Question. Write a PROLOG program read_file(F,L) that reads a file F and returns the list L of ascii codes that compose the file.

The basic idea for this question is to write an auxiliary predicate, read_text(L,Channel), using the get0(+Stream, -Char), to get each ascii code for each character. We got the following result from a test file.

```
1 ?- read_file('test.txt',L).
L = [104, 101, 108, 108, 111, 32, 119, 111, 114|...].
```

As it shown, there is a limit on the depth to prevent too long output. To solve this problem, we use the predicate set_prolog_flag/1.

```
?- set_prolog_flag(toplevel_print_options, [quoted(true), portray(true), max_depth(100), priority(699)]).
  true.
?- read_file('test.txt',L).L =
[104,101,108,108,111,32,119,111,114,108,100,44,32,119,104,97,116,32,97,114,101,32,121,111,
117,32,100,111,105,110,103,63,32,116,111,100,97,121,32,105,115,32,67,104,114,105,115,116,1
09,97,115,32,68,97,121,46,10,104,105,44,32,97,110,111,116,104,101,114,32,111,110,101,46,10,
119,104,97,116,32,97,98,111,117,116,32,116,104,105,115,63].
```

Question. Using the DCG(Definite Clause Grammar) formalism, build a syntactical parser detect_noccs_string(F,S,N) that unifies N with the number of occurrences of a string S in a file F.

The basic idea for this question is to solve it in several steps.
Step1. transfer the file into ascii using the auxiliary predicate, read_file('test1.txt',L).

```
?- read_file('test1.txt',L).
L =
[104,101,108,108,111,32,119,111,114,108,100,44,32,104,111,119,32,97,114,101,32,121,111,117
,63]
```

Step2. use lists of list to store each word. Every sublist is the ascii of the word, HERE we remove the some special signs '?' '.' '\n' ' ' ',':',etc.' (In test1.txt, we wrote, ” hello world, how are you?”)

```
?- read_file('test1.txt',L),list_word_ascii(L,Result).
L =
[104,101,108,108,111,32,119,111,114,108,100,44,32,104,111,119,32,97,114,101,32,121,111,117
,63],
Result =
[[104,101,108,108,111],[119,111,114,108,100],[],[104,111,119],[97,114,101],[121,111,117,63]] ;
```

Step3. Transfer the detected string into ascii by using name(). For example, we want to the number of occurrences of a string ‘world’ in a file test1.txt

```
?- name('world',ListString).
ListString = [119,111,114,108,100].
```

Step4. Count the number of occurrences of same word.

```
?-
read_file('test1.txt',L),list_word_ascii(L,Result),name('world',ListString),nbword2(Result,ListString
,N).
```

```
L =
[104,101,108,108,111,32,119,111,114,108,100,44,32,104,111,119,32,97,114,101,32,121,111,117
,63],
Result =
[[104,101,108,108,111],[119,111,114,108,100],[],[104,111,119],[97,114,101],[121,111,117,63]],
ListString = [119,111,114,108,100],
N = 1 ;
```

We define two predicate to count the number of occurrences of a string, nbword(L,S,N) and nbword2(L,S,N). The nbword2(L,S,N) is tail prolog.

```
nbword([],_,0):-!.
nbword([S|R],S,N):- !, nbword(R,S,T),N is T+1.
nbword([_|R],S,N):- nbword(R,S,N).

% nbword2 is tail prolog.
nbword2(L,S,N):- nbword2(L,S,N,0).
nbword2([S|R],S,N,T):- !, P is T +1, nbword2(R,S,N,P).
nbword2([_|R],S,N,T):- nbword2(R,S,N,T).
nbword2([],_,P,P).
```

To conclude, the following is the complete code.

```
detect_noccs_string(F,S,N):-
                                    open(F, read, Channel),
                                    read_file(F,ListTxt), % step1.
                                    list_word_ascii(ListTxt,Result), % step2.
                                    name(S,ListString), % step3.
                                    nbword2(Result,ListString,N),% nbword2 is tail prolog.% step4.
                                    close(Channel).
list_word_ascii([],[]).
list_word_ascii([L|Ls],[Result|Results]):-list2lol(L,Ls,Ys,Result),list_word_ascii(Ys,Results).
list2lol(X,[],[],[X]).
list2lol(X,[Y|Ys],[Y|Ys],[]) :- member(X, "! \n,?.:"),!.
list2lol(X,[Z|Xs],Ys,[X|Zs]) :- list2lol(Z,Xs,Ys,Zs).

nbword([],_,0):-!.
nbword([S|R],S,N):- !, nbword(R,S,T),N is T+1.
nbword([_|R],S,N):- nbword(R,S,N).

% nbword2 is tail prolog.
nbword2(L,S,N):- nbword2(L,S,N,0).
nbword2([S|R],S,N,T):- !, P is T +1, nbword2(R,S,N,P).
nbword2([_|R],S,N,T):- nbword2(R,S,N,T).
nbword2([],_,P,P).
```

