LAB SHEET 7

SEMANTIC ANALYSIS

1. Try the following program for evaluating expressions. (The Semantic actions are included in each production). Include the necessary modifications in the program as and when required.
You are supposed to write the main program and the input file.

Jlex File

```
import java_cup.runtime.Symbol;
import java_cup.runtime.Scanner;
%%
%cup
%%
"+" {System.out.println("LA "+yytext());return new Symbol(sym.PLUS);}
"-" {System.out.println("LA "+yytext());return new Symbol(sym.MINUS);}
"*" {System.out.println("LA "+yytext());return new Symbol(sym.TIMES);}
"/" {System.out.println("LA "+yytext());return new Symbol(sym.DIVIDE);}
"%" {System.out.println("LA "+yytext());return new Symbol(sym.MOD);}
";" {System.out.println("LA "+yytext());return new Symbol(sym.SEMI);}
"(" {System.out.println("LA "+yytext());return new Symbol(sym.LPAREN);}
")" {System.out.println("LA "+yytext());return new Symbol(sym.RPAREN);}
[0-9]+ {System.out.println("LA "+yytext());return new Symbol(sym.NUMBER ,new
Integer(yytext()));}
```

Cup File

```
import java_cup.runtime.*;
import java.util.*;

init with {: :};
scan with {: return getScanner().next_token(); :};
terminal          SEMI, PLUS, MINUS, TIMES, DIVIDE, MOD, NL;
terminal          UMINUS, LPAREN, RPAREN;
terminal Integer   NUMBER;

non terminal            st, expr_list, expr_part;
non terminal Integer    expr;

precedence left PLUS, MINUS;
precedence left TIMES, DIVIDE, MOD;
precedence left UMINUS;
expr_list ::= expr_list expr_part
                |
             expr_part
                ;

expr_part ::= expr:e
                {: System.out.println("= " + e); :}
             SEMI
                ;
```

```
expr      ::= expr:e1 PLUS expr:e2
                {: RESULT = new Integer(e1.intValue() + e2.intValue()); :}
                |
            expr:e1 MINUS expr:e2
            {: RESULT = new Integer(e1.intValue() - e2.intValue()); :}
                |
            expr:e1 TIMES expr:e2
                {: RESULT = new Integer(e1.intValue() * e2.intValue()); :}
                |
            expr:e1 DIVIDE expr:e2
                {: RESULT = new Integer(e1.intValue() / e2.intValue()); :}
                |
            expr:e1 MOD expr:e2
                {: RESULT = new Integer(e1.intValue() % e2.intValue()); :}
                |
            NUMBER:n
                {: RESULT = n; :}
                |
            MINUS expr:e
                {: RESULT = new Integer(0 - e.intValue()); :}
                %prec UMINUS
                |
            LPAREN expr:e RPAREN
                {: RESULT = e; :}
                ;
```

2. Modify the program to include
a. Expressions in multiple line.
b. Declaration statements of the form type id,id,id;
b. Assignment Statements of the form id=expression;id=string;id=number;...
c. Write necessary semantic actions to give an error message while incompataible operands are being assigned to a variable.

****