Help for Lab 2, Pascal Parser in Prolog

The lab exercise consists of 3 parts:-

(i) reading the source code from a file
(ii) lexical analysis
(iii) syntax analysis.

The Prolog code should be as below:-

```
lab2(File, Result) :-
  read_in(File, L), lexer(L, Tokens), parser(Tokens, Result).
```

Reading the Pascal source file may be done using the code from
Programming in Prolog, (4th Edition, Clocksin & Mellish 1994) Ch. 5

(see the lab spec web page– Clocksin & Mellish Lexical Analysis)
NOTE: this code has been modified for this lab (but not 100%!).

but with some modifications since the source code is read in from a file (done) and in addition **":=" should be handled as a special symbol.**

```
read_in(File, [W|Ws]) :-
see(File), get0(C), readword(C, W, C1), restsent(W, C1, Ws), seen.
```

How to handle the ':' (58) possibly followed by the '=' (61):

```
readword(C, W, C2) :- C = 58, get0(C1),  readwordaux(C, W, C1, C2).
```

```
readwordaux(C, W, C1, C2) :- C1  = 61, name(W, [C, C1]), get0(C2).
readwordaux(C, W, C1, C2) :- C1 \= 61, name(W, [C]), C1 = C2.
```

Check that the result from read_in is a list of lexemes.

The **Lexical Analyser's** task id to transform the list of lexemes, **L**, comprising keywords, special symbols, variable names and integers which is produced by **read_in(File, L)**, to a list of tokens.

The tokens are represented as integers.

The predicate lexer processes one lexeme at a time. This lexeme is always the first element in the list.

```
lexer([  ], [  ]).
lexer([H|T], [F|S]) :-match(H, F), lexer(T,S).
```

The **match** predicate had several definitions (around 20), one for each **keyword** and **special symbol** and one for matching of a **variable name** and finally a predicate to check that an **integer** is composed only of digits.

```
match(L, F) :-          L = 'program',    F is 256.
...
match(L, F) :-          L = ';',          F is 59.
...
match(L, F) :-          L = ':=',         F is 271.
...
match(L, F) :-
   name(L, [T|S]), char_type(T, digit), match_digit(S),    F is 272.

match_digit([  ]).
match_digit([H|T]) :-   char_type(H, digit), match_digit(T).
```

The Parser checks that what the lexical analyser delivers, follows the grammar rules of Pascal. These grammar rules must be written according to the Prolog syntax.

```
/*** Terminals = 'Facts' ***/
program      → [256].
...
scolon       → [59].
...
assign_op    → [271].


/*** Non-Terminals = 'Rules' ***/
prog         → header, var_part, stat_part.
...
parser(Tokens, Res) :-
  (prog(Tokens, Res), Res = [ ], write('Parse Succeed!'));
    write('Parse Fail!').
```

Testing of all the Pascal programs may be carried out using the following example:

```
testa :-   parseFiles( [ 'testok1.pas', 'testok2.pas','testok3.pas' ] ).

parseFiles([ ]).
parseFiles([H|T]) :-
     write('Testing '), write(H), nl,
     read_in(H,L), lexer(L, Tokens),  parser(Tokens, Result),
     nl, write(H), write(' end'), nl, nl,
     parseFiles(T).
```