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:

\[
\begin{align*}
\text{fib}(1,1). \\
\text{fib}(2,1). \\
\text{fib}(N,F) : - \\
& \quad N > 2, \\
& \quad N2 \text{ is } N-2, \ N1 \text{ is } N-1, \\
& \quad \text{fib}(N2,F2), \ \text{fib}(N1,F1), \\
& \quad F \text{ is } F1+F2.
\end{align*}
\]

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:

\[
\begin{align*}
\text{minElem}([\text{Min}], \text{Min}). \\
\text{minElem}([\text{Head}|\text{Tail}], \text{Min} ) : - \\
& \quad \text{minElem}(\text{Tail}, \text{TailMin}), \\
& \quad \text{Head} =< \text{TailMin}, \\
& \quad \text{Min} \text{ is } \text{Head}. \\
\text{minElem}([\text{Head}|\text{Tail}], \text{Min} ) : - \\
& \quad \text{minElem}(\text{Tail}, \text{TailMin}), \\
& \quad \text{Head} > \text{TailMin}, \\
& \quad \text{Min} \text{ is } \text{TailMin}.
\end{align*}
\]

Implement the same predicate \texttt{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

\[
\text{edge}(a,b).
\]

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