```
%{
      #include <malloc.h>
      #include <stdlib.h>
      #include <stdio.h>
      #include <math.h>
      int   VAL[26];
%}

%token NEWLINE PLUS MINUS MULTIPLICATION DIVISION CM SM IDENTIFIER NUMBER
NUMB STRING LP RP IF GT LT LB RB IDENTIFIER1 EQUAL ELSE MOD DOUBLE SIN
COS TAN LOG LPP RPP LOOP TO LBP RBP FOR
%left PLUS MINUS
%left MULTIPLICATION DIVISION
%left TO
%left  LT GT
%right EQUAL
%right MOD
%left SIN COS TAN LOG


%%
START:

      | START stmt
      ;

stmt: NEWLINE

      | exp NEWLINE          { if($1>floor($1) && $1<ceil($1))
printf("%lf\n",$1); else if($1==-1); else printf("%d\n",$1); }

      | type var SM          {printf("Valid identifier\n");}

      | IDENTIFIER1 EQUAL exp SM    { VAL[$1] = $3; }

      | FOR LP NUMB RP LB exp RB {
                                      int i;
                                      for(i=0 ; i<$3 ; i++)
{printf("value of the loop: %d expression value: %d\n", i,$6);}

                                      }

      | IF LP exp RP   exp SM                 {
                                                    if($3)
                                                    {
                                                          printf("\nvalue
of expression in IF: %d\n",$5);
                                                    }

                                 }

      | IFELSE
      | IF LP exp RP    IF LP exp RP exp SM ELSE exp SM    SM ELSE exp SM
{
                                                    if($3)
                                                    {

                                                         if($7)
                                                         {

      printf("value of expression in Nested  IF: %d\n",$9);
                                                         }
                                                         else
                                                         {

      printf("value of expression in Nested else: %d\n",$12);
                                                         }
                                                    }
                                                    else
                                                    {
                                                         printf("value of
expression in Outer ELSE: %d\n",$16);
                                                    }
                                                 }


      ;
IFELSE: IF LP exp RP exp SM ELSE exp SM {
                                                    if($3)
                                                    {
                                                         printf("value of
expression in IF: %d\n",$5);
                                                    }
                                                    else
                                                    {
                                                         printf("value of
expression in ELSE: %d\n",$8);
                                                    }
                                                 }
           ;

var: var CM IDENTIFIER
      |IDENTIFIER
      ;
type:NUMBER
      |STRING
      ;

exp:      NUMB                      { $$ = $1;  }
           |DOUBLE                            { $$ = $1;  }

           | IDENTIFIER1                   {$$ = VAL[$1]}

        | exp PLUS exp              { $$ = $1 + $3; }

        | exp MINUS exp             { $$ = $1 - $3; }
        | exp MULTIPLICATION exp  { $$ = $1 * $3; }

        | exp DIVISION exp              { if($3!=0) $$ = $1 / $3; else{
printf("Divide by zero error");$$=-1;$1=-1;}}

        | exp GT exp                    { $$ = $1 > $3; }

        | exp LT exp                    { $$ = $1 < $3; }

        | exp MOD exp                   { $$ = $1 % $3;}

        | SIN exp                             {printf("Value of Sin(%d)
is %lf\n",$2,sin($2*3.1416/180)); $$=-1}

        | COS exp                             {printf("Value of Cos(%d)
is %lf\n",$2,cos($2*3.1416/180)); $$=-1}

        | TAN exp                             {printf("Value of Tan(%d)
is %lf\n",$2,tan($2*3.1416/180)); $$=-1}

        | LOG exp                             {printf("Value of Log(%d)
is %lf\n",$2,log($2)/2.303); $$=-1}

        | LPP exp RPP                   { $$ = $2;  }


      ;

%%


int yyerror(char *s)
{
      printf("%s\n",s);
      return(0);
}


int main(void)
{
      freopen("in.txt","r",stdin);
      freopen("out.txt","w",stdout);
      yyparse();
      exit(0);
}
```