```
   (1992 ACM Mid-Central Regional Programming Contest
 Sample Solution to Problem #6 - Cha-Ching! Calculator)

program math ( input, output );
type
    string_40 = string[40];

const

     ten  : array[0..9] of string_40  = ( 'ten',
                                          'eleven',
                                          'twelve',
                                          'thirteen',
                                          'fourteen',
                                          'fifteen',
                                          'sixteen',
                                          'seventeen',
                                          'eighteen',
                                          'nineteen' );

     tens : array[2..9] of string_40  = ( 'twenty',
                                          'thirty',
                                          'forty',
                                          'fifty',
                                          'sixty',
                                          'seventy',
                                          'eighty',
                                          'ninety' );

     units : array[0..9] of string_40  = ( 'zero',
                                           'one',
                                           'two',
                                           'three',
                                           'four',
                                           'five',
                                           'six',
                                           'seven',
                                           'eight',
                                           'nine' );
var

   str   : string_40;
   num   : integer;
   total : integer;
   token : string_40;
   ifile,ofile: text;



procedure lookup ( s : string_40; VAR num : integer );
var
   index : integer;
   done  : boolean;

begin

     done := FALSE;

     index := -1;
```

```
      while ( NOT done ) and ( index < 9 ) do
      begin
            inc ( index );
            if ( ten[index] = s ) then
               done := TRUE;
      end;


      if ( done ) then
         num := num + 10 + index;

      index := 1;
      done := false;

      while ( NOT done ) and ( index < 9 ) do
      begin
            inc ( index );
            if ( tens[index] = s ) then
               done := TRUE;
      end;

      if ( done ) then
         num := num + 10*index;

      index := -1;
      done := false;

      while ( NOT done ) and ( index < 9 ) do
      begin
            inc ( index );
            if ( units[index] = s ) then
               done := TRUE;
      end;

      if ( done ) then
         num := num + index;

end;



function string_val ( s : string ) : integer;
var
   first  : string_40;
   second : string_40;
   place  : integer;
   number : integer;

begin

      place := pos('-',s);
      first := '';
      second := '';
      number := 0;

      if ( place > 0 ) then
      begin
         first := copy ( s, 1, place - 1 );
         second := copy ( s, place + 1, length(s) - place );
```

```
      end
      else
          second := s;

      if ( first <> '' ) then
         lookup ( first, number );

      lookup ( second, number );

      string_val := number;

end;



function val_string ( number : integer ) : string;
var
   temp : string[80];
   first : integer;
   second : integer;
   third : integer;
begin

     first := number div 10;
     second := number mod 10;

     if ( first > 1 ) then
     begin
          temp := tens[first];
          if ( second > 0 ) then
               temp := temp + '-' + units[second];
     end
     else
         if ( first = 1 ) then
            temp := ten[second]
         else
             temp := units[second];

     val_string := temp;

end;


procedure get_token ( VAR s : string_40 );
var
   temp : string_40;
   done : boolean;
   ch   : char;
begin

     temp := '';
     done := false;

     while ( not eoln(ifile) ) and ( not done ) do
     begin
          read (ifile, ch );
          if ( ch = ' ' ) then
             done := TRUE
          else
              temp := temp + ch;
```

```
     end;

     s := temp;

end;

{ the following are stubs to allow mutual recursion }

procedure term;forward;
procedure factor;forward;

procedure expression;
var
   temp   : integer;
   token2 : string_40;
begin

     term;

     while ( ( token = 'plus' ) or ( token = 'minus' ) ) do
     begin

          temp := total;
          token2 := token;

          term;

          if ( token2 = 'plus' ) then
             total := temp + total
          else
              total := temp - total;

     end;

end;


procedure term;
var
   token2 : string_40;
   temp   : integer;
begin

     factor;

     while ( ( token = 'times' ) or ( token = 'div' ) ) do
     begin

          token2 := token;
          temp := total;

          factor;

          if ( token2 = 'times' ) then
             total := temp * total
          else
              total := temp div total;

     end;
```

```
end;


procedure factor;
begin

     get_token ( token );

     if ( token = '(' ) then
     begin

          expression;
          get_token ( token );

     end
     else
     begin
         total := string_val ( token );
         get_token ( token );
     end;

end;


begin
     assign (ifile,'math.in');
     reset(ifile);
     assign (ofile,'math.out');
     rewrite(ofile);
     while not eof(ifile) do
     begin

          expression;
          readln(ifile);
          if ( total < 0 ) or ( total > 99 ) then
             writeln (ofile, 'a suffusion of yellow' )
          else
             writeln (ofile, val_string ( total ) );

     end;
     close(ifile);
     close(ofile);

end.
```