Name: Shruti Dalvi Class: D12A Div: Roll No:

Subject: SPCC Topic: Date: Page No:

Aim : Implement LL(1) Parser in
a) Generate the Predictive Parsing Table
b) Perform Parsing action for valid and invalid inputs based on Parsing Table generated.

Theory :

LL(1) Parsing

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

Rules for generation of predictive parsing table.

1. First check all the essential conditions required :

1. The grammar should be 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.

Algorithm for parsing action in predictive parsing table.

Predictive Parsers has the following components:

a) Input Buffer: The input buffer includes the string to be parsed followed by an end marker $ to denote end of string.
b) Stack: It contains a combination of grammar symbols with $ on bottom of stack.
At start of Parsing, stack contains the start symbol of Grammar followed by $
c) Parsing table: It is a 2D Array of matrix M[A,a] where A is non-terminal and 'a' is terminal symbol.

All terminals are written column-wise and non-terminals are written row-wise.

Parsing program:

The parsing program performs some action by comparing the symbol on top of the stack and current input symbol to be read on input buffer.

Actions: Parsing program takes various actions depending upon symbol on top of the stack and current input symbol.

Name: _____ Class: ____ Div: ____ Roll No: ____

Subject: _____ Topic: _____ Date: _____ Page No: _____

Various actions taken are given below:

	Description	Top of stack	Current symbol.	Action.
1.	If stack is empty	$	$	Parsing successful and will be halted.
2.	If symbol at top of stack and current input symbol to be read are both terminals and same.	$a ↑	$ba ↑	Pop a from stack and advance to next input symbol.
3.	If both top of stack and current input symbol are terminals and they are not equal.	$a ↑	$b ↑	Error.
4.	If top of stack is non-terminal and input symbol is terminal.	$X ↑	$a.	Refer to entry M[X, a] in parsing table. If M[X, a] ⇒ X → AB then Pop X from stack push B, A onto stack.

Algorithm:

Input : Context free Grammar.
Output : Predictive Parsing Table M.
Method : For the production $A \rightarrow \alpha$ of Grammar G.

For each terminal, a in FIRST(α) add $A \rightarrow \alpha$ to M[A, a]
If ϵ is in FIRST(α) and b is in FOLLOW(A), then
add $A \rightarrow \alpha$ to M[A, b]
If ϵ is in FIRST(α) and $ is in FOLLOW(A), then
add $A \rightarrow \alpha$ to M[A, $]
All remaining entries in Table M are error.