Homework: Lambda Calculus

1. Understand lambda calculus, theory of functional programming
2. Understand β-reduction, church encoding

Instructions:

- Total points: 53 pt
- Early deadline: Oct 24 (Wed) 2018 at 6:00 PM; Regular deadline: Oct 26 (Fri) 2018 at 6:00 PM (or till TAs start grading the homework)
- Submit one pdf file to Canvas under Assignments, Homework 6. You are encouraged to use latex. But we will accept a scanned copy as well.

Questions:

1. (9 pt) Perform β-reduction for the following λ expressions.

   (a) (3 pt) $((\lambda(x)\, x)((\lambda(y)\, y)(((\lambda(v)(\lambda(w)\, w))\, a)\, b)))$

   (b) (3 pt) $(((\lambda(x)(\lambda(y)(x\, y)))((\lambda(w)\, w)\, a))\, b)$

   (c) (3 pt) $(((\lambda(x)(\lambda(y)(y\, y)))(\lambda(a)\, a))\, b)$

   Sol

   (a) (3pt)

$$
\begin{aligned}
& ((\lambda(x)x)((\lambda(y)\, y)(((\lambda(v)(\lambda(w)w))\, a)\, b))) && (1)\\
= \; & ((\lambda(x)x)((\lambda(y)\, y)((\lambda(w)w)\, b))) && (2)\\
= \; & ((\lambda(x)x)((\lambda(y)\, y)\, b)) && (3)\\
= \; & ((\lambda(x)x)b) && (4)\\
= \; & b && (5)
\end{aligned}
$$

   (b) (3 pt)

$$
\begin{aligned}
& (((\lambda(x)(\lambda(y)(x\, y)))((\lambda(w)w)a))b) && (1)\\
= \; & (((\lambda(x)(\lambda(y)(x\, y)))a)b) && (2)\\
= \; & ((\lambda(y)(a\, y))b) && (3)\\
= \; & (a\, b) && (4)
\end{aligned}
$$

(c) (3pt)

$$
\begin{aligned}
& (((\lambda(x)(\lambda(y)(y\,y)))(\lambda(z)(z\,z)))b) && (1)\\
= \; & ((\lambda(y)(y\,y))b) && (2)\\
= \; & (b\,b) && (3)
\end{aligned}
$$

2. (6 pt) The goal of this problem is to help you understand the evaluation order of lambda calculus. In the following, show the steps of β-reduction for the lambda expression using two types of evaluation orders

$$((\lambda(x)\,p)((\lambda(y)(y\,y))(\lambda(z)(z\,z))))$$

Sol

(a) (5pt)Define the logic Boolean operations

$$
\begin{aligned}
& ((\lambda(x)p)((\lambda(y)(y\,y))(\lambda(z)(z\,z)))) && (1)\\
= \; & p && (2)
\end{aligned}
$$

(b) (5pt)

$$
\begin{aligned}
& ((\lambda(x)p)((\lambda(y)(y\,y))(\lambda(z)(z\,z)))) && (1)\\
= \; & ((\lambda(x)p)((\lambda(z)(z\,z))\;(\lambda(z)(z\,z)))) && (2)\\
= \; & ((\lambda(x)p)((\lambda(z)(z\,z))\;(\lambda(z)(z\,z)))) && (3)\\
= \; & ((\lambda(x)p)((\lambda(z)(z\,z))\;(\lambda(z)(z\,z)))) && (4)\\
= \; & ... && (5)
\end{aligned}
$$

3. (3 pt) Define the logic Boolean operations of *or a b* using *true*, *false* and *ite* given in the lecture. **Sol**

(a) OR:

```
(ite a
        true
        (ite  b  true  false))
```

4. (20 pt) Using the Church numeral encoding and also *succ*, *true*, *false* provided in the lecture, answer the following two questions:

(a) (5 pt) What is the result of $((\lambda(z)((three\,f)\,z))\,two)$?

(b) Suppose we define *third*: $(\lambda(x)(\lambda(y)(\lambda(z)\,z)))$ and *g*: $(\lambda(n)((n\,third)\,true))$, what is the result of:

i. (4 pt) $(g\,zero)$

ii. (3 pt) $(g\,one)$

iii. (3 pt) $(g\,two)$

iv. (5 pt) What mathematical/logical operation is computed by g?

Sol

(a) (5pt) This function will apply f on two three times. $(f(f(f\ two))$

(b) i. (4 pt) $(((\lambda(f)(\lambda(x)x))third)true)$ reduces to:$((\lambda(x)x)true)$ which is true

ii. (3 pt) It will always be false $(((\lambda(f)(\lambda(x)(f\ \ x)))third)true)$ beta reduces to $(third\ \ true)$ which written out as $((\lambda(x)(\lambda(y)(\lambda(z)z)))true)$ this beta reduces to $(\lambda(y)(\lambda(z)z))$ which is the semantics of false. Making n larger will just increase the number of times third is given one argument which will always output false.

iii. (3 pt) As it was stated in the previous answer, the function third is applied three times with the same result, false.

iv. (5 pt) $(= n\ zero)$

5. (15 pt) Given:
true: $(\lambda(x)(\lambda(y)\,x))$
false: $(\lambda(x)(\lambda(y)\,y))$
g: $(\lambda(n)((n(\lambda(x)\,false))\,true))$
zero: $(\lambda(f)(\lambda(x)\,x))$
one: $(\lambda(f)(\lambda(x)(f\,x)))$.

(a) (3 pt) What is the result of $(g\ zero)$?

(b) (3 pt) What is the result of $(g\ one)$?

(c) (3 pt) What computation does g performs?

(d) (6 pt) Suppose we define *ite*: $(\lambda(c)(\lambda(t)(\lambda(e)((c\,t)\,e))))$ to represent if then else ((if c t) e). Write a lambda calculus expression that uses g and *ite* to define *IsEqual* that tests if two numbers m and n have the equal values.

Sol.

(a) **true**

(b) **false**

(c) **check if the argument is zero**

(d) **Sol 1**

```
f = (λ(m) (λ(n)
(((ite (g m))
(((ite (g n)) true) false))
((f (sub m 1)) (sub n 1)))))
```

Sol 2

$(\lambda(m)(\lambda(n)(((if(g\ (sub\,m\,n))(g\ (sub\,n\,m)))\ false)))$