```
%{

/* chad c d clark   < clarkch @ cpsc . ucalgary . ca >
 *
 * cpsc 411     lec ??
 * winter 2002  lab 02
 *
 * assignment #1 - a first stab.
 *
 * file: as1lex.l
 * purpose: a basic lexer
 *
 * assumtions:
 *    - all keywords are lowercase.
 *    - identifiers are case sensitive, start with an alphabetic character,
 *      and consist of alphabetic and numeric characters.
 *    - whitespace is to be ignored. (syntax oriented, not line oriented.)
 *
 */

#include "as1tokens.h"
#include "as1globals.h"
#include "as1tree.h"


%}


IF      "if"
THEN    "then"
WHILE   "while"
DO      "do"
UNTIL   "until"
READ    "read"
ELSE    "else"
BEGIN   "begin"
END     "end"
PRINT   "print"
ID      [a-zA-Z][0-9a-zA-Z]*
NUM     [0-9]+
ADD     "+"
SUB     "-"
MUL     "*"
DIV     "/"
LPAR    "("
RPAR    ")"
SEMICOLON       ";"
ASSIGN  ":="
MULTISTART      "/*"
MULTIEND        "*/"


%x COMMENT

%%

{MULTISTART}            {BEGIN COMMENT;}
<COMMENT>{MULTIEND}     {BEGIN 0;}
<COMMENT>\n             { /* multiline comment */ }
<COMMENT>.              { /* multiline comment */ }

"%".*\n         { /* single line comment */ }

[ ]     { /* whitespace */ }
\t      { /* whitespace */ }
```

```
\n      { /* whitespace */ }

{IF}    { return (T_IF); }
{THEN}  { return (T_THEN); }
{WHILE} { return (T_WHILE); }
{DO}    { return (T_DO); }
{UNTIL} { return (T_UNTIL); }
{READ}  { return (T_READ); }
{ELSE}  { return (T_ELSE); }
{BEGIN} { return (T_BEGIN); }
{END}   { return (T_END); }
{PRINT} { return (T_PRINT); }
{ID}    { return (T_ID); }
{NUM}   { return (T_NUM); }
{ADD}   { return (T_ADD); }
{SUB}   { return (T_SUB); }
{MUL}   { return (T_MUL); }
{DIV}   { return (T_DIV); }
{LPAR}  { return (T_LPAR); }
{RPAR}  { return (T_RPAR); }
{SEMICOLON}     { return (T_SEMICOLON); }
{ASSIGN}        { return (T_ASSIGN); }
.       { return (T_ERROR); }

%%


/* chad c d clark   < clarkch @ cpsc . ucalgary . ca >
 *
 * cpsc 411     lec ??
 * winter 2002  lab 02
 *
 * assignment #1 - a first stab.
 *
 * function: as1main.c
 * purpose: tests the lexer's return values.
 *
 *
 */

struct stree_node * prog(void);

int main(int argc, char **argv) {

        struct stree_node *STree = NULL;

        if(!FINAL) printf("\n");

        curr_token = yylex();

        STree = prog();

        if(PRINT_STREE) {       /* see as1globals.h */
                printf("\nSyntax Tree:\n");
                print_stree(STree, 0);
        }

        /*
        * printf("\n\nSeaching for PRINT_NODE's ... ");
        * if (find_node(PRINT_NODE, STree))
        *       printf("success!\n");
        * else
        *       printf("failed!\n");
        */
```

```
        if(!FINAL) printf("\nGenerated Code:\n");
        gen_code(STree);

        delete_stree(STree);

        if(!FINAL) printf("\n");
        return(0);
}
```