Assignment No 2

Aim:
Implement Recursive Descent Parser for some language.

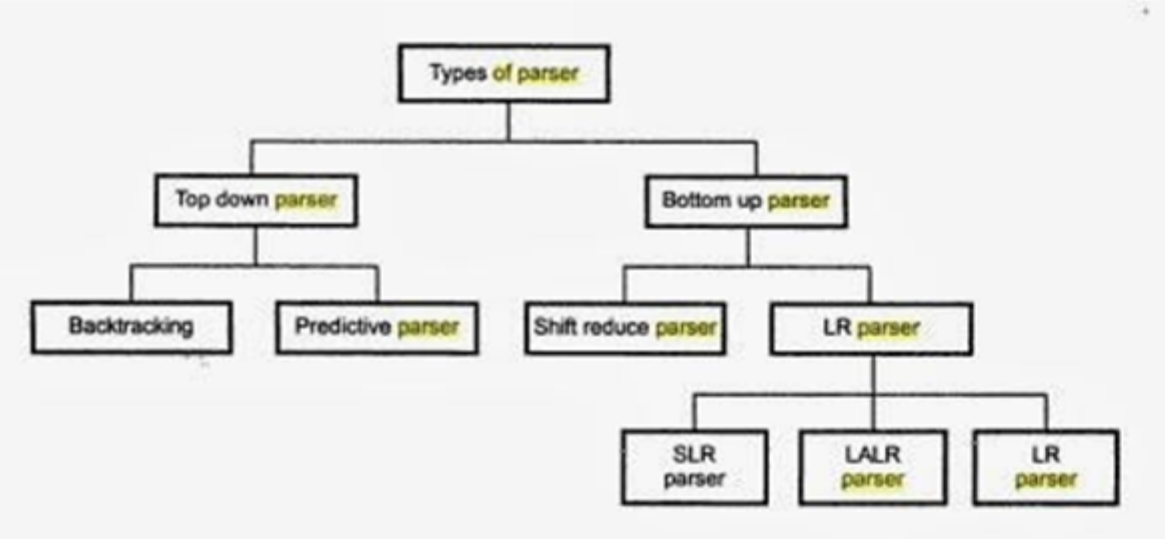
Software:

1. Linux Operating System
2. GCCcompiler

MATHEMATICALMODEL:

S=s,e,i,o,f,DD,NDD,success,failure
s=Start of program
e=The end of program
i=Input String to be parsed. o=Output of the parser, ie,
ValidString(if given input string is valid) OR Parse error (if given input string is invalid)
Success-Parser gives expected output
Failure-Power Failure, Insufficient Memory. f=E, Eprime, T, Tprime, F

Theory:
Recursive Descent Parser


Types of parser
Top down parser
Bottom up parser
Backtracking
Predictive parser
Shift reduce parser
LR parser
SLR parser
LALR parser
LR parser

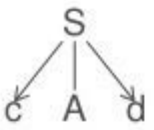
This is a top-down parser in which the parser attempts to verify that the syntax of the input stream is correct as it is read from left to right. A basic operation necessary for this involves reading characters from the input stream and matching then with terminals from the grammar that describes the syntax of the input.RDP will look ahead one character and advance the input stream reading pointer when proper matches occur.

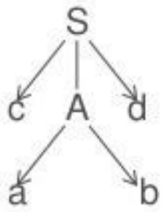
RDP consists of a set of procedures, one for each non terminal. Execution begins with the procedure of the start symbol, which halts and announces success if its procedure body scans the entire input string (valid).

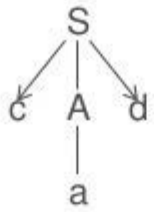
RDP may require back tracking; that is, it may require repeated scans over the input. Parse tree with all possible alternatives. If one alternative doesnt workout, backtrack and use another alternative. If even after using allalternatives, required string is not formed, string is invalid.Consider the following grammar.The required string is cad:

S -> c A d

A -> a b | a







c a d

Advantage:
1.Construction of RDP is easy.

Disadvantages:
1.Backtracking(Not allowed in many programming languages)
2.For left recursive grammar RDP parser may fall in infinite loop.
3.Large space to be traversed (multi level backtracking as parse tree is derived blindly)

Construction of Recursive Descent Parser:

1. If input symbol is non terminal then a call to the procedure corresponding to the non terminal is made.
2. If input symbol is terminal then it is matched with the look ahead from input. The look ahead pointer has to be advanced on matching of the input symbol.
3. If the production rule has many alternates then all these alternates has to be combined into a single body of procedure.
4. The parser should be activated by a procedure corresponding to the start symbol.

Command:
$gccRDescent.c
$./a.out

CONCLUSION:
Thus we have constructed Recursive Descent Parser for given grammar.

Roll No	*Name of Student*	*Date of performance*	*Date of Checking*	*Signature of Sta*
*BECOC*357	Sunny Shah	07 / 07 / 2017	07 / 07 / 2017	

1 PLAGARISM REPORT :

Plagiarism Report

No plagiarism detected

Aim : Implement Recursive Descent Parser for some language. Software Required: Linux Operating Systems, GCC Input: Input data as string to be parsed. Output: Shows if input string is valid or not. Mathematical Model: S= { s, e, i, o, f, DD, NDD, success, failure} s=Start of program e = The end of program i= Input String to be parsed. o = Output of the parser, ie, Valid String (if given input string is valid) OR Parse error (if given input string is invalid) Success- Parser gives expected output Failure- Power Failure, Insufficient Memory.

f={E,Eprime,T,Tprime,F} THEORY : Recursive Descent Parser: Figure 1: Fig: Classification of parsers. This is a top-down parser in which the parser attempts to verify that the syntax of the input stream is correct as it is read from left to right. A basic operation necessary for this involves reading characters from the input stream and matching then with terminals from the grammar that describes the syntax of the input. RDP will look ahead one character and advance the input stream reading pointer when proper matches occur. RDP consists of a set of procedures, one for each nonterminal. Execution begins with the procedure of the start symbol, which halts and announces success if its procedure body scans the entire input string (valid).RDP may require backtracking; that is, it may require repeated scans over the input .

Parse tree with all possible alternatives. If one alternative doesnt work out, backtrack and use another alternative. If even after using all alternatives, required string is not formed, string is invalid. Consider the following grammar. The required string is "cad": Advantage: 1. Construction of RDP is easy. Disadvantages: 1. Backtracking (Not allowed in many programming languages) 2. For left recursive grammar RDP parser may fall in infinite loop.3. Large space to be traversed (multilevel backtracking as parse tree is derived blindly) Construction of Recursive Descent Parser: 1.

If input symbol is nonterminal then a call to the procedure correspond- ing to the nonterminal is made. 2. If input symbol is terminal then it is matched with the lookahead from input. pointer has to be advanced on matching of the input symbol. The lookahead 3. If the production rule has many alternates then all these alternates has to be combined into a single body of procedure. 4. The parser should be activated by a procedure corresponding to the start symbol. Command: $ gcc RDescent.c $ ./a.out CONCLUSION: Thus we have constructed Recursive Descent Parser for given grammar. [Warning: Draw object ignored]

Figure 1: Plagarism Checker www.smallseotools.com/plagarism-checker