Question. Generalize this parser to allow the evaluation of the occurrences of all the strings belonging to a list LS of strings. The resulting program detect_noccs_strings(F,LS,R) unify R with an association list that associates to each string of LS its number of occurrences.

The basic idea for this question is similar with the previous one. The only change is in nblwords(_,[],[]), we use list rather than number to be output.

```
?- detect_noccs_strings('test1.txt',['hello','are'],R).
R = [[hello,1],[are,1]] ;
```

```
detect_noccs_strings(F,LS,R):-
                                        open(F,read,Channel),
                                        read_file(F,ListTxt),
                                        list_word_ascii(ListTxt,Result),% Result is the lists of list,
which each one is the ascii of every word.
                                        nblwords(Result,LS,R),
                                        close(Channel).
nblwords(_,[],[]).
nblwords(Result,[L|Lr],[[L,N]|Nr]):- name(L,ListString),
      nbword2(Result,ListString,N), nblwords(Result,Lr,Nr).
```

Question. A tail recursion and Pretty print.

I already try my best to do the predicate in tail prolog.

In this part, I use the predicate write() to solve this problem.

```
pretty_print(L,N):-

      write('**********************************************************************')
,nl,
                                write('You can find that'),
                                write('    '),
                                write(N),
                                write('     '),
                                write('"'),
                                write(L),
                                write('"'),
                                write('    '),
                                write('occur in this novel.'),
                                nl.
```

When we test the code in a novel file, we can get the following results.

```
?- detect_noccs_strings('The Adventures of Huckleberry Finn_Mark
Twain.txt',['morning','love','flower'],N).
```

```
Welcome! you are opening this novel: The Adventures of Huckleberry Finn_Mark Twain.txt


This is a syntactical parser in PROLOG! Enjoy!
**********************************************************************
You can find that    2      "flower"    occur in this novel.
**********************************************************************
You can find that    9      "love"    occur in this novel.
**********************************************************************
You can find that    45      "morning"    occur in this novel.
N = [[morning,45],[love,9],[flower,2]] ;
```

