Computer Science and Engineering Department
I.I.T Kharagpur
Compilers Laboratory: CS39003
3rd year CSE: 5th Semester

Assignment - 7 Marks: 30
Assignment Out: 10/Oct/2013 Report on or before: 27/Oct/2013

1. Consider the following grammar with the start symbol P and the set of production rules as given below. This grammar generates the expressions for vector operation, which you have already encountered during the Lab Test. Terminal Symbols = {+ * X # ( ) [ ] NORM . id , }

S → VE AE
| SE

VE → VE + T
| VE * SE
| T

AE → # V
| . V
| ε

SE → NORM V
| id

T → T X V
| V

V → (VE)
| VEC

VEC → [ID]

ID → ID, id
| id

Lexical Elements are as follows:

id	(-?num)
num	(digit digit*)
digit	([0-9])

2. Manually transform the grammar to LL(1) (without changing the language) by removing *left-recursion* and *left factoring*. You may have to introduce new non-terminals in the process. Write the LL(1) grammar in a file called "**Grammar.txt**".

3. Manually compute the **FIRST()** set of all production body and the non-terminals. Also compute the **FOLLOW()** set of non-terminals. Store this information in another file called "**Fst-Fol.txt**".

4. Write a C program to implement a *table driven predictive parser* for the grammar given in 1. This implementation should contain two different modules. The first module should construct the LL(1) parsing table from the (a) input grammar ("**Grammar.txt**") and (b) computed **FIRST, FOLLOW** set ("**Fst-Fol.txt**"). The representation of parse table is upto you (either in a file or in a temporary data structure). In the second module, take an input expression and parse it using the parse table. The output of the program should print "Accepted" along with the ordered sequence of productions (essentially a leftmost derivation) in case the input expression satisfies the given grammar. In the event of errors, try to do as clear error reporting as possible. Both lexical and syntactic errors should be clearly reported. Use Lex to generate a scanner for the input expression and recognize tokens (Feel free to reuse the code of Lab Test 1).

Let the name of the lex file be <*group-no*>**.7.l** and name of the C program file be <*group-no*>**.7.c.**

5. **Reporting Scheme**: In case of a valid expression, your parser should print "Accepted" and display the ordered sequence of productions. In case of invalid expression, error message should be generated.

6. Write a Makefile to compile <*group-no*>**.7.l** to the corresponding scanner (**lex.yy.c**), and finally to compile <*group-no*>**.7.c** and **lex.yy.c** to an executable (call it **llparser**).

7. **Deliverables**: A tar-archive with the name <*group-no*>**.7.tar** containing the Makefile, <*group-no*>**.7.l**, <*group-no*>**.7.c**, **y.tab.h** and other supporting files (**Grammar.txt, Parsing table, FIRST, FOLLOW** etc).

Sample Example:

Input Expression (w): [9, 4, 2] + [1, 2, 3]