lex/yacc example

Consider the following simple expression grammar;

```
s  -->  e $
e  -->  e + t | t
t  -->  t * f | f
f  -->  ID | NUM | ( e )
```

I. lex rule file (rules.txt);

```
A                   [A-Za-z]
D                   [0-9]
%%
{A}({A}|{D})*       { toknum++; return (ID); }
{D}+                { toknum++; return (NUM); }
"+"                 { toknum++; return (PLUS); }
"*"                 { toknum++; return (MULT); }
")"                 { toknum++; return (RPAREN); }
"("                 { toknum++; return (LPAREN); }
.                   ;
%%
int yyerror ()
 { printf (" lex/yacc error at token %d\n", toknum); return(1);
 }
```

II. yacc file (expr.y);

```
%{
#include <stdio.h>
int toknum = 0;
%}
%token      ID 1 NUM 2 PLUS 3 MULT 4 LPAREN 5 RPAREN 6
%start  s
%%
s :     e ;
e :     e PLUS t
  |     t ;
t :     t MULT f
  |     f ;
f :     ID
  |     NUM
  |     LPAREN e RPAREN ;
%%
#include "lexyy.c"
void main ()
```

```
{ if (!yyparse())
      printf (" success!\n");
  else printf (" failure\n");
}
```

III. construction and execution (MS);

```
c:> flex rules.txt
c:> yacc expr
c:> gcc expr.c                    /* or any other compiler... */
c:> expr <source_expr             /* source file containing expression to be parsed */
```