Exercises for the Course
Logic Programming Engineering
(Author Dr. W. Nauber)
Dr Paola Bruscoli, Peter Steinke, Amin Timany

Practical 3
Lists, Recursion II

Exercise 3.1
Define a predicate `delfirstn(List, N, DL)` which deletes the first `N` elements of a list `List`, and gives the result as list `DL`.
Example: `-delfirstn([a, b, c, d, e], 3, DL). gives DL = [d, e]`.

Exercise 3.2
Define a predicate `average(List, Av)` where `Av` is the average over the numeric elements of a list `List`.
Hints: Write an auxiliary predicate `sum(List, Sum)` for the sum of elements of list `List`, and use the system predicate `length(List, N)` which computes the number `N` of the elements of list `List`.
For a more efficient solution write an auxiliary predicate `sum_number(List, Sum, Number)` which calculates the sum and the number of elements of a list going only once through the list.
Example: `-average([2, 3, 5, 1, 6], Av). gives Av = 3.4`.

Exercise 3.3
Define a predicate `maxmember(List, Max)` where `Max` is the maximum of the elements of list `List`.
Example: `-maxmember([3, 1, 5, 2], Max). gives Max = 5`.

Exercise 3.4
Define predicates for the greatest common divisor (using EUCLID’s algorithm) and the lowest common multiple (`lcm(N1, N2) := (N1/gcd(N1, N2)) * N2`) of two non-zero integers `N1` and `N2`.
EUCLID’s algorithm for `gcd(N1, N2)` in pseudocode:

```
while N2>0 do
    Remainder:=N1 mod N2
    N1:=N2
    N2:=Remainder
end_while
gcd:=N1
```
Hint: Use the system operators `mod` and `//` for the modulo operation and the integer division.
Example: `-gcd(24, 18, D). gives D = 6 and lcm(24, 18, M). gives M = 72`.

Exercise 3.5
Define a predicate `horner(X, List, Y)` which for a given polynomial func-
tion in $X$ with coefficients represented in the list $List$ computes the function value $Y$ using HORNER’s rule

(e.g. $Y = 3 \times X^3 + 2 \times X^2 - 5 \times X + 4 = ((3 \times X + 2) \times X - 5) \times X + 4$).

Solve it a) without an accumulator;
   b) using an accumulator.

Hints for b):
To do this define an auxiliary predicate $\text{horn}(X,TList,Sum,Y)$ where $Sum$ is used as an accumulator (initialized with the first coefficient of list $List$ as a first sum). The list $TList$ is the tail of list $List$. The value $Y$ equals $Sum$ if $TList$ is empty - implement this as a fact.

Example: $? - \text{horner}(4,[3,2,-5,4],Y)$. gives $Y = 208$. 