Christian McGovern, The problem for this lab was fixing 3 operators. (), *, and a negative number (unary) in yacc and lex.

Files created by yacc: y.tab.c and t.tab.h

Files created by lex: lex.yy.c

We do not compile Lex output directly here because we put

#include "lex.yy.c"

which implicitly compiles it to our yacc program.

Yacc code:

```
%{

/*
 *                                   **** CALC ****
 *
 * This routine will function like a desk calculator
 * There are 26 integer registers, named 'a' thru 'z'
 *
 */

/* This calculator depends on a LEX description which outputs either VARIABLE or INTEGER.
   The return type via yylval is integer

   When we need to make yylval more complicated, we need to define a pointer type for yylval
   and to instruct YACC to use a new type so that we can pass back better values

   The registers are based on 0, so we substract 'a' from each single letter we get.
   based on context, we have YACC do the correct memmory look up or the storage depending
   on position


   Shaun Cooper
    January 2015


   problems  fix unary minus, fix parenthesis, add multiplication
   problems  make it so that verbose is on and off with an input argument instead of compiled in
*/




           /* begin specs */
#include <stdio.h>
#include <ctype.h>
#include "lex.yy.c"


int regs[26];
int base, debugsw;


void yyerror (s)  /* Called by yyparse on error */
     char *s;
{
  printf ("%s\n", s);
}




%}
/*  defines the start symbol, what values come back from LEX and how the operators are associated  */
```

```
%start list

%token INTEGER
%token  VARIABLE

%left '|'
%left '&'
%left '+' '-'
%left '*' '/' '%'
%left UMINUS


%%      /* end specs, begin rules */

list    :       /* empty */
        |       list stat '\n'
        |       list error '\n'
                        { yyerrok; }
        ;

stat    :       expr
                        { fprintf(stderr,"the answer is %d\n", $1); }
        |       VARIABLE '=' expr
                        { regs[$1] = $3; }
        ;

expr    :       '(' expr ')'
                        { $$ = $2; }
        |       expr '-' expr
                        { $$ = $1 - $3; }
        |       expr '+' expr
                        { $$ = $1 + $3; }
        |       expr '/' expr
                        { $$ = $1 / $3; }
        //added multiplication rules
        |   expr '*' expr
                        { $$ = $1 * $3; }
        |       expr '%' expr
                        { $$ = $1 % $3; }
        |       expr '&' expr
                        { $$ = $1 & $3; }
        |       expr '|' expr
                        { $$ = $1 | $3; }
        |   '-' expr  %prec UMINUS //removed the expr before the '-'
                        { $$ = -$2; }
        |       VARIABLE
                        { $$ = regs[$1]; fprintf(stderr,"found a variable value = %d\n",$1); }
        |       INTEGER {$$=$1; fprintf(stderr,"found an integer\n");}
        ;
%%      /* end of rules, start of program */

main()
{ yyparse();
}
```

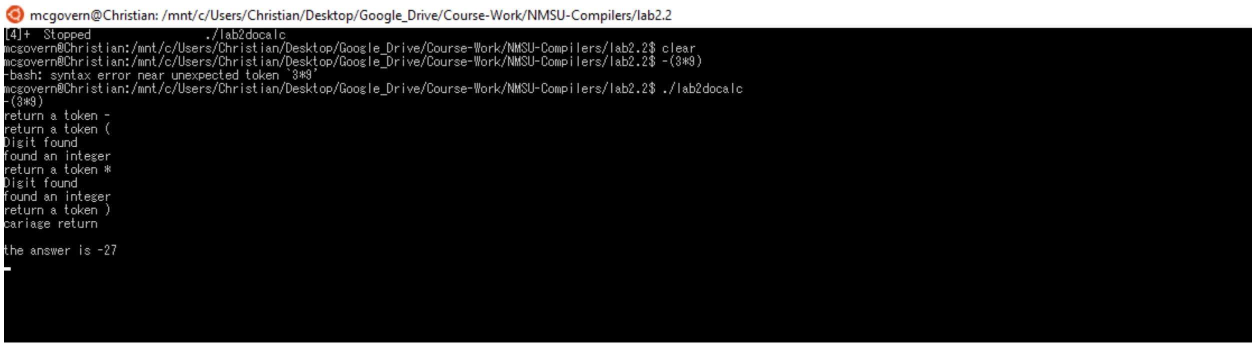
Lex code:

```
/*              Small LEX routine which returns two formal tokens (INTEGER and VARIABLE)
```

```
          along with single string elements like '+'.


           This LEX definition is the companion to the docalc.y YACC routine which
          is a simple calculator
           Shaun Cooper
           January 2015
*/
%{
int mydebug=1;
#include "y.tab.h"
%}
%%
[a-z]           {if (mydebug) fprintf(stderr,"Letter found\n");
            yylval=*yytext-'a'; return(VARIABLE);}
[0-9][0-9]*     {if (mydebug) fprintf(stderr,"Digit found\n");
            yylval=atoi((const char *)yytext); return(INTEGER);}
[ \t]           {if (mydebug) fprintf(stderr,"Whitespace found\n");}
[=()\-+*/%&|]   { if (mydebug) fprintf(stderr,"return a token %c\n",*yytext); //added () to set
            return (*yytext);}
\n              { if (mydebug) fprintf(stderr,"cariage return %c\n",*yytext);
            return (*yytext);}

%%
int yywrap(void)
{ return 1;}
```



mcgovern@Christian: /mnt/c/Users/Christian/Desktop/Google_Drive/Course-Work/NMSU-Compilers/lab2.2

```
[4]+  Stopped                 ./lab2docalc
mcgovern@Christian:/mnt/c/Users/Christian/Desktop/Google_Drive/Course-Work/NMSU-Compilers/lab2.2$ clear
mcgovern@Christian:/mnt/c/Users/Christian/Desktop/Google_Drive/Course-Work/NMSU-Compilers/lab2.2$ -(3*9)
-bash: syntax error near unexpected token `3*9'
mcgovern@Christian:/mnt/c/Users/Christian/Desktop/Google_Drive/Course-Work/NMSU-Compilers/lab2.2$ ./lab2docalc
-(3*9)
return a token -
return a token (
Digit found
found an integer
return a token *
Digit found
found an integer
return a token )
cariage return

the answer is -27
```