Programming Paradigms
Seminar 3

**Exercise 1. (Finding an Element in a List)** Give a definition of {Member Xs Y} that tests whether Y is an element of Xs. For this assignment you have to use the truth values true and false. The equality test (that is ==) returns truth values and a function returning truth values can be used as condition in an if-expression. For example, the call {Member [a b c] b} should return true, whereas {Member [a b c] d} should return false.

**Exercise 2. (Taking and Dropping Elements)** Write two functions {Take Xs N} and {Drop Xs N}. The call {Take Xs N} returns the first N elements of Xs whereas the call {Drop Xs N} returns Xs without its first N elements. For example, {Take [1 4 3 6 2] 3} returns [1 4 3] and {Drop [1 4 3 6 2] 3} returns [6 2].

**Exercise 3. (Zip and UnZip)** The operation a # b constructs a tuple with label '#' and fields a and b which is also known as a pair. We can use it to implement lists-of-pairs, e.g [a#1 b#2 c#3]. A different view of this data structure is known as a pair-of-lists, e.g [a b c]#[1 2 3]. Two important functions that convert list-of-pairs to pair-of-lists and vice versa are Zip and UnZip.

a) Implement a function Zip that takes a pair Xs#Ys of two lists Xs and Ys (of the same length) and returns a pairlist, where the first field of each pair is taken from Xs and the second from Ys. For example, {Zip [a b c]#[1 2 3]} returns the pairlist [a#1 b#2 c#3].
b) The function UnZip does the inverse, for example {UnZip [a#1 b#2 c#3]} returns [a b c]#[1 2 3]. Give a specification and implementation of UnZip.

**Exercise 4. (Finding the Position of an Element in a List)** Write a function {Position Xs Y} that returns the first position of Y in the list Xs. The positions in a list start with 1. For example, {Position [a b c] c} returns 3 and {Position [a b c b] b} returns 2.
Try two versions:

1) one that assumes that Y is an element of Xs and
2) one that returns 0, if Y does not occur in Xs.

**Exercise 5. (Arithmetic Expressions Evaluation)** Suppose that you are given an arithmetic expression described by a tree constructed from tuples as follows:

1. An integer is described by a tuple int(N), where N is an integer.
2. An addition is described by a tuple add(X Y), where both X and Y are arithmetic expressions.
3. A multiplication is described by a tuple mul(X Y), where both X and Y are arithmetic expressions.

Implement a function Eval that takes an arithmetic expression and returns its value. For example, add(int(1) mul(int(3) int(4))) is an arithmetic expression and its evaluation returns 13.