SL V

Assignment No. 9.

[33215]

Title :- Recursive descent Parser (RDP)

Problem Statement :- Write a program to implement Recursive Decent Parser.

Objective :-
1) To understand the basic principle of top-down parsing.
2) To study RDP

Theory :- A recursive decent parser is a top-down parser, so called because it builds a parser tree from top (the start symbol) down, and from left to right. (The actual tree is not constructed but is implicit in a sequence of ~~symbols~~ function calls).

This type of parser was very popular for real compilers in the past, but is not as popular now. The parser is usually written entirely by hand and does not require any sophisticated tools. It is not as powerful as shift reduce parsers.

This parser uses a recursive function corresponding to each grammar rule (that is corresponding to non-terminal symbol in the language)

For simplicity one can just use the non-terminal as the name of function. The body of the each recursive function mirror the right side of corresponding rule.

Ex. Grammar

$S \rightarrow aBC$
$B \rightarrow ab / c$
$C \rightarrow c / \epsilon$

R.D.P will be

```
parse_S () {
    if (next == 'a').
        { consume ('a');
          parse_B();
          parse_C();
        }

parse_B() {
    if (next == 'a').
      {
        consume('a');
          if (next == 'b')
              consume ('b');
      }
    else
      {
        parse_C();
      }
```

33213

```
parse_C()
{
  if (next == 'c')
  { return consume('c');
  }
  return true;
}
```

Conclusion:- In this way, we have studied and implemented RDP.