Assignment No : B7

Roll no : 4351

1 Title

Recursive Descent Parser

2 Problem Statement

Write a program to generate Recursive descent Parser

3 Learning Objectives

1. To learn the concepts of Compiler Design
2. To develop a recursive-descent parser for a given grammar.
3. To generate a syntax tree as an output of the parser
4. To handle syntax errors

4 Learning Outcome

1. Compiler design techniques were successfully learnt.
2. Implementation of Recursive Descent Parser
3. Syntax Tree was generated.

5 Theory

Recursive Descent Parser

A recursive descent parser is a kind of top-down parser built from a set of mutually-recursive procedures (or a non-recursive equivalent) where each such procedure usually implements one of the production rules of the grammar. Thus the structure of the resulting program closely mirrors that of the grammar it recognizes.
This 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. Our recursive descent parsers will look ahead one character and advance the input stream reading pointer when proper matches occur. What a recursive descent parser actually does is to perform a depth-first search of the derivation tree for the string being parsed. This provides the 'descent' portion of the name. The 'recursive' portion comes from the parser's form, a collection of recursive procedures.

Algorithm:

- Make grammar suitable for parsing i.e. remove left recursion (if required).
- Write a function for each production with error handler.
- Given input is said to be valid if input is scanned completely and no error function is called.

Routine :

```
Procedure T()
Begin
If (inputsymbol=() then
next();
E();
If (inputsymbol=)) then
next();
end
```

else If (inputsymbol=x) then
next();
else
Errorhandler();
END

6 Mathematical Model

Let S be the solution perspective of the system such that
S={ s, e , X, Y, Fme , DD, NDD ,Ffriend | ϕs }
where
DD = Deterministic Data
NDD = Non - Deterministic Data = User Input

ϕs = Constraints on System S

s is start state

e is end state=Syntax Tree is generated

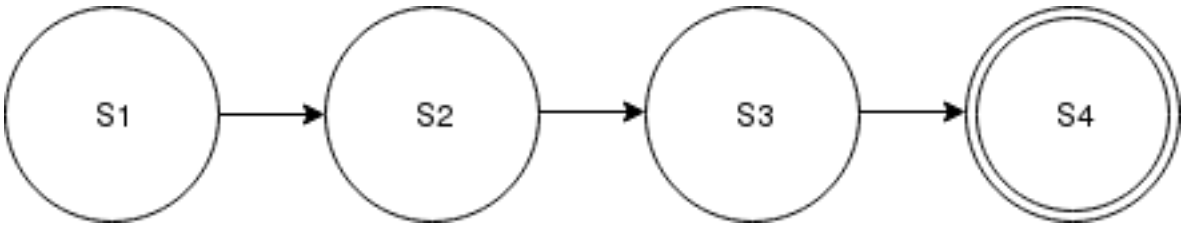
X=set of input parameters
X = X1
where, X1=Expression which is to be parsed

Y=set of output parameters
Y = Y1
where Y1 = Parsed Tree
Fme is the set of main functions
Fme = f1, f2
f1 = Parse the input
f2 = Generate Syntax Tree

Success Case = Right Solution

Failure Case = Wrong solution

7 State Diagram



s1 is the start state
s2 is the input state
s3 is the parsing syntax tree generating state
s4 is the final state

8 Conclusion

We successfully implemented Recursive Descent Parser