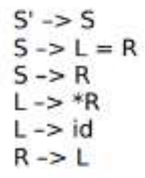
Compiler Design CSE-3151

(FISAC-2) -Take Home Assignment

1. The following is an ambiguous grammar:

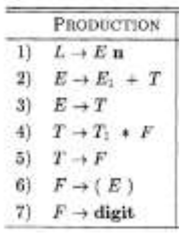
   T→PT | q
   P→TP | r
   Construct for this grammar its collection of sets of LR (0) items. If we try to build an LR-parsing table, there are certain conflicting actions. What are they? Suppose we tried to use the parsing table by non-deterministically choosing a possible action whenever there is a conflict. Show all the possible sequences of actions on input "qrqr".

2. Design a tiny grammar that contains left recursion, and use it to demonstrate that left recursion is not a problem for LR parsing. Then show a small example comparing growth of the LR parse stack with left recursion versus right recursion.
3. Construct an LR (0) automaton for the given grammar where S' is the augmented grammar symbol.



```
S' -> S
S -> L = R
S -> R
L -> *R
L -> id
R -> L
```

4. Show that the grammar given in Q3. Is not SLR (1) but CLR (1) by constructing respective parse tables.
5. Illustrate using an example, how does Shift Reduce parsers resolve conflicts for an expression grammar using precedence an associativity rule.
6. Write the general structure of LEX program in detail by clearly describing various LEX functions and LEX variables. Also write a LEX program to various generate tokens for the C grammar.
7. Translate the given arithmetic expression into a syntax tree, quadruple, triple and indirect triple. "a + - (b + c)".
8. For the grammar given below, construct the annotated parse tree for the expression: "(3+4)*(5+6)n" and write the semantic rules.



	Production
1)	$L \rightarrow E$ **n**
2)	$E \rightarrow E_1 + T$
3)	$E \rightarrow T$
4)	$T \rightarrow T_1 * F$
5)	$T \rightarrow F$
6)	$F \rightarrow (E)$
7)	$F \rightarrow$ **digit**

9. Construct the syntax tree and the corresponding DAG for the following expressions.
   i. a+b+a+b
   ii. a+b+(a+b)
   iii. ((x+y)-((x+y)*(x-y)))+((x+y)*(x-y))
10. Discuss the various Peephole Optimization techniques in compiler design with an example for each.