Week 1. Problem set (solutions)

1. Which of the following λ-terms are *closed* [1, §5.1]? Justify your answer.

   (a) $\lambda a.(\lambda b.a\ b)\ a$
   Answer: Closed.
   Justification: Both occurrences of a are bound by the outermost λ-abstraction. The only occurrence of b is bound by the only other λ-abstraction.

   (b) $\lambda d.x\ (\lambda d.d)$
   Answer: Not closed (open).
   Justification: The only occurrence of x is not bound by any λ-abstraction.

   (c) $\lambda x.(\lambda x.x)\ x$
   Answer: Closed.
   Justification: The leftmost occurrence of x is bound by the inner λ-abstraction, while the rightmost is bound by the outer λ-abstraction. The corresponding binders are shown with indices here: $\lambda x_1.(\lambda x_2.x_2)\ x_1$

2. Write down the *call-by-value* evaluation sequence for the following λ-terms. Each step of the evaluation must correspond to a single β-reduction or an α-conversion. You may introduce aliases for subterms.

   (a) $(\lambda x.\lambda y.x)\ (\lambda z.y)\ (\lambda z.z)\ w$
   Solution. First, we need to rename bound variable y, then we can proceed with β-reduction.

$$
\begin{aligned}
& (\lambda x.\lambda y.x)\ (\lambda z.y)\ (\lambda z.z)\ w && (1)\\
& \overset{\alpha}{=} \underline{(\lambda x.\lambda b.x)\ (\lambda z.y)}\ (\lambda z.z)\ w && (2)\\
& \xrightarrow{\beta} \underline{(\lambda b.\lambda z.y)\ (\lambda z.z)}\ w && (3)\\
& \xrightarrow{\beta} \underline{(\lambda z.y)\ w} && (4)\\
& \xrightarrow{\beta} y && (5)
\end{aligned}
$$

   (b) $(\lambda b.\lambda x.\lambda y.b\ y\ x)\ (\lambda x.\lambda y.y)$
   Solution. Here, we stop after one β-reduction, since in **call-by-value**, we do not reduce under λ-abstraction:

$$
\begin{aligned}
& \underline{(\lambda b.\lambda x.\lambda y.b\ y\ x)\ (\lambda x.\lambda y.y)} && (6)\\
& \xrightarrow{\beta} \lambda x.\lambda y.(\lambda x.\lambda y.y)\ y\ x && (7)
\end{aligned}
$$

   (c) $(\lambda s.\lambda z.s\ (s\ z))\ (\lambda b.\lambda x.\lambda y.b\ y\ x)\ (\lambda x.\lambda y.y)$
   Solution. Here, we stop after three β-reductions, since in **call-by-value**, we do not reduce under λ-abstraction:

$$
\begin{aligned}
& \underline{(\lambda s.\lambda z.s\ (s\ z))\ (\lambda b.\lambda x.\lambda y.b\ y\ x)}\ (\lambda x.\lambda y.y) && (8)\\
& \xrightarrow{\beta} \underline{(\lambda z.(\lambda b.\lambda x.\lambda y.b\ y\ x)\ ((\lambda b.\lambda x.\lambda y.b\ y\ x)\ z))\ (\lambda x.\lambda y.y)} && (9)\\
& \xrightarrow{\beta} \underline{(\lambda b.\lambda x.\lambda y.b\ y\ x)\ ((\lambda b.\lambda x.\lambda y.b\ y\ x)\ (\lambda x.\lambda y.y))} && (10)\\
& \xrightarrow{\beta} \lambda x.\lambda y.(\lambda b.\lambda x.\lambda y.b\ y\ x)\ (\lambda x.\lambda y.y)\ y\ x && (11)
\end{aligned}
$$

3. Recall that with Church booleans [1, §5.2] we have the following encoding:

$$\begin{aligned} \mathsf{tru} &= \lambda t.\lambda f.t \\ \mathsf{fls} &= \lambda t.\lambda f.f \end{aligned}$$

   (a) Using only bare λ-calculus (variables, λ-abstraction and application), write down a λ-term for logical equivalence (eq) of two Church booleans. You may **not** use aliases.
   Solution. The following pseudocode corresponds to checking logical equivalence:

```
function EQ(x, y):
  if x
    then y
    else
      if y
        then FALSE
        else TRUE
```

   Assuming arguments are Church-encoded booleans (and so behave like if with a built-in condition), we can construct the following λ-term for EQ:

$$\begin{aligned} \mathsf{eq} &= \lambda x.\lambda y.x\ y\ (y\ \mathsf{fls}\ \mathsf{tru}) \\ &= \lambda x.\lambda y.x\ y\ (y\ (\lambda t.\lambda f.f)\ (\lambda t.\lambda f.t)) \end{aligned}$$