Appendix (code)

```
% Ex.1 write a PROLOG program read_file(F,L) that reads a file F and returns the list L of ascii
codes that compose the file.

% in these codes, it will display all of the ascii of the letters no matter it is '\n' or 'space'
read_file(F, L) :-
                                open(F, read, Channel),
                                read_text(L, Channel),
                                close(Channel).
read_text(L,Channel) :-
                                get0(Channel, C),
                                read_text(L, Channel, C).
read_text([], _, -1):-!. /* end of file*/

read_text([C|L], Channel, C) :-
                                        get0(Channel, Cs),
                                        read_text(L, Channel, Cs).

% There is a limit on the depth to prevent too long output. You can change it with
set_prolog_flag/1
% ?- current_prolog_flag(toplevel_print_options,A).
% A = [quoted(true), portray(true), max_depth(10), spacing(next_argument)]. % it is the reason
there are 9 numbers in the list L.
% You can set it as follows.
% ?- set_prolog_flag(toplevel_print_options, [quoted(true), portray(true), max_depth(100),
priority(699)]).
% true.
%?-read_file('test.txt',L).L=
[104,101,108,108,111,119,111,114,108,100,44,119,104,97,116,97,114,101,121,111,117,100,111,
105,110,103,63,116,111,100,97,121,105,115,67,104,114,105,115,116,109,97,115,68,97,121,46,1
04,105,44,97,110,111,116,104,101,114,111,110,101,46,119,104,97,116,97,98,111,117,116,116,1
04,105,115,63].

% Ex.2 Using the DCG(Definite Clause Grammar) formalism, build a syntactical parser
detect_noccs_string(F,S,N) that unifies N with the number of occurrences of a string S in a file F.


% my idea:
% step1. transfer the file into ascii
% step2. use lists of list to store each word. every sublist is the ascii of the word, HERE  we
remove the some special signs '?' '.' '\n' ' ' ',':',etc.'
% step3. transfer the detected string into ascii
% step4. count the same word.
```

```
detect_noccs_string(F,S,N):-
                                        open(F, read, Channel),
                                        read_file(F,ListTxt), % step1.
                                        list_word_ascii(ListTxt,Result), % step2.
                                        name(S,ListString), % step3.
                                        nbword2(Result,ListString,N),% nbword2 is tail prolog.%
step4.
                                        close(Channel).


% when the input is: % L = [104,32,101,10,100,100,46,10,119,119,63,10,101], The output should
be R=[[104],[101],[100,100],[119,119],[101]].
% L is :'h e\ndd.\nww?\ne'. % we remove the '?' '.' '\n' ' ' ',' '"' ':' '-'
list_word_ascii([],[]).
list_word_ascii([L|Ls],[Result|Results]):-list2lol(L,Ls,Ys,Result),list_word_ascii(Ys,Results).
list2lol(X,[],[],[X]).
list2lol(X,[Y|Ys],[Y|Ys],[]) :- member(X, "! \n,?.:"),!.
list2lol(X,[Z|Xs],Ys,[X|Zs]) :- list2lol(Z,Xs,Ys,Zs).

nbword([],_,0):-!.
nbword([S|R],S,N):- !,
                        nbword(R,S,T),
                        N is T+1.
nbword([_|R],S,N):-
                            nbword(R,S,N).

% nbword2 is tail prolog.
nbword2(L,S,N):- nbword2(L,S,N,0).

nbword2([S|R],S,N,T):- !,
                        P is T +1,
                        nbword2(R,S,N,P).
nbword2([_|R],S,N,T):-
                            nbword2(R,S,N,T).
nbword2([],_,P,P).

%%% Ex3 generalize this parser to allow the evaluation of the occurrences of all the strings
belonging to a list LS of strings. The resulting program detect_noccs_strings(F,LS,R) unify R with
an association list that associates to each string of LS its number of occurrences.
% detect_noccs_strings('p.txt',['my','hehe'],N). % N = [2,1].
% the basic idea is the same as detect_noccs_string(F,S,N). the only change is nblwords. we use
list rather than number.
detect_noccs_strings(F,LS,R):-
                                        open(F,read,Channel),
```

```
                                        nl,
                                        write('Welcome! you are opening this novel: '),
                                        write(F),nl,nl,
                                        write('This is a syntactical parser in PROLOG! Enjoy!'),nl,
                                        read_file(F,ListTxt),
                                        list_word_ascii(ListTxt,Result),% Result is the lists of list,
which each one is the ascii of every word.
                                        nblwords(Result,LS,R),
                                        close(Channel).

nblwords(_,[],[]).
nblwords(Result,[L|Lr],[[L,N]|Nr]):-
                                                name(L,ListString),
                                                nbword(Result,ListString,N),
                                                nblwords(Result,Lr,Nr),
                                                pretty_print(L,N).
% I already try my best to do the predicate in tail prolog.

% some test and result:
% ?- read_file('test.txt',L).L =
[104,101,108,108,111,32,119,111,114,108,100,44,32,119,104,97,116,32,97,114,101,32,121,111,
117,32,100,111,105,110,103,63,32,116,111,100,97,121,32,105,115,32,67,104,114,105,115,116,1
09,97,115,32,68,97,121,46,10,104,105,44,32,97,110,111,116,104,101,114,32,111,110,101,46,10,
10,10,119,104,97,116,32,97,98,111,117,116,32,116,104,105,115,63].

% ?- detect_noccs_string('The Adventures of Huckleberry Finn_Mark Twain.txt','morning',N).
% N = 45 ;
% false.

% ?- detect_noccs_strings('The Adventures of Huckleberry Finn_Mark
Twain.txt',['morning','love','flower'],N).
% N = [[morning, 45], [love, 9], [flower, 2]]
% pretty print
pretty_print(L,N):-

      write('**********************************************************************')
,nl,
                                write('You can find that'),
                                write('    '),
                                write(N),
                                write('      '),
                                write('"'),
                                write(L),
                                write('"'),
```

```
        write('    '),
        write('occur in this novel.'),
        nl.
```