```
----------- EXPRESSION GRAMMAR ----------

----- LL(1) grammar specification -------
Number of grammar variables? 5
More than one character for a variable supported
Variable 1: E
Variable 2: E'
Variable 3: T
Variable 4: T'
Variable 5: F
Number of terminals? 6
Please consider epsilon as e. You may enter more than one
character for each terminal
Terminal 1: id
Terminal 2: (
Terminal 3: )
Terminal 4: +
Terminal 5: *
Terminal 6: e
Number of rules?8
Please (in RHS) give a whitespace between successive entities
(terminals, non-terminals, or terminal non-terminal) else
the code will match largest prefix of RHS to terminal/variable
Rule 1 LHS variable: E
Rule 1 RHS string: T E'
Rule 2 LHS variable: E'
Rule 2 RHS string: + T E'
Rule 3 LHS variable: E'
Rule 3 RHS string: e
Rule 4 LHS variable: T
Rule 4 RHS string: F T'
Rule 5 LHS variable: T'
Rule 5 RHS string: * F T'
Rule 6 LHS variable: T'
Rule 6 RHS string: e
Rule 7 LHS variable: F
Rule 7 RHS string: ( E )
Rule 8 LHS variable: F
Rule 8 RHS string: id
Start symbol: E
------- LL(1) grammar specifics entered -------
Variables: E E' T T' F
Terminals: id ( ) + * e
Grammar rules:
E -> T E'
E' -> + T E'
E' -> e
T -> F T'
T' -> * F T'
T' -> e
```

```
F -> ( E )
F -> id
--------- FIRST SET ----------------
E: ( id
E': + e
T: ( id
T': * e
F: ( id
--------- FOLLOW SET ----------------
E: $ )
E': E E' $ ) E E'
T: + e  + e
T': T T' + e  + e   T T'  ) $
F: * e  * e
--------- NUMBERED PRODUCTIONS ------------
0: E -> T E'
1: E' -> + T E'
2: E' -> e
3: T -> F T'
4: T' -> * F T'
5: T' -> e
6: F -> ( E )
7: F -> id
---------- PREDICTIVE PARSING TABLE ------------
    id  (    )   +   *   e   $
E   0    0   -1  -1  -1  -1  -1
E'  -1   -1  2   1   -1  2   2
T   3    3   -1  -1  -1  -1  -1
T'  -1   -1  5   5   4   5   5
F   7    6   -1  -1  -1  -1  -1
----------- INPUT STRING ----------
Enter the input string: id + id * id $
E$       id+id*id$
TE'$     id+id*id$
FT'E'$       id+id*id$
idT'E'$      id+id*id$
T'E'$        +id*id$
eE'$     +id*id$
E'$      +id*id$
+TE'$        +id*id$
TE'$     id*id$
FT'E'$       id*id$
idT'E'$      id*id$
T'E'$        *id$
*FT'E'$      *id$
FT'E'$       id$
idT'E'$      id$
T'E'$        $
eE'$     $
E'$      $
```

```
e$      $
$       $
        $
Underflow
----------- INPUT STRING ----------
Enter the input string: id
E$      id
TE'$    id
FT'E'$      id
idT'E'$     id
----------- INPUT STRING ----------
Enter the input string: id * ( id + id ) $
E$T'E'      id*(id+id)$
TE'$T'E'    id*(id+id)$
FT'E'$T'E'      id*(id+id)$
idT'E'$T'E'     id*(id+id)$
T'E'$T'E'       *(id+id)$
*FT'E'$T'E'     *(id+id)$
FT'E'$T'E'      (id+id)$
(E)T'E'$T'E'    (id+id)$
E)T'E'$T'E'     id+id)$
TE')T'E'$T'E'       id+id)$
FT'E')T'E'$T'E'     id+id)$
idT'E')T'E'$T'E'    id+id)$
T'E')T'E'$T'E'      +id)$
eE')T'E'$T'E'       +id)$
E')T'E'$T'E'    +id)$
+TE')T'E'$T'E'      +id)$
TE')T'E'$T'E'       id)$
FT'E')T'E'$T'E'     id)$
idT'E')T'E'$T'E'    id)$
T'E')T'E'$T'E'      )$
eE')T'E'$T'E'       )$
E')T'E'$T'E'    )$
e)T'E'$T'E'     )$
)T'E'$T'E'      )$
T'E'$T'E'       $
eE'$T'E'    $
E'$T'E'     $
e$T'E'      $
$T'E'       $
T'E'    $
eE'     $
E'      $
e       $
        $
        $
Underflow
----------- INPUT STRING ----------
Enter the input string: id + $
```

```
E$       id+$
TE'$     id+$
FT'E'$       id+$
idT'E'$      id+$
T'E'$        +$
eE'$     +$
E'$      +$
+TE'$        +$
TE'$     $
Error in parsing
----------- INPUT STRING ----------
Enter the input string: id $
E$E'$        id$
TE'$E'$      id$
FT'E'$E'$        id$
idT'E'$E'$       id$
T'E'$E'$     $
eE'$E'$      $
E'$E'$       $
e$E'$        $
$E'$     $
E'$      $
e$       $
$        $
         $
Underflow
----------- INPUT STRING ----------
Enter the input string: * $
E$       *$
Error in parsing
----------- INPUT STRING ----------
Enter the input string:
```