Programming Languages and Types
Assignment 3

In addition to group exercises, there are compulsory homework assignments which you are to work on in groups of three students. To be admitted to the exam, you have to hand in reasonable solutions for all but two assignments, and you have to present your homework in one of the exercises classes. Please hand in your homework by email to \texttt{mailto:pllecture@informatik.uni-marburg.de} When handing in, please CC your mail to the other members of your team. Note: group exercises do not have to be handed in, only the homework assignments.

The exercise descriptions are published on \url{http://www.uni-marburg.de/fb12/ps/teaching/ws10/plt} on Thursday evenings or Friday after the lecture, and have to be handed in until Tuesday of the following week. The deadline for this assignment is November 09, 2010.

G3.1 Church encodings

Implement the following data types as Church encodings:

1. the natural numbers
2. finite lists
3. binary trees

Write an identity function for each of the encodings.

G3.2 Using Church encodings

Write the following functions using the Church encodings of the previous exercise:

1. the factorial function of type \( Nat \rightarrow Nat \)
2. the addition function of type \( Nat \rightarrow Nat \rightarrow Nat \)
3. the functions that computes the length of a list
4. the functions that computes the depth of a tree
G3.3 Fixpoint combinator

Consider the following call-by-value fixpoint combinator:

\[ Z = \lambda f. (\lambda x. f (\lambda y. x x y)) (\lambda x. f (\lambda y. x x y)) \]

Use \( Z \) to define

1. the factorial function

2. the mutually recursive functions \( \text{even} \) and \( \text{odd} \), defined as follows:

\[
\begin{align*}
\text{even} \ 0 &= \top \\
\text{even} \ n &= \text{odd} \ (n - 1) \\
\text{odd} \ 0 &= \bot \\
\text{odd} \ n &= \text{even} \ (n - 1)
\end{align*}
\]

H3.1 Using church encodings

Write the following functions using the Church encodings from exercise G3.1.

1. the multiplication function of type \( \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat} \)

2. the exponentiation function of type \( \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat} \)

3. the functions that builds the sum of all numbers in a list, type \( \text{List Nat} \rightarrow \text{Nat} \)

4. the functions that builds the sum of all numbers in a binary tree, type \( \text{Tree Nat} \rightarrow \text{Nat} \)

H3.2 Total functional programming

Read the article “Total functional programming” by David Turner (link is on the course website) and answer the following questions.

1. What are the advantages of total functional programming?

2. What is the difference between data and codata?

3. In your opinion, why is total functional programming not applied in practice?