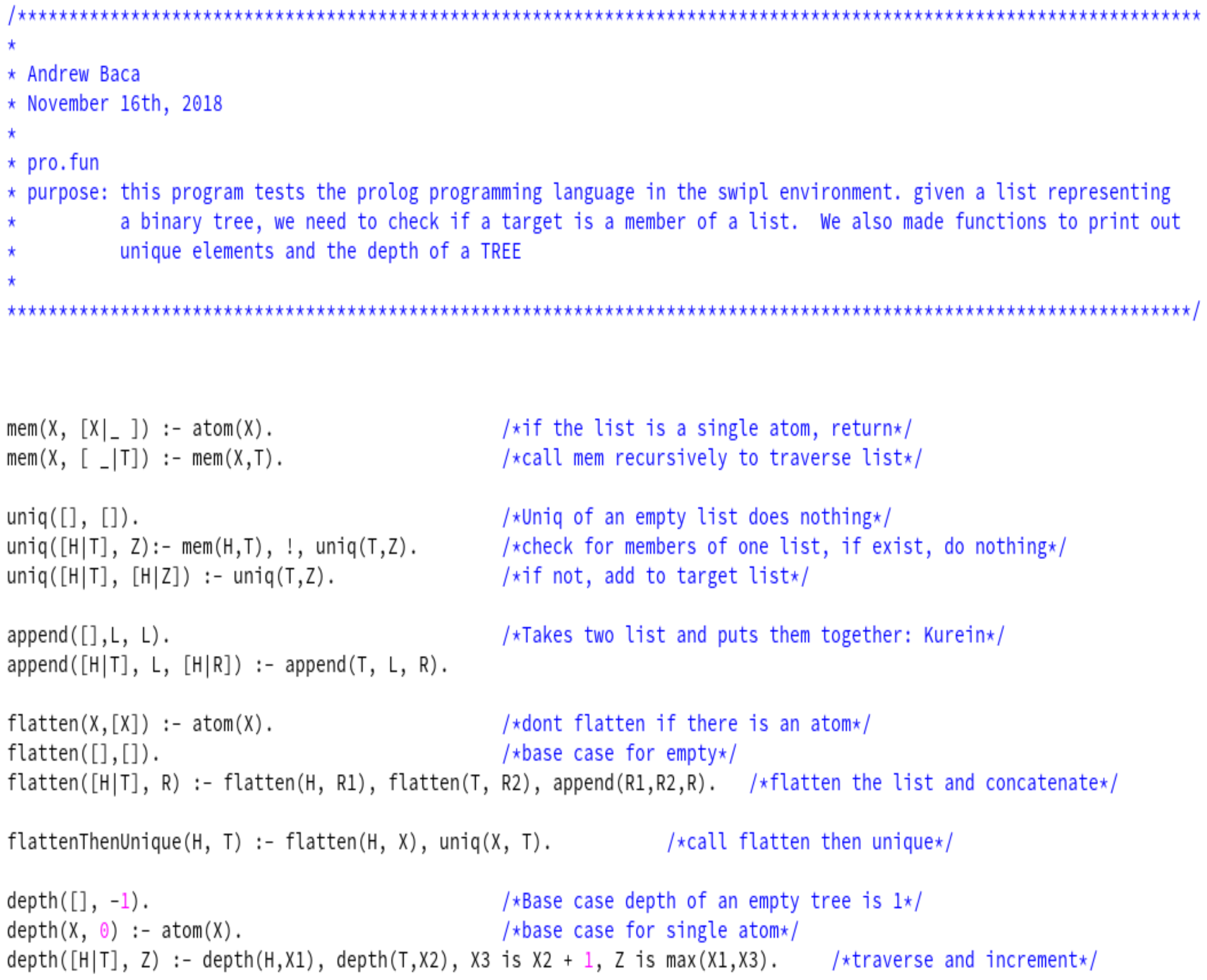
Andrew Baca
November 16th, 2018
CS471
Prolog programming assignment

Purpose:
This program is intended to extract a list in prolog in the swipl environment. Given a list representing a binary tree, we need to see if a target is a member of a list, as well as be able to flatten a list and see what elements are unique in a list, and print the depth of a list.

