CS2104 Programming Language Concepts

Tutorial 02,(Chapter 1)

1 Warm Up

Given is the following declaration and assignment.

```
declare
R=r(1:[a b c] 4:[d [e [f]]] z:q(g h [10 11 [12 13]]))
```

Draw the value as a tree. In the following you have to give an expression composed of dot, width, and label functions that return the desired value. For example, for the value `g` the expression is `R.z.1` and for `r` it is `{Label R}`. If there is more than one possibility, give at least two expressions.

1. `b`	2. `q`	3. `12`	4. `nil`
5. `´|´`	6. `3`	7. `h`	8. `2`

2 Pattern Matching for Head and Tail

Give definitions for `Head` and `Tail` that use pattern matching.

3 Length of a List

Try both versions of `Length` as presented in the lecture.

4 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.

5 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] c}` returns `2`.
Try two versions: one that assumes that `Y` is an element of `Xs` and one that returns `0`, if `Y` does not occur in `Xs`.

6 Taking and Dropping Elements

Write two functions `{Take Xs N}` and `{Drop Xs N}`. `{Take Xs N}` returns the first `N` elements of `Xs` whereas `{Drop Xs N}` returns `Xs` without its first `N` elements.

7 Appending Two Lists

Given are two lists `Xs` and `Ys`. Give an inductive definition for a function that appends the elements of `Ys` to `Xs`.
Given an implementation `Append` of this function.

8 Reverting a List

Write a function `{Reverse Xs}` that returns a list with the elements of `Xs` in reverse order.
Use `Append` from the previous section.

9 Changing Head and Tail

Consider the following definitions of `Head` and `Tail`:

```
fun {Head Xs} X Xr in Xs=X|Xr X end
```

and

```
fun {Tail Xs} X Xr in Xs=X|Xr Xr end
```

Let your teaching assistent explain local declarations to you.
Then explain how these definitions differ from those given in the lecture. You might want to try some examples, consider in particular (try them one by one):

```
declare
Xs {Browse Xs} Ys={Tail Xs} Zs={Tail Ys} a={Head Zs}
```

10 A Different Encoding of Lists: Records

Lists are encoded with the atom `nil` and binary tuples (that is, a tuple of width 2) with label `´|´`.
Give an encoding of lists that uses the atom `el` (for empty lists) and binary records with label `cons` and features `head` and `tail`. The head and tail of a cons should be stored under the feature with that name.
Write `Length`, `Member` and `Append` that use this list encoding (use pattern matching).
Which encoding do you like better?

Do you have an idea how one could write programs that can compute with lists independent of the particular encoding?