

### Prolog Tutorial 3

1. Define and test the following predicates according to the specification given below:

a) `mysort(L,SL)`

SL is list L sorted and all duplicates removed. So, for example:

```
| ?- mysort([3,2,4,1,5,3,2], [1,2,3,4,5]).      gets the answer yes.  
| ?- mysort([22, 11, 22, 10], X).              gets the answer X = [10,11,22].
```

Use *setof* and *member*. Prolog has an inbuilt predicate *sort*. Do not use it for this exercise.

b) `rev(L, RevL)`

RevL is list L with the order of its elements reversed. So, for example:

```
|?- rev([1,2,3],R).          gets the answer R=[3,2,1].  
|?- rev([1,pears,[],[2,3]],R). gets the answer R=[[2,3],[],pears,1]
```

Prolog has an inbuilt predicate *reverse*. Do not use it for this exercise.

For this exercise give two different definitions for `rev`, one non-tail-recursive and one tail-recursive.

c) `followedBy(X,Y,L)`

X is followed by Y on list L. So, for example:

```
| ?- followedBy(4,6,[1,3,4,6,7]).      gets the answer yes.  
| ?- followedBy(4,X,[1,3,4,6,7]).      gets the answer X = 6.  
| ?- followedBy(X,Y,[1,3,4,6,7]).      gets the answers  
X = 1, Y = 3 ? ; X = 3, Y = 4 ? ;  
X = 4, Y = 6 ? ; X = 6, Y = 7 ? .
```

Here are some other queries you could try:

```
| ?- followedBy(1,2,[X, Y, Z]).      gets the answers  
X = 1, Y = 2 ? ; Y = 1, Z = 2.  
  
| ?- followedBy(1,2,X).              gets the answers  
X = [1,2|_A] ;  
X = [_A,1,2|_B] ;  
X = [_A,_B,1,2|_C] ;  
X = [_A,_B,_C,1,2|_D] etc.
```

d) `nextTo(X,Y,L)`

X and Y are next to one another on list L. So, for example:

`nextTo(3,6,[12,6,3,1,7])` and `nextTo(6,3,[12,6,3,1,7])` both get the answer yes.

e) `sumList(L,S)`

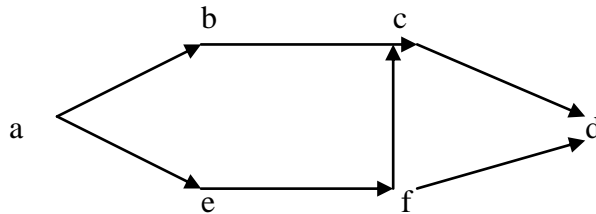
S is the sum of all integers on list L. Assume L is a list of positive or negative integers. So, for example:

```
?- sumList([1,3,4,6], S).              gets the answers S=14.
```

- f. Write Prolog clauses for the relation  $last(E,L)$  that finds the last element  $E$  of a list  $L$ .

2.

- a) Describe the graph below by a set of Prolog facts for the relation  $edge(X,Y)$  stating that there is an edge from node  $X$  to node  $Y$ .



- b) Using the relation  $edge$  write a Prolog program for the relation  $path(X,Y)$  that determines if there is path from node  $X$  to node  $Y$ .

- c) Modify the definition of relation  $path$  to define a new relation  $path/3$  such that  $path(X,Y,P)$  succeeds when  $P$  is a path from node  $X$  to node  $Y$ .

<https://tutorcs.com>

3. Write a Prolog program for the relation  $max(E,L)$  that determines the maximum element  $E$  of a list  $L$ .

WeChat: cstutorcs

4. Using the relation  $max$  and any other auxiliary relations you need to define, write a Prolog program for the relation  $max\_of\_all(E, Ls)$  to find the maximum element  $E$  of a list of lists  $Ls$ . So for example the query  $max\_of\_all(E, [[1],[2,4,1], [3,45,6,4]])$  succeeds with  $E=45$ . You can assume that in any call  $Ls$  is a list of elements of the same type, e.g. all numbers.