

## Programming Paradigms

### Seminar 3

**Exercise 1. (Finding an Element in a List)** Give a definition of  $\{\text{Member } Xs \ Y\}$  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  $\{\text{Member } [a \ b \ c] \ b\}$  should return `true`, whereas  $\{\text{Member } [a \ b \ c] \ d\}$  should return `false`.

**Exercise 2. (Taking and Dropping Elements)** Write two functions  $\{\text{Take } Xs \ N\}$  and  $\{\text{Drop } Xs \ N\}$ . The call  $\{\text{Take } Xs \ N\}$  returns the first  $N$  elements of  $Xs$  whereas the call  $\{\text{Drop } Xs \ N\}$  returns  $Xs$  without its first  $N$  elements. For example,  $\{\text{Take } [1 \ 4 \ 3 \ 6 \ 2] \ 3\}$  returns  $[1 \ 4 \ 3]$  and  $\{\text{Drop } [1 \ 4 \ 3 \ 6 \ 2] \ 3\}$  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,  $\{\text{Zip } [a \ b \ c]\#[1 \ 2 \ 3]\}$  returns the pairlist  $[a\#1 \ b\#2 \ c\#3]$ .
- b) The function `UnZip` does the inverse, for example  $\{\text{UnZip } [a\#1 \ b\#2 \ c\#3]\}$  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  $\{\text{Position } Xs \ Y\}$  that returns the first position of  $Y$  in the list  $Xs$ . The positions in a list start with 1. For example,  $\{\text{Position } [a \ b \ c] \ c\}$  returns 3 and  $\{\text{Position } [a \ b \ c \ b] \ b\}$  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  $\text{int}(N)$ , where  $N$  is an integer.
2. An addition is described by a tuple  $\text{add}(X \ Y)$ , where both  $X$  and  $Y$  are arithmetic expressions.
3. A multiplication is described by a tuple  $\text{mul}(X \ Y)$ , where both  $X$  and  $Y$  are arithmetic expressions.

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