Functional and Logic Programming Fall 2017

S07 : Prolog (Introduction)

Professor : Le Peutrec Stephane
Assistant : Lauper Jonathan

Exercise 1 : Fibonacci

```
% fibonacci(+N:int,?F), succeed if F is the Nth Fibonnaci number
fibonacci(0,0).
fibonacci(1,1).
fibonacci(N,F) :-
    N > 1,
    N1 is N - 1,
    N2 is N - 2,
    fibonacci(N1,F1),
    fibonacci(N2,F2),
    F is F1 + F2.
```

Exercise 2

myLength

```
% myLength(?N:list,?L:int), succeed if L is the length of N
myLength(0,[]).
myLength(Res,[_|Xs]) :-
    myLength(N,Xs),
    Res is N + 1.
```

myAppend

```
% myAppend(?L1:list,?L2:list,?L3:list), succeed if L3 is the concatenation of L1 and L2
myAppend([],L,L).
myAppend([X|Xs],Ys,[X|L3]) :-
    myAppend(Xs,Ys,L3).
```

myLast

```
% myLast(?E,?L:list), succeed if E is the last element of L
myLast(E,[E|[]]).
myLast(E,[_|Xs]) :- myLast(E,Xs).
```

myReverse

```
% myReverse(?L1:list,?L2:list), succeed if L1 is the reverse of L2
myReverse([],[]). % not necessary, myReverse([],[],L3) would unify to myReverse([],[],[]).
myReverse(L1,L2) :- myReverse(L1,L2,[]).

% recursive execution, using accumulator to reverse L1 from myReverse/2
myReverse([],Acc,Acc).
myReverse([X|Xs],L2,Acc) :- myReverse(Xs,L2,[X|Acc]).
```

mySelect

```
% mySelect(?E,?L1:list,?L2:list), succeed if L2 is (L1 \ E)
mySelect(E,[E|Xs],Xs).
mySelect(E,[X|Xs],[X|Ys]) :- mySelect(E,Xs,Ys).
```

atoms

```
% atoms(?L:list),succeedif L contains only atom
% if L is [], we consider that L contains only atom
atoms([]).
atoms([X|Xs]) :-
    atom(X),
    atoms(Xs).
```