MakeFile

```
# Name: Zachary Neeley
# Lab: 2.2
# Date: 1/18/19
# Question 1: File names created by yacc: y.tab.c
# Question 2: File names created by lex: lex.yy.c
# Question 3: We do not directly compile LEX output because it only gives tokens and not code.
#

all:
	yacc -d lab2docalc.y
	lex lab2docalc.l
	gcc y.tab.c -o lab2docalc
```

lab2docal.y

```
%{

/*
 * Name: Zachary Neeley
 * Date: 1/25/19
 * Purpose: The prupose of this program is to create a calculator using LEX and YACC
 *        to learn the basic process of tokens, expressions and basic trees.
 *
 *     Input: Numbers and characters ex: -2*(5+3)
 *     Output: Show each token read in the input and the the digit found. Finially
 *           showing an answer.
 *     Assumption: The input is assumed to be a only numbers and no variables in the
 *              input for example: y = 2^2x(2-1)
 *     Changes: Fixed the Unary minus by making it so that it only has 1 input and the - symbol
 *                                    Fixed multiplication by creating the rule allowing the multiplication to take place.
 */
```

```
/*
 *                          **** 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; }
        |       expr '%' expr
                { $$ = $1 % $3; }
        |       expr '&' expr
                { $$ = $1 & $3; }
        |       expr '|' expr
                { $$ = $1 | $3; }
  |     expr '*' expr
                { // Added the entire expression for multiplication
                $$ = $1 * $3; }
        |       '-' expr  %prec UMINUS
                { // Fixed the minus by removing the $1
                $$ = -$2; }
        |       VARIABLE
                { $$ = 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();
}
```

Lab2docalc.l

```
/*
 * Name: Zachary Neeley
 * Date: 1/25/19
 * Purpose: The prupose of this program is to create a calculator using LEX and YACC
 *         to learn the basic process of tokens, expressions and basic trees.
 *
 *     Input: Numbers and characters ex: -2*(5+3)
 *     Output: Token steam of the incoming input.
 *     Assumption: The input is assumed to be a only numbers and no variables in the
 *                 input for example: y = 2^2x(2-1)
 *     Changes: Fixed the () by adding them to the LEX token output.
 */

/*              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");}
[=\-+*/%&|()]       { //Added () to the symbols
                            if (mydebug) fprintf(stderr,"return a token %c\n",*yytext);
                    return (*yytext);}
\n                  { if (mydebug) fprintf(stderr,"cariage return %c\n",*yytext);
                    return (*yytext);}

%%

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

Output