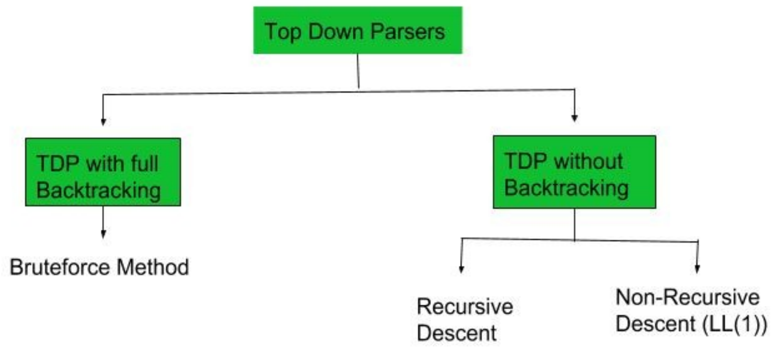
Experiment No:7

AIM: To implement a program to check whether the given grammar is LL(1) or not.
THEORY:

A top-down parser builds the parse tree from the top down, starting with the start non-terminal. There are two types of Top-Down Parsers:

1. Top-Down Parser with Backtracking
2. Top-Down Parsers without Backtracking

Top-Down Parsers without backtracking can further be divided into two parts:



Here the 1st **L** represents that the scanning of the Input will be done from Left to Right manner and the second **L** shows that in this parsing technique we are going to use Left most Derivation Tree. And finally, the **1** represents the number of look-ahead, which means how many symbols are you going to see when you want to make a decision.

Essential conditions to check first are as follows:

1. The grammar is free from left recursion.
2. The grammar should not be ambiguous.
3. The grammar has to be left factored in so that the grammar is deterministic grammar.

These conditions are necessary but not sufficient for proving a LL(1) parser.

ALGORITHM:

Step 1: First check all the essential conditions mentioned above and go to step 2.

Step 2: Calculate First() and Follow() for all non-terminals.

1. First(): If there is a variable, and from that variable, if we try to drive all the strings then the beginning Terminal Symbol is called the First.
2. Follow(): What is the Terminal Symbol which follows a variable in the process of derivation.

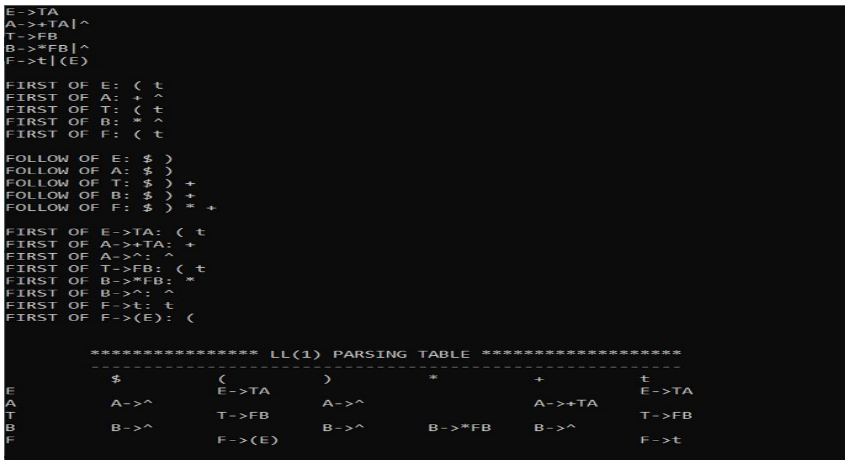
Step 3: For each production A –> α. (A tends to alpha)

1. Find First(α) and for each terminal in First(α), make entry A –> α in the table.
2. If First(α) contains ε (epsilon) as terminal than, find the Follow(A) and for each terminal in Follow(A), make entry A –> α in the table.
3. If the First(α) contains ε and Follow(A) contains $ as terminal, then make entry A –> α in the table for the $.
   To construct the parsing table, we have two functions:

In the table, rows will contain the Non-Terminals and the column will contain the Terminal Symbols. All the **Null Productions** of the Grammars will go under the Follow elements and the remaining productions will lie under the elements of the First set.

COMPUTING ENVRONMENT
Platform: ubuntu
Programming Language: C / C++ / Java

Expacted Output:



```
E->TA
A->+TA|^
T->FB
B->*FB|^
F->t|(E)

FIRST OF E: ( t
FIRST OF A: + ^
FIRST OF T: ( t
FIRST OF B: * ^
FIRST OF F: ( t

FOLLOW OF E: $ )
FOLLOW OF A: $ )
FOLLOW OF T: $ ) +
FOLLOW OF B: $ ) +
FOLLOW OF F: $ ) * +

FIRST OF E->TA: ( t
FIRST OF A->+TA: +
FIRST OF A->^: ^
FIRST OF T->FB: ( t
FIRST OF B->*FB: *
FIRST OF B->^: ^
FIRST OF F->t: t
FIRST OF F->(E): (


        **************** LL(1) PARSING TABLE *******************
        ---------------------------------------------------------
          $         (         )         *         +         t
E                   E->TA                                   E->TA
A         A->^                A->^                A->+TA
T                   T->FB                                   T->FB
B         B->^                B->^      B->*FB    B->^
F                   F->(E)                                  F->t
```

Conclusion: Thus the program to check whether the given grammar is LL(1) or not.

Viva Voce Questions:

1. **Define A Context Free Grammar.**

Answer :

A context free grammar G is a collection of the following
· V is a set of non terminals
· T is a set of terminals

· S is a start symbol
· P is a set of production rules
G can be represented as G = (V,T,S,P)
Production rules are given in the following form
Non terminal → (V U T)*

2. **Briefly Explain The Concept Of Derivation.**

Answer :

Derivation from S means generation of string w from S. For constructing derivation two things are important.
i) Choice of non terminal from several others.
ii) Choice of rule from production rules for corresponding non terminal.
Instead of choosing the arbitrary non terminal one can choose
i) either leftmost derivation – leftmost non terminal in a sentinel form.
ii) or rightmost derivation – rightmost non terminal in a sentinel form.

3. **Define Ambiguous Grammar.**

Answer :

A grammar G is said to be ambiguous if it generates more than one parse tree for some sentence of language L(G).i.e. both leftmost and rightmost derivations are same for the given sentence.

4. What is LL(1) parser?

Answer: A top-down parser that uses a one-token lookahead is called an LL(1) parser.The first L indicates that the input is read from left to right.The second L says that it produces a left-to-right derivation.And the 1 says that it uses one lookahead token. (Some parsers look ahead at the next 2 tokens, or even more than that.

-