CMPSC 461: Programming Language Concepts
Assignment 1 Solution

Problem 1 [9pt] Add parentheses to the following lambda terms so that the grouping of sub-terms becomes explicit. For example, the term $\lambda x.\ x\ \lambda y.\ y$ with parentheses is $\lambda x.\ (x\ (\lambda y.\ y))$.

a) (3pt) $\lambda x.\lambda y.\ x\ y\ \lambda z.\ x\ z$

Solution: $\lambda x.\Big(\lambda y.\ ((x\ y)\ (\lambda z.\ (x\ z)))\Big)$

b) (3pt) $\lambda x.\lambda y.\ x\ \lambda x.\ x\ y\ y$

Solution: $\lambda x.\Big(\lambda y.\ (x\ (\lambda x.\ ((x\ y)\ y)))\Big)$

c) (3pt) $\lambda x.\ x\ \lambda y.\ x\ x\ y$

Solution: $\lambda x.\ \Big(x\ (\lambda y.\ ((x\ x)\ y))\Big)$

Problem 2 [6pt] For each of following terms, connect all bound variables to their definitions with lines. For example, the answer for $\lambda x.\ x\ x\ y$ should be $\lambda\, x.\ x\ \ x\ \ y$.

a) (3pt) $\lambda x.\ y\ x\ \lambda x.\ x\ y$

Solution: $\lambda\, x.\ y\ x\ \lambda\, x.\ x\ \ y$

b) (3pt) $\lambda y.\ \lambda z.\ z\ y\ \lambda y.\ x$

Solution: $\lambda\, y.\ \lambda\, z.\ z$

Problem 3 [10pt] Fully ev ction is possible.

a) (5pt) $((\lambda x\ y.\ x\ y)\ (\lambda x.\ y))\ u$

Solution:

$$
\begin{aligned}
& ((\lambda x\ y.\ x\ y)\ (\lambda x.\ y))\ u \\
&= ((\lambda x.\ (\lambda y.\ x\ y))\ (\lambda x.\ y))\ u && \text{(desugar term } \lambda x\ y.\ x) \text{)} \\
&= ((\lambda x.\ (\lambda z.\ x\ z))\ (\lambda x.\ y))\ u && (\alpha - \text{reduction}) \\
&= (\lambda z.\ (\lambda x.\ y)\ z)\ u && (\beta - \text{reduction}) \\
&= (\lambda x.\ y)\ u && (\beta - \text{reduction}) \\
&= y && (\beta - \text{reduction})
\end{aligned}
$$

b) (5pt) $((\lambda x.\ x)\ (\lambda y\ z.\ y))\ z$

Solution:

$$
\begin{aligned}
& ((\lambda x.\ x)\ (\lambda y\ z.\ y))\ z \\
&= (\lambda y\ z.\ y)\ z && (\beta - \text{reduction}) \\
&= (\lambda y.\ (\lambda z.\ y))\ \ z && \text{(desugar term } \lambda y\ z.\ y\text{)} \\
&= (\lambda y.\ (\lambda u.\ y))\ z && (\alpha - \text{reduction}) \\
&= \lambda u.\ z && (\beta - \text{reduction})
\end{aligned}
$$

Problem 4 [9pt] Recall that under Church encoding, we have the following definitions:

$$\texttt{IF} \triangleq \lambda b\ t\ f.\ b\ t\ f \qquad \texttt{TRUE} \triangleq \lambda t\ f.\ t \qquad \texttt{FALSE} \triangleq \lambda t\ f.\ f$$

a) (4pt) Fully evaluate $(\lambda x.\ (x\ y\ \texttt{TRUE}))\ \texttt{FALSE}$ so that no further β-reduction is possible.. **Solution**:

$$\begin{aligned}
& (\lambda x.\ (x\ y\ \texttt{TRUE}))\ \texttt{FALSE} & \\
= & (\texttt{FALSE}\ y\ \texttt{TRUE}) & (\beta - \text{reduction}) \\
= & (\lambda t\ f.\ f)\ y\ \texttt{TRUE} & (\text{definition of } \texttt{FALSE}) \\
= & \texttt{TRUE} & (\beta - \text{reduction})
\end{aligned}$$

b) (5pt) Show that $(\texttt{IF FALSE TRUE FALSE}) = \texttt{FALSE}$ under such encoding. **Solution**:

$$\begin{aligned}
& (\texttt{IF FALSE TRUE FALSE}) & \\
= & (\lambda b\ t\ f.\ b\ t\ f)\ \texttt{FALSE TRUE FALSE} & (\text{definition of } \texttt{IF}) \\
= & \texttt{FALSE TRUE FALSE} & (\beta - \text{reduction}) \\
= & (\lambda t\ f.\ f)\ \texttt{TRUE FALSE} & (\text{definition of } \texttt{TRUE}) \\
= & \texttt{FALSE} & (\beta - \text{reduction})
\end{aligned}$$

Problem 5 [16pt] In the lecture, we have used the pure λ-calculus to construct natural numbers and encoded some operations on them. In such encoding, a number n is encoded as a function $\lambda f\ z.\ f^n\ z$ (denoted as $\underline{n}$). In this problem, we will defi
try to proceed by assuming th ven in lectures and previous quest

a) (4pt) Define a function ISZERO in λ-calculus, so that given a number $\underline{n}$, it returns TRUE (the encoding of true) if $\underline{n} = \underline{0}$; FALSE (the encoding of false) if $\underline{n} \neq \underline{0}$. H plying the second term multiple times to the first term returns FAL eturns TRUE.
Solution:
$\texttt{ISZERO} \triangleq \lambda n.\ n\ (\lambda f.\ \texttt{FALSE})\ (\texttt{TRUE})$

b) (4pt) Define a function PRED in λ-calculus, so that given a number $\underline{n}$, the function returns its predecessor, assuming the predecessor of $\underline{0}$ is $\underline{0}$ Hint: follow the idea in Problem 4a. You might need to use PAIR.
Solution:
$\texttt{PRED} \triangleq \lambda n.\ \texttt{RIGHT}\ (n\ (\lambda p.\ \texttt{PAIR}\ (\texttt{SUCC}\ (\texttt{LEFT}\ p))\ (\texttt{LEFT}\ p))\ (\texttt{PAIR}\ \underline{0}\ \underline{0}))$

c) (4pt) Use your encoding of PRED to define a subtraction function MINUS, so that MINUS $\underline{n_1}$ $\underline{n_2}$ returns $\underline{n_1 - n_2}$ when $n_1 \geq n_2$, and $\underline{0}$ otherwise.
Solution:
$\texttt{MINUS} \triangleq \lambda n_1\ n_2.\ n_2\ \texttt{PRED}\ n_1$

d) (4pt) Recall that in lecture note 2, we have defined MULT (the encoding of $\times$) based on PLUS (the encoding of +): $\texttt{MULT} \triangleq \lambda n_1\ n_2.\ (n_1\ (\texttt{PLUS}\ n_2)\ \underline{0})$. Try to define MULT on Church numerals without using PLUS (replacing PLUS with an equivalent term under α/β reduction doesn't count as a solution). Hint: by definition, we have $\underline{n}\ f = \lambda z.\ f^n\ z$, which can be interpreted as repeating an arbitrary function f for n times to parameter z. The goal here is to repeat an arbitrary function f for $n_1 \times n_2$ times (according to the definition of $\underline{n_1 \times n_2}$).
Solution:
$\texttt{MULT} \triangleq \lambda n_1\ n_2\ f.\ (n_2\ (n_1\ f))$