Programming Paradigms Exercise 4 - Prolog 3

Theodoros Chondrogiannis

2nd Semester 2016/17

1. In the previous exercise you implemented a program to compute the Nth Fibonacci number:

```
fib(1,1).
fib(2,1).
fib(N,F) :-
   N > 2,
   N2 is N-2, N1 is N-1,
   fib(N2,F2), fib(N1,F1),
   F is F1+F2.
```

   For large values, this version takes too long. Use accumulators to implement a faster version. Why is the version with accumulators so much faster?

2. In the previous exercise you also implemented a program to find the minimal element of a list:

```
minElem([Min], Min).
minElem([Head|Tail], Min ) :-
minElem(Tail, TailMin),
Head =< TailMin,
Min is Head.
minElem([Head|Tail], Min) :-
minElem(Tail, TailMin),
Head > TailMin,
Min is TailMin.
```

   Implement the same predicate `minElem` which return the minimal element of a list using accumulators.

3. The fictitious country of Elbonia issues stamps in the denominations of 15¢, 7¢, 3¢, and 1¢. Write a Prolog program that gets as input the total postage You have to pay and outputs how many stamps of each denomination you need to reach this total.

4. A directed graph can be represented in Prolog by listing the edges between nodes as facts. An edge from node `a` to node `b` would be represented by

```
edge(a,b).
```

Define a predicate `path(S,T,P)` that returns `true` if there is a simple (acyclic) path from `S` to `T`. Otherwise it returns `false`. (hint: you can use the `member/2` predicate of Prolog as `member(X, [One])` which returns true if the given list contains element `X`.)