   (b) Verify your implementation of eq by writing down evaluation sequence for the term $\mathsf{eq}\ \mathsf{fls}\ \mathsf{tru}$. You must expand this term and then evaluate **without** aliases.
   Solution. The following sequence of reductions follows *call-by-value* strategy:

$$\begin{aligned}
&\mathsf{eq}\ \mathsf{fls}\ \mathsf{tru} \tag{12}\\
&= \underline{(\lambda x.\lambda y.x\ y\ (y\ (\lambda t.\lambda f.f)\ (\lambda t.\lambda f.t)))\ (\lambda t.\lambda f.f)}\ (\lambda t.\lambda f.t) \tag{13}\\
&\xrightarrow{\beta} \underline{(\lambda y.(\lambda t.\lambda f.f)\ y\ (y\ (\lambda t.\lambda f.f)\ (\lambda t.\lambda f.t)))\ (\lambda t.\lambda f.t)} \tag{14}\\
&\xrightarrow{\beta} \underline{(\lambda t.\lambda f.f)\ (\lambda t.\lambda f.t)}\ ((\lambda t.\lambda f.t)\ (\lambda t.\lambda f.f)\ (\lambda t.\lambda f.t)) \tag{15}\\
&\xrightarrow{\beta} (\lambda f.f)\ (\underline{(\lambda t.\lambda f.t)\ (\lambda t.\lambda f.f)}\ (\lambda t.\lambda f.t)) \tag{16}\\
&\xrightarrow{\beta} (\lambda f.f)\ (\underline{(\lambda f.\lambda t.\lambda f.f)\ (\lambda t.\lambda f.t)}) \tag{17}\\
&\xrightarrow{\beta} \underline{(\lambda f.f)\ (\lambda t.\lambda f.f)} \tag{18}\\
&\xrightarrow{\beta} \lambda t.\lambda f.f \tag{19}
\end{aligned}$$

   The following sequence of reductions follows *call-by-name* strategy:

$$\begin{aligned}
&\mathsf{eq}\ \mathsf{fls}\ \mathsf{tru} \tag{20}\\
&= \underline{(\lambda x.\lambda y.x\ y\ (y\ (\lambda t.\lambda f.f)\ (\lambda t.\lambda f.t)))\ (\lambda t.\lambda f.f)}\ (\lambda t.\lambda f.t) \tag{21}\\
&\xrightarrow{\beta} \underline{(\lambda y.(\lambda t.\lambda f.f)\ y\ (y\ (\lambda t.\lambda f.f)\ (\lambda t.\lambda f.t)))\ (\lambda t.\lambda f.t)} \tag{22}\\
&\xrightarrow{\beta} \underline{(\lambda t.\lambda f.f)\ (\lambda t.\lambda f.t)}\ ((\lambda t.\lambda f.t)\ (\lambda t.\lambda f.f)\ (\lambda t.\lambda f.t)) \tag{23}\\
&\xrightarrow{\beta} \underline{(\lambda f.f)\ ((\lambda t.\lambda f.t)\ (\lambda t.\lambda f.f)\ (\lambda t.\lambda f.t))} \tag{24}\\
&\xrightarrow{\beta} \underline{(\lambda t.\lambda f.t)\ (\lambda t.\lambda f.f)}\ (\lambda t.\lambda f.t) \tag{25}\\
&\xrightarrow{\beta} \underline{(\lambda f.\lambda t.\lambda f.f)\ (\lambda t.\lambda f.t)} \tag{26}\\
&\xrightarrow{\beta} \lambda t.\lambda f.f \tag{27}
\end{aligned}$$

   The result is equal to fls by definition, as expected.

4. Recall that with Church numerals [1, §5.2] we have the following encoding:

$$\begin{aligned}
\mathsf{c}_0 &= \lambda s.\lambda z.z \\
\mathsf{c}_1 &= \lambda s.\lambda z.sz \\
\mathsf{c}_2 &= \lambda s.\lambda z.s\ (s\ z) \\
\mathsf{c}_3 &= \lambda s.\lambda z.s\ (s\ (s\ z)) \\
&\dots
\end{aligned}$$

(a) Using only bare λ-calculus (variables, λ-abstraction and application), write down a single λ-term for each of the following functions on natural numbers. You may **not** use aliases.

i. $n \mapsto 2n + 1$

ii. $n \mapsto 2^{n+1}$

Solution.

i. $f_{\mathrm{i}} = \lambda n.\lambda s.\lambda z.s\ (n\ (\lambda z.s\ (s\ z))\ z)$

ii. $f_{\mathrm{ii}} = \lambda n.(\lambda s.\lambda z.n\ s\ (s\ z))\ (\lambda s.\lambda z.s\ (s\ z))$

(b) Verify each your implementations of the functions above by writing down a *full β-reduction* sequence for each of them, when applied to c_2. You may use aliases.