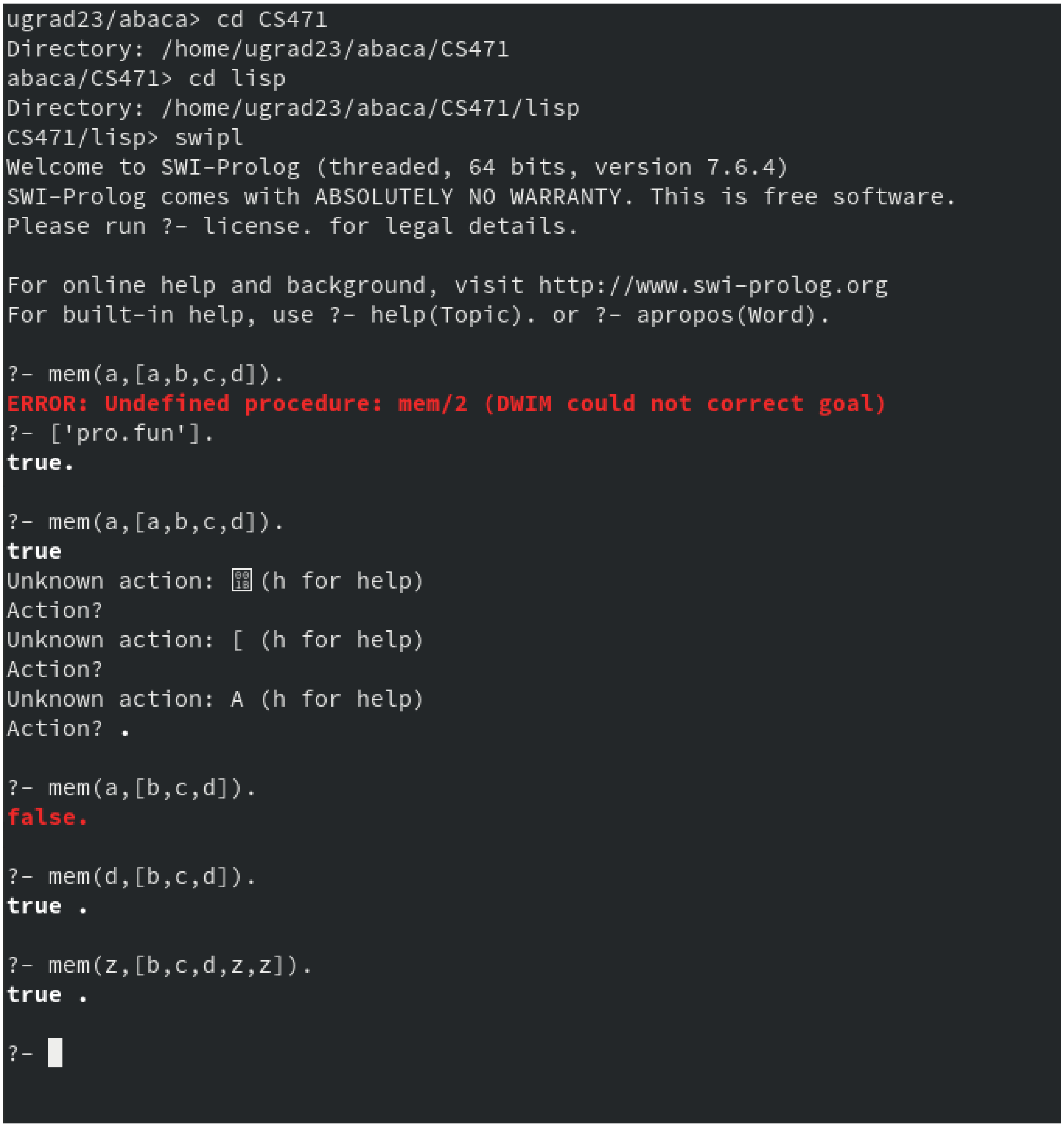
Code:



```
/****************************************************************************************************************
*
* Andrew Baca
* November 16th, 2018
*
* pro.fun
* purpose: this program tests the prolog programming language in the swipl environment. given a list representing
*          a binary tree, we need to check if a target is a member of a list.  We also made functions to print out
*          unique elements and the depth of a TREE
*
****************************************************************************************************************/



mem(X, [X|_ ]) :- atom(X).                        /*if the list is a single atom, return*/
mem(X, [ _|T]) :- mem(X,T).                       /*call mem recursively to traverse list*/

uniq([], []).                                     /*Uniq of an empty list does nothing*/
uniq([H|T], Z):- mem(H,T), !, uniq(T,Z).          /*check for members of one list, if exist, do nothing*/
uniq([H|T], [H|Z]) :- uniq(T,Z).                  /*if not, add to target list*/

append([],L, L).                                  /*Takes two list and puts them together: Kurein*/
append([H|T], L, [H|R]) :- append(T, L, R).

flatten(X,[X]) :- atom(X).                        /*dont flatten if there is an atom*/
flatten([],[]).                                   /*base case for empty*/
flatten([H|T], R) :- flatten(H, R1), flatten(T, R2), append(R1,R2,R).   /*flatten the list and concatenate*/

flattenThenUnique(H, T) :- flatten(H, X), uniq(X, T).          /*call flatten then unique*/

depth([], -1).                                    /*Base case depth of an empty tree is 1*/
depth(X, 0) :- atom(X).                           /*base case for single atom*/
depth([H|T], Z) :- depth(H,X1), depth(T,X2), X3 is X2 + 1, Z is max(X1,X3).     /*traverse and increment*/
```

