Homework 2

Problem 1. Find λ terms representing the logical or and not functions.

Solution. Consider $\lambda xy.x\mathbf{t}y$ for or, then we have the following:
or $\mathbf{t}\mathbf{t} \twoheadrightarrow_\beta \mathbf{t}\mathbf{t}\mathbf{t} \rightarrow_\beta (\lambda y.\mathbf{t})\mathbf{t} \rightarrow_\beta \mathbf{t}$,
or $\mathbf{t}\mathbf{f} \twoheadrightarrow_\beta \mathbf{t}\mathbf{t}\mathbf{f} \rightarrow_\beta (\lambda y.\mathbf{t})\mathbf{f} \rightarrow_\beta \mathbf{t}$,
or $\mathbf{f}\mathbf{t} \twoheadrightarrow_\beta \mathbf{f}\mathbf{t}\mathbf{t} \rightarrow_\beta (\lambda y.y)\mathbf{t} \rightarrow_\beta \mathbf{t}$,
or $\mathbf{f}\mathbf{f} \twoheadrightarrow_\beta \mathbf{f}\mathbf{t}\mathbf{f} \rightarrow_\beta (\lambda y.y)\mathbf{f} \rightarrow_\beta \mathbf{f}$.
Consider $\lambda x.x\mathbf{f}\mathbf{t}$ for not, then we have:
not $\mathbf{t} \rightarrow_\beta \mathbf{t}\mathbf{f}\mathbf{t} \rightarrow_\beta (\lambda y.\mathbf{f})\mathbf{t} \rightarrow_\beta \mathbf{f}$,
not $\mathbf{f} \rightarrow_\beta \mathbf{f}\mathbf{f}\mathbf{t} \rightarrow_\beta (\lambda y.y)\mathbf{t} \rightarrow_\beta \mathbf{t}$.

Problem 2. Prove that
(a) $\mathbf{add}\,\overline{m}\,\overline{n} \twoheadrightarrow_\beta \overline{m+n}$.
(b) $\mathbf{mult}\,\overline{m}\,\overline{n} \twoheadrightarrow_\beta \overline{m\cdot n}$.

Solution.
(a) $\mathbf{add}\,\overline{m}\,\overline{n} \equiv (\lambda mnfx.nf(mfx))\overline{m}\,\overline{n} \twoheadrightarrow_\beta \lambda fx.\overline{n}f(\overline{m}fx)$
$\equiv \lambda fx.(\lambda fx.f^n x)f((\lambda fx.f^m x)fx) \twoheadrightarrow_\beta \lambda fx.(\lambda fx.f^n x)f(f^m x)$
$\twoheadrightarrow_\beta \lambda fx.(f^n(f^m x)) \equiv \lambda fx.f^{m+n}x \equiv \overline{m+n}$
(b) $\mathbf{mult}\,\overline{m}\,\overline{n} \equiv (\lambda mnf.n(mf))\overline{m}\,\overline{n} \twoheadrightarrow_\beta \lambda f.\overline{n}(\overline{m}f)$
$\equiv \lambda f.(\lambda fx.f^n x)((\lambda fx.f^m x)f) \rightarrow_\beta \lambda f.(\lambda fx.f^n x)(\lambda x.f^m x)$
$\rightarrow_\beta \lambda f.(\lambda x.(\lambda x.f^m x)^n x) \equiv \lambda fx.(\lambda x.f^m x)^{n-1}((\lambda x.f^m x)x)$
$\rightarrow_\beta \lambda fx.(\lambda x.f^m x)^{n-1}(f^m x) \equiv \lambda fx.(\lambda x.f^m x)^{n-2}((\lambda x.f^m x)(f^m x))$
$\rightarrow_\beta \lambda fx.(\lambda x.f^m x)^{n-2}(f^m(f^m x)) \equiv \lambda fx.(\lambda x.f^m x)^{n-2}(f^{2m}x)$
$\twoheadrightarrow_\beta \lambda fx.(\lambda x.f^m x)^{n-k}(f^{k\cdot m}x)[k=1,2,...,n]$
$\twoheadrightarrow_\beta \lambda fx.f^{n\cdot m}x \equiv \overline{m\cdot n}$

Problem 3. Compute the β-normal forms of the following terms. Are they strongly normalizable?
(a) $(\lambda xy.yx)((\lambda x.xx)(\lambda x.xx))(\lambda xy.y)$.
(b) $(\lambda xy.yx)(\mathbf{kk})(\lambda x.xx)$.

Solution.

(a) $(\lambda xy.yx)((\lambda x.xx)(\lambda x.xx))(\lambda xy.y) \to_\beta (\lambda y.y((\lambda x.xx)(\lambda x.xx)))(\lambda xy.y)$
$\to_\beta (\lambda xy.y)((\lambda x.xx)(\lambda x.xx)) \to_\beta \lambda y.y \equiv \mathbf{i}$, which is a normal form.
It is not strongly normalizable, because $(\lambda x.xx)(\lambda x.xx) \to_\beta (\lambda x.xx)(\lambda x.xx)$, so the whole term can be β-reduced to itself if we conduct the reduction on the $((\lambda x.xx)(\lambda x.xx))$ part first, creating an infinite sequence of reductions.

(b) $(\lambda xy.yx)(\mathbf{kk})(\lambda x.xx) \twoheadrightarrow_\beta (\lambda x.xx)(\mathbf{kk}) \equiv (\lambda x.xx)((\lambda xy.x)\mathbf{k})$
$\to_\beta (\lambda x.xx)(\lambda y.\mathbf{k}) \to_\beta (\lambda y.\mathbf{k})(\lambda y.\mathbf{k}) \to_\beta \mathbf{k}$, which is a normal form.
It is strongly normalizable. Easily we can list all the ways of β-reduction of the term, which all end in finite steps.

