33304

Assignment - 09

* **Problem Statement** :- Study of Recursive Decent Parser

* **Objective** :- 1. To understand the basic principle of top - down parsing
2. To study recursive decent parsers.

* **Theory** :-

A recursive decent parser is a top - down parser, so called because it builds a parse tree from top (the start symbol) down, and from left to right, using an input sentence as a target as it is scanned from left to right.

This type of parser was very popular in the past. It is a simple and effective technique, but is not as popular as some of the shift - reduce parsers.

This parser uses a recursive function corresponding to each grammar rule. The body of each function mirrors the right side of the corresponding rules. In order for this method to work, one must be able to decide which function to call based on the next input symbol.

Perhaps, the hardest part of RDP is the scanning; repeatedly fetching the next token. It is tricky to decide when to scan; the parser doesn't work if there's an extra scan or a missing scan.

* **Algorithm** :- 1. Apply left recursion removal method and remove left recursive grammar.
2. Apply left factoring
3. Compute first set.

33304

Grammar

i) $S \rightarrow TL$

ii) $L \rightarrow +S / e$

iii) $T \rightarrow UM$

iv) $M \rightarrow * T/e$

v) $U \rightarrow (S) / V$

vi) $V \rightarrow 0/1/ \dots 9$

4. Compute first sets

i) First (V) = {0, ... 9}

ii) First (U) = First (S) ∪ First (V) = {(} ∪ {(, 0 ... 9} = {(, 0 ... 9}

iii) First (M) = First (*T) ∪ First (e) = {* , e}

iv) First (T) = First (UM) = First (U) = {(, 0 ... 9}

v) First (L) = First (+S) ∪ First (e) = {+ , e}

5. First (S) = First (TL) = First (T)

6. First (S) = First (TL) = First (T) = {(, 0 ... 9}

Recursive Decent Parser

```
parse_S() {
    //S → TL
     parse_T();
     parse_L();
}
parse_L() {
    //L → +S
    if (lookahead == +) {
    match (+);
    parse_S();
}  }
    //L
    else ....
}
```

DATE __/__/___
PAGE NO ______

33304

```
parse_T() {
        // T → VM
        parse_V();
        parse_M();
}
parse_L() {
       // L → +S
       if (lookahead == +) {
              match (+);
              parse_S();
       }
       else .... ;
}
parse_T() {
        // T → VM
        parse_V();
        parse_M();
}
parse_V() {
        if (lookahead == "(") {
                // V → (S)
                match ("(");
                parse_S();
                match (")");
        }
        else parse_V();
}
parse_V() {
       if (lookahead == "0") {
              // V → 0
              match ("0");
       }
       else if (lookahead == "1") {      // V → 1
              match ("1");
        }
        :match (a);
        } else error();
2
```

DATE __/__/__
PAGE NO ______

33304

* Conclusion → Thus in this assignment we learnt about RDP and implemented a program based on RDP.