Problem Set 1 (Solution)

Programming Paradigms

Fall 2023

Submitted by: Abu Huraira (a.huraira@innopolis.university)

2nd September, 2023

1. For each of the following λ-terms write down an α-equivalent term without shadowed variables (i.e. where different variables have distinct names):

   Solution:

   a. λx.(λx.x) x

      Answer: λx.(λy.y) x

   b. λc.(λx.c x) c

      Answer: λc.(λx.c x) c

   c. λd.d (λd.d)

      Answer: λe.e (λf.f)

   d. λz.(λz.z) z

      Answer: λz.(λy.y) z

   e. (λx.λy.y) z x

      Answer: (λx.λy.y) z x

2. Write down 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.

   Solution:

   a. (λx.λy.y) y z

      ⇒ (λx.λy.y) y z

      ⇒ (λp.p) z

      ⇒ z

      Answer: z

   b. (λx.λy.y) z (λz.y) w

      ⇒ (λx.λy.y) z (λz.y) w

      ⇒ (λy.y) (λz.y) w

      ⇒ (λz.y) w

⇒ y

Answer: y

c. (λb.λx.λy.b y x) (λx.λy.x)

⇒ λx.λy.(λx.λy.x) y x

⇒ λx.λy.(λu.y) x

⇒ λx.λy.y

Answer: λx.λy.y

d. (**λs**.λz.**s** (**s** (**s** z))) **(λb.λx.λy.b y x)** (λx.λy.y)

⇒ (**λz.**(λb.λx.λy.b y x) ((λb.λx.λy.b y x) ((λb.λx.λy.b y x) **z**))) (λx.λy.y)

⇒ (**λb**.λx.λy.**b** y x) **((λb.λx.λy.b y x) ((λb.λx.λy.b y x) (λx.λy.y))**

⇒ λx.λy.(**λb**.λx.λy.**b** y x) **((λb.λx.λy.b y x) (λx.λy.y))** y x

⇒ λx.λy.(**λx**.λy.(λb.λx.λy.b y x) (λx.λy.y) y x) **y** x

⇒ λx.λy.(**λp.**(λb.λx.λy.b y x) (λx.λy.y) **p** y) x

⇒ λx.λy.(**λb**.λq.λy.**b** y q) **(λq.λy.y)** x y

⇒ λx.λy.(**λq**.λy.(λq.λy.y) y q) x y

⇒ λx.λy.(**λy.**(λr.λy.y) y x) **y**

⇒ λx.λy.(**λr**.λy.y) **y** x

⇒ λx.λy.(λs.s) x

Answer: λx.λy.(λs.s) x

e. (λs.λz.s (s z)) (λs.λz.s (s z)) (λb.λy.λx.b x y) (λy.λx.x)

⇒ (**λs**.λz.**s** (**s** z)) **(λs.λz.s (s z))** (λb.λy.λx.b x y) (λy.λx.x)

⇒ (**λz.**(λs.λz.s (s **z**)) ((λs.λz.s (s z)) **z**)) (λb.λy.λx.b x y) (λy.λx.x)

⇒ (**λs.**λz.**s** (**s** z)) (**(λs.λz.s (s z)) (λb.λy.λx.b x y)**) (λy.λx.x)

⇒ (**λz.**(λs.λz.s (s z)) (λb.λy.λx.b x y) ((λs.λz.s (s z)) (λb.λy.λx.b x y) **z**)) (λy.λx.x)

⇒ **λs**.λz.**s** (**s** z)) **(λb.λy.λx.b x y)** ((λs.λz.s (s z)) (λb.λy.λx.b x y) (λy.λx.x))

⇒ (**λz**.(λb.λy.λx.b x y) ((λb.λy.λx.b x y) **z**)) ((λs.λz.s(s z)) (λb.λy.λx.b x y) (λy.λx.x))

⇒ (**λb**.λy.λx.b x y) **((λb.λy.λx.b x y) ((λs.λz.s (s z))** (**λb**.λy.λx.b x y) **(λy.λx.x)**))

⇒ λy.λx.(**λb**.λy.λx.**b** x y) **((λs.λz.s (s z)) (λb.λy.λx.b x y) (λy.λx.x))** x y

