SIT 211: LOGIC PROGRAMMING

Q&A

---

1. **Write a predicate named count that counts how many elements are in a nested list, at all levels. The first argument is the nested list and the second is the number of elements. Example:**

   ?- count ([a, [[b, c], d], e], R).
   R = 5

   Sol

   ```
   count([]).
   count([H|T],N):- is_list(H),!,count(H,N1), count(T,N2), N is N1+N2.
   count([_|T],N):- count(T,N1), N is N+1.
   ```

2. **Write a predicate twice(In,Out) whose left argument is a list, and whose right argument is a list consisting of every element in the left list written twice.**

   Examples:
   the query
   twice([a,4,buggle],X).
   should return
   X = [a,a,4,4,buggle,buggle]).

   the query
   twice([1,2,1,1],X).
   should return
   X = [1,1,2,2,1,1,1,1].

   Sol

   ```
   twice([],[]).
   twice([Ha|Ta],[Ha|Tb]):- twice(Ta,Tb).
   ```

3. **Define a predicate length (L, N) which holds iff N is the length of the list L.**

   Sol

   ```
   :- redefine_system_predicate(length( _, _ )).
   Length ([], 0).
   Length ([ _ | L], N) :- length (L, M), N is M+1.
   ```

4. **Define a predicate writeall that writes out the elements of a list, one per line.**

For example
?- writeall([alpha,'this is a string',20,[a,b,c]]).
should return:
alpha
this is a string
20
[a,b,c]
yes

Sol

```
writeall([]).
writeall([A|L]):- write(A), nl, writeall(L).
```

5. **Write a predicate named add_one that increments by 1 all the numbers in a binary tree. The first argument is the initial tree and the second argument is the result. Assume that the initial tree is given and that it can contain numbers or atoms. If a key is not a number, it remains unchanged in the result. For example :**
?- add_one(t(1,t(a,nil,nil),t(3,nil,t(b,nil,nil))), NewT).
NewT = t(2, t(a, nil, nil), t(4, nil, t(b, nil, nil)))

Sol

```
add_one(nil, nil).
add_one(t(X, L, R), t(X1, NL, NR)):-
number(X), !, X1 is X + 1, add_one(L, NL), add_one(R, NR).
add_one(t(X, L, R), t(X, NL, NR)):-
add_one(L, NL), add_one(R, NR).
```

6. **Write this predicate in Prolog: scan(L, Q) which succeeds if L and Q are lists of numbers, and Q is obtained by summing each possible prefix of L. Example: scan([2,3,5,7,11], Q) succeeds with Q = [0,2,5,10,17,28].**

Sol

```
scan([ ],[0]).

scan([H|T],[0|Z]) :- scan(T,Y), addToAll(H,Y,Z).

addToAll( _,[ ],[ ]).

addToAll(X,[H|T],[A|B]) :- A is X+H, addToAll(X,T,B).
```

7. **Define a predicate *add_up_list(L,K)* which, given a list of integers L, returns a list of integers in which each element is the sum of all the elements in L up to the same position. For example:**

```
?- add_up_list([1,2,3,4],K).

K = [1,3,6,10];

no
```

8.

Sol

```
add_up_list(L,K) :- aux(L,K,0).

aux([],[],_).

aux([X|L],[Y|K],Z) :- Y is Z+X, aux(L,K,Y).
```