```
%{
        void yyerror(char *s);
        #include <stdio.h>
        #include <stdlib.h>

        int symbols[52];
        int symbolVal(char symbol);
        void updateSymbolVal(char symbol, int val);

%}

%union  {       int num; char id;       }
%start  line
%token  print
%token  exit_command
%token  <num>   number
%token  <id>    identifier
%type   <num>   line exp term factor
%type   <id>    assignment

%%

line    :       assignment ';'  {       ;       }
                |       exit_command ';'        {       exit(EXIT_SUCCESS);     }
                |       print exp ';'   {       printf("Result : %d/n", $2);    }
                |       line assignment ';'     {       ;       }
                |       line exit_command ';'   {       exit(EXIT_SUCCESS);     }
                |       line print exp ';'      {       printf("Result : %d\n",
$3);    }
                ;

assignment      :       identifier '=' exp      {       updateSymbolVal($1, $3);        }
                        ;

exp     :       term    {       $$ = $1;        }
        |       exp '+' term    {       $$ = $1 + $3;   }
        ;
term   : factor                          {$$ = $1;}
       | term '*' factor                 {$$ = $1 * $3;}
       ;

factor :        number                         {$$ = $1;}
       |        '(' exp ')'                    {$$ = $2;}
       |    identifier  {       $$ = symbolVal($1);     }
       ;

%%

int computeSymbolIndex(char token)
{
        int indx = -1;
        if(islower(token))
                indx = token - 'a' + 26;
        else if(isupper(token))
                indx = token - 'A';

        return indx;
}

int symbolVal(char symbol)
{
        return symbols[computeSymbolIndex(symbol)];
}

void updateSymbolVal(char symbol, int val)
{
        symbols[computeSymbolIndex(symbol)] = val;
}

int main(void)
{
        int i;
        for(i = 0; i < 52; i++)
                symbols[i] = 0;
        return yyparse();
}

void yyerror(char *s)
{
        printf("%s\n", s);
}
```