First 10 problems of P-99: Ninety-Nine Prolog Problems

– manisha-deshpande

Problem 1: Find the last element of a list.

Code:

```
last_ele(X,[X]).
last_ele(X,[_H|T]) :- last_ele(X,T).
```

Output:

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

?- last_ele(X,[b,3,a,4]).
X = 4 ;
```

Problem 2: Find the last but one element of a list.

Code:

```
sec_last_ele(X,[X|T]) :- length(T,1).
sec_last_ele(X,[_H|T]) :- sec_last_ele(X,T).
```

Output:

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

?- sec_last_ele(X,[2,3]).
X = 2 .
```

Problem 3: Find the K'th element of a list.

Code:

```
ele_at(X,[X|_T],1).

ele_at(X,[_H|T],I) :-
    I>1,
    I2 is I-1,
    ele_at(X,T,I2).
```

Output:

```
?- ele_at(X,[a,b,c,d,e],3).
X = c .

?- ele_at(X,[1,2,3,4,5,6],5).
X = 5 .
```

Problem 4: Find the number of elements of a list.

Code:

```
list_length(0,[]).

list_length(Res,[_H|T]) :-
    list_length(Temp,T),
    Res is Temp+1.
```

Output:

```
?- list_length(X,[]).
X = 0.

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

Problem 5: Reverse a list.

Code:

```
list_reverse(Res,[],Res).

list_reverse(X,[H|T],Temp) :- list_reverse(X,T,[H|Temp]).

list_reverse(X,List) :- list_reverse(X,List,[]).
```

Output:

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

?- list_reverse(X,[1,2,a,b,c]).
X = [c, b, a, 2, 1].
```

Problem 6: Find out whether a list is a palindrome.

Code:

```
list_reverse(Res,[],Res).

list_reverse(X,[H|T],Temp) :- list_reverse(X,T,[H|Temp]).

list_reverse(X,List) :- list_reverse(X,List,[]).

list_palindrome(List) :-
    list_reverse(X,List),
    X = List.
```

Output:

```
?- list_palindrome([a,b]).
false.

?- list_palindrome([m,a,d,a,m]).
true.
```

Problem 7: Flatten a nested list structure.

Code:

```
list_join(L1,L2,Res):- append(L1,L2,Res).

list_flatten([],[]).

list_flatten(X,[X]):- not(is_list(X)).

list_flatten([H|T],Res):-
    list_flatten(H,Temp1),
    list_flatten(T,Temp2),
    list_join(Temp1,Temp2,Res).
```

Output:

```
?- list_flatten([a,b,[1,2],c],X).
X = [a, b, 1, 2, c] .

?- list_flatten([a,b,[1,2,[3,4]],c],X).
X = [a, b, 1, 2, 3, 4, c] .
```

Problem 8: Eliminate consecutive duplicates of list elements.

Code:

```
list_eliminate_cons_dupl([],[]).
list_eliminate_cons_dupl([X],[X]).

list_eliminate_cons_dupl([H,H|T],Res):-
    list_eliminate_cons_dupl([H|T],Res).

list_eliminate_cons_dupl([H1,H2|T1],[H1|T2]):-
    H1\=H2,
    list_eliminate_cons_dupl([H2|T1],T2).
```

Output:

```
?- list_eliminate_cons_dupl([1,1,1,2],X).
X = [1, 2] .

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

Problem 9: Pack consecutive duplicates of list elements into sublists.

Code:

```
list_pack_cons_dupl([],[]).
list_pack_cons_dupl([X],[[X]]).

list_pack_cons_dupl([H,H|T1],[[H|T2]|T3]):-
    list_pack_cons_dupl([H|T1],[T2|T3]).

list_pack_cons_dupl([H1,H2|T1],[[H1]|T2]):-
    H1\=H2,
    list_pack_cons_dupl([H2|T1],T2).
```

Output:

```
?- list_pack_cons_dupl([1,2,3,4],X).
X = [[1], [2], [3], [4]] .

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

Problem 10: Run-length encoding of a list.

Code:

```
list_encode_helper([],[]).
list_encode_helper([[X|Y]|T1],[[Length,X]|T2]):-
    length([X|Y],Length),
    list_encode_helper(T1, T2).

list_encode(List,Res):-
    list_pack_cons_dupl(List,Temp),
    list_encode_helper(Temp,Res).
```

Output:

```
?- list_encode([1,2,3,4],X).
X = [[1, 1], [1, 2], [1, 3], [1, 4]] .

?- list_encode([a,a,b,b,c,d,a,a,e],X).
X = [[2, a], [2, b], [1, c], [1, d], [2, a], [1, e]] .
```
