Exp14 : RECURSIVE DESCENT PARSER

1. Start
2. Given Grammar
   E -> T E'
   E' -> + T E' | ε
   T -> F T'
   T' -> * F T' | ε
   F -> id | (E)
3. input[100] , i = 0 , error = 0
4. E()
5. if (strlen(input) = i & error = 0) then
   1. print “Accepted”
6. else
   1. print “Rejected”

7. void E()
   T()
   E_dash()

8. void E_dash()
   1. if (input[i] = +) then
      i++
      T()
      E_dash()

9. void T()
   F()
   T_dash()

10. void T_dash()
    1. if (input[i] = *) then
       i++
       F()
       T_dash()

11. void F()
    1. if (isalnum(input[i])) then
       i++
    2. else if (input[i] = ( ) then
       i++
       E()
       if (input[i] =)) then
          i++
       else
          error = 1
    3. else
       error = 1

12. Stop

Recursive Descent Parser for a given Grammar (C)

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>

char input[100];
int i, error;
void E();
void T();
void E_dash();
void T_dash();
void F();

int main(){
   i = 0;
   error = 0;
   printf("The Grammar implemented in this code\n E -> T E' \n E' -> + T E' | ε \n T -> F T' \n T' ->
* F T' | ε \n F -> id | ( E ) \n");
   printf("Enter an arithmetic expression (either use numbers or single alphabets) : ");
   gets(input);

   E(); //first call the starting symbol
   if(strlen(input) == i && error == 0){
      printf("Accepted\n");
   }else{
      printf("Rejected\n");
   }

   return 0;
}

void E(){
   // E -> T E'
   T();
   E_dash();
}

void E_dash(){
   // E' -> + T E'
   if (input[i] == '+'){
      i++;
      T();
      E_dash();
   }
}

void T(){
   // T -> F T'
   F();
   T_dash();
}
```

```
void T_dash(){
   // T' -> * F T'
   if(input[i] == '*'){
      i++;
      F();
      T_dash();
   }
}

void F(){
   if(isalnum(input[i])){
      // F -> id
      // Here id can be alpha-numeric
      i++;
   }else if(input[i] == '('){
      // F -> ( E )
      i++;
      E();
      if(input[i] == ')'){
         i++;
      }else{
         error = 1;
      }
   }else{
      error = 1;
   }
}
```

output

```
deadpool@daredevil:~/Desktop/s7-CD/9 Recursive Descent Parser$ gcc recursive.c
recursive.c: In function ‘main’:
recursive.c:18:5: warning: implicit declaration of function ‘gets’; did you mean ‘fgets’? [-Wimplicit-function-declaration]
   18 |     gets(input);
      |     ^~~~
      |     fgets
/usr/bin/ld: /tmp/cc2NY01R.o: in function `main':
recursive.c:(.text+0x4f): warning: the `gets' function is dangerous and should not be used.
deadpool@daredevil:~/Desktop/s7-CD/9 Recursive Descent Parser$ ./a.out
The Grammar implemented in this code
 E -> T E'
 E' -> + T E' | ε
 T -> F T'
 T' -> * F T' | ε
 F -> id | ( E )
Enter an arithmetic expression (either use numbers or single alphabets) : 3+4*(5)
Accepted
deadpool@daredevil:~/Desktop/s7-CD/9 Recursive Descent Parser$ ./a.out
The Grammar implemented in this code
 E -> T E'
 E' -> + T E' | ε
 T -> F T'
 T' -> * F T' | ε
 F -> id | ( E )
Enter an arithmetic expression (either use numbers or single alphabets) : a+b*c+(d)
Accepted
deadpool@daredevil:~/Desktop/s7-CD/9 Recursive Descent Parser$ ./a.out
The Grammar implemented in this code
 E -> T E'
 E' -> + T E' | ε
 T -> F T'
 T' -> * F T' | ε
 F -> id | ( E )
Enter an arithmetic expression (either use numbers or single alphabets) : a+b**
Rejected
deadpool@daredevil:~/Desktop/s7-CD/9 Recursive Descent Parser$
```