Problem 4. Find a representation of the following functions on integers

(a) $f(n) = \begin{cases} \text{true} & n \text{ is even,} \\ \text{false} & n \text{ is odd.} \end{cases}$

(b) $\exp(n, m) = n^m$.

(c) $\text{pred}(n) = \begin{cases} 0 & \text{if } n = 0, \\ n-1 & \text{otherwise.} \end{cases}$ (Hard)

Solution.

(a) Consider $f \equiv \lambda n.n\, \text{not}\, \mathbf{t} \equiv \lambda n.n(\lambda x.x\mathbf{ft})\mathbf{t}$, then we have:

$f(\overline{n}) \equiv \overline{n}\, \text{not}\, \mathbf{t} \equiv (\lambda fx.f^n x)\text{not}\, \mathbf{t} \to_\beta \text{not}^n\mathbf{t} \twoheadrightarrow_\beta \begin{cases} \mathbf{t} & n \text{ is even,} \\ \mathbf{f} & n \text{ is odd.} \end{cases}$

(b) Consider $\exp \equiv \lambda nm.m(\mathbf{mult}\, n)\overline{1} \equiv \lambda nm.m((\lambda nmf.n(mf))n)(\lambda fx.fx)$, then we have:
$\exp \overline{n}\,\overline{m} \equiv \overline{m}(\mathbf{mult}\,\overline{n})\overline{1} \equiv (\lambda fx.f^m x)(\mathbf{mult}\,\overline{n})\overline{1} \equiv (\mathbf{mult}\overline{n})^m\overline{1} \twoheadrightarrow_\beta \overline{n^m}$

(c) To construct a predecessor function, we first define the following functions that form a pair of terms (a, b).

$$\begin{aligned} \mathbf{pair} &= \lambda abs.sab \\ \mathbf{first} &= \lambda p.p\mathbf{t} \\ \mathbf{second} &= \lambda p.p\mathbf{f} \end{aligned}$$

By such way, we can use $p = \mathbf{pair}\, ab$ to denote $p = (a, b)$, and $\mathbf{first}\, p$ and $\mathbf{second}\, p$ to denote the first and second element of the pair p respectively, as $\mathbf{first}(\mathbf{pair}\, ab) = \mathbf{pair}\, ab\mathbf{t} = \mathbf{t}ab = a$ and $\mathbf{second}(\mathbf{pair}\, ab) = \mathbf{pair}\, ab\mathbf{f} = \mathbf{f}ab = b$.
What we want to do is to define a series of pairs like $(\overline{0}, \overline{0}), (\overline{0}, \overline{1}), ..., (\overline{n-1}, \overline{n})$ which the $(n+1)$-th pair contains the predecessor of n. Inductively,

we can define the construction function of the "next" pair in this pair sequence as follows:

$$\mathbf{nextpair} = \lambda p.\mathbf{pair}\,(\mathbf{second}\,p)\,(\mathbf{succ}(\mathbf{second}\,p))$$

With which we can verify $\mathbf{nextpair}(\mathbf{pair}\,\overline{0}\,\overline{0}) = \mathbf{nextpair}(\mathbf{pair}\,\overline{0}\,\overline{1})$, and $\mathbf{nextpair}(\mathbf{pair}\,\overline{n-1}\,\overline{n}) = \mathbf{nextpair}(\mathbf{pair}\,\overline{n}\,\overline{n+1}), n \geq 1$.
Then for the predecessor for n, we only need the $(n+1)$-th pair of the sequence, which happens to be the result of applying the **nextpair** function n times on the first pair $(\overline{0}, \overline{0})$. So we have:

$$\text{pred} = \lambda n.\mathbf{first}\,(n\,\mathbf{nextpair}\,(\mathbf{pair}\,\overline{0}\,\overline{0}))$$

Problem 5. Suppose two binary relations $\to_1$ and $\to_2$ *commute*, that is, $s \to_1 t_1$ and $s \to_2 t_2$ implies that there exists t such that $t_1 \to_2 t$ and $t_2 \to_1 t$. Let $\to_{12}$ be the union of $\to_1$ and $\to_2$. Prove that if $\to_1$ and $\to_2$ satisfy the diamond property, then so is $\twoheadrightarrow_{12}$.

Solution. Say $s \to_{12} u, s \to_{12} v$, then either $s \to_1 u$ or $s \to_2 u$, also $s \to_2 v$ or $s \to_2 v$. If $\exists i \in \{1, 2\}\; s.t.\; s \to_i u, s \to_i v$, then by the diamond property of $\to_i$, $\exists t\; s.t.\; u \to_i t, v \to_i t$, then $u \to_{12} t, v \to_{12} t$. In the other case, when u and v are reduced by different relations, say $u \to_1 t, v \to_2 t$, then by the *commute* property, $\exists t\; s.t.\; u \to_2 t, v \to_1 t$, which implies $u \to_{12} t, v \to_{12} t$. Thus $\to_{12}$ satisfies the diamond property, which means $\twoheadrightarrow_{12}$ satisfies also.

Problem 6. (Optional) Write an algorithm computing the factorial function in Python without using explicit recursion. Sample codes are provided in `lambda.py`. Note that the use of parenthesis in Python for function application is different from the mathematical way. For example, the term xyz used in classes as an abbreviation for $((xy)z)$ should be written as $x(y)(z)$ in Python in order to be consistent with the Python function call convention.