Compiler Design Lab

Recursive Descent Parser

1. 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. 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.

   Rule for immediate left recursion removal.

   - For a rule like:
     - A -> A α|β
   - We may replace it with
     - A -> β A'
     - A' -> α A' |ε

   (a) Write a program to remove the recursion from the following grammars:
   Grammar 1 : E -> E+T|T; T -> T*F|F; F ->(E)|id 2 marks
   Grammar 2 : S -> Aa | b; A -> Ac | Sd | ε 3 marks

   (b) Recursive Descent Parser:
   - Consists of a set of procedures, one for each nonterminal
   - Execution begins with the procedure for start symbol
   - A typical procedure for a non-terminal A is gien by:

```
void A() {
    choose an A-production, A->X1X2..Xk
    for (i=1 to k) {
        if (Xi is a nonterminal)
            call procedure Xi();
        else if (Xi equals the current input symbol a)
            advance the input to the next symbol;
        else /* an error has occurred */
    }
}
```

   Implement a recursive descent parser and show the results (Whether the string is valid or not and the parse tree generated) for the grammar 1 above and test for input strings: (id+id)*id,)id+id*id 5 marks