Computer Science & Engineering Department
I. I. T. Kharagpur

Principles of Programming Languages: CS40032

Elective

Assignment – 1: λ-Calculus *Marks: 25*

Assign Date: *17^{th} January, 2020* Submit Date: *23:55, 24^{th} January, 2020*

Instructions: Please solve the questions using pen and paper and scan the images. Every image should contain your roll number and name.

1. Fully parenthesize the following λ-expressions: **[1.5 * 3 = 4.5]**

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

   (b) $(\lambda x.\ x\ z)\ \lambda y.\ w\ \lambda w.\ w\ y\ z\ x$

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

   BEGIN SOLUTION

λx.xz λy.xy	→ **(λx.((x z) (λy.(x y))))**
(λx.xz) λy.w λw.wyzx	→ **((λx.(x z)) (λy.(w (λw.((((w y) z) x))))))**
λx.xy λx.yx	→ **(λx.((x y) (λx.(y x))))**

   END SOLUTION

2. Mark the free variables in the following λ-expressions: **[1.5 * 3 = 4.5]**

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

   (b) $(\lambda x.\ x\ z)\ \lambda y.\ w\ \lambda w.\ w\ y\ z\ x$

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

   BEGIN SOLUTION

λx.x z λy.x y	→ **(λx.((x z) (λy.(x y))))**
(λx. x z) λy. w λw. w y z x	→ **((λx.(x z)) (λy.(w (λw.((((w y) z) x))))))**
λx. x y λx. y x	→ **(λx.((x y) (λx.(y x))))**

   END SOLUTION

3. Prove the following using encoding in λ-calculus: **[2 * 8 = 16]**

   (a) $NOT(NOT\ TRUE) = TRUE$

   Given:

$$NOT = \lambda x.\ ((x\ FALSE)\ TRUE)$$

$$TRUE = \lambda x.\ \lambda y.\ x$$

$$FALSE = \lambda x.\ \lambda y.\ y$$

   BEGIN SOLUTION

not (not true)	**// replacing 1st not w/ encoding**
= λx.((x false) true) (not true)	**// β-reduction: x → not true**
= ((not true) false) true	**// replacing not w/ encoding**
= ((λx.((x false) true) true) false) true	**// β-reduction: x → true**
= (((true false) true) false) true	**// replace true w/ encoding**
= ((((λx.λy.x) false) true) false) true	**// β-reduction: 1st x → false**
= (((λy.false) true) false) true	**// β-reduction: y → true**
= ((false) false) true	**// replace false w/ encoding**
= ((λx.λy.y) false) true	**// β-reduction: x → false**
= (λy.y) true	**// β-reduction: y → true**
= true	**// not (not true) = true**

   END SOLUTION

(b) $OR\ FALSE\ TRUE = TRUE$

Given:

$$OR = \lambda x.\ \lambda y.\ ((x\ TRUE)\ y)$$

$$TRUE = \lambda x.\ \lambda y.\ x$$

$$FALSE = \lambda x.\ \lambda y.\ y$$

BEGIN SOLUTION

```
or false true                              // replacing or w/ encoding
= λx. λy. ((x true) y) false true          // β-reduction: x → false
= λy. ((false true) y) true                // β-reduction: y → true
= (false true) true                        // replace 1st false w/ encoding
= ((λx.λy.y) true) true                    // β-reduction: x → false
= (λy.y) true                              // β-reduction: y → true
= true                                     // or false true = true
```

END SOLUTION

(c) $SUCC\ 2 = 3$

Given:

$$2 = \lambda f.\ \lambda y.\ f\ (f\ y)$$

$$3 = \lambda f.\ \lambda y.\ f\ (f\ (f\ y))$$

$$SUCC = \lambda z.\ \lambda f.\ \lambda y.\ f\ (z\ f\ y)$$

BEGIN SOLUTION

```
succ 2                                     // replacing succ w/ encoding
= (λz.λf.λy.f (z f y)) 2                   // β-reduction: z → 2
= λf.λy.f (2 f y)                          // expanding 2 w/ encoding
= λf.λy.f ((λf.λy.f (f y)) f y)            // β-reduction: 1st f → f
= λf.λy.f ((λy.f (f y)) y)                 // β-reduction: 1st y → y
= λf.λy.f  (f (f y))                       // apply encoding for 3
= 3                                        // succ 2 = 3
```

END SOLUTION

(d) $(Y\ FACT)\ 2 = 2$

Given:

$$Y = \lambda f.\ (\lambda x.\ f\ (x\ x))\ (\lambda x.\ f\ (x\ x))$$

$$FACT = \lambda f.\ \lambda n.\ IF\ n = 0\ THEN\ 1\ ELSE\ n\ *\ (f\ (n\ -\ 1))$$

BEGIN SOLUTION

```
Given:
    Y = λf.(λx.f (x x)) (λx.f (x x))
    fact = λf. λn.if n = 0 then 1 else n * (f (n-1))
Proof:
    (Y fact) 2                                     // replacing Y w/ encoding
    = (λf.(λx.f (x x)) (λx.f (x x)) fact) 2        // β-reduction: 1st f → fact
    = (λx.fact (x x)) (λx.fact (x x)) 2         // β-reduction: 1st x → λx.fact (x x)
    = (fact ((λx.fact (x x)) (λx.fact (x x)))) 2
            // apply encoding for (Y fact)
            // ((λx.fact (x x)) (λx.fact (x x))) → (Y fact)
            // we know this is the encoding for (Y fact) from 3rd line of proof
    = (fact (Y fact)) 2                            // apply encoding for fact
    = (λf. λn.if n = 0 then 1 else n * (f (n-1)) (Y fact) 2
                                                   // β-reduction: 1st f →  (Y fact)
    = (λn.if n = 0 then 1 else n * ((Y fact) (n-1))) 2// β-reduction: n → 2
    = if 2=0 then 1 else 2 * ((Y fact) (2-1))      // apply if
    = 2 * ((Y fact) 1)                             // showed in class (Y fact) 1 = 1
    = 2 * 1                                        // apply *
    = 2
```

END SOLUTION

(e) Show: $exp\ \overline{0}\ \overline{n} = \overline{1}$

Given:

$$exp = \lambda m.\lambda n.(\ m\ \ n)$$

(f) Solve: $add\ \overline{6}\ \overline{2}$

Given: $add = \lambda n.\lambda m.\lambda f.\lambda x.\ n\ f\ (m\ f\ x)$

(g) $IF\ FALSE\ THEN\ x\ ELSE\ y = y$

Given:

$$IF\ a\ THEN\ b\ ELSE\ c = a\ b\ c$$

$$TRUE = \lambda x.\ \lambda y.\ x$$

$$FALSE = \lambda x.\ \lambda y.\ y$$

(h) Prove: add and mul are associative

Given:

$mul = \lambda n.\lambda m.\lambda x.\ (n\ (m\ x))$

$mul = \lambda n.\lambda m.\lambda f.\ n\ (m\ f)$

$add = \lambda n.\lambda m.\lambda f.\lambda x.\ n\ f\ (m\ f\ x)$