow S + S \mid S - S \mid S*S \mid S/S \mid S\hat{}S \mid ID=S \mid NUM \mid ID \mid (S)\}$

Symbols	Precedence	Associatively
NUM, ID, (	1 (highest)	Left to Right
^	2	Right to Left
/, *	3	Left to Right
+, –	4	Left to Right
=	5 (lowest)	Right to Left

B) State whether the following statements are true or false. Justify your answer **[6]**

i. Every L-attributed definition is S-attributed definition.
ii. A non-left recursive and left-factored grammar may not be LL(1).
iii. SLR(1) parser has same number of states as LALR(1) parser

C) Trace code generation and register allocation using register and address descriptor for following IR code: **[8]**

```
t1 := a + b;
t2:= a - c ;
t3: = t1 + t2;
a:= t3
```

Q-5 Answer the following questions: **[16]**

A) What is advantage of register allocation using Graph coloring? Explain using register allocation for flow graph given in Figure 1. **[10]**

B) Design a Syntax Directed definition to identify undeclared and re-declared variables. **[6]**

OR

B) Considering variable usage count, which variables will get register in flow graph of Figure 2? (assume register bank size is 3) **[6]**

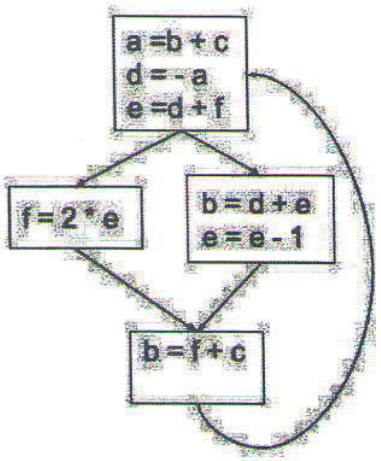

Figure 1

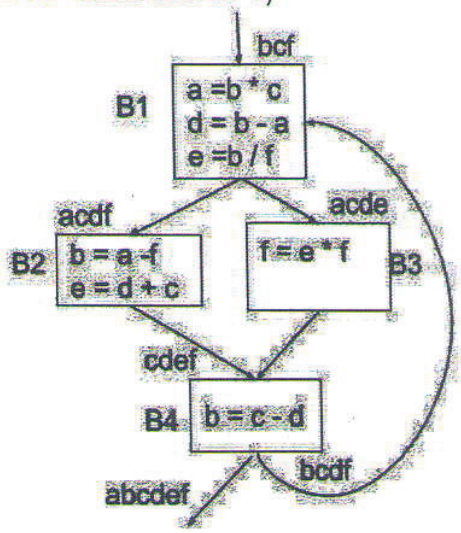

Figure 2

Q-6 Answer the following questions: **[16]**

A) Translate following 'C' code fragment to IR. Convert it into flow graph. **[6]**

```
while(a<b) {
  if(c<d) x=a[i]+2;
  else  x=a[i]*3;}
```

OR

A) Explain any four code optimization approaches. **[6]**

B) Write top-down evaluation code for following SDD: **[6]**

$A \rightarrow B$	*{print(B.n0), print(B.n1)}*
$B \rightarrow 0\ B_1$	*{B.n0=B_1.n0+1, B.n1=B_1.n1}*
$B \rightarrow 1\ B_1$	*{B.n0=B_1.n0, B.n1=B_1.n1+1}*
$B \rightarrow \varepsilon$	*{B.n0=0, B.n1=0}*

C) Explain shift/reduce conflict, reduce/reduce conflict using proper example. **[4]**