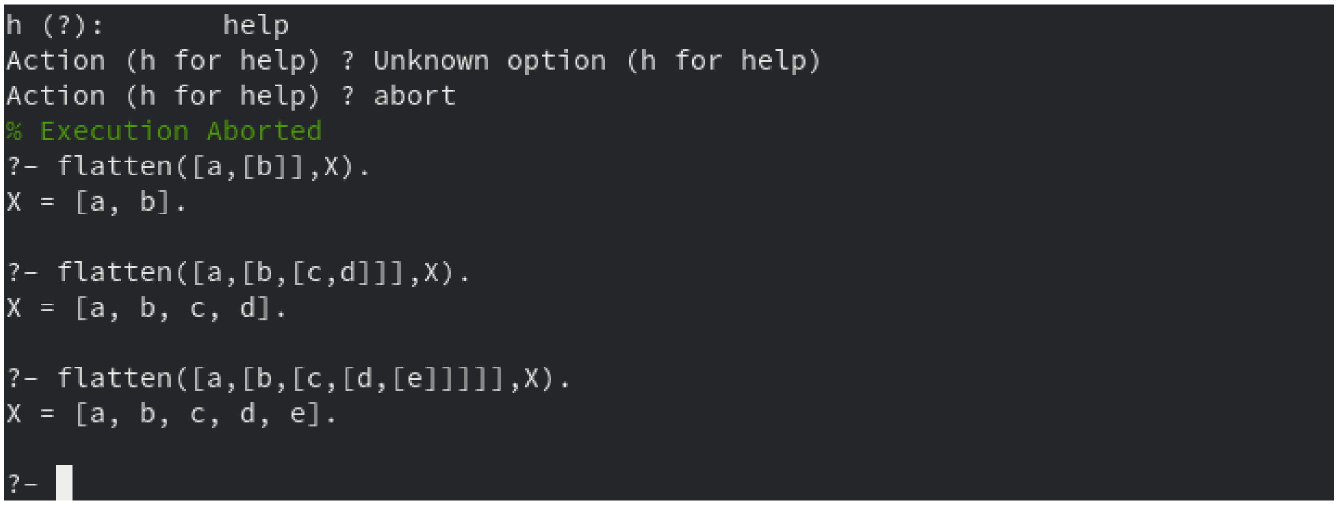
Output:

member:



