Bryan Duong

013104973

Fall 2018

CECS 424 Assignment 5

1. Evaluate the following λ expressions
   1. ((λx. λy.(y x) λp.λq.p) λi.i)

(λy.(y λp.λq.p) λi.i)

(λi.i λp.λq.p)

λp.λq.p

* 1. (((λx.λy.λz.((x y) z) λf.λa.(f a)) λi.i) λj.j)

(((λy.λz.((λf.λa.(f a) y) z)) λi.i) λj.j)

(((λy.λz.(λa.(y a)) z) λi.i) λj.j)

((λy.λz.(y z) λi.i) λj.j)

(λz.(λi.i z) λj.j)

(λi.i λj.j)

λj.j

* 1. (λh.((λa.λf.(f a) h) h) λf.(f f))

(λh.(λf.(f h) h) λf.(f f))

(λh.(h h) λf.(f f))

(λf.(f f) λf.(f f)) – Infinite Loop

* 1. ((λp.λq.(p q) (λx.x λa.λb.a)) λk.k)

(λq.((λx.x λa.λb.a) q) λk.k)

(λq.(λa.λb.a q) λk.k)

(λq.λb.q λk.k)

λb.λk.k

* 1. (((λf.λg.λx.(f (g x)) λs.(s s)) λa.λb.b) λx.λy.x)

((λg.λx.(λs.(s s) (g x)) λa.λb.b) λx.λy.x)

((λg.λx.((g x) (g x)) λa.λb.b) λx.λy.x)

(λx.((λa.λb.b x) (λa.λb.b x)) λx.λy.x)

(λa.λb.b λx.λy.x) (λa.λb.b λx.λy.x)

(λa.λx.λy.x) (λa.λx.λy.x)

1. Define a function:

def make triplet = …

which is like make pair but constructs a triplet from a sequence of three arguments so that any one of the arguments may be selected by the subsequent application of a triplet to a selector function. Define selector functions:

def triplet first = …

def triplet second = …

def triplet third = …

which will select the first, second or third item from a triplet respectively.

def make triplet = λf.λs.λt.λfunc.(((func f) s) t)

def triplet first = λfirst.λsecond.λthird.first

def triplet second = λfirst.λsecond.λthird.second

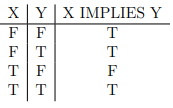
def triplet third = λfirst.λsecond.λthird.third

1. Use α conversion to ensure unique names in the expressions in each of the following λ expressions:
   1. λx.λy.(λx.y λy.x) 🡪 λx’.λy’.(λx.y’ λy.x’)
   2. λx.(x (λy.(λx.x y) x)) 🡪 λx.(x (λy.(λx’.x’ y) x))
   3. λa.(λb.a λb.(λa.a b)) 🡪 λa.(λb.a λb’.(λa’.a’ b’))
   4. (λfree.bound λbound.(λfree.free bound)) 🡪

(λfree.bound’ λbound.(λfree’.free’ bound))

* 1. λp.λq.(λr.(p (λq.(λp.(r q)))) (q p)) 🡪 λp.λq.(λr.(p (λq’.(λp’.(r q’)))) (q p))

1. The boolean operation implication is defined by the following truth table:



Define a λ calculus representation for implication:

x ? y : True

def implies = λx.λy.(((cond y) True) x)

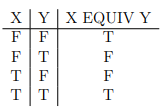
def implies = λx.λy.(((λe1.λe2.λc.((c e1) e2) y) True) x)

def implies = λx.λy.(((λe2.λc.(c y) e2) True) x)

def implies = λx.λy.((λc.(c y) True) x)

def implies = λx.λy.((x y) True)

1. The boolean operation equivalence is defined by the following truth table:



Define a λ calculus representation for equivalence:

x ? y : Not y

def equiv = λx.λy.(((cond y) Not y) x)

def equiv = λx.λy.(((λe1.λe2.λc.((c e1) e2) y) Not y) x)

def equiv = λx.λy.(((λe2.λc.(c y) e2) Not y) x)

def equiv = λx.λy.((λc.(c y) Not y) x)

def equiv = λx.λy.((x y) Not y)

1. Write a function that finds the product of the numbers between n and one:

prod n = ...

in λ calculus is equivalent to:

n \* n-1 \* n-2 \* ... \* 1

in normal arithmetic. Assume the function isone n is defined

prod n =

if isone n then 1

else mult n (prod pred n)