FLOLAC 2012, Lambda Calculus
Assignment 1, Solution

0. Identify the free and bound variables in the following term:

```
 lambda u. lambda x. [x (lambda y.x y) y u]
                                    ^
                     this is the only free var.
```

The other y is bound by the lambda y. Remember that free/bound is dependent on the term in question. Here's another example:

(lambda x. x(lambda x.x)) t

This term reduces to t(lambda x.x). The outer x is FREE relative to the body of the entire lambda term.

1. Determine if the following pairs of terms are alpha-equivalent.

a. lambda x.lambda y. (x y x) and
   lambda x.lambda y. (y x y)

   NO - the first is 1 2 1 , the second is 2 1 2

b. lambda x. lambda y. x (lambda y.x) y and
   lambda a. lambda b. a (lambda u.a) b

   YES - the inner lambda y can be converted to lambda u

For the rest of the problems, find the beta-normal forms of the following terms. Remember: you can rename bound variables to avoid clash, but never free variables.

2. (lambda x.lambda y. (y x)) u (lambda x.x)
  -> (lambda y. (y u)) (lambda x.x)
  -> (lambda x.x) u
  -> u

>> WARNING: I will sometimes combine up to 2 steps (-->) in the following solutions. I'll also use the book's notation lambda xy to mean lambda x.lambda y - use the form you're comfortable with.

3. given S, K, I as they are in the handout, find the normal forms of II, KIK, SIK, and S(IK). Application associates to the left, and the order is important.

>> S = (lambda xyz.(x z)(y z))
   K = (lambda xy.x)

I = lambda x.x

II = (lambda x.x) (lambda y.y) -> lambda y.y = I

KIK = [(lambda xy.x) (lambda u.u)] K
-> (lambda y.lambda u.u) K
-> lambda u.u
(there's no y in the body (lambda u.u), so it doesn't change)

SIK = (lambda xyz.(x z)(y z)) I K
-> (lambda yz.(I z) (y z)) K
-> lambda z.(I z) (K z)   (I will choose to reduce (I z) first
= lambda z.([lambda u.u] z) ([lambda ab.a] z)
-> lambda z.z ([lambda ab.a] z)
-> lambda z.z (lambda b.z)

S(IK) = S ((lambda a.a) K)
-> S K
= (lambda xyz. x z (y z)) (lambda ab.a)
-> lambda yz. (lambda ab.a) z (y z)
--> lambda yz.z
(here, I combined two steps,
(lambda ab.a) z (y z) -> (lambda b.z) (y z) -> z)

4. given A = lambda m.lambda n.lambda f.lambda x. m f (n f x),
T = lambda f.lambda x.f (f x)

>> ATT = (lambda mnfx.m f (n f x)) T T
--> lambda fx. T f (T f x)
= lambda fx. [(lambda gy.g (g y)) f (T f x)] - I choose call by name
-> lambda fx. [(lambda y.f (f y)) (T f x)]
-> lambda fx. f (f (T f x))
= lambda fx. f (f ([lambda gy.g (g y)] f x))
-> lambda fx. f (f ([lambda y.f (f y)] x))
-> lambda fx. f (f (f (f x)))

** Call by value alternative: **
... lambda fx. T f (T f x)
= lambda fx. T f ([lambda gy.g (g y)] f x)
-> lambda fx. T f ([lambda y.f (f y)] x)
-> lambda fx. T f (f (f x))
= lambda fx. (lambda gy. g (g y)) f (f (f x)) - careful with the ()'s!
-> lambda fx. (lambda y. f (f y)) (f (f x))
-> lambda fx. f (f (f (f x)))

find the normal form of (A T T).
Conjecture what would happen if T = lambda f.lambda x.f (f (f x)).

(T is the "Church representation" of the number 2, and A represents
Addition - Church originally formulated the lambda calculus as a

symbolic foundation for all of mathematics.)

5. Given T = lambda x. lambda y. x
F = lambda x. lambda y. y
A = lambda p.lambda q. (p q F)   (where F is as above)

Find the normal forms of:

a. (A F F)
b. (A F T)
c. (A T F)
d. (A T T)

AFF = (lambda pq.p q F) F F
-> (lambda q.F q F) F
-> F F F
= (lambda xy.y) F F
-> (lambda y.y) F
-> F

AFT = (lambda pq.p q F) F T
--> F T F
= (lambda xy.y) T F
--> F

ATF = (lambda pq.p q F) T F
--> T F F
= (lambda xy.x) F F
-> (lambda y.F) F
-> F

ATT = (lambda pq.p q F) T T
--> T T F
= (lambda xy.x) T F
--> T

As you should know by now, A is the lambda combinator for boolean "and"

In class I defined and as (lambda pq. if p q F), but this
beta-reduces to (lambda pq.p q f) since if is just (lambda abc.a b c)

6. Let T and F be as they were in the above problem, and let
N = lambda x. (x F T)

what are the normal forms of

a. (N F)
b. (N T)
c. (N (N T))

>> NF = lambda x.(x F T) F
-> F F T
= (lambda xy.y) F T
-> (lambda y.y) T
-> T

NT = lambda x.(x F T) T
-> T F T
= (lambda xy.x) F T
--> F

N(NT) --> NF (by reduction above, which shows that NT-->F)
---> T (by first reduction above, which shows that NF--->T)