

COMP210 –Introduction to Artificial Intelligence

Answers

If you encounter any problems or have any suggestions please email me tbc@csc.liv.ac.uk.

1. In the lecture member was defined as follows.

```
member(H, [H|Tail]).
```

```
member(X, [_|Tail]):-  
    member(X, Tail).
```

(a) 3 is a member of the list [1,2,3,4,5];

```
| ?- member(3, [1,2,3,4,5]).
```

yes

(b) 7 is a member of the list [1,2,3,4,5];

```
| ?- member(7, [1,2,3,4,5]).
```

no

(c) a is a member of the list [l,a,b,o,r,a,t,o,r,y]

```
| ?- member(a, [l,a,b,o,r,a,t,o,r,y]).
```

yes

(d) z is a member of the list [l,a,b,o,r,a,t,o,r,y]

```
| ?- member(z, [l,a,b,o,r,a,t,o,r,y]).
```

no

2. The output from tracing the first of the above is.

```
| ?- trace.
```

```
{The debugger will first creep -- showing everything (trace)}
```

```
yes
```

```
{trace}
```

```
| ?- member(3, [1,2,3,4,5]).
```

```
1      1 Call: member(3, [1,2,3,4,5]) ?
```

```
2      2 Call: member(3, [2,3,4,5]) ?
```

```
3      3 Call: member(3, [3,4,5]) ?
```

```
?      3      3 Exit: member(3, [3,4,5]) ?
```

```
?      2      2 Exit: member(3, [2,3,4,5]) ?
```

```
?      1      1 Exit: member(3, [1,2,3,4,5]) ?
```

```
yes
{trace}
```

3. In the lecture append was given as

```
/******  
% append(L1,L2,L3)  
% takes two lists L1 and L2 and returns a  
% list L3 which is the result of appending  
% L2 to L1  
*****/  
  
append([],L2,L2).  
  
append([H1|L1],L2,[H1|L3]):-  
    append(L1,L2,L3).
```

```
(a) append([3,4,5],[1,2],X)  
    | ?- append([3,4,5],[1,2],X).  
  
    X = [3,4,5,1,2] ?
```

```
yes
```

```
(b) append([1,2],[3,4,5],X)  
    | ?- append([1,2],[3,4,5],X).  
  
    X = [1,2,3,4,5] ?
```

```
yes
```

```
(c) append([3,4,5],[],X)  
    append([3,4,5],[],X).  
  
    X = [3,4,5] ?
```

```
yes
```

```
(d) append([], [3,4,5],X)  
    | ?- append([], [3,4,5],X).  
  
    X = [3,4,5] ?
```

```
yes
```

Try get them to look at the difference between (c) and (d), i.e. (c) requires 4 calls to append whereas (d) only requires 1.

4. There is a built in definition length so call this listlength or something similar. The definition of listlength is as follows

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

```
listlength([H|Tail],Len1):-
    listlength(Tail,Len2),
    Len1 is Len2 + 1.
```

(a) listlength([1,2,3,4,5],Y).
 | ?- listlength([1,2,3,4,5],Y).

Y = 5 ? ;

yes

(b) listlength([l,a,b,o,r,a,t,o,r,y],Z);
 | ?- listlength([l,a,b,o,r,a,t,o,r,y],Z).

Z = 10 ?

yes

(c) listlength([],Z);
 | ?- listlength([],Z).

Z = 0 ?

yes

(d) listlength(a,Z);
 | ?- listlength(a,Z).

no

The last example fails because “a” is not a list.

5. sumlist([],0).

```
sumlist([H|Tail],Z):-
    sumlist(Tail,Z1),
    Z is Z1 + H.
```

6. deleteone([],_,[]).

```
deleteone([H|Tail],H,Tail).
```

```
deleteone([H|Tail],Y,[H|Z]):-
    deleteone(Tail,Y,Z).
```

note if you use a variable in the first part of the definition then the prolog interpreter will complain about the singleton variable.

7. deleteall([],_,[]).

```
deleteall([H|Tail],H,Z):-
    deleteall(Tail,H,Z).
```

```
deleteall([H|Tail],Y,[H|Z]):-
    deleteall(Tail,Y,Z).
```

```
8. reverselist(Initial,Final):-
    reverselist(Initial, [],Final).

reverselist([],Final,Final).

reverselist([H|Tail],Rev,Final):-
    reverselist(Tail, [H|Rev], Final).
```

or using append.

```
reverselist1([], []).

reverselist1([H|Tail],Final):-
    reverselist1(Tail,Rev),
    append(Rev, [H],Final).
```

```
9.
oddeven([],even).

oddeven([_|Tail],Final):-
    oddeven1(Tail,Final).

oddeven1([],odd).

oddeven1([_|Tail],Final):-
    oddeven(Tail,Final).
```