Solution.

i.

$$\begin{aligned}
&f_{\mathrm{i}}\ \mathsf{c}_2 && (28)\\
&= (\lambda n.\lambda s.\lambda z.s\ (n\ (\lambda z.s\ (s\ z))\ z))\ (\lambda s.\lambda z.s\ (s\ z)) && (29)\\
&\xrightarrow{\beta} \lambda s.\lambda z.s\ ((\lambda s.\lambda z.s\ (s\ z))\ (\lambda z.s\ (s\ z))\ z) && (30)\\
&\xrightarrow{\beta} \lambda s.\lambda z.s\ ((\lambda z.(\lambda z.s\ (s\ z))\ ((\lambda z.s\ (s\ z))\ z))\ z) && (31)\\
&\xrightarrow{\beta} \lambda s.\lambda z.s\ ((\lambda z.s\ (s\ z))\ ((\lambda z.s\ (s\ z))\ z)) && (32)\\
&\xrightarrow{\beta} \lambda s.\lambda z.s\ (s\ (s\ ((\lambda z.s\ (s\ z))\ z))) && (33)\\
&\xrightarrow{\beta} \lambda s.\lambda z.s\ (s\ (s\ (s\ (s\ z)))) && (34)\\
&= \mathsf{c}_5 && (35)
\end{aligned}$$

ii.