⇒ λy.λx.(**λy**.λx.(λs.λz.s (s z)) (λb.λy.λx.b x y) (λy.λx.x) x y) **x** y

⇒ λy.λx.(**λu**.(λs.λz.s (s z)) (λb.λy.λx.b x y) (λy.λx.x) **u** x) **y**

⇒ λy.λx.(**λs**.λz.**s** (**s** z)) **(λb.λy1.λx.b x v)** (λv.λx.x) y x

⇒ λy.λx.(**λz**.(λb.λv.λx.b x v) ((λb.λv.λx.b x v) **z**)) (λv.λx.x) y x

⇒ λy.λx.(**λb**.λv.λx.b x v) **((λb.λv.λx.b x y1) (λv.λx.x))** y x

⇒ λy.λx.(**λv**.λx.(λb.λv.λx.b x v) (λv.λx.x) x **v**) y x

⇒ λy.λx.(**λx**.(λb.λw.λx.b x w) (λw.λx.x) x y) **x**

⇒ λy.λx.(**λb**.λw.λx.b x w) **(λw.λx.x)** x y

⇒ λy.λx.(**λw**.λx.(λw.λx.x) x **w**) x y

⇒ λy.λx.(λp.(**λp**.λx.x) **p** x) y

⇒ λy.λx.(**λp**.λx.x) **y** x

⇒ λy.λx.(**λx**.x) **x**

⇒ λy.λx.x

Answer: λy.λx.x

3. Recall that with Church booleans we have the following encoding:

`tru` = λt.λf.t

`fls` = λt.λf.f

a. Using only bare λ-calculus (variables, λ-abstraction and application), write down a λ-term for logical NAND (`nand`) of two Church booleans. You may **not** use aliases.
b. Verify your implementation of nand by writing down evaluation sequence for the term nand fls tru. You must expand this term and then evaluate **without** aliases.

Solution:

a. From the slide we can derive AND as:
   λa.λb.a b fls
   We can derive NOT as:
   λa.a fls tru

   Thus, we can obtain NAND as:
   - ⇨ λa.λb.(NOT (AND a b))
   - ⇨ λa.λb.(λp.p fls tru)(AND a b)
   - ⇨ λa.λb.and a b fls tru
   - ⇨ λa.λb.(λp.λq.p q p) a b fls tru
   - ⇨ λa.λb.(λq.a q a) b fls tru
   - ⇨ λa.λb.a b a fls tru
   - ⇨ λa.λb.a b a (λx.λy.y) tru
   - ⇨ **λa.λb.a b a (λx.λy.y) (λx.λy.x)**

   Answer: λa.λb.a b a (λx.λy.y) (λx.λy.x)

   Let us write down the evaluation sequence for the term **nand fls tru**.

   - ⇨ NAND fls tru
   - ⇨ λa.λb.a b a (λx.λy.y) (λx.λy.x) fls tru
   - ⇨ (λb.fls b fls (λx.λy.y) (λx.λy.x)) tru
   - ⇨ fls tru fls (λx.λy.y) (λx.λy.x)
   - ⇨ (λx.λy.y) tru fls (λx.λy.y) (λx.λy.x)
   - ⇨ (λy.y) fls (λx.λy.y) (λx.λy.x)
   - ⇨ fls (λx.λy.y) (λx.λy.x)
   - ⇨ (λx.λy.y) (λx.λy.y) (λx.λy.x)
   - ⇨ (λy.y) (λx.λy.x)
   - ⇨ λx.λy.x
   - ⇨ **tru**

   We can observe the implemented NAND λ-term returns the correct output.

4. Recall that with Church numerals we have the following encoding:

c0 = λs.λz.z

c1 = λs.λz.s z

c2 = λs.λz.s (s z)

c3 = λs.λz.s (s (s z))

. . .

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. $n \to 2(n+1)$

ii. $n \to (n+1)^2$

iii. $n \to n^{n+2}$

iv. $n \to 2^{2^n}$

b. Verify each your implementations of the functions above by writing down evaluation sequence for each of them, when applied to c2. You may use aliases.

Solution:

i. Let's recall definition of **inc** and **times** from lab:

inc := λn. λs. λz. s (n s z)

plus := λm. λn. λs. λz. m s (n s z)

times := λm. λn. m (plus n) c0

For $n \to 2\ (n+1)$, we can use

times c2 (plus n c1)

⇨ λm. λn. λs. λz. m s (n s z) λp.λq.p q (λm. λn. λs. λz. m s (n s z) r λs.λz.z)

Answer: **λm. λn. λs. λz. m s (n s z) λp.λq.p q (λm. λn. λs. λz. m s (n s z) r λs.λz.z)**

ii. times (inc n) (inc n)
iii. …
iv. …