```
%{

        #include <stdio.h>
        #include <stdlib.h>
        #include <string.h>

        void yyerror(char *s);

        enum TreeType   {       OperatorNode, NumberNode, VariableNode  };

        typedef struct Tree
        {
                enum TreeType NodeType;
                union
                {
                        struct
                        {
                                struct Tree *left;
                                struct Tree *right;
                                char operator;
                        }anOperator;

                        int number;
                        char variable;
                } body;
        } Tree;



static Tree *toOperator(Tree *l, char op, Tree *r)
{
        Tree *node = (Tree*)malloc(sizeof(Tree));

        node->NodeType = OperatorNode;
        node->body.anOperator.left = l;
        node->body.anOperator.operator = op;
        node->body.anOperator.right = r;

        return node;
}

static Tree *toNumber(int n)
{
        Tree *node = (Tree*)malloc(sizeof(Tree));

        node->NodeType = NumberNode;
        node->body.number = n;

        return node;
}

static Tree *toVariable(char v)
{
        Tree *node = (Tree*)malloc(sizeof(Tree));

        node->NodeType = VariableNode;
        node->body.variable = v;

        return node;
}

static void displayTree(Tree *tr, int lvl)
{
        int gap = 2;
        if(tr)  //tree is not null
        {
                switch(tr->NodeType)
                {
                        case OperatorNode:
                                displayTree(tr->body.anOperator.right, lvl+gap);
                                printf("%*c%c\n", lvl, ' ', tr->body.anOperator.operator);
                                displayTree(tr->body.anOperator.left, lvl+gap);
                                break;

                        case NumberNode:
                                printf("%*c%d\n", lvl, ' ', tr->body.number);
                                break;

                        case VariableNode:
                                printf("%*c%c\n", lvl, ' ', tr->body.variable);
                                break;
                }
        }
}




%}

%union
{
        int number;
        char variable;
        struct Tree *tr;
}

%start  line

%token  exit_command
%token  <number>        num
%token  <variable>      var
%type   <tr>            exp     term    factor

%%

line    :       exp     ';'                     {       displayTree($1, 1);             }
                |       exit_command ';'        {       exit(EXIT_SUCCESS);     }
                |       line exit_command ';'   {       exit(EXIT_SUCCESS);     }
                ;

exp             :       '+' term                {       $$ = $2;        }
                |       '-' term                {       $$ = toOperator(NULL, '~', $2); }
                |       term                    {       $$ = $1;        }
                |       exp '+' term    {       $$ = toOperator($1, '+', $3);   }
                |       exp '-' term    {       $$ = toOperator($1, '-', $3);   }
                ;

term    :       factor                  {       $$ = $1;        }
                |       term '*' factor {       $$ = toOperator($1, '*', $3);   }
                |       term '/' factor {       $$ = toOperator($1, '/', $3);   }
                ;

factor  :       num                     {       $$ = toNumber($1);      }
                |       var                     {       $$ = toVariable($1);    }
                |       '('     exp     ')'             {       $$ = $2;        }
                ;

%%

int main(void)
{
        return yyparse();
}

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