$$
\begin{aligned}
& f_{\text{ii}}\ \mathsf{c}_2 && (36)\\
&= (\lambda n.(\lambda s.\lambda z.n\ s\ (s\ z))\ (\lambda s.\lambda z.s\ (s\ z)))\ (\lambda s.\lambda z.s\ (s\ z)) && (37)\\
&= (\lambda n.(\lambda s.\lambda z.n\ s\ (s\ z))\ \mathsf{c}_2)\ \mathsf{c}_2 && (38)\\
&\xrightarrow{\beta} (\lambda s.\lambda z.\mathsf{c}_2\ s\ (s\ z))\ \mathsf{c}_2 && (39)\\
&\xrightarrow{\beta} \lambda z.\mathsf{c}_2\ \mathsf{c}_2\ (\mathsf{c}_2\ z) && (40)\\
&= \lambda z.(\lambda s.\lambda z.s\ (s\ z))\ \mathsf{c}_2\ (\mathsf{c}_2\ z) && (41)\\
&\xrightarrow{\beta} \lambda z.(\lambda z.\mathsf{c}_2\ (\mathsf{c}_2\ z))\ (\mathsf{c}_2\ z) && (42)\\
&\xrightarrow{\beta} \lambda z.\mathsf{c}_2\ (\mathsf{c}_2\ (\mathsf{c}_2\ z)) && (43)\\
&= \lambda z.(\lambda s.\lambda z.s\ (s\ z))\ (\mathsf{c}_2\ (\mathsf{c}_2\ z)) && (44)\\
&\overset{\alpha}{=} \lambda z.(\lambda s.\lambda x.s\ (s\ x))\ (\mathsf{c}_2\ (\mathsf{c}_2\ z)) && (45)\\
&\xrightarrow{\beta} \lambda z.\lambda x.\mathsf{c}_2\ (\mathsf{c}_2\ z)\ ((\mathsf{c}_2\ (\mathsf{c}_2\ z))\ x) && (46)\\
&= \lambda z.\lambda x.(\lambda s.\lambda z.s\ (s\ z))\ (\mathsf{c}_2\ z)\ ((\mathsf{c}_2\ (\mathsf{c}_2\ z))\ x) && (47)\\
&\overset{\alpha}{=} \lambda z.\lambda x.(\lambda s.\lambda y.s\ (s\ y))\ (\mathsf{c}_2\ z)\ ((\mathsf{c}_2\ (\mathsf{c}_2\ z))\ x) && (48)\\
&\xrightarrow{\beta} \lambda z.\lambda x.(\lambda y.\mathsf{c}_2\ z\ (\mathsf{c}_2\ z\ y))\ ((\mathsf{c}_2\ (\mathsf{c}_2\ z))\ x) && (49)\\
&\xrightarrow{\beta} \lambda z.\lambda x.\mathsf{c}_2\ z\ (\mathsf{c}_2\ z\ (\mathsf{c}_2\ (\mathsf{c}_2\ z)\ x)) && (50)\\
&= \lambda z.\lambda x.(\lambda s.\lambda z.s\ (s\ z))\ z\ (\mathsf{c}_2\ z\ (\mathsf{c}_2\ (\mathsf{c}_2\ z)\ x)) && (51)\\
&\overset{\alpha}{=} \lambda z.\lambda x.(\lambda s.\lambda w.s\ (s\ w))\ z\ (\mathsf{c}_2\ z\ (\mathsf{c}_2\ (\mathsf{c}_2\ z)\ x)) && (52)\\
&\xrightarrow{\beta} \lambda z.\lambda x.(\lambda w.z\ (z\ w))\ (\mathsf{c}_2\ z\ (\mathsf{c}_2\ (\mathsf{c}_2\ z)\ x)) && (53)\\
&\xrightarrow{\beta} \lambda z.\lambda x.z\ (z\ (\mathsf{c}_2\ z\ (\mathsf{c}_2\ (\mathsf{c}_2\ z)\ x))) && (54)\\
&= \lambda z.\lambda x.z\ (z\ ((\lambda s.\lambda z.s\ (s\ z))\ z\ (\mathsf{c}_2\ (\mathsf{c}_2\ z)\ x))) && (55)\\
&\overset{\alpha}{=} \lambda z.\lambda x.z\ (z\ ((\lambda s.\lambda w.s\ (s\ w))\ z\ (\mathsf{c}_2\ (\mathsf{c}_2\ z)\ x))) && (56)\\
&\xrightarrow{\beta} \lambda z.\lambda x.z\ (z\ ((\lambda w.z\ (z\ w))\ (\mathsf{c}_2\ (\mathsf{c}_2\ z)\ x))) && (57)\\
&\xrightarrow{\beta} \lambda z.\lambda x.z\ (z\ (z\ (z\ (\mathsf{c}_2\ (\mathsf{c}_2\ z)\ x)))) && (58)\\
&= \lambda z.\lambda x.z\ (z\ (z\ (z\ ((\lambda s.\lambda z.s\ (s\ z))\ (\mathsf{c}_2\ z)\ x)))) && (59)\\
&\overset{\alpha}{=} \lambda z.\lambda x.z\ (z\ (z\ (z\ ((\lambda s.\lambda y.s\ (s\ y))\ (\mathsf{c}_2\ z)\ x)))) && (60)\\
&\xrightarrow{\beta} \lambda z.\lambda x.z\ (z\ (z\ (z\ ((\lambda y.\mathsf{c}_2\ z\ (\mathsf{c}_2\ z\ y))\ x)))) && (61)\\
&\xrightarrow{\beta} \lambda z.\lambda x.z\ (z\ (z\ (z\ (\mathsf{c}_2\ z\ (\mathsf{c}_2\ z\ x))))) && (62)\\
&= \lambda z.\lambda x.z\ (z\ (z\ (z\ ((\lambda s.\lambda z.s\ (s\ z))\ z\ (\mathsf{c}_2\ z\ x))))) && (63)\\
&\overset{\alpha}{=} \lambda z.\lambda x.z\ (z\ (z\ (z\ ((\lambda s.\lambda w.s\ (s\ w))\ z\ (\mathsf{c}_2\ z\ x))))) && (64)\\
&\xrightarrow{\beta} \lambda z.\lambda x.z\ (z\ (z\ (z\ ((\lambda w.z\ (z\ w))\ (\mathsf{c}_2\ z\ x))))) && (65)\\
&\xrightarrow{\beta} \lambda z.\lambda x.z\ (z\ (z\ (z\ (z\ (z\ (\mathsf{c}_2\ z\ x)))))) && (66)\\
&= \lambda z.\lambda x.z\ (z\ (z\ (z\ (z\ (z\ ((\lambda s.\lambda z.s\ (s\ z))\ z\ x)))))) && (67)\\
&\overset{\alpha}{=} \lambda z.\lambda x.z\ (z\ (z\ (z\ (z\ (z\ ((\lambda s.\lambda w.s\ (s\ w))\ z\ x)))))) && (68)\\
&\xrightarrow{\beta} \lambda z.\lambda x.z\ (z\ (z\ (z\ (z\ (z\ ((\lambda w.z\ (z\ w))\ x)))))) && (69)\\
&\xrightarrow{\beta} \lambda z.\lambda x.z\ (z\ (z\ (z\ (z\ (z\ (z\ (z\ x))))))) && (70)\\
&= \mathsf{c}_8 && (71)
\end{aligned}
$$

References

[1] B.C. Pierce. *Types and Programming Languages*. The MIT Press. MIT Press, 2002. ISBN: 9780262162098. URL: https://books.google.ru/books?id=ULT4DwAAQBAJ.