Recursive Descent Parser

Parsing is the process to determine whether the start symbol can derive the program or not. If the Parsing is successful then the program is a valid program otherwise the program is invalid.

There are generally two types of Parsers:

1. **Top-Down Parsers:**

   In this Parsing technique we expand the start symbol to the whole program.

   Recursive Descent and LL parsers are the Top-Down parsers.

2. **Bottom-Up Parsers:**

   In this Parsing technique we reduce the whole program to start symbol.

   Operator Precedence Parser, LR(0) Parser, SLR Parser, LALR Parser and CLR Parser are the Bottom-Up parsers.

Recursive Descent Parser:

It is a kind of Top-Down Parser. A top-down parser builds the parse tree from the top to down, starting with the start non-terminal. A *Predictive Parser* is a special case of Recursive Descent Parser, where no Back Tracking is required.

By carefully writing a grammar means eliminating left recursion and left

factoring from it, the resulting grammar will be a grammar that can be parsed by a recursive descent parser.

Example:

Before removing left recursion	After removing left recursion
E –> E + T | T	E –> T E’
T –> T * F | F	E’ –> + T E’ | e
F –> (E) | id	T –> F T’
	T’ –> * F T’ | e
	F –> (E) | id

**Here e is Epsilon

For Recursive Descent Parser, we are going to write one program for every variable.

Example:

```
Grammar: E --> i E'

E' --> + i E' | e
```

```c
int main()
{
    // E is a start symbol.
    E();

    // if lookahead = $, it represents the end of the string
    // Here l is lookahead.
    if (l == '$')
        printf("Parsing Successful");
}

// Definition of E, as per the given production
E()
{
    if (l == 'i') {
        match('i');
        E'();
    }
}

// Definition of E' as per the given production
E'()
{
    if (l == '+') {
        match('+');
        match('i');
        E'();
    }
    else
        return ();
}

// Match function
match(char t)
{
    if (l == t) {
        l = getchar();
    }
    else
        printf("Error");
}
```