

### Exercise 1 ( $\beta$ -reduction)

A term  $t$  is in  $\beta$ -normal form if there is no term  $t'$  such that  $t \rightarrow_{\beta} t'$ . List all terms  $t$  such that:

$$(\lambda x. (\lambda x y. x) z y) ((\lambda x. x x) (\lambda x. x x) ((\lambda x y. x) y)) \rightarrow_{\beta}^* t$$

Which are normal forms?

### Exercise 2 (Lists in $\lambda$ -calculus)

Specify  $\lambda$ -terms for `nil`, `cons`, `hd`, `tl` and `null`, that encode lists in the  $\lambda$ -calculus. Show that your terms satisfy the following conditions:

$$\begin{array}{llll} \text{null nil} & \rightarrow_{\beta}^* \text{true} & \text{hd (cons } x \text{ l)} & \rightarrow_{\beta}^* x \\ \text{null (cons } x \text{ l)} & \rightarrow_{\beta}^* \text{false} & \text{tl (cons } x \text{ l)} & \rightarrow_{\beta}^* l \end{array}$$

*Hint:* Use pairs.

### Homework 3 (Substitution Lemma)

Show that, given  $x \neq y$  and  $x \notin \text{FV}(u)$ :

$$s[t/x][u/y] = s[u/y][t[u/y]/x]$$

### Homework 4 (Trees in $\lambda$ -calculus)

Encode a datatype of binary trees in lambda calculus. Specify the operations `tip` and `node` that construct trees, as well as `isTip`, `left`, `right`, and `value`. Each `tip` should carry a value, whereas each `node` should consist of two subtrees.

Show that the following holds:

$$\begin{aligned} \text{isTip (tip } a) &\rightarrow_{\beta}^* \text{true} \\ \text{isTip (node } x \ y) &\rightarrow_{\beta}^* \text{false} \\ \text{value (tip } a) &\rightarrow_{\beta}^* a \\ \text{left (node } x \ y) &\rightarrow_{\beta}^* x \\ \text{right (node } x \ y) &\rightarrow_{\beta}^* y \end{aligned}$$

### Homework 5 (Alternative Encoding of Lists)

In this exercise, we consider an alternative encoding of lists. The list  $[x, y, z]$ , for instance, will now be encoded as:  $\lambda cn. cx(cy(czn))$  (speaking in terms of functional programming, each list now encodes its corresponding *fold*). As in the tutorial, define the functions `nil`, `cons`, `hd`, and `null` for this encoding and show that they satisfy the same conditions. You do not need to define `tl`.

### Homework 6 (Multiplication)

Define multiplication as a closed  $\lambda$ -term using `add` but no other helper functions and demonstrate its correctness on an example.