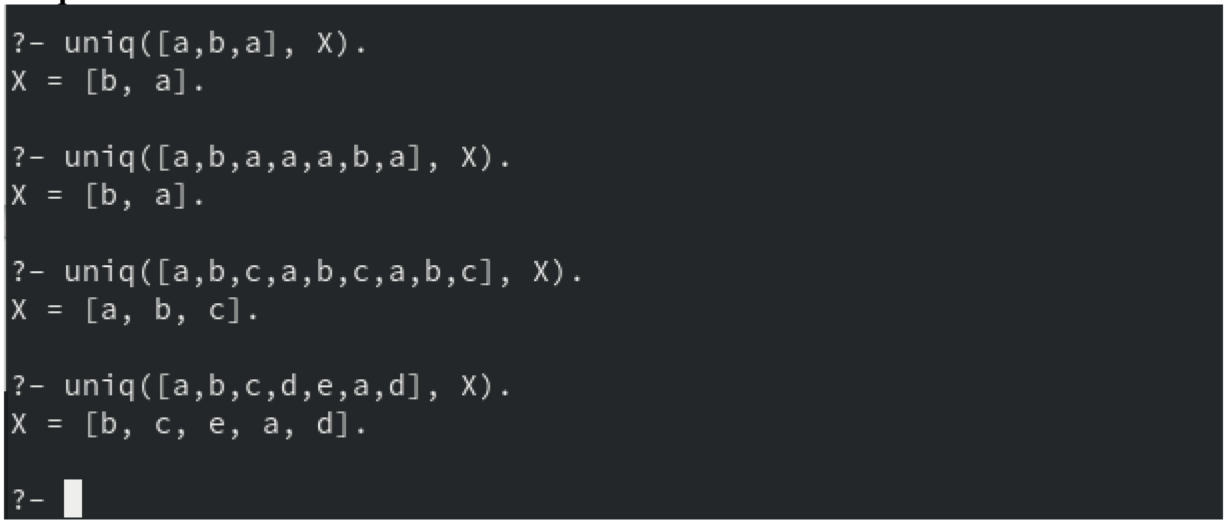
```
ugrad23/abaca> cd CS471
Directory: /home/ugrad23/abaca/CS471
abaca/CS471> cd lisp
Directory: /home/ugrad23/abaca/CS471/lisp
CS471/lisp> swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- mem(a,[a,b,c,d]).
ERROR: Undefined procedure: mem/2 (DWIM could not correct goal)
?- ['pro.fun'].
true.

?- mem(a,[a,b,c,d]).
true
Unknown action: ␛ (h for help)
Action?
Unknown action: [ (h for help)
Action?
Unknown action: A (h for help)
Action? .

?- mem(a,[b,c,d]).
false.

?- mem(d,[b,c,d]).
true .

?- mem(z,[b,c,d,z,z]).
true .

?- 
```

flatten:



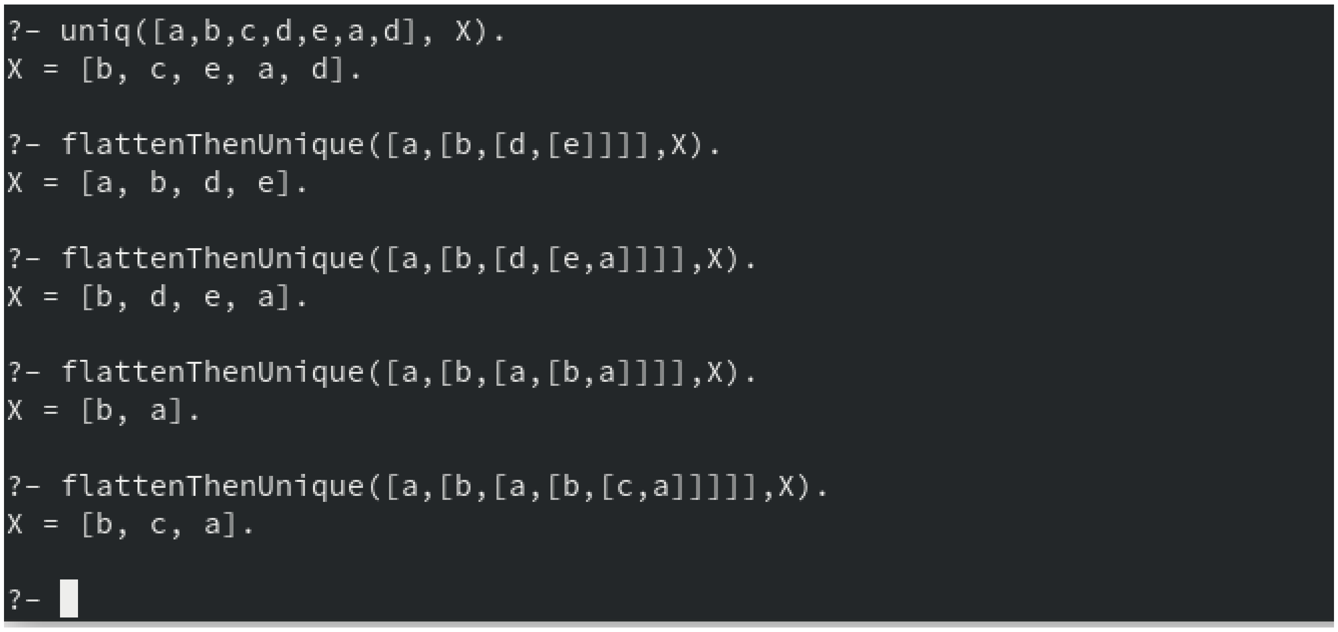
```
h (?):        help
Action (h for help) ? Unknown option (h for help)
Action (h for help) ? abort
% Execution Aborted
?- flatten([a,[b]],X).
X = [a, b].

?- flatten([a,[b,[c,d]]],X).
X = [a, b, c, d].

?- flatten([a,[b,[c,[d,[e]]]]],X).
X = [a, b, c, d, e].

?-
```

unique:



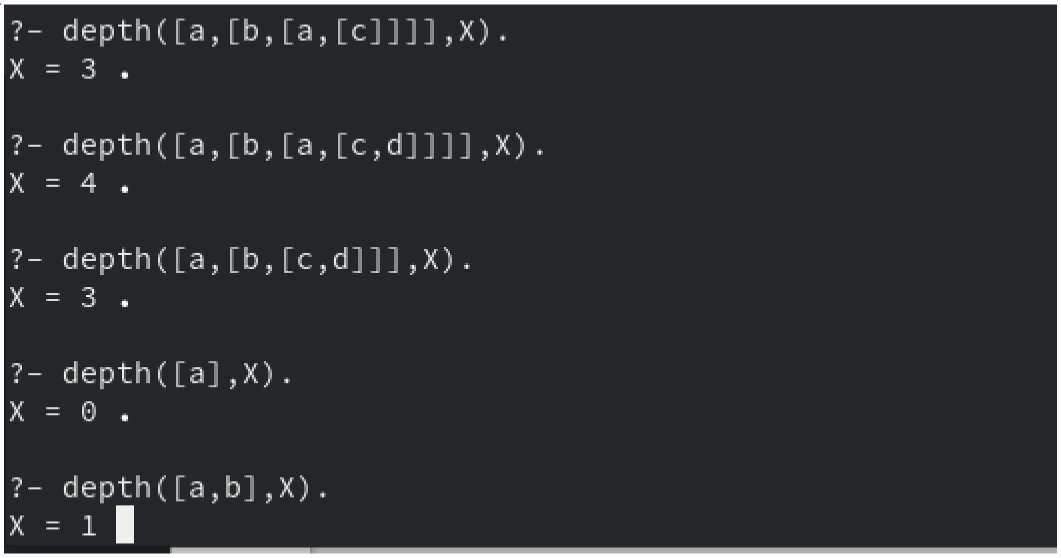
```
?- uniq([a,b,a], X).
X = [b, a].

?- uniq([a,b,a,a,a,b,a], X).
X = [b, a].

?- uniq([a,b,c,a,b,c,a,b,c], X).
X = [a, b, c].

?- uniq([a,b,c,d,e,a,d], X).
X = [b, c, e, a, d].

?-
```

flattenThenUnique:



```
?- uniq([a,b,c,d,e,a,d], X).
X = [b, c, e, a, d].

?- flattenThenUnique([a,[b,[d,[e]]]],X).
X = [a, b, d, e].

?- flattenThenUnique([a,[b,[d,[e,a]]]],X).
X = [b, d, e, a].

?- flattenThenUnique([a,[b,[a,[b,a]]]],X).
X = [b, a].

?- flattenThenUnique([a,[b,[a,[b,[c,a]]]]],X).
X = [b, c, a].

?-
```

depth:



```
?- depth([a,[b,[a,[c]]]],X).
X = 3 .

?- depth([a,[b,[a,[c,d]]]],X).
X = 4 .

?- depth([a,[b,[c,d]]],X).
X = 3 .

?- depth([a],X).
X = 0 .

?- depth([a,b],X).
X = 1
```