1. (10 points) Evaluate the following λ expressions:

(a) (2 points) ((λx.λy.(y x) λp.λq.p) λi.i)

=>((λy.(y λp.λq.p)) λi.i)

=>λi.i λp.λq.p

=> λp.λq.p

b) (((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)

=> (λz.((λf.λa.(f a) λi.i) z) λj.j)

=> ((λf.λa.(f a) λi.i) λj.j)

=> ((λa.(λi.i a)) λj.j)

=> λi.i λj.j

=> λj.j

c) (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)

=> λf.(f f) λf.(f f)

=> λf.(f f) λf.(f f) … infinte loop

d)((λp.λq.(p q) (λx.x λa.λb.a)) λk.k)

=> ((λp.λq.(p q) λa.λb.a) λk.k)

=>(λq.(λa.λb.a q) λk.k)

=> λa.λb.a λk.k

=> λb.λk.k

e) (((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) // x => λa.λb.b g => λx.λy.x

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

=> (λb.b λb.b)

=> λb.b

2) def make_triplet = λfirst.λsecond.λthird.λs.(((s first) second) third)

def triplet_first = λfirst.λsecond.λthird.first

def triplet_second = λfirst.λsecond.λthird.second

def triplet_third = λfirst.λsecond.λthird.third

3)
(a) (2 points) λx.λy.(λx.y λy.x) // collapsing in λx in λx and λy in λy => chang λx and λy
=> λx.λy.(λa.y λb.x)

(b) (2 points) λx.(x (λy.(λx.x y) x)) // collapsing in λx.x with λx outside => change λx.x
=> λx.(x (λy.(λa.a y) x))

(c) (2 points) λa.(λb.a λb.(λa.a b)) // collapsing in λa.a with λa outside => change λa.a
=> λa.(λb.a λb.(λx.x b))

(d) (2 points) (λfree.bound λbound.(λfree.free bound))
=>(λfree.bound λbound.bound)
=> λbound.bound => no change needed

(e) (2 points) λp.λq.(λr.(p (λq.(λp.(r q)))) (q p)) // collapsing in λq. with λq. q and λp λp
=> λp.λq.(λr.(p (λa.(λb.(r a)))) (q p))

4) def implies = λx.λy.((x y) true)

Implies	(x y) true	=>
False False	(False False) True	True
False True	(False True) True	True
True False	(True False) True	False
True True	(True True) True	True

5) def equiv = λx.λy.((x y) (not y))

equiv	(x y) not y	=>
False False	(False False) (not False)	True
False True	(False True) (not True)	False
True False	(True False) (not False)	False

True True	(True True) (not True)	True

6) rec prod n =
    if isone n then one
    else mult n (